



## **SDK6 API USB**

Version: 1.7

21 Aug 2015



Confidentiality Notice:

Copyright © 2015 Ambarella, Inc.

The contents of this document are proprietary and confidential information of Ambarella, Inc.

The material in this document is for information only. Ambarella assumes no responsibility for errors or omissions and reserves the right to change, without notice, product specifications, operating characteristics, packaging, ordering, etc. Ambarella assumes no liability for damage resulting from the use of information contained in this document. All brands, product names, and company names are trademarks of their respective owners.

#### **US**

3101 Jay Street  
Ste. 110  
Santa Clara, CA 95054, USA  
Phone: +1.408.734.8888  
Fax: +1.408.734.0788

#### **Hong Kong**

Unit A&B, 18/F, Spectrum Tower  
53 Hung To Road Kwun Tong, Kowloon Phone:  
+85.2.2806.8711

#### **Korea**

6 Floor, Hanwon-Bldg.  
Sunae-Dong, 6-1, Bundang-Gu SeongNam-City, Kyunggi-Do  
Republic of Korea 463-825  
Phone: +031.717.2780

#### **China - Shanghai**

9th Floor, Park Center  
1088 Fangdian Road, Pudong New District  
Shanghai 201204, China

#### **Taiwan**

Suite C1, No. 1, Li-Hsin Road 1  
Science-Based Industrial Park  
Hsinchu 30078, Taiwan  
Phone: +886.3.666.8828  
Fax: +886.3.666.1282

#### **Japan - Yokohama**

Shin-Yokohama Business Center Bldg. 5th Floor  
3-2-6 Shin-Yokohama, Kohoku-ku,  
Yokohama, Kanagawa, 222-0033,  
Japan Phone: +81.45.548.6150

#### **China - Shenzhen**

Unit E, 5th Floor  
No. 2 Finance Base  
8 Ke Fa Road  
Shenzhen, 518057, China  
Phone: +86.755.3301.0366



## Contents

|   |           |
|---|-----------|
| <b>CONTENTS.....</b>                                      | <b>II</b> |
| <b>1. OVERVIEW.....</b>                                   | <b>1</b>  |
| 1.1    INTRODUCTION .....                                 | 2         |
| <b>2. USB DEVICE APIs .....</b>                           | <b>3</b>  |
| 2.1.1 <i>Data Structures</i> .....                        | 4         |
| 2.1.2 <i>Callback functions</i> .....                     | 8         |
| 2.1.3 <i>AmbaUSB_DeviceSystemSetup</i> .....              | 17        |
| 2.1.4 <i>AmbaUSB_System_GetVcsVersion</i> .....           | 19        |
| 2.1.5 <i>AmbaUSBD_System_SetMemoryPool</i> .....          | 20        |
| 2.1.6 <i>AmbaUSBD_System_ChargeDetection</i> .....        | 21        |
| 2.1.7 <i>AmbaUSBD_System_ControlRead</i> .....            | 22        |
| 2.1.8 <i>AmbaUSBD_System_ControlWrite</i> .....           | 23        |
| 2.1.9 <i>AmbaUSBD_System_DataContactDetection</i> .....   | 24        |
| 2.1.10 <i>AmbaUSBD_System_DumpRegisters</i> .....         | 25        |
| 2.1.11 <i>AmbaUSBD_System_GetConnectSpeed</i> .....       | 26        |
| 2.1.12 <i>AmbaUSBD_System_GetDataConnWithVbus</i> .....   | 27        |
| 2.1.13 <i>AmbaUSBD_System_GetVbusPin</i> .....            | 28        |
| 2.1.14 <i>AmbaUSBD_System_GetVbusPinStatus</i> .....      | 29        |
| 2.1.15 <i>AmbaUSBD_System_GetVbusStatus</i> .....         | 30        |
| 2.1.16 <i>AmbaUSBD_System_IsConfigured</i> .....          | 31        |
| 2.1.17 <i>AmbaUSBD_System_RegisterVbusCallback</i> .....  | 32        |
| 2.1.18 <i>AmbaUSBD_System_RegisterVendorRequest</i> ..... | 34        |
| 2.1.19 <i>AmbaUSBD_System_ClassHook</i> .....             | 37        |
| 2.1.20 <i>AmbaUSBD_System_SetConnect</i> .....            | 38        |
| 2.1.21 <i>AmbaUSBD_System_SetDataConn</i> .....           | 40        |
| 2.1.22 <i>AmbaUSBD_System_SetDataConnWithVbus</i> .....   | 41        |
| 2.1.23 <i>AmbaUSBD_System_SetSoftwareConnect</i> .....    | 42        |
| 2.1.24 <i>AmbaUSBD_System_SetUsbOwner</i> .....           | 43        |
| 2.1.25 <i>AmbaUSBD_System_SetIsrTaskInfo</i> .....        | 44        |



|        |  |     |
|--------|--|-----|
| 2.1.26 | <i>AmbaUSBD_System_GetIsltTaskInfo</i>             | 45  |
| 2.1.27 | <i>AmbaUSBD_System_SetVbusTimerTaskInfo</i>        | 46  |
| 2.1.28 | <i>AmbaUSBD_System_GetVbusTimerTaskInfo</i>        | 48  |
| 2.1.29 | <i>AmbaUSBD_System_SetBulkInTimeout</i>            | 49  |
| 2.1.30 | <i>AmbaUSB_System_SetDeviceConfigDetectTimeout</i> | 50  |
| 2.2    | CUSTOMIZE APIs                                     | 51  |
| 2.2.1  | <i>Data Structures</i>                             | 52  |
| 2.2.2  | <i>AmbaUSBD_Descriptor_Init</i>                    | 53  |
| 2.2.3  | <i>AmbaUSBD_Descriptor_SetCustomize</i>            | 55  |
| 2.3    | MASS STORAGE CLASS APIs                            | 57  |
| 2.3.1  | <i>Data Structures</i>                             | 58  |
| 2.3.2  | <i>Callback functions</i>                          | 59  |
| 2.3.3  | <i>AmbaUSBD_Msc_Init</i>                           | 68  |
| 2.3.4  | <i>AmbaUSBD_Msc_SetInfo</i>                        | 70  |
| 2.3.5  | <i>AmbaUSBD_Msc_SetProp</i>                        | 73  |
| 2.3.6  | <i>AmbaUSBD_Msc_Mount</i>                          | 75  |
| 2.3.7  | <i>AmbaUSBD_Msc_UnMount</i>                        | 77  |
| 2.3.8  | <i>AmbaUSBD_Msc_SetReadCacheMissTimeout</i>        | 79  |
| 2.3.9  | <i>AmbaUSBD_Msc_GetReadCacheMissTimeout</i>        | 80  |
| 2.3.10 | <i>AmbaUSBD_Msc_SetReadCacheTaskInfo</i>           | 81  |
| 2.3.11 | <i>AmbaUSBD_Msc_GetReadCacheTaskInfo</i>           | 83  |
| 2.3.12 | <i>AmbaUSBD_Msc_SetCbwCallback</i>                 | 84  |
| 2.3.13 | <i>AmbaUSBD_Msc_SendBulkInData</i>                 | 88  |
| 2.3.14 | <i>AmbaUSBD_Msc_RecvBulkOutData</i>                | 89  |
| 2.3.15 | <i>AmbaUSBD_Msc_GetReadCacheTaskInfo</i>           | 90  |
| 2.3.16 | <i>AmbaUSBD_Msc_GetReadCacheTaskInfo</i>           | 91  |
| 2.3.17 | <i>AmbaUSBD_Msc_GetReadCacheTaskInfo</i>           | 92  |
| 2.4    | MTP/PTP CLASS APIs                                 | 93  |
| 2.4.1  | <i>Data Structures</i>                             | 94  |
| 2.4.2  | <i>Callback functions</i>                          | 100 |
| 2.4.3  | <i>AmbaUSBD_Mtp_AddEvent</i>                       | 129 |
| 2.4.4  | <i>AmbaUSBD_Mtp_SetInfo</i>                        | 131 |
| 2.4.5  | <i>AmbaUSBD_Mtp_VendorOperation</i>                | 135 |
| 2.4.6  | <i>AmbaUSBD_Mtp_EnableDebug</i>                    | 136 |



|        |  |     |
|--------|--|-----|
| 2.4.7  | <i>AmbaUSBD_Mtp_DisableDebug</i> .....                 | 137 |
| 2.4.8  | <i>AmbaUSBD_Mtp_IsDebugEnabled</i> .....               | 138 |
| 2.4.9  | <i>AmbaUSBD_Mtp_SetReadCacheTaskInfo</i> .....         | 139 |
| 2.4.10 | <i>AmbaUSBD_Mtp_GetReadCacheTaskInfo</i> .....         | 141 |
| 2.4.11 | <i>AmbaUSBD_Mtp_SetSupportedEvents</i> .....           | 142 |
| 2.5    | PICTBRIDGE CLASS APIs.....                             | 144 |
| 2.5.1  | <i>Data Structures and Defines</i> .....               | 145 |
| 2.5.2  | <i>Callback functions</i> .....                        | 151 |
| 2.5.3  | <i>AmbaUSBD_Pictbridge_SetInfo</i> .....               | 156 |
| 2.5.4  | <i>AmbaUSBD_Pictbridge_RegisterCb</i> .....            | 158 |
| 2.5.5  | <i>AmbaUSBD_Pictbridge_CheckCapability</i> .....       | 160 |
| 2.5.6  | <i>AmbaUSBD_Pictbridge_ConfigJob</i> .....             | 162 |
| 2.5.7  | <i>AmbaUSBD_Pictbridge_IsPrinterServiceReady</i> ..... | 164 |
| 2.5.8  | <i>AmbaUSBD_Pictbridge_AbortJob</i> .....              | 165 |
| 2.5.9  | <i>AmbaUSBD_Pictbridge_ContinueJob</i> .....           | 166 |
| 2.5.10 | <i>AmbaUSBD_Pictbridge_SetMainTaskInfo</i> .....       | 167 |
| 2.5.11 | <i>AmbaUSBD_Pictbridge_GetMainTaskInfo</i> .....       | 168 |
| 2.6    | CDC ACM CLASS APIs .....                               | 169 |
| 2.6.1  | <i>AmbaUSBD_CDC_ACN_TerminalOpen</i> .....             | 170 |
| 2.6.2  | <i>AmbaUSBD_CDC_ACN_Write</i> .....                    | 172 |
| 2.6.3  | <i>AmbaUSBD_CDC_ACN_Read</i> .....                     | 174 |
| 2.6.4  | <i>AmbaUSBD_CDC_ACN_Multi_TerminalOpen</i> .....       | 176 |
| 2.6.5  | <i>AmbaUSBD_CDC_ACN_Multi_Write</i> .....              | 178 |
| 2.6.6  | <i>AmbaUSBD_CDC_ACN_Multi_Read</i> .....               | 180 |
| 2.7    | SIMPLE CLASS APIs.....                                 | 182 |
| 2.7.1  | <i>AmbaUSBD_Simple_Init</i> .....                      | 183 |
| 2.7.2  | <i>AmbaUSBD_Simple_Read</i> .....                      | 184 |
| 2.7.3  | <i>AmbaUSBD_Simple_Write</i> .....                     | 186 |
| 2.8    | UVC CLASS APIs .....                                   | 188 |
| 2.8.1  | <i>Data Structures and Defines</i> .....               | 189 |
| 2.8.2  | <i>Callback functions</i> .....                        | 191 |
| 2.8.3  | <i>AmbaUSBD_UVC_BulkSend</i> .....                     | 199 |
| 2.8.4  | <i>AmbaUSBD_UVC_BulkSendEx</i> .....                   | 200 |
| 2.8.5  | <i>AmbaUSBD_UVC_IsoSend</i> .....                      | 201 |



|           |  |            |
|-----------|--|------------|
| 2.8.6     | <i>AmbaUSBD_UVC_GetHostConfig</i> .....              | 202        |
| 2.8.7     | <i>AmbaUSBD_UVC_GetInputTerminalAttribute</i> .....  | 203        |
| 2.8.8     | <i>AmbaUSBD_UVC_GetProcessingUnitAttribute</i> ..... | 204        |
| 2.8.9     | <i>AmbaUSBD_UVC_SetInputTerminalAttribute</i> .....  | 205        |
| 2.8.10    | <i>AmbaUSBD_UVC_SetProcessingUnitAttribute</i> ..... | 206        |
| 2.8.11    | <i>AmbaUSBD_UVC_RegisterCallback</i> .....           | 207        |
| 2.8.12    | <i>AmbaUSBD_UVC_GetBrightnessSetting</i> .....       | 209        |
| 2.8.13    | <i>AmbaUSBD_UVC_SetBrightnessSetting</i> .....       | 210        |
| 2.8.14    | <i>AmbaUSBD_UVC_GetSaturationSetting</i> .....       | 211        |
| 2.8.15    | <i>AmbaUSBD_UVC_SetSaturationSetting</i> .....       | 212        |
| 2.8.16    | <i>AmbaUSBD_UVC_GetContrastSetting</i> .....         | 213        |
| 2.8.17    | <i>AmbaUSBD_UVC_SetContrastSetting</i> .....         | 214        |
| 2.8.18    | <i>AmbaUSBD_UVC_GetHueSetting</i> .....              | 215        |
| 2.8.19    | <i>AmbaUSBD_UVC_SetHueSetting</i> .....              | 216        |
| 2.8.20    | <i>AmbaUSBD_UVC_GetSharpnessSetting</i> .....        | 217        |
| 2.8.21    | <i>AmbaUSBD_UVC_SetSharpnessSetting</i> .....        | 218        |
| 2.8.22    | <i>AmbaUSBD_UVC_GetPowerLineSetting</i> .....        | 219        |
| 2.8.23    | <i>AmbaUSBD_UVC_SetPowerLineSetting</i> .....        | 220        |
| 2.9       | STREAM CLASS APIs .....                              | 221        |
| 2.9.1     | <i>AmbaUSBD_Stream_Read</i> .....                    | 222        |
| 2.9.2     | <i>AmbaUSBD_Stream_Write</i> .....                   | 224        |
| 2.9.3     | <i>AmbaUSBD_Stream_NativeRead</i> .....              | 226        |
| 2.9.4     | <i>AmbaUSBD_Stream_NativeWrite</i> .....             | 228        |
| <b>3.</b> | <b>USB HOST APIs.....</b>                            | <b>230</b> |
| 3.1       | SYSTEM AND FLOW CONTROL APIs .....                   | 231        |
| 3.1.1     | <i>Data Structures</i> .....                         | 232        |
| 3.1.2     | <i>AmbaUSB_HostSystemSetup</i> .....                 | 234        |
| 3.1.3     | <i>AmbaUSBH_System_ClassHook</i> .....               | 236        |
| 3.1.4     | <i>AmbaUSBH_System_SetPhy0Owner</i> .....            | 237        |
| 3.1.5     | <i>AmbaUSBH_System_SetUsbOwner</i> .....             | 240        |
| 3.1.6     | <i>AmbaUSBH_System_ClassUnHook</i> .....             | 241        |
| 3.1.7     | <i>AmbaUSBH_System_SetMemoryPool</i> .....           | 242        |
| 3.1.8     | <i>AmbaUSBH_System_SetIsoSlopDelay</i> .....         | 243        |



|        |   |     |
|--------|---|-----|
| 3.2    | STORAGE CLASS APIs .....                      | 244 |
| 3.2.1  | <i>AmbaUSBH_Storage_SetSlotInfo</i> .....     | 245 |
| 3.2.2  | <i>AmbaUSBH_Storage_GetStatus</i> .....       | 247 |
| 3.2.3  | <i>AmbaUSBH_Storage_Read</i> .....            | 249 |
| 3.2.4  | <i>AmbaUSBH_Storage_Write</i> .....           | 250 |
| 3.2.5  | <i>AmbaUSBH_Msc_SetMainTaskInfo</i> .....     | 251 |
| 3.2.6  | <i>AmbaUSBH_Msc_GetMainTaskInfo</i> .....     | 252 |
| 3.3    | MTP CLASS APIs .....                          | 253 |
| 3.3.1  | <i>Data Structures</i> .....                  | 254 |
| 3.3.2  | <i>Callback functions</i> .....               | 259 |
| 3.3.3  | <i>AmbaUSBH_Mtp_SessionOpen</i> .....         | 262 |
| 3.3.4  | <i>AmbaUSBH_Mtp_SessionClose</i> .....        | 264 |
| 3.3.5  | <i>AmbaUSBH_Mtp_RegisterCallback</i> .....    | 265 |
| 3.3.6  | <i>AmbaUSBH_Mtp_DeviceInfoGet</i> .....       | 267 |
| 3.3.7  | <i>AmbaUSBH_Mtp_StorageIdsGet</i> .....       | 270 |
| 3.3.8  | <i>AmbaUSBH_Mtp_StorageInfoGet</i> .....      | 272 |
| 3.3.9  | <i>AmbaUSBH_Mtp_ObjectNumberGet</i> .....     | 274 |
| 3.3.10 | <i>AmbaUSBH_Mtp_ObjectHandlesGet</i> .....    | 276 |
| 3.3.11 | <i>AmbaUSBH_Mtp_ObjectInfoGet</i> .....       | 279 |
| 3.3.12 | <i>AmbaUSBH_Mtp_ObjectGet</i> .....           | 282 |
| 3.3.13 | <i>AmbaUSBH_Mtp_ObjectInfoSend</i> .....      | 286 |
| 3.3.14 | <i>AmbaUSBH_Mtp_ObjectSend</i> .....          | 288 |
| 3.3.15 | <i>AmbaUSBH_Mtp_ObjectCopy</i> .....          | 292 |
| 3.3.16 | <i>AmbaUSBH_Mtp_ObjectMove</i> .....          | 293 |
| 3.3.17 | <i>AmbaUSBH_Mtp_ObjectDelete</i> .....        | 294 |
| 3.3.18 | <i>AmbaUSBH_Mtp_ObjectTransferAbort</i> ..... | 295 |
| 3.3.19 | <i>AmbaUSBH_Mtp_ThumbGet</i> .....            | 296 |
| 3.3.20 | <i>AmbaUSBH_Mtp_DeviceReset</i> .....         | 300 |
| 3.3.21 | <i>AmbaUSBH_Mtp_VendorCommandSend</i> .....   | 301 |
| 3.3.22 | <i>AmbaUSBH_Mtp_VendorCommandGet</i> .....    | 305 |
| 3.3.23 | <i>AmbaUSBH_Mtp_GetResponseCode</i> .....     | 309 |
| 3.3.24 | <i>AmbaUSBH_Mtp_SetResponseCode</i> .....     | 311 |
| 3.4    | VIDEO CLASS (UVC) APIs.....                   | 312 |
| 3.4.1  | <i>Data Structures</i> .....                  | 313 |



|        |  |     |
|--------|--|-----|
| 3.4.2  | <i>Callback Functions</i> .....                    | 327 |
| 3.4.3  | <i>AmbaUSBH_Uvc_GetDeviceInfo</i> .....            | 331 |
| 3.4.4  | <i>AmbaUSBH_Uvc_GetItCtrlInfo</i> .....            | 335 |
| 3.4.5  | <i>AmbaUSBH_Uvc_GetPuCtrlInfo</i> .....            | 339 |
| 3.4.6  | <i>AmbaUSBH_Uvc_PrintDeviceValues</i> .....        | 344 |
| 3.4.7  | <i>AmbaUSBH_Uvc_ProbeAndCommit</i> .....           | 345 |
| 3.4.8  | <i>AmbaUSBH_Uvc_GetCurrentProbeSetting</i> .....   | 346 |
| 3.4.9  | <i>AmbaUSBH_Uvc_SetCurValues</i> .....             | 347 |
| 3.4.10 | <i>AmbaUSBH_Uvc_StreamingStart</i> .....           | 348 |
| 3.4.11 | <i>AmbaUSBH_Uvc_StreamingStop</i> .....            | 349 |
| 3.4.12 | <i>AmbaUSBH_Uvc_Read</i> .....                     | 350 |
| 3.4.13 | <i>AmbaUSBH_Uvc_ReadBlock</i> .....                | 351 |
| 3.4.14 | <i>AmbaUSBH_Uvc_ReadAppend</i> .....               | 353 |
| 3.4.15 | <i>AmbaUSBH_Uvc_GetMaxPacketSize</i> .....         | 354 |
| 3.4.16 | <i>AmbaUSBH_Uvc_RegisterCallback</i> .....         | 355 |
| 3.4.17 | <i>AmbaUSBH_Uvc_GetBacklightCtrlInfo</i> .....     | 356 |
| 3.4.18 | <i>AmbaUSBH_Uvc_SetBacklightCtrl</i> .....         | 357 |
| 3.4.19 | <i>AmbaUSBH_Uvc_GetBrightnessCtrlInfo</i> .....    | 358 |
| 3.4.20 | <i>AmbaUSBH_Uvc_SetBrightnessCtrl</i> .....        | 359 |
| 3.4.21 | <i>AmbaUSBH_Uvc_GetContrastCtrlInfo</i> .....      | 360 |
| 3.4.22 | <i>AmbaUSBH_Uvc_SetContrastCtrl</i> .....          | 361 |
| 3.4.23 | <i>AmbaUSBH_Uvc_GetHueCtrlInfo</i> .....           | 362 |
| 3.4.24 | <i>AmbaUSBH_Uvc_SetHueCtrl</i> .....               | 363 |
| 3.4.25 | <i>AmbaUSBH_Uvc_GetSaturationCtrlInfo</i> .....    | 364 |
| 3.4.26 | <i>AmbaUSBH_Uvc_SetSaturationCtrl</i> .....        | 365 |
| 3.4.27 | <i>AmbaUSBH_Uvc_GetSharpnessCtrlInfo</i> .....     | 366 |
| 3.4.28 | <i>AmbaUSBH_Uvc_SetSharpnessCtrl</i> .....         | 367 |
| 3.4.29 | <i>AmbaUSBH_Uvc_GetGammaCtrlInfo</i> .....         | 368 |
| 3.4.30 | <i>AmbaUSBH_Uvc_SetGammaCtrl</i> .....             | 369 |
| 3.4.31 | <i>AmbaUSBH_Uvc_GetWBTemperatureCtrlInfo</i> ..... | 370 |
| 3.4.32 | <i>AmbaUSBH_Uvc_SetWBTemperatureCtrl</i> .....     | 371 |
| 3.4.33 | <i>AmbaUSBH_Uvc_GetWBComponentCtrlInfo</i> .....   | 372 |
| 3.4.34 | <i>AmbaUSBH_Uvc_SetWBComponentCtrl</i> .....       | 373 |
| 3.4.35 | <i>AmbaUSBH_Uvc_GetGainCtrlInfo</i> .....          | 374 |



|        |   |     |
|--------|---|-----|
| 3.4.36 | <i>AmbaUSBH_Uvc_SetGainCtrl</i> .....                   | 375 |
| 3.4.37 | <i>AmbaUSBH_Uvc_GetCurLineFrequencyCtrl</i> .....       | 376 |
| 3.4.38 | <i>AmbaUSBH_Uvc_SetLineFrequencyCtrl</i> .....          | 377 |
| 3.4.39 | <i>AmbaUSBH_Uvc_GetHueAutoCtrlInfo</i> .....            | 378 |
| 3.4.40 | <i>AmbaUSBH_Uvc_SetHueAutoCtrl</i> .....                | 379 |
| 3.4.41 | <i>AmbaUSBH_Uvc_GetWBTempAutoCtrlInfo</i> .....         | 380 |
| 3.4.42 | <i>AmbaUSBH_Uvc_SetWBTempAutoCtrl</i> .....             | 381 |
| 3.4.43 | <i>AmbaUSBH_Uvc_GetWBCompAutoCtrlInfo</i> .....         | 382 |
| 3.4.44 | <i>AmbaUSBH_Uvc_SetWBCompAutoCtrl</i> .....             | 383 |
| 3.4.45 | <i>AmbaUSBH_Uvc_GetDigitalMultiplierCtrlInfo</i> .....  | 384 |
| 3.4.46 | <i>AmbaUSBH_Uvc_SetDigitalMultiplierCtrl</i> .....      | 385 |
| 3.4.47 | <i>AmbaUSBH_Uvc_GetDigitalMultiplierLimitInfo</i> ..... | 386 |
| 3.4.48 | <i>AmbaUSBH_Uvc_SetDigitalMultiplierLimit</i> .....     | 387 |
| 3.4.49 | <i>AmbaUSBH_Uvc_GetCurAnalogVideoStandard</i> .....     | 388 |
| 3.4.50 | <i>AmbaUSBH_Uvc_SetAnalogVideoStandard</i> .....        | 389 |
| 3.4.51 | <i>AmbaUSBH_Uvc_GetCurAnalogLockStatus</i> .....        | 390 |
| 3.4.52 | <i>AmbaUSBH_Uvc_SetAnalogLockStatus</i> .....           | 391 |
| 3.4.53 | <i>AmbaUSBH_Uvc_H264Enable</i> .....                    | 392 |
| 3.4.54 | <i>AmbaUSBH_Uvc_IsH264Supported</i> .....               | 393 |
| 3.4.55 | <i>AmbaUSBH_Uvc_H264GetVideoConfigInfo</i> .....        | 394 |
| 3.4.56 | <i>AmbaUSBH_Uvc_H264SetVideoConfig</i> .....            | 396 |
| 3.4.57 | <i>AmbaUSBH_Uvc_H264RequestIdr</i> .....                | 397 |
| 3.4.58 | <i>AmbaUSBH_Uvc_H264GetRateCtrlModeInfo</i> .....       | 398 |
| 3.4.59 | <i>AmbaUSBH_Uvc_H264SetRateCtrlMode</i> .....           | 400 |
| 3.4.60 | <i>AmbaUSBH_Uvc_H264GetTemporalScaleModelInfo</i> ..... | 401 |
| 3.4.61 | <i>AmbaUSBH_Uvc_H264SetTemporalScaleMode</i> .....      | 402 |
| 3.4.62 | <i>AmbaUSBH_Uvc_H264GetSpatialScaleModelInfo</i> .....  | 403 |
| 3.4.63 | <i>AmbaUSBH_Uvc_H264SetSpatialScaleMode</i> .....       | 404 |
| 3.4.64 | <i>AmbaUSBH_Uvc_H264GetFrameRateConfigInfo</i> .....    | 405 |
| 3.4.65 | <i>AmbaUSBH_Uvc_H264SetFrameRateConfig</i> .....        | 406 |
| 3.4.66 | <i>AmbaUSBH_Uvc_GetDeviceInfoEx</i> .....               | 407 |
| 3.4.67 | <i>AmbaUSBH_Uvc_GetActiveDeviceEx</i> .....             | 409 |
| 3.4.68 | <i>AmbaUSBH_Uvc_GetItControlInfoEx</i> .....            | 410 |
| 3.4.69 | <i>AmbaUSBH_Uvc_GetPuControlInfoEx</i> .....            | 412 |



|         |  |     |
|---------|--|-----|
| 3.4.70  | <i>AmbaUSBH_Uvc_PrintDeviceValuesEx</i>        | 414 |
| 3.4.71  | <i>AmbaUSBH_Uvc_ProbeAndCommitEx</i>           | 415 |
| 3.4.72  | <i>AmbaUSBH_Uvc_GetCurrentProbeSettingEx</i>   | 417 |
| 3.4.73  | <i>AmbaUSBH_Uvc_SetCurValuesEx</i>             | 419 |
| 3.4.74  | <i>AmbaUSBH_Uvc_StreamingStartEx</i>           | 421 |
| 3.4.75  | <i>AmbaUSBH_Uvc_StreamingStopEx</i>            | 422 |
| 3.4.76  | <i>AmbaUSBH_Uvc_GetAlternateSettingEx</i>      | 423 |
| 3.4.77  | <i>AmbaUSBH_Uvc_ReadBlockEx</i>                | 425 |
| 3.4.78  | <i>AmbaUSBH_Uvc_ReadAppendEx</i>               | 427 |
| 3.4.79  | <i>AmbaUSBH_Uvc_GetMaxPacketSizeEx</i>         | 428 |
| 3.4.80  | <i>AmbaUSBH_Uvc_RegisterCallbackEx</i>         | 429 |
| 3.4.81  | <i>AmbaUSBH_Uvc_GetBacklightCtrlInfoEx</i>     | 431 |
| 3.4.82  | <i>AmbaUSBH_Uvc_SetBacklightCtrlEx</i>         | 433 |
| 3.4.83  | <i>AmbaUSBH_Uvc_GetBrightnessCtrlInfoEx</i>    | 434 |
| 3.4.84  | <i>AmbaUSBH_Uvc_SetBrightnessCtrlEx</i>        | 436 |
| 3.4.85  | <i>AmbaUSBH_Uvc_GetContrastCtrlInfoEx</i>      | 437 |
| 3.4.86  | <i>AmbaUSBH_Uvc_SetContrastCtrlEx</i>          | 439 |
| 3.4.87  | <i>AmbaUSBH_Uvc_GetHueCtrlInfoEx</i>           | 440 |
| 3.4.88  | <i>AmbaUSBH_Uvc_SetHueCtrlEx</i>               | 442 |
| 3.4.89  | <i>AmbaUSBH_Uvc_GetSaturationCtrlInfoEx</i>    | 443 |
| 3.4.90  | <i>AmbaUSBH_Uvc_SetSaturationCtrlEx</i>        | 445 |
| 3.4.91  | <i>AmbaUSBH_Uvc_GetSharpnessCtrlInfoEx</i>     | 446 |
| 3.4.92  | <i>AmbaUSBH_Uvc_SetSharpnessCtrlEx</i>         | 448 |
| 3.4.93  | <i>AmbaUSBH_Uvc_GetGammaCtrlInfoEx</i>         | 449 |
| 3.4.94  | <i>AmbaUSBH_Uvc_SetGammaCtrlEx</i>             | 451 |
| 3.4.95  | <i>AmbaUSBH_Uvc_GetWBTemperatureCtrlInfoEx</i> | 452 |
| 3.4.96  | <i>AmbaUSBH_Uvc_SetWBTemperatureCtrlEx</i>     | 454 |
| 3.4.97  | <i>AmbaUSBH_Uvc_GetWBComponentCtrlInfoEx</i>   | 455 |
| 3.4.98  | <i>AmbaUSBH_Uvc_SetWBComponentCtrlEx</i>       | 457 |
| 3.4.99  | <i>AmbaUSBH_Uvc_GetGainCtrlInfoEx</i>          | 458 |
| 3.4.100 | <i>AmbaUSBH_Uvc_SetGainCtrlEx</i>              | 460 |
| 3.4.101 | <i>AmbaUSBH_Uvc_GetCurLineFrequencyCtrlEx</i>  | 461 |
| 3.4.102 | <i>AmbaUSBH_Uvc_SetLineFrequencyCtrlEx</i>     | 463 |
| 3.4.103 | <i>AmbaUSBH_Uvc_GetHueAutoCtrlInfoEx</i>       | 464 |



|         |   |     |
|---------|---|-----|
| 3.4.104 | <i>AmbaUSBH_Uvc_SetHueAutoCtrlEx</i> .....                | 466 |
| 3.4.105 | <i>AmbaUSBH_Uvc_GetWBTempAutoCtrlInfoEx</i> .....         | 467 |
| 3.4.106 | <i>AmbaUSBH_Uvc_SetWBTempAutoCtrlEx</i> .....             | 469 |
| 3.4.107 | <i>AmbaUSBH_Uvc_GetWBCompAutoCtrlInfoEx</i> .....         | 470 |
| 3.4.108 | <i>AmbaUSBH_Uvc_SetWBCompAutoCtrlEx</i> .....             | 472 |
| 3.4.109 | <i>AmbaUSBH_Uvc_GetDigitalMultiplierCtrlInfoEx</i> .....  | 473 |
| 3.4.110 | <i>AmbaUSBH_Uvc_SetDigitalMultiplierCtrlEx</i> .....      | 475 |
| 3.4.111 | <i>AmbaUSBH_Uvc_GetDigitalMultiplierLimitInfoEx</i> ..... | 476 |
| 3.4.112 | <i>AmbaUSBH_Uvc_SetDigitalMultiplierLimitEx</i> .....     | 478 |
| 3.4.113 | <i>AmbaUSBH_Uvc_GetCurAnalogVideoStandardEx</i> .....     | 479 |
| 3.4.114 | <i>AmbaUSBH_Uvc_SetAnalogVideoStandardEx</i> .....        | 481 |
| 3.4.115 | <i>AmbaUSBH_Uvc_GetCurAnalogLockStatusEx</i> .....        | 482 |
| 3.4.116 | <i>AmbaUSBH_Uvc_SetAnalogLockStatusEx</i> .....           | 484 |
| 3.4.117 | <i>AmbaUSBH_Uvc_H264EnableEx</i> .....                    | 485 |
| 3.4.118 | <i>AmbaUSBH_Uvc_IsH264SupportedEx</i> .....               | 486 |
| 3.4.119 | <i>AmbaUSBH_Uvc_H264GetVideoConfigInfoEx</i> .....        | 487 |
| 3.4.120 | <i>AmbaUSBH_Uvc_H264SetVideoConfigEx</i> .....            | 489 |
| 3.4.121 | <i>AmbaUSBH_Uvc_H264RequestIdrEx</i> .....                | 491 |
| 3.4.122 | <i>AmbaUSBH_Uvc_H264GetRateCtrlModeInfoEx</i> .....       | 493 |
| 3.4.123 | <i>AmbaUSBH_Uvc_H264SetRateCtrlModeEx</i> .....           | 495 |
| 3.4.124 | <i>AmbaUSBH_Uvc_H264GetTemporalScaleModeInfoEx</i> .....  | 497 |
| 3.4.125 | <i>AmbaUSBH_Uvc_H264SetTemporalScaleModeEx</i> .....      | 499 |
| 3.4.126 | <i>AmbaUSBH_Uvc_H264GetSpatialScaleModeInfoEx</i> .....   | 501 |
| 3.4.127 | <i>AmbaUSBH_Uvc_H264SetSpatialScaleModeEx</i> .....       | 503 |
| 3.4.128 | <i>AmbaUSBH_Uvc_H264GetFrameRateConfigInfoEx</i> .....    | 505 |
| 3.4.129 | <i>AmbaUSBH_Uvc_H264SetFrameRateConfigEx</i> .....        | 507 |
| 3.5     | SIMPLE CLASS APIs.....                                    | 509 |
| 3.5.1   | <i>Callback functions</i> .....                           | 510 |
| 3.5.2   | <i>AmbaUSBH_Simple_RegisterCallback</i> .....             | 512 |
| 3.5.3   | <i>AmbaUSBH_Simple_SetPidVid</i> .....                    | 513 |
| 3.5.4   | <i>AmbaUSBH_Simple_GetBulkInEndpointList</i> .....        | 514 |
| 3.5.5   | <i>AmbaUSBH_Simple_GetBulkOutEndpointList</i> .....       | 516 |
| 3.5.6   | <i>AmbaUSBH_Simple_ControlRequest</i> .....               | 518 |
| 3.5.7   | <i>AmbaUSBH_Simple_BulkRead</i> .....                     | 520 |



|                    |                                  |            |
|--------------------|----------------------------------|------------|
| 3.5.8              | AmbaUSBH_Simple_BulkWrite .....  | 522        |
| <b>APPENDIX 1.</b> | <b>ADDITIONAL RESOURCES.....</b> | <b>524</b> |
| <b>APPENDIX 2.</b> | <b>IMPORTANT NOTICE .....</b>    | <b>525</b> |
| <b>APPENDIX 3.</b> | <b>REVISION HISTORY .....</b>    | <b>527</b> |

For PROTRULY Confidential Only



## 1. Overview

This document lists and describes ThreadX APIs for both the USB Device and the USB Host. Please note that some chips support both the USB Device and Host mode, and some do not. Please check your target platform before using these APIs.

For PROTRULY Confidential Only



## 1.1 Introduction

This document details the Universal Serial Bus (USB) library Application Programming Interfaces (APIs) as follows:

- Chapter 2 “USB Device APIs”
- Chapter 3 “USB Host APIs”

For PROTRULY Confidential Only



## 2. USB Device APIs

This chapter describes APIs for the USB device mode.

### 2.1 System and Flow Control APIs

This section describes APIs for system and flow control.

For Confidential  
PROTRULY Only



## 2.1.1 Data Structures

### 2.1.1.1 USB\_SYSTEM\_INIT\_s

| Type    | Parameter           | Description  |
|---------|---------------------|--|
| UINT8 * | <b>MemPoolPtr</b>   | [Input] Memory for USBX Device driver stack            |
| UINT32  | <b>TotalMemSize</b> | [Input] Total memory size for USBX device system stack |

Table 2-1. Definition of **USB\_SYSTEM\_INIT\_s**.

### 2.1.1.2 USB\_CLASS\_INIT\_s

| Type        | Parameter                 | Description   |
|-------------|---------------------------|---|
| UDC_CLASS_e | <b>classID</b>            | [Input] USB Class ID:<br><b>UDC_CLASS_NONE</b> - No USB class<br><b>UDC_CLASS_MSC</b> - Mass Storage class<br><b>UDC_CLASS_MTP</b> - PTP/MTP class<br><b>UDC_CLASS_PICT</b> - Pictbridge class<br><b>UDC_CLASS_CDC_ACM</b> - RS232 over USB class (single instance)<br><b>UDC_CLASS_CUSTOM</b> - Custom class<br><b>UDC_CLASS_STREAM</b> - Stream class<br><b>UDC_CLASS_HID</b> - HID class<br><b>UDC_CLASS_SIMPLE</b> - Simple class<br><b>UDC_CLASS_UVC</b> - USB Video class<br><b>UDC_CLASS_MIX_STG</b> - MTP+MSC class<br><b>UDC_CLASS_CDC_ACM_MULTI</b> - RS232 over USB class (multiple instances) |
| UINT32      | <b>ClassTaskStaciSize</b> | [Input] Stack size for class driver tasks   |
| UINT32      | <b>ClassTaskPriority</b>  | [Input] Task priority for class driver tasks  |



|        |                              |  |
|--------|------------------------------|--|
| UINT32 | <b>ClassTaskAffinityMask</b> | [Input] Affinity Mask for class driver tasks |
|--------|------------------------------|--|

Table 2-2. Definition of **USB\_CLASS\_INIT\_s**.

#### 2.1.1.3 USB\_DEV\_VBUS\_CB\_s

| Type             | Parameter               | Description   |
|------------------|-------------------------|---|
| Function pointer | <b>VbusConnectCB</b>    | This pointer is called when the VBUS is connected.<br>Please refer to Section 2.1.2.2 for more details.   |
| Function pointer | <b>VbusDisconnectCB</b> | It is called when the VBUS is disconnected. Please refer to Section 0 for more details.                   |
| Function pointer | <b>SystemStart</b>      | It is called when the USB is going to start. Please refer to Section 2.1.2.4 for more details.            |
| Function pointer | <b>SystemRelease</b>    | It is called when the USB is going to stop. Please refer to Section 2.1.2.5 for more details.             |
| Function pointer | <b>SystemConfigured</b> | It is called when the USB device is configured by Host. Please refer to Section 2.1.2.6 for more details. |
| Function pointer | <b>SystemSuspended</b>  | It is called when the USB device is suspended by Host. Please refer to Section 2.1.2.7 for more details.  |
| Function pointer | <b>SystemResumed</b>    | It is called when the USB device is resumed by Host. Please refer to Section 2.1.2.8 for more details.    |
| Function pointer | <b>SystemReset</b>      | It is called when the USB device is reset by Host. Please refer to Section 2.1.2.9 for more details.      |

Table 2-3. Definition of **USB\_DEV\_VBUS\_CB\_s**.

#### 2.1.1.4 USB\_VENDOR\_REQUEST\_INFO\_s

| Type         | Parameter      | Description  |
|--------------|----------------|--|
| UDC_CLAS_S_e | <b>classID</b> | [Input] USB Class ID:<br><b>UDC_CLASS_NONE</b> - No USB class<br><b>UDC_CLASS_MSC</b> - Mass Storage class<br><b>UDC_CLASS_MTP</b> - PTP/MTP class |



| Type             | Parameter                         | Description   |
|------------------|-----------------------------------|---|
|                  |                                   | <b>UDC_CLASS_PICT</b> - Pictbridge class<br><b>UDC_CLASS_CDC_ACM</b> - RS232 over USB class (single instance)<br><b>UDC_CLASS_CUSTOM</b> - Custom class<br><b>UDC_CLASS_STREAM</b> - Stream class<br><b>UDC_CLASS_HID</b> - HID class<br><b>UDC_CLASS_SIMPLE</b> - Simple class<br><b>UDC_CLASS_UVC</b> - USB Video class<br><b>UDC_CLASS_MIX_STG</b> - MTP+MSC class<br><b>UDC_CLASS_CDC_ACM_MULTI</b> - RS232 over USB class (multiple instances) |
| Function pointer | <b>InterfaceVendorRequestFunc</b> | Please refer to section 2.1.2.1 for more details.   |

Table 2-4. Definition of **USB\_VENDOR\_REQUEST\_INFO\_s**.

#### 2.1.1.5 UDC\_TASKINFO\_s

| Type   | Parameter           | Description  |
|--------|---------------------|--|
| UINT32 | <b>Priority</b>     | Task priority  |
| UINT32 | <b>AffinityMask</b> | Task Affinity Mask. The affinity mask is represented as a bitmask, with the lowest order bit corresponding to the first logical CPU and the highest order bit corresponding to the last logical CPU. Not all CPUs may exist on a given system but a mask may specify more CPUs than are present. The masks are typically given in hexadecimal. For example:<br>0x00000001 is processor #0<br>0x00000003 is processors #0 and #1<br>0xFFFFFFFF is all processors (#0 through #31) |
| UINT32 | <b>StackSize</b>    | The stack size of this task  |

Table 2-5. Definition of **UDC\_TASKINFO\_s**.



#### 2.1.1.6 UHC\_TASKINFO\_s

| Type   | Parameter           | Description  |
|--------|---------------------|--|
| UINT32 | <b>Priority</b>     | Task priority.   |
| UINT32 | <b>AffinityMask</b> | Task Affinity Mask. The affinity mask is represented as a bitmask, with the lowest order bit corresponding to the first logical CPU and the highest order bit corresponding to the last logical CPU. Not all CPUs may exist on a given system but a mask may specify more CPUs than are present. The masks are typically given in hexadecimal. For example:<br>0x00000001 is processor #0<br>0x00000003 is processors #0 and #1<br>0xFFFFFFFF is all processors (#0 through #31) |
| UINT32 | <b>StackSize</b>    | The stack size of this task  |

Table 2-6. Definition of **UHC\_TASKINFO\_s**.



## 2.1.2 Callback functions

### 2.1.2.1 InterfaceVendorRequestFunc

#### API Syntax:

```
INT32 InterfaceVendorRequestFunc (UINT32 bmRequestType, UINT32 bRequest,  
UINT32 wValue, UINT32 wIndex, UINT32 wLength)
```

#### Function Description:

This function is called when an USB vendor request is received.

#### Parameters:

| Type   | Parameter            | Description                                    |
|--------|----------------------|--|
| UINT32 | <b>bmRequestType</b> | The bmRequestType field in USB device requests |
| UINT32 | <b>bRequest</b>      | The bRequest field in USB device requests      |
| UINT32 | <b>wValue</b>        | The wValue field in USB device requests        |
| UINT32 | <b>wIndex</b>        | The wIndex field in USB device requests        |
| UINT32 | <b>wLength</b>       | The wLength field in USB device requests       |

Table 2-7. Parameters for AmbaUSB API **InterfaceVendorRequestFunc()**.

#### Return:

| Return | Description |
|--------|-------------|
| 0      | Success     |
| -1     | Failure     |

Table 2-8. Returns for AmbaUSB API **InterfaceVendorRequestFunc()**.

#### Example:

None

#### See Also:

[InterfaceVendorRequestFunc\(\)](#)



### 2.1.2.2 VbusConnectCB

#### API Syntax:

```
void VbusConnectCB (void)
```

#### Function Description:

This function is called when the VBUS is connected.

#### Parameters:

None

#### Return:

None

#### Example:

None

#### See Also:

None



### 2.1.2.3 VbusDisconnectCB

#### API Syntax:

```
void VbusDisconnectCB (void)
```

#### Function Description:

This function is called when the VBUS is disconnected.

#### Parameters:

None

#### Return:

None

#### Example:

None

#### See Also:

None

### 2.1.2.4 SystemStart

#### API Syntax:

```
void SystemStart (void)
```

#### Function Description:

This function is called when the USB system is going to start. Application must allocate memory and call **AmbaUSB\_System\_DeviceSystemSetup()** to initialize USB device system in this callback.

**Parameters:**

None

**Return:**

None

**Example:**

```
int AppUsbd_SystemInit(void)
{
    USB_SYSTEM_INIT_s Sysconfig = {0};
    int nRet = 0;
    if (UsbxSystemMemory.pMemAlignedBase != 0) {
        AmbaPrint("%s(): free memory for USBX.", __func__);
        nRet = AmbaKAL_MemFree(&UsbxSystemMemory);
        if (nRet != OK) {
            AmbaPrint("[Error] %s(): can't free memory for USBX, 0x%X.",
                      __func__, AMBA_USB_MEM_BUF_SIZE, nRet);
        }
        memset(&UsbxSystemMemory, 0, sizeof(UsbxSystemMemory));
    }

    nRet = AmbaKAL_MemAllocate(&AmbaBytePool_Cached,
                               &UsbxSystemMemory, AMBA_USB_MEM_BUF_SIZE,
                               32);
    if (nRet != OK) {
        AmbaPrint("[Error] %s(): can't allocate %d bytes memory for USBX,
0x%X.",
                  __func__, AMBA_USB_MEM_BUF_SIZE, nRet);
        return nRet;
    }

    /* Initialization of USB Class */
    Sysconfig.MemPoolPtr = UsbxSystemMemory.pMemAlignedBase;
    Sysconfig.TotalMemSize = AMBA_USB_MEM_BUF_SIZE;
```



```
    return AmbaUSB_System_DeviceSystemSetup(&Sysconfig);  
}  
  
static void USB_DeviceSystemStart(void)  
{  
    AmbaPrint("DeviceSystemStart");  
    AppUsbd_SystemInit();  
}
```

**See Also:**

None

**2.1.2.5    SystemRelease**

**API Syntax:**

```
void SystemRelease (void)
```

**Function Description:**

This function is called when the USB system is going to stop. Application must release the memory allocated in **SystemStart()** callback.

**Parameters:**

None

**Return:**

None

**Example:**

```
int AppUsbd_SystemDestroy(void)  
{  
    int nRet = 0;  
    if (UsbxSystemMemory.pMemAlignedBase != 0) {  
        nRet = AmbaKAL_MemFree(&UsbxSystemMemory);  
    }  
}
```



```
if (nRet != OK) {
    AmbaPrint("[Error] %s(): can't free memory for USBX.",
              __func__, AMBA_USB_MEM_BUF_SIZE);
}

memset(&UsbxSystemMemory, 0, sizeof(UsbxSystemMemory));
}

return nRet;
}

static void USB_DeviceSystemRelease(void)
{
    AmbaPrint("DeviceSystemRelease");
    AppUsbd_SystemDestroy();
}
```

#### See Also:

None

#### 2.1.2.6 [SystemConfigured](#)

#### API Syntax:

```
void SystemConfigured(UINT16 index)
```

#### Function Description:

This function is called when the USB device is configured by Host.

#### Parameters:

| Type   | Parameter | Description                         |
|--------|-----------|-------------------------------------|
| UINT16 | index     | The configuration index set by Host |

Table 2-9. Parameters for AmbaUSB API **SystemConfigured()**.

#### Return:

None



**Example:**

None

**See Also:**

None

**2.1.2.7    SystemSuspended**

**API Syntax:**

```
void SystemSuspended (void)
```

**Function Description:**

This function is called when the USB device is suspended by Host.

**Parameters:**

None

**Return:**

None

**Example:**

None

**See Also:**

None

**2.1.2.8    SystemResumed**

**API Syntax:**

```
void SystemResumed(void)
```

**Function Description:**

This function is called when the USB device is resumed by Host.

**Parameters:**

None

**Return:**

None

**Example:**

None

**See Also:**

None

### 2.1.2.9    [SystemReset](#)

**API Syntax:**

```
void SystemReset (void)
```

**Function Description:**

This function is called when the USB device is reset by Host.

**Parameters:**

None

**Return:**

None

**Example:**

None



**See Also:**

None

For PROTRULY Confidential Only



### 2.1.3 AmbaUSB\_DeviceSystemSetup

#### API Syntax:

```
UINT32 AmbaUSB_DeviceSystemSetup (USB_SYSTEM_INIT_s *config)
```

#### Function Description:

This function is used to initialize the USBX deviec system and the USB device driver stack.

#### Parameters:

| Type               | Parameters | Description   |
|--------------------|------------|---|
| USB_SYSTEM_INIT_s* | config     | Define USB system initial parameters. Please refer to Section 2.1.1.1 USB_SYSTEM_INIT_s for more details. |

Table 2-10. Parameters for AmbaUSB API **AmbaUSB\_DeviceSystemSetup()**.

#### Return:

| Return     | Description |
|------------|-------------|
| 0          | Success     |
| 0xFFFFFFFF | Failure     |

Table 2-11. Returns for AmbaUSB API **AmbaUSB\_DeviceSystemSetup()**.

#### Example:

```
USB_SYSTEM_INIT_s Sysconfig = {0};  
int nRet = 0;  
  
nRet = AmbaKAL_MemAllocate(&AmbaBytePool_Cached,  
                           &UsbxSystemMemory,  
                           AMBA_USB_MEM_BUF_SIZE,  
                           32);  
  
if (nRet != OK) {  
    AmbaPrint("[Error] %s(): can't allocate %d bytes memory for USBX, 0x%X.",  
             __func__,  
             AMBA_USB_MEM_BUF_SIZE,
```



```
    nRet);  
  
    return nRet;  
}  
  
/* Initialization of USB device system */  
Sysconfig.MemPoolPtr = UsbxSystemMemory.pMemAlignedBase;  
Sysconfig.TotalMemSize = AMBA_USB_MEM_BUF_SIZE;  
return AmbaUSB_DeviceSystemSetup(&Sysconfig);
```

**See Also:**

None

For PROTRULY Confidential Only



## 2.1.4 AmbaUSB\_System\_GetVcsVersion

### API Syntax:

```
char *AmbaUSB_System_GetVcsVersion (void)
```

### Function Description:

This function is used to get the version information of Version Control System (VCS) in the current USB library.

### Parameters:

None

### Return:

| Return | Description                 |
|--------|-----------------------------|
| char * | Version information string. |

Table 2-12. Returns for AmbaUSB API **AmbaUSB\_System\_GetVcsVersion()**.

### Example:

```
char *version = AmbaUSB_System_GetVcsVersion ();
AmberPrint( "SVN: %s.", version );
```

### See Also:

None



## 2.1.5 AmbaUSBD\_System\_SetMemoryPool

### API Syntax:

```
UINT32 AmbaUSBD_System_SetMemoryPool (TX_BYTE_POOL *cached_pool,  
TX_BYTE_POOL *noncached_pool)
```

### Function Description:

This function is used to set the memory pool for the USB Device system. The USB Device system will allocate memory from the memory pool when running.

It should be called before calling **AmbaUSB\_DeviceSystemSetup()**.

### Parameters:

| Type           | Parameter             | Description                    |
|----------------|-----------------------|--------------------------------|
| TX_BYTE_POOL * | <b>cached_pool</b>    | [Input] cached memory pool.    |
| TX_BYTE_POOL * | <b>noncached_pool</b> | [Input] noncached memory pool. |

Table 2-13. Returns for AmbaUSB API **AmbaUSBD\_System\_SetMemoryPool()**.

### Return:

| Return | Description |
|--------|-------------|
| 0      | Success     |
| others | Failure     |

Table 2-14. Returns for AmbaUSB API **AmbaUSBD\_System\_SetMemoryPool()**.

### Example:

None

### See Also:

None



## 2.1.6 AmbaUSBD\_System\_ChargeDetection

### API Syntax:

```
UINT32 AmbaUSBD_System_ChargeDetection (void)
```

### Function Description:

It is used to detect the charger after the VBUS is detected. Please note that it can only detect chargers which support USB Battery Charger 1.1. For chargers which do not support USB Battery Charger 1.1, the result of this API is unknown.

### Parameters:

None

### Return:

| Return | Description          |
|--------|----------------------|
| 0      | No charger detected. |
| 1      | Charger detected.    |

Table 2-15. Returns for AmbaUSB API **AmbaUSBD\_System\_ChargeDetection()**.

### Example:

None

### See Also:

None



## 2.1.7 AmbaUSBD\_System\_ControlRead

### API Syntax:

```
UINT32 AmbaUSBD_System_ControlRead (UINT8 *buff, UINT32 size)
```

### Function Description:

This function is used to read data from USB Host by the control endpoint. Note that it can only be called during data phase in vendor requests, that is, applications must call **AmbaUSBD\_System\_RegisterVendorRequest()** to register a callback for vendor requests first, and then call this function in the *InterfaceVendorRequestFunc()* to read control data. Calling it in other places will cause undefined behavior.

### Parameters:

| Type    | Parameter   | Description                       |
|---------|-------------|-----------------------------------|
| UINT8 * | <b>buff</b> | [Input] The buffer for reading    |
| UINT32  | <b>size</b> | [Input] The data size for reading |

Table 2-16. Returns for AmbaUSB API **AmbaUSBD\_System\_ControlRead()**.

### Return:

| Return     | Description |
|------------|-------------|
| 0          | Success     |
| 0xFFFFFFFF | Failure     |

Table 2-17. Returns for AmbaUSB API **AmbaUSBD\_System\_ControlRead()**.

### Example:

None

### See Also:

**AmbaUSBD\_System\_RegisterVendorRequest()**



## 2.1.8 AmbaUSBD\_System\_ControlWrite

### API Syntax:

```
UINT32 AmbaUSBD_System_ControlWrite (UINT8 *buff, UINT32 size)
```

### Function Description:

This function is used to write data to USB Host by the control endpoint. Note that it can only be called during data phase in vendor requests, that is, applications must call

**AmbaUSBD\_System\_RegisterVendorRequest()** to register a callback for vendor requests first, and then call this function in the **InterfaceVendorRequestFunc()** to write control data.

Calling it in other places will cause undefined behavior.

### Parameters:

| Type    | Parameter   | Description                     |
|---------|-------------|---------------------------------|
| UINT8 * | <b>buff</b> | [Input] The buffer to write     |
| UINT32  | <b>size</b> | [Input] The data size for write |

Table 2-18. Returns for AmbaUSB API **AmbaUSBD\_System\_ControlWrite()**.

### Return:

| Return     | Description |
|------------|-------------|
| 0          | Success     |
| 0xFFFFFFFF | Failure     |

Table 2-19. Returns for AmbaUSB API **AmbaUSBD\_System\_ControlWrite()**.

### Example:

None

### See Also:

**AmbaUSBD\_System\_RegisterVendorRequest()**



## 2.1.9 **AmbaUSBD\_System\_DataContactDetection**

### API Syntax:

UINT32 **AmbaUSBD\_System\_DataContactDetection** (UINT32 t1, UINT32 t2)

### Function Description:

It is used to detect the data contact(D+) after VBUS is detected. Applications can use this API to detect the USB Host type (Charger or PC) after the VBUS is connected.

### Parameters:

| Type   | Parameter | Description  |
|--------|-----------|--|
| UINT32 | t1        | The time (ms) for detection taking effect. 0 means "detect right now." |
| UINT32 | t2        | The detection period (ms)  |

Table 2-20. Parameters for AmbaUSB API **AmbaUSBD\_System\_DataContactDetection()**.

### Return:

| Return | Description                                     |
|--------|---|
| 0      | No data contacted. The host may be a charger.   |
| 1      | Data contact detected. The host should be a PC. |

Table 2-21. Returns for AmbaUSB API **AmbaUSBD\_System\_DataContactDetection()**.

### Example:

None

### See Also:

None



## 2.1.10 AmbaUSBD\_System\_DumpRegisters

### API Syntax:

```
void AmbaUSBD_System_DumpRegisters (void)
```

### Function Description:

This function is used to dump USB device registers to UART0. It is only for debug purpose.

### Parameters:

None

### Return:

None

### Example:

```
AmbaUSBD_System_DumpRegisters ();
```

### See Also:

None



## 2.1.11 AmbaUSBD\_System\_GetConnectSpeed

### API Syntax:

```
UINT32 AmbaUSBD_System_GetConnectSpeed (void)
```

### Function Description:

This function is used to get the current USB device connection speed.

### Parameters:

None

### Return:

| Return | Description |
|--------|-------------|
| 0      | Full speed  |
| 1      | High speed  |

Table 2-22. Returns for AmbaUSB API `AmbaUSBD_System_GetConnectSpeed()`.

### Example:

None

### See Also:

None



## 2.1.12 AmbaUSBD\_System\_GetDataConnWithVbus

### API Syntax:

```
UINT32 AmbaUSBD_System_GetDataConnWithVbus (void)
```

### Function Description:

This function is used to get the current setting of the device data connection. Device data connection will take effect when VBUS is detected.

### Parameters:

None

### Return:

| Return | Description   |
|--------|---|
| 0      | Data connect will NOT be enabled when VBUS is detected. USB Host will not recognize it until <i>AmbaUSBD_System_SetDataConn(1)</i> is called. |
| 1      | Data connect will be enabled when VBUS is detected. USB Host will recognize it immediately.   |

Table 2-23. Returns for AmbaUSB API *AmbaUSBD\_System\_GetDataConnWithVbus()*.

### Example:

None

### See Also:

None



## 2.1.13 AmbaUSBD\_System\_GetVbusPin

### API Syntax:

```
AMBA_GPIO_PIN_ID_e AmbaUSBD_System_GetVbusPin (void)
```

### Function Description:

This function is used to get the alternate GPIO pin for VBUS detection; some applications don't use formal VBUS pin for USB VBUS detection.

### Parameters:

None

### Return:

| Return             | Description                                 |
|--------------------|---|
| AMBA_GPIO_PIN_ID_e | The GPIO pin for alternative VBUS detection |

Table 2-24 Returns for AmbaUSB API *AmbaUSBD\_System\_Get\_VbusPin()*.

### Example:

None

### See Also:

None



## 2.1.14 **AmbaUSBD\_System\_GetVbusPinStatus**

### API Syntax:

```
UINT32 AmbaUSBD_System_GetVbusPinStatus (void)
```

### Function Description:

This function is used to get current hardware status of VBUS. Please note, this function is only for test purpose, applications should call **AmbaUSBD\_SystemGetVbusStatus()** to get VBUS status.

### Parameters:

None

### Return:

| Return | Description                      |
|--------|----------------------------------|
| 0      | VBUS connection is not detected. |
| 1      | VBUS connection is detected.     |

Table 2-25. Returns for AmbaUSB API **AmbaUSBD\_System\_GetVbusPinStatus()**.

### Example:

None

### See Also:

None



## 2.1.15 AmbaUSBD\_System\_GetVbusStatus

### API Syntax:

```
UINT32 AmbaUSBD_System_GetVbusStatus (void)
```

### Function Description:

This function is used to check if the USB cable is inserted or not by detecting the VBUS connection.

### Parameters:

None

### Return:

| Return | Description            |
|--------|------------------------|
| 0      | USB cable is removed.  |
| 1      | USB cable is inserted. |

Table 2-26. Returns for AmbaUSB API **AmbaUSBD\_System\_GetVbusStatus()**.

### Example:

None

### See Also:

None



## 2.1.16 AmbaUSBD\_System\_IsConfigured

### API Syntax:

```
UINT32 AmbaUSBD_System_IsConfigured (void)
```

### Function Description:

This function is used to report if the device is recognized and configured by the host PC.

### Parameters:

None

### Return:

| Return | Description            |
|--------|------------------------|
| 0      | USB is NOT configured. |
| 1      | USB is configured.     |

Table 2-27. Returns for AmbaUSB API **AmbaUSBD\_System\_IsConfigured()**.

### Example:

None

### See Also:

None



## 2.1.17 AmbaUSBD\_System\_RegisterVbusCallback

### API Syntax:

```
UINT32 AmbaUSBD_System_RegisterVbusCallback (USB_DEV_VBUS_CB_s *VbusCb)
```

### Function Description:

This function is used to register the callback functions for handling the VBUS and system status changes.

### Parameters:

| Type              | Parameter | Description  |
|-------------------|-----------|--|
| USB_DEV_VBUS_CB_s | VbusCb    | [Input] a collection of callback functions.<br>Please refer to Section 2.1.1.3 for more details. |

Table 2-28. Parameters for AmbaUSB API **AmbaUSBD\_System\_RegisterVbusCallback()**.

### Return:

| Return     | Description |
|------------|-------------|
| 0          | Success     |
| 0xFFFFFFFF | Failure     |

Table 2-29. Returns for AmbaUSB API **AmbaUSBD\_System\_RegisterVbusCallback()**.

### Example:

```
static void USB_VbusConnect(void) {  
    AmbaPrint("Vbus Connect!!");  
}  
  
static void USB_VbusDisconnect(void) {  
    AmbaPrint("Vbus Disconnect!!");  
}  
  
USB_DEV_VBUS_CB_s UsbVbusCb = {  
    USB_VbusConnect,  
    USB_VbusDisconnect
```



```
};

static UINT32 USB_RegisterVbusCallback(void){
    return AmbaUSBD_System_RegisterVbusCallback(&UsbVbusCb);
}
```

**See Also:**

None

For PROTRULY Confidential Only



## 2.1.18 AmbaUSBD\_System\_RegisterVendorRequest

### API Syntax:

```
UINT32 AmbaUSBD_System_RegisterVendorRequest  
(USB_VENDOR_REQUEST_INFO_s *info)
```

### Function Description:

This function is used to register a callback function for processing USB vendor requests.

### Parameters:

| Type                      | Parameter | Description                                       |
|---------------------------|-----------|---|
| USB_VENDOR_REQUEST_INFO_S | *info     | Please refer to section 2.1.1.4 for more details. |

Table 2-30. Parameters for AmbaUSB API **AmbaUSBD\_System\_RegisterVendorRequest()**.

### Return:

| Return     | Description |
|------------|-------------|
| 0          | Success     |
| 0xFFFFFFFF | Failure     |

Table 2-31. Returns for AmbaUSB API **AmbaUSBD\_System\_RegisterVendorRequest()**.

### Example:

```
static INT Simple_VendorRequest(UINT32 RequestType, UINT32 request,  
UINT32 value,  
                          UINT32 index,  UINT32 len)  
{  
    INT status = 0;  
    switch (request) {  
        case SIMPLE_REQUEST_IN:  
            // Control-In request.  
            AmbaPrint("%s(): request_in=0x%x, request_tye=0x%x,  
                      request_value=0x%x,
```



```
request_index=0x%x, request_length = %d",
__func__, request, RequestType, value, index, len);
{
    int i = 0;
    for (i=0; i<BUFF_LEN; i++) {
        SimpleBuff[i] = i;
    }
}
AmбаUSBD_System_ControlWrite(SimpleBuff, len);
break;
case SIMPLE_REQUEST_OUT:
// Control-Out request.
AmбаPrint("%s(): request_out=0x%x, request_tye=0x%x,
request_value=0x%x, request_index=0x%x, request_length =
%d",
__func__, request, RequestType, value, index, len);
AmбаUSB_System_ControlRead(SimpleBuff, len);
{
    int i = 0;
    for (i=0; i<len; i++) {
        AmбаPrint("data [0x%x] = 0x%x", i, SimpleBuff[i]);
    }
}
break;
default:
AmбаPrint("%s(): Don't support STD Control Request 0x%X", __func__,
request);
/* Unknown function. Stall the endpoint. */
status = -1;
break;
}
return status;
}
```



```
USB_VENDOR_REQUEST_INFO_s SimpleIfno = {  
    UDC_CLASS_SIMPLE,  
    Simple_VendorRequest  
};  
AmbaUSBD_System_RegisterVendorRequest (&SimpleIfno);
```

**See Also:**

None

For Confidential PROTRULY Only



## 2.1.19 AmbaUSBD\_System\_ClassHook

### API Syntax:

```
UINT32 AmbaUSBD_System_ClassHook (USB_CLASS_INIT_s* config)
```

### Function Description:

This function initializes the Ambarella USB device and the specified USB class. The application should call **AmbaUSBD\_Descriptor\_Init()** to input the customer's device information into the USB descriptors before it is called.

### Parameters:

| Type             | Parameter | Description  |
|------------------|-----------|--|
| USB_CLASS_INIT_s | config    | [Input] configuration information. Please refer to Section 2.1.1.2 for more details. |

Table 2-32. Parameters for AmbaUSB API **AmbaUSBD\_System\_ClassHook()**.

### Return:

| Return     | Description |
|------------|-------------|
| 0          | Success     |
| 0xFFFFFFFF | Failure     |

Table 2-33. Returns for AmbaUSB API **AmbaUSBD\_System\_ClassHook()**.

### Example:

```
USB_CLASS_INIT_s ClassConfig = {UDC_CLASS_NONE, 0, 0};  
/* Pre Initialization of USB Class*/  
AmbaUSB_Custom_SetDeviceInfo(AmbaUsbDeviceClass);  
AmbaUSBD_System_ClassHook (&ClassConfig); /* Hook USB Class*/
```

### See Also:

None



## 2.1.20 AmbaUSBD\_System\_SetConnect

### API Syntax:

```
UINT32 AmbaUSBD_System_SetConnect (UINT32 enable)
```

### Function Description:

This function is used to simulate the VBUS plug/unplug. This is used only for test purposes; users are not to call the function in the normal operation flow.

### Parameters:

| Type   | Parameter | Description  |
|--------|-----------|--|
| UINT32 | enable    | <p>[Input]<br/>0 - Simulate VBUS unplug<br/>1 - Simulate VBUS plug</p> |

Table 2-34. Parameters for AmbaUSB API **AmbaUSBD\_System\_SetConnect()**.

### Return:

| Return     | Description |
|------------|-------------|
| 0          | Success     |
| 0xFFFFFFFF | Failure     |

Table 2-35. Returns for AmbaUSB API **AmbaUSBD\_System\_SetConnect()**.

### Example:

```
if (input == 1) {  
    if (AmbaUSB_GetCurClass() == AMBAUSB_CLASS_NONE) {  
        AmbaUSB_ClassInit(AMBAUSB_CLASS_MSC);  
    } else {  
        AmbaUSBD_System_SetConnect(1);  
    }  
} else if ( input == 0) {  
    AmbaUSBD_System_SetConnect (0);  
}
```



See Also :

None

For PROTRULY Confidential Only



### 2.1.21 AmbaUSBD\_System\_SetDataConn

#### API Syntax:

```
UINT32 AmbaUSBD_System_SetDataConn (UINT32 value)
```

#### Function Description:

This function is used to enable/disable the device data connection. It is equivalent to pull-up/pull-down D+ for USB insertion/removal and speed identification. Please refer to USB 2.0 specification, Section 7.1.5, Device Speed Identification for detailed information.

#### Parameters:

| Type   | Parameter | Description   |
|--------|-----------|---|
| UINT32 | value     | <p>[Input]</p> <p>0: Data connect will NOT be enabled. USB Host will consider an USB device that is disconnected immediately.</p> <p>1: Data connect will be enabled. USB Host will recognize it immediately.</p> |

Table 2-36. Parameters for AmbaUSB API **AmbaUSBD\_System\_SetDataConn()**.

#### Return:

| Return     | Description |
|------------|-------------|
| 0          | Success     |
| 0xFFFFFFFF | Failure     |

Table 2-37. Returns for AmbaUSB API **AmbaUSBD\_System\_SetDataConn()**.

#### Example:

None

#### See Also:

None



## 2.1.22 AmbaUSBD\_System\_SetDataConnWithVbus

### API Syntax:

```
UINT32 AmbaUSBD_System_SetDataConnWithVbus (UINT32 value)
```

### Function Description:

This function is used to set the current setting of device data connection. Device data connection will take effect when VBUS is detected.

### Parameters:

| Type   | Parameter | Description   |
|--------|-----------|---|
| UINT32 | value     | <p>[Input]</p> <p>0: Data connect will NOT be enabled when VBUS is detected. USB Host will not recognize it until <b>AmbaUSBD_System_SetDataConn</b> (1) is called.</p> <p>1: Data connect will be enabled when VBUS is detected. USB Host will recognize it immediately.</p> |

Table 2-38. Parameters for AmbaUSB API **AmbaUSBD\_System\_SetDataConnWithVbus()**.

### Return:

| Return     | Description |
|------------|-------------|
| 0          | Success     |
| 0xFFFFFFFF | Failure     |

Table 2-39. Returns for AmbaUSB API **AmbaUSBD\_System\_SetDataConnWithVbus()**.

### Example:

None

### See Also:

None



### 2.1.23 AmbaUSBD\_System\_SetSoftwareConnect

#### API Syntax:

```
void AmbaUSBD_System_SetSoftwareConnect (UINT32 enable, UINT32 connect)
```

#### Function Description:

This function is used to select the software VBUS source and then simulate VBUS connect/disconnect. This is used only for test purposes; users are not to call the function in the normal operation flow.

#### Parameters:

| Type   | Parameter      | Description  |
|--------|----------------|--|
| UINT32 | <b>enable</b>  | [Input] Enable software VBUS source                                |
| UINT32 | <b>connect</b> | [Input] Software VBUS source status<br>0: Disconnect<br>1: Connect |

Table 2-40. Parameters for AmbaUSB API **AmbaUSBD\_System\_SetSoftwareConnect()**.

#### Return:

None

#### Example:

None

#### See Also:

None



## 2.1.24 AmbaUSBD\_System\_SetUsbOwner

### API Syntax:

```
void AmbaUSBD_System_SetUsbOwner (USB_IRQ_OWNER_e owner, int update)
```

### Function Description:

This function is used to set the USB device IRQ owner. Because there might be more than one operating system running on one Ambarella chip and only one of them can access USB at the same time, applications must call this function to assign USB owner to one of these operating systems.

Note that this function can only set the USB device IRQ owner to RTOS. To set the USB device IRQ owner to others, please refer to corresponding API documents.

### Parameters:

| Type            | Parameter     | Description  |
|-----------------|---------------|--|
| USB_IRQ_OWNER_e | <b>owner</b>  | [Input]<br>USB_IRQ_RTOS - IRQ owner is RTOS (ThreadX.)                                     |
| int             | <b>update</b> | [Input]<br>0: Disable IRQ after owner is assigned<br>1: Enable IRQ after owner is assigned |

Table 2-41. Parameters for AmbaUSB API **AmbaUSBD\_System\_SetUsbOwner()**.

### Return:

None

### Example:

None

### See Also:

None



## 2.1.25 AmbaUSBD\_System\_SetIsrTaskInfo

### API Syntax:

```
UINT32 AmbaUSBD_System_SetIsrTaskInfo (UDC_TASKINFO_s *info)
```

### Function Description:

This function is used to set the information for creating an ISR task in USB driver. An ISR task handles and dispatches all USB interrupts to corresponding functions.

It should be called before **AmbaUSB\_System\_DeviceSystemSetup()**.

### Parameters:

| Type             | Parameter   | Description   |
|------------------|-------------|---|
| UDC_TASKINFO_s * | <b>info</b> | [Input] Task information. Please refer to Section 2.1.1.5 for more details. |

Table 2-42. Parameters for AmbaUSB API **AmbaUSBD\_System\_SetIsrTaskInfo()**.

### Return:

| Return | Description |
|--------|-------------|
| 0      | Success     |
| -1     | Failure     |

Table 2-43. Returns for AmbaUSB API **AmbaUSBD\_System\_SetIsrTaskInfo()**.

### Example:

```
// set the stack size to 1KB, priority to 10, and AffinityMask to CORE 0.  
UDC_TASKINFO_s ti;  
ti.StackSize = 1024;  
ti.Priority = 10;  
ti.AffinityMask = 0x01;  
AmbaUSBD_System_SetIsrTaskInfo (&ti);
```

### See Also:

None



## 2.1.26 AmbaUSBD\_System\_GetIsrTaskInfo

### API Syntax:

```
UINT32 AmbaUSBD_System_GetIsrTaskInfo (UDC_TASKINFO_s *info)
```

### Function Description:

This function is used to get the information for creating USB driver ISR task.

### Parameters:

| Type             | Parameter | Description  |
|------------------|-----------|--|
| UDC_TASKINFO_s * | info      | [Output] Task information. Please refer to Section 2.1.1.5 for more details. |

Table 2-44. Parameters for AmbaUSB API **AmbaUSBD\_System\_GetIsrTaskInfo()**.

### Return:

| Return | Description |
|--------|-------------|
| 0      | Success     |
| -1     | Failure     |

Table 2-45. Returns for AmbaUSB API **AmbaUSBD\_System\_GetIsrTaskInfo()**.

### Example:

```
UDC_TASKINFO_s ti;  
AmbaUSBD_System_GetIsrTaskInfo (&ti);
```

### See Also:

None



## 2.1.27 AmbaUSBD\_System\_SetVbusTimerTaskInfo

### API Syntax:

```
UINT32 AmbaUSBD_System_SetVbusTimerTaskInfo (UDC_TASKINFO_s *info)
```

### Function Description:

This function is used to set the information for creating USB Vbus Timer task. Vbus Timer task checks VBUS status periodically and reports to Applications when VBUS status changes.

It should be called before **AmbaUSBD\_System\_RegisterVbusCallback()**.

### Parameters:

| Type             | Parameter   | Description   |
|------------------|-------------|---|
| UDC_TASKINFO_s * | <b>info</b> | [Input] Task information. Please refer to Section 2.1.1.5 for more details. |

Table 2-46. Parameters for AmbaUSB API **AmbaUSBD\_System\_SetVbusTimerTaskInfo()**.

### Return:

| Return     | Description |
|------------|-------------|
| 0          | Success     |
| 0xFFFFFFFF | Failure     |

Table 2-47. Returns for AmbaUSB API **AmbaUSBD\_System\_SetVbusTimerTaskInfo()**.

### Example:

```
// set the stack size to 1KB, priority to 255, and AffinityMask to CORE 0.  
UDC_TASKINFO_s ti;  
ti.StackSize = 1024;  
ti.Priority = 255;  
ti.AffinityMask = 0x01;  
AmbaUSBD_System_SetVbusTimerTaskInfo (&ti);
```



See Also:

None

For PROTRULY Confidential Only



## 2.1.28 AmbaUSBD\_System\_GetVbusTimerTaskInfo

### API Syntax:

```
UINT32 AmbaUSBD_System_GetVbusTimerTaskInfo (UDC_TASKINFO_s *info)
```

### Function Description:

This function is used to get the information for creating USB Vbus Timer task.

### Parameters:

| Type             | Parameter | Description  |
|------------------|-----------|--|
| UDC_TASKINFO_s * | info      | [Output] Task information. Please refer to Section 2.1.1.5 for more details. |

Table 2-48. Parameters for AmbaUSB API **AmbaUSBD\_System\_GetVbusTimerTaskInfo()**.

### Return:

| Return     | Description |
|------------|-------------|
| 0          | Success     |
| 0xFFFFFFFF | Failure     |

Table 2-49. Returns for AmbaUSB API **AmbaUSBD\_System\_GetVbusTimerTaskInfo()**.

### Example:

```
UDC_TASKINFO_s ti;  
AmbaUSBD_System_GetVbusTimerTaskInfo (&ti);
```

### See Also:

None



## 2.1.29 AmbaUSBD\_System\_SetBulkInTimeout

### API Syntax:

```
void AmbaUSBD_System_SetBulkInTimeout (UINT32 value)
```

### Function Description:

This function is used to set the timeout value for bulk-in transfer.

### Parameters:

| Type   | Parameter | Description                                     |
|--------|-----------|---|
| UINT32 | value     | [Input] timeout value. The unit is mili-second. |

Table 2-50. Parameters for AmbaUSB API *AmbaUSBD\_System\_SetBulkInTimeout()*.

### Return:

None

### Example:

None

### See Also:

None



## 2.1.30 AmbaUSB\_System\_SetDeviceConfigDetectTimeout

### API Syntax:

```
UINT32 AmbaUSB_System_SetDeviceConfigDetectTimeout (UINT32 value)
```

### Function Description:

This function is used to set the timeout value for device configuration detection. When VBUS connection is detected, USB Vbus Timer task will wait for occurrence of device configuration set by Host. If it times out, callback function **SystemConfigured(0xFFFF)** is called.

Please refer to section 2.1.2.6 for the information of callback function **SystemConfigured()**.

### Parameters:

| Type   | Parameter | Description                                     |
|--------|-----------|---|
| UINT32 | value     | [Input] timeout value. The unit is mili-second. |

Table 2-51. Parameters for AmbaUSB API **AmbaUSB\_System\_SetDeviceConfigDetectTimeOut()**.

### Return:

| Return     | Description |
|------------|-------------|
| 0          | Success     |
| 0xFFFFFFFF | Failure     |

Table 2-52. Returns for AmbaUSB API **AmbaUSB\_System\_SetDeviceConfigDetectTimeout()**.

### Example:

None

### See Also:

None



## 2.2      Customize APIs

This section describes APIs for system customization.

For PROTRULY Confidential Only



## 2.2.1 Data Structures

### 2.2.1.1 CLASS\_STACK\_INIT\_INFO\_s

| Type    | Parameter              | Description                       |
|---------|------------------------|-----------------------------------|
| UINT8 * | <b>DescFrameworkFs</b> | USB full speed descriptor         |
| UINT8 * | <b>DescFrameworkHs</b> | USB high speed descriptor         |
| UINT8 * | <b>StrFramework</b>    | USB string descriptor             |
| UINT8 * | <b>LangIDFramework</b> | USB language descriptor           |
| UINT32  | <b>DescSizeFs</b>      | Size of USB full speed descriptor |
| UINT32  | <b>DescSizeHs</b>      | Size of USB high speed descriptor |
| UINT32  | <b>StrSize</b>         | Size of USB string descriptor     |
| UINT32  | <b>LangIDSize</b>      | Size of USB language descriptor   |

Table 2-53. Definition of **CLASS\_STACK\_INIT\_INFO\_s**.

### 2.2.1.2 USB\_DEV\_INFO\_s

| Type    | Parameter                | Description   |
|---------|--------------------------|---|
| UINT16  | <b>idVendor</b>          | The idVendor filed in the Standard Device Descriptor    |
| UINT16  | <b>idProduct</b>         | The idProduct filed in the Standard Device Descriptor   |
| UINT16  | <b>bcdDevice</b>         | The bcdDevice filed in the Standard Device Descriptor   |
| UINT16  | <b>LangID</b>            | The supported Language ID in the String Descriptor      |
| UINT8 * | <b>StrFramework</b>      | The buffer contains String Descriptors                  |
| UINT32  | <b>StrFrameWorkSize</b>  | The total length of String Descriptors                  |
| UINT8   | <b>MaxPowerConfig1</b>   | The bMaxPower filed in 1st Configuration Descriptor     |
| UINT8   | <b>AttributesConfig1</b> | The bmAttributes filed in 1st Configuration Descriptor  |
| UINT8   | <b>MaxPowerConfig2</b>   | The bMaxPower filed in 2nd Configuration Descriptor.    |
| UINT8   | <b>AttributesConfig2</b> | The bmAttributes filed in 2nd Configuration Descriptor. |

Table 2-54. Definition of **USB\_DEV\_INFO\_s**.



## 2.2.2 AmbaUSBD\_Descriptor\_Init

### API Syntax:

```
UINT32 AmbaUSBD_Descriptor_Init (CLASS_STACK_INIT_INFO_s *info)
```

### Function Description:

This API is used for applications to customize USB descriptors and should be called before the **AmbaUSB\_DeviceSystemSetup()** function is invoked.

### Parameters:

| Type                     | Parameter | Description   |
|--------------------------|-----------|---|
| CLASS_STACK_INIT_INFO_s* | info      | USB descriptor information. Please refer to Section 2.2.1.1 for more details. |

Table 2-55. Parameters for AmbaUSB API **AmbaUSBD\_Descriptor\_Init()**.

### Return:

| Return | Description |
|--------|-------------|
| 0      | Success     |
| 0xFF   | Failure     |

Table 2-56. Returns for AmbaUSB API **AmbaUSBD\_Descriptor\_Init()**.

### Example:

```
CLASS_STACK_INIT_INFO_s Classinfo = {  
    .DescFrameworkFs = MscDescFs,  
    .DescFrameworkHs = MscDescHs,  
    .StrFramework    = MscStr,  
    .LangIDFramework = LangID,  
    .DescSizeFs      = sizeof(MscDescFs),  
    .DescSizeHs      = sizeof(MscDescHs),  
    .StrSize         = sizeof(MscStr),  
    .LangIDSize     = sizeof(LangID),  
};  
AmbaUSBD_Descriptor_Init (&Classinfo);
```



See Also:

None

For PROTRULY Confidential Only



### 2.2.3 AmbaUSBD\_Descriptor\_SetCustomize

#### API Syntax:

```
UINT32 AmbaUSBD_Descriptor_SetCustomize (UDC_CLASS_e UdcClass,  
USB_DEV_INFO_s *desc)
```

#### Function Description:

This API is used for applications to customize some fields in USB descriptors and should be called before the **AmbaUSB\_DeviceSystemSetup()** function is invoked.

#### Parameters:

| Type             | Parameter | Description   |
|------------------|-----------|---|
| UDC_CLASS_e      | UdcClass  | [Input] USB Class ID:<br><b>UDC_CLASS_NONE</b> - No USB class<br><b>UDC_CLASS_MSC</b> - Mass Storage class<br><b>UDC_CLASS_MTP</b> - PTP/MTP class<br><b>UDC_CLASS_PICT</b> - Pictbridge class<br><b>UDC_CLASS_CDC_ACM</b> - RS232 over USB class (single instance)<br><b>UDC_CLASS_CUSTOM</b> - Custom class<br><b>UDC_CLASS_STREAM</b> - Stream class<br><b>UDC_CLASS_HID</b> - HID class<br><b>UDC_CLASS_SIMPLE</b> - Simple class<br><b>UDC_CLASS_UVC</b> - USB Video class<br><b>UDC_CLASS_MIX_STG</b> - MTP+MSC class<br><b>UDC_CLASS_CDC_ACM_MULTI</b> - RS232 over USB class (multiple instances) |
| USB_DEV_INFO_s * | desc      | [Input] USB descriptor information. Please refer to Section 2.2.1.1 for more details.   |

Table 2-57. Parameters for AmbaUSB API **AmbaUSBD\_Descriptor\_SetCustomize()**.



**Return:**

| Return | Description |
|--------|-------------|
| 0      | Success     |
| 0xFF   | Failure     |

Table 2-58. Returns for AmbaUSB API *AmbaUSBD\_Descriptor\_SetCustomize()*.

**Example:**

None

**See Also:**

None



## 2.3 Mass Storage Class APIs

This section describes APIs for Mass Storage Class.

For PROTRULY Confidential Only



## 2.3.1 Data Structures

### 2.3.1.1 MSC\_FSLIB\_OP\_s

| Type             | Parameter           | Description  |
|------------------|---------------------|--|
| Function pointer | <b>SectorRead</b>   | Callback function for media to read. Please refer to Section 2.3.2.1 for more details.             |
| Function pointer | <b>SectorWrite</b>  | Callback function for media to write. Please refer to Section 2.3.2.2 for more details.            |
| Function pointer | <b>GetMediaInfo</b> | Callback function for getting media information. Please refer to Section 2.3.2.3 for more details. |

Table 2-59. Definition of **MSC\_FSLIB\_OP\_s**.

### 2.3.1.2 AMBA\_SCM\_STATUS\_s

| Type   | Parameter           | Description   |
|--------|---------------------|---|
| UINT8  | <b>CardPresent</b>  | Check whether a card is present or not.<br>0: Not present<br>1: Present |
| UINT8  | <b>BusWidth</b>     | Data bus width  |
| UINT8  | <b>WriteProtect</b> | Checks whether card is write protected or not.                          |
| UINT8  | <b>OS</b>           | Local/remote OS   |
| UINT32 | <b>CardType</b>     | Type of card  |
| UINT32 | <b>ManfID</b>       | Manufacture ID  |
| UINT32 | <b>Version</b>      | Version   |
| char   | <b>Name[64]</b>     | Name of card (if any)   |
| int    | <b>Format</b>       | File System Format of card  |
| UINT32 | <b>SecsCnt</b>      | Number of sectors   |
| UINT16 | <b>SecSize</b>      | Sector size in bytes  |
| char   | <b>Speed[16]</b>    | Current card speed  |
| void * | <b>Extra</b>        | Pointer to extra information  |

Table 2-60. Definition of **AMBA\_SCM\_STATUS\_s**.



## 2.3.2 Callback functions

### 2.3.2.1 SectorRead

#### API Syntax:

```
INT32 SectorRead (int SlotID, UINT8 *pBuf, UINT32 Sector, UINT32 Sectors)
```

#### Function Description:

This function is called when the MSC “Read” command is received.

#### Parameters:

| Type    | Parameter      | Description   |
|---------|----------------|---|
| int     | <b>SlotID</b>  | [Input] Target disk drive:<br><b>SCM_SLOT_FL0</b> - NAND Flash driver (A)<br><b>SCM_SLOT_FL1</b> - NAND Flash driver (B)<br><b>SCM_SLOT_SD0</b> - SD/MMC slot 1 (C)<br><b>SCM_SLOT_SD1</b> - SD/MMC slot 2 (D)<br><b>SCM_SLOT_RD</b> - RAM disk (E)<br><b>SCM_SLOT_IDX</b> - Video Index partition (F)<br><b>SCM_SLOT_PRF</b> - Preference partition (G)<br><b>SCM_SLOT_CAL</b> - Calibration partition (H)<br><b>SCM_SLOT_USB</b> - USB Host (I)<br><b>SCM_SLOT_RF1</b> - ROM File system 1 (Y)<br><b>SCM_SLOT_RF2</b> - ROM File system 1 (Z) |
| UINT8 * | <b>pBuf</b>    | [Output] The buffer to read   |
| UINT32  | <b>Sector</b>  | [Input] The start sector to read  |
| UINT32  | <b>Sectors</b> | [Input] The total sectors to read   |

Table 2-61. Parameters for AmbaUSB API **SectorRead()**.



**Return:**

| Return | Description |
|--------|-------------|
| 0      | Success     |
| -1     | Failure     |

Table 2-62. Returns for AmbaUSB API **SectorRead()**.

**Example:**

None

**See Also:**

None



### 2.3.2.2 SectorWrite

#### API Syntax:

```
INT32 SectorWrite (int SlotID, UINT8 *pBuf, UINT32 Sector, UINT32 Sectors)
```

#### Function Description:

This function is called when the MSC “Write” command is received.

#### Parameters:

| Type    | Parameter      | Description   |
|---------|----------------|---|
| int     | <b>SlotID</b>  | [Input] Target disk drive:<br><b>SCM_SLOT_FL0</b> - NAND Flash driver (A)<br><b>SCM_SLOT_FL1</b> - NAND Flash driver (B)<br><b>SCM_SLOT_SD0</b> - SD/MMC slot 1 (C)<br><b>SCM_SLOT_SD1</b> - SD/MMC slot 2 (D)<br><b>SCM_SLOT_RD</b> - RAM disk (E)<br><b>SCM_SLOT_IDX</b> - Video Index partition (F)<br><b>SCM_SLOT_PRF</b> - Preference partition (G)<br><b>SCM_SLOT_CAL</b> - Calibration partition (H)<br><b>SCM_SLOT_USB</b> - USB Host (I)<br><b>SCM_SLOT_RF1</b> - ROM File system 1 (Y)<br><b>SCM_SLOT_RF2</b> - ROM File system 1 (Z) |
| UINT8 * | <b>pBuf</b>    | [Output] The buffer for the write function  |
| UINT32  | <b>Sector</b>  | [Input] The start sector to write   |
| UINT32  | <b>Sectors</b> | [Input] The total number of sectors to write  |

Table 2-63. Parameters for AmbaUSB API **SectorWrite()**.

#### Return:

| Return | Description |
|--------|-------------|
| 0      | Success     |



| Return | Description |
|--------|-------------|
| -1     | Failure     |

Table 2-64. Returns for AmbaUSB API **SectorWrite()**.

**Example:**

None

**See Also:**

None



### 2.3.2.3 GetMediaInfo

#### API Syntax:

```
INT32 GetMediaInfo (int SlotID, AMBA_SCM_STATUS_s *pStatus)
```

#### Function Description:

This function is called when the MSC driver wants to know the status of a disk drive.

#### Parameters:

| Type                | Parameter | Description   |
|---------------------|-----------|---|
| int                 | SlotID    | [Input] Target disk drive:<br><b>SCM_SLOT_FL0</b> - NAND Flash driver (A)<br><b>SCM_SLOT_FL1</b> - NAND Flash driver (B)<br><b>SCM_SLOT_SD0</b> - SD/MMC slot 1 (C)<br><b>SCM_SLOT_SD1</b> - SD/MMC slot 2 (D)<br><b>SCM_SLOT_RD</b> - RAM disk (E)<br><b>SCM_SLOT_IDX</b> - Video Index partition (F)<br><b>SCM_SLOT_PRF</b> - Preference partition (G)<br><b>SCM_SLOT_CAL</b> - Calibration partition (H)<br><b>SCM_SLOT_USB</b> - USB Host (I)<br><b>SCM_SLOT_RF1</b> - ROM File system 1 (Y)<br><b>SCM_SLOT_RF2</b> - ROM File system 1 (Z) |
| AMBA_SCM_STATUS_s * | pStatus   | [Output] media status information. Please refer to Section 2.3.1.2.   |

Table 2-65. Parameters for AmbaUSB API **GetMediaInfo()**.

#### Return:

| Return | Description |
|--------|-------------|
| 0      | Success     |
| -1     | Failure     |

Table 2-66. Returns for AmbaUSB API **GetMediaInfo()**.



**Example:**

None

**See Also:**

None

For PROTRULY Confidential Only



#### 2.3.2.4 UDC\_MSC\_CBK\_PROCESSCBW

##### API Syntax:

```
INT32 UDC_MSC_CBK_PROCESSCBW (UINT8 *cbwcb, UINT32 cbwcb_length, UINT8  
*status)
```

##### Function Description:

It is called when MSC driver receives a Command Block Wrapper(CBW) when it is registered by `AmbaUSBD_Msc_SetCbwCallback()`. Note that if status is set to `UDC_MSC_CSW_STS_GOOD` and the return value is `UDC_MSC_CBW_PROCESS_DONE`, then MSC driver will send a Command Status Wrapper(CSW), otherwise, no CSW is sent.

Applications should not wait for a long time or disable USB function in this callback, otherwise USB Host operating system will consider the MSC device is corrupted due to no response on MSC protocol.



Below is the CBW format defined in USB Mass Storage Class:

| Byte               | bit | 7                             | 6 | 5 | 4                   | 3              | 2 | 1 | 0 |  |  |  |  |
|--------------------|-----|-------------------------------|---|---|---------------------|----------------|---|---|---|--|--|--|--|
| 0-3                |     | <i>dCBWSignature</i>          |   |   |                     |                |   |   |   |  |  |  |  |
| 4-7                |     | <i>dCBWTag</i>                |   |   |                     |                |   |   |   |  |  |  |  |
| 8-11<br>(08h-0Bh)  |     | <i>dCBWDataTransferLength</i> |   |   |                     |                |   |   |   |  |  |  |  |
| 12<br>(0Ch)        |     | <i>bmCBWFlags</i>             |   |   |                     |                |   |   |   |  |  |  |  |
| 13<br>(0Dh)        |     | Reserved (0)                  |   |   |                     | <i>bCBWLUN</i> |   |   |   |  |  |  |  |
| 14<br>(0Eh)        |     | Reserved (0)                  |   |   | <i>bCBWCBLength</i> |                |   |   |   |  |  |  |  |
| 15-30<br>(0Fh-1Eh) |     | <i>CBWCB</i>                  |   |   |                     |                |   |   |   |  |  |  |  |

Figure 2-1 Command Block Wrapper.

#### Parameters:

| Type    | Parameter           | Description  |
|---------|---------------------|--|
| UINT8 * | <b>cbwcb</b>        | [Input] Command block data of CBW. Please refer to <b>CBWCB</b> field of Figure 2-1.   |
| UINT32  | <b>cbwcb_length</b> | [Input] The length of cbwcb. It also represents the <b>bCBWCBLength</b> field of Figure 2-1.   |
| UINT8 * | <b>status</b>       | [Output] App must set it to one of the following value:<br>UDC_MSC_CSW_STS_GOOD: Good status<br>UDC_MSC_CSW_STS_CMD_FAILED: Command failed.<br>UDC_MSC_CSW_STS_PHASE_ERR: Phase error<br>UDC_MSC_CSW_STS_SENDED: Status data has sended.<br>UDC_MSC_CSW_STS_USB_ERR: USB sends/receives error. |

Table 2-67. Parameters for AmbaUSB API **UDC\_MSC\_CBK\_PROCESSCBW()**.



**Return:**

| Return                       | Description  |
|------------------------------|--|
| UDC_MSC_CBW_PROCESS_DONE     | Callback function has processed the CBW. MSC class driver will not handle it.            |
| UDC_MSC_CBW_PROCESS_CONTINUE | Callback function has not processed the CBW. MSC class driver will handle it afterwards. |

Table 2-68. Returns for AmbaUSB API **UDC\_MSC\_CBK\_PROCESSCBW()**.

**Example:**

None

**See Also:**

None



### 2.3.3 AmbaUSBD\_Msc\_Init

#### API Syntax:

```
UINT32 AmbaUSBD_Msc_Init (MSC_FSLIB_OP_s *op)
```

#### Function Description:

This function is used to register storage device operation functions including read, write, and media information function. This is a low-level glue API that works with FIO API. Since the FIO API would not change in the future, this hook function may be fixed in the library in the near future.

#### Parameters:

| Type             | Parameter | Description   |
|------------------|-----------|---|
| MSC_FSLIB_OP_s * | op        | Callback functions for media operations.<br>Please refer to Section 2.3.1.1 for more details. |

Table 2-69. Parameters for AmbaUSB API **AmbaUSBD\_Msc\_Init()**.

#### Return:

| Return | Description |
|--------|-------------|
| 0      | Success     |
| -1     | Failure     |

Table 2-70. Returns for AmbaUSB API **AmbaUSBD\_Msc\_Init()**.

#### Example:

```
static MSC_FSLIB_OP_s MscFsOp = {  
    AmbaSCM_Read, /* Read API exported by card manager */  
    AmbaSCM_Write, /* Write API exported by card manager */  
    AmbaSCM_GetSlotStatus /* Slot status API exported by card manager */  
};  
AmbaUSBD_Msc_Init (&MscFsOp);
```



See Also :

None

For PROTRULY Confidential Only



### 2.3.4      **AmbaUSBD\_Msc\_SetInfo**

#### API Syntax:

```
UINT32 AmbaUSBD_Msc_SetInfo (UINT8 *vendor, UINT8 *product, UINT8 *revision)
```

#### Function Description:

This command is used to specify the descriptive information of the USB mass storage drive. They represent Vendor Identification, Product Identification, and Product Revision Level fields correspondingly in the Standard INQUIRY data in the SCSI primary commands. Note that the SCSI primary commands specification defines the maximal length of these three strings. The maximal length of vendor, product, and revision are 8, 16, and 4 respectively. If the specified string is too long, it will be truncated.



**Table 81 — Standard INQUIRY data format**

| Bit<br>Byte | 7                       | 6                          | 5                   | 4                      | 3                    | 2                | 1                | 0                   |
|-------------|-------------------------|----------------------------|---------------------|------------------------|----------------------|------------------|------------------|---------------------|
| 0           | PERIPHERAL QUALIFIER    |                            |                     | PERIPHERAL DEVICE TYPE |                      |                  |                  |                     |
| 1           | RMB                     | Reserved                   |                     |                        |                      |                  |                  |                     |
| 2           | VERSION                 |                            |                     |                        |                      |                  |                  |                     |
| 3           | Obsolete                | Obsolete                   | NORMACA             | HiSUP                  | RESPONSE DATA FORMAT |                  |                  |                     |
| 4           | ADDITIONAL LENGTH (n-4) |                            |                     |                        |                      |                  |                  |                     |
| 5           | SCCS                    | ACC                        | TPGS                |                        | 3PC                  | Reserved         |                  | PROTECT             |
| 6           | BQUE                    | ENCSERV                    | VS                  | MULTIP                 | MCHNGR               | Obsolete         | Obsolete         | ADDR16 <sup>a</sup> |
| 7           | Obsolete                | Obsolete                   | WBUS16 <sup>a</sup> | SYNC <sup>a</sup>      | LINKED               | Obsolete         | CMDQUE           | VS                  |
| 8           | (MSB)                   | T10 VENDOR IDENTIFICATION  |                     |                        |                      |                  |                  | (LSB)               |
| 15          |                         |                            |                     |                        |                      |                  |                  |                     |
| 16          | (MSB)                   | PRODUCT IDENTIFICATION     |                     |                        |                      |                  |                  | (LSB)               |
| 31          |                         |                            |                     |                        |                      |                  |                  |                     |
| 32          | (MSB)                   | PRODUCT REVISION LEVEL     |                     |                        |                      |                  |                  | (LSB)               |
| 35          |                         |                            |                     |                        |                      |                  |                  |                     |
| 36          |                         | Vendor specific            |                     |                        |                      |                  |                  |                     |
| 55          |                         |                            |                     |                        |                      |                  |                  |                     |
| 56          | Reserved                |                            |                     | CLOCKING <sup>a</sup>  |                      | QAS <sup>a</sup> | IUS <sup>a</sup> |                     |
| 57          | Reserved                |                            |                     |                        |                      |                  |                  |                     |
| 58          | (MSB)                   | VERSION DESCRIPTOR 1       |                     |                        |                      |                  |                  | (LSB)               |
| 59          |                         |                            |                     |                        |                      |                  |                  |                     |
|             |                         | :                          |                     |                        |                      |                  |                  |                     |
| 72          | (MSB)                   | VERSION DESCRIPTOR 8       |                     |                        |                      |                  |                  | (LSB)               |
| 73          |                         |                            |                     |                        |                      |                  |                  |                     |
| 74          |                         | Reserved                   |                     |                        |                      |                  |                  |                     |
| 95          |                         | Vendor specific parameters |                     |                        |                      |                  |                  |                     |
| 96          |                         | Vendor specific            |                     |                        |                      |                  |                  |                     |
| n           |                         |                            |                     |                        |                      |                  |                  |                     |

<sup>a</sup> The meanings of these fields are specific to SPI-5 (see 6.4.3). For SCSI transport protocols other than the SCSI Parallel Interface, these fields are reserved.

*Table 2-71 Standard Inquiry Data Format.*



**Parameters:**

| Type    | Parameter       | Description                |
|---------|-----------------|----------------------------|
| UINT8 * | <b>Vendor</b>   | Vendor name (8 bytes)      |
| UINT8 * | <b>Product</b>  | Product name (16 bytes)    |
| UINT8 * | <b>Revision</b> | Product revision (4 bytes) |

Table 2-72. Parameters for AmbaUSB API **AmbaUSBD\_Msc\_SetInfo()**.

**Return:**

| Return     | Description |
|------------|-------------|
| 0          | Success     |
| 0xFFFFFFFF | Failure     |

Table 2-73. Returns for AmbaUSB API **AmbaUSBD\_Msc\_SetInfo()**.

**Example:**

```
UINT8    AmbaStorageVendorId[]="Ambarella";           // 8 Bytes
UINT8    AmbaStorageProductId[]="A9 DSC Platform ";   // 16 Bytes
UINT8    AmbaStorageProductVer[]="1000";               // 4 Bytes
AmbaUSBD_Msc_SetInfo (AmbaStorageVendorId, AmbaStorageProductId,
AmbaStorageProductVer);
```

**See Also:**

None



### 2.3.5 AmbaUSBD\_Msc\_SetProp

#### API Syntax:

UINT32 **AmbaUSBD\_Msc\_SetProp** (UINT8 drive, UINT8 removal, UINT8 wp, UINT8 type)

#### Function Description:

This function is used to configure the Mass Storage drive property.

#### Parameters:

| Type  | Parameter      | Description   |
|-------|----------------|---|
| UINT8 | <b>drive</b>   | [Input] Target disk drive:<br><b>SCM_SLOT_FL0</b> - NAND Flash driver (A)<br><b>SCM_SLOT_FL1</b> - NAND Flash driver (B)<br><b>SCM_SLOT_SD0</b> - SD/MMC slot 1 (C)<br><b>SCM_SLOT_SD1</b> - SD/MMC slot 2 (D)<br><b>SCM_SLOT_RD</b> - RAM disk (E)<br><b>SCM_SLOT_IDX</b> - Video Index partition (F)<br><b>SCM_SLOT_PRF</b> - Preference partition (G)<br><b>SCM_SLOT_CAL</b> - Calibration partition (H)<br><b>SCM_SLOT_USB</b> - USB Host (I)<br><b>SCM_SLOT_RF1</b> - ROM File system 1 (Y)<br><b>SCM_SLOT_RF2</b> - ROM File system 1 (Z) |
| UINT8 | <b>removal</b> | [Input] Indicates if it is a removal drive. It represents the <b>RMB</b> field in the Standard INQUIRY data in the SCSC primary command.  |
| UINT8 | <b>wp</b>      | [Input] Indicates if it is a write protect drive. If it is 1, then this drive is read only.   |
| UINT8 | <b>type</b>    | [Input] Device type of this drive. It represents the <b>Peripheral Device Type</b> field in the Standard INQUIRY data in the SCSC primary commands. The workshop already has some definitions for it:<br><b>MSC_MEDIA_FAT_DISK</b> - A FAT/FAT32/EXFAT disk   |



| Type | Parameter | Description  |
|------|-----------|--|
|      |           | drive.<br><b>MSC_MEDIA_CDROM</b> - A CD-ROM disk drive.<br><b>MSC_MEDIA_OPTICAL_DISK</b> - A Optical disk drive other than the CD-ROM. |

Table 2-74. Parameters for AmbaUSB API **AmbaUSBD\_Msc\_SetProp()**.

**Return:**

| Return     | Description |
|------------|-------------|
| 0          | Success     |
| 0xFFFFFFFF | Failure     |

Table 2-75. Returns for AmbaUSB API **AmbaUSBD\_Msc\_SetProp()**.

**Example:**

```
AmbaUSBD_Msc_SetProp (SCM_SLOT_SD, 0x80, 0, MSC_MEDIA_FAT_DISK);
```

**See Also:**

None



### 2.3.6 AmbaUSBD\_Msc\_Mount

#### API Syntax:

```
UINT32 AmbaUSBD_Msc_Mount (UINT8 drive)
```

#### Function Description:

This function is used to mount the mass storage device. All mass storage configurations should be completed before calling this API. Note that the mounted drive will not be removed from file system and might cause data inconsistency. To avoid data inconsistency, the application should call corresponding FIO APIs to unmount it from the file system.

#### Parameters:

| Type  | Parameter | Description   |
|-------|-----------|---|
| UINT8 | drive     | [Input] Target disk drive:<br><b>SCM_SLOT_FL0</b> - NAND Flash driver (A)<br><b>SCM_SLOT_FL1</b> - NAND Flash driver (B)<br><b>SCM_SLOT_SD0</b> - SD/MMC slot 1 (C)<br><b>SCM_SLOT_SD1</b> - SD/MMC slot 2 (D)<br><b>SCM_SLOT_RD</b> - RAM disk (E)<br><b>SCM_SLOT_IDX</b> - Video Index partition (F)<br><b>SCM_SLOT_PRF</b> - Preference partition (G)<br><b>SCM_SLOT_CAL</b> - Calibration partition (H)<br><b>SCM_SLOT_USB</b> - USB Host (I)<br><b>SCM_SLOT_RF1</b> - ROM File system 1 (Y)<br><b>SCM_SLOT_RF2</b> - ROM File system 1 (Z) |

Table 2-76. Parameters for AmbaUSB API **AmbaUSBD\_Msc\_Mount()**.

#### Return:

| Return | Description |
|--------|-------------|
| 0      | Success     |
| -1     | Failure     |

Table 2-77. Returns for AmbaUSB API **AmbaUSBD\_Msc\_Mount()**.



**Example:**

```
/* Register File system Operation. */
AmBaUSBD_Msc_Init(&MscFsOp);

/*Mount SCM_SLOT_SD*/
AmBaUSBD_Msc_SetInfo(AmBaStorageVendorId, AmBaStorageProductId,
AmBaStorageProductVer);
AmBaUSBD_Msc_SetProp(SCM_SLOT_SD, 0x80, 0, MSC_MEDIA_FAT_DISK);
AmBaUSBD_Msc_Mount (SCM_SLOT_SD);
```

**See Also:**

None



### 2.3.7 AmbaUSBD\_Msc\_UnMount

#### API Syntax:

```
UINT32 AmbaUSBD_Msc_UnMount (UINT8 drive)
```

#### Function Description:

This function is used to unmount a storage device from the Mass Storage class. Note that the unmounted drive will not be added to file system. To avoid this, the application should call corresponding FIO APIs to mount it to the file system.

#### Parameters:

| Type  | Parameter | Description   |
|-------|-----------|---|
| UINT8 | drive     | [Input] Target disk drive:<br><b>SCM_SLOT_FL0</b> - NAND Flash driver (A)<br><b>SCM_SLOT_FL1</b> - NAND Flash driver (B)<br><b>SCM_SLOT_SD0</b> - SD/MMC slot 1 (C)<br><b>SCM_SLOT_SD1</b> - SD/MMC slot 2 (D)<br><b>SCM_SLOT_RD</b> - RAM disk (E)<br><b>SCM_SLOT_IDX</b> - Video Index partition (F)<br><b>SCM_SLOT_PRF</b> - Preference partition (G)<br><b>SCM_SLOT_CAL</b> - Calibration partition (H)<br><b>SCM_SLOT_USB</b> - USB Host (I)<br><b>SCM_SLOT_RF1</b> - ROM File system 1 (Y)<br><b>SCM_SLOT_RF2</b> - ROM File system 1 (Z) |

Table 2-78. Parameters for AmbaUSB API **AmbaUSBD\_Msc\_UnMount()**.

#### Return:

| Return     | Description |
|------------|-------------|
| 0          | Success     |
| 0xFFFFFFFF | Failure     |

Table 2-79. Returns for AmbaUSB API **AmbaUSBD\_Msc\_UnMount()**.



**Example:**

```
AmbaUSBD_Msc_UnMount (SCM_SLOT_SD);
```

**See Also:**

None

For PROTRULY Confidential Only



### 2.3.8 AmbaUSBD\_Msc\_SetReadCacheMissTimeout

#### API Syntax:

```
UINT32 AmbaUSBD_Msc_SetReadCacheMissTimeout (UINT32 timeout)
```

#### Function Description:

This function is used to specify a timeout value for MSC cache read.

#### Parameters:

| Type   | Parameter | Description           |
|--------|-----------|-----------------------|
| UINT32 | timeout   | [Input] Timeout value |

Table 2-80. Parameters for AmbaUSB API *AmbaUSBD\_Msc\_SetReadCacheMissTimeout()*.

#### Return:

| Return     | Description |
|------------|-------------|
| 0          | Success     |
| 0xFFFFFFFF | Failure     |

Table 2-81. Returns for AmbaUSB API *AmbaUSBD\_Msc\_SetReadCacheMissTimeout()*.

#### Example:

```
// set the timeout value to 500 ms.  
AmbaUSBD_Msc_SetReadCacheMissTimeout (500);
```

#### See Also:

None



### 2.3.9 AmbaUSBD\_Msc\_GetReadCacheMissTimeout

#### API Syntax:

```
UINT32 AmbaUSBD_Msc_GetReadCacheMissTimeout (void)
```

#### Function Description:

This function is used to get the timeout value for MSC cache read.

#### Parameters:

None

#### Return:

| Return | Description    |
|--------|----------------|
| UINT32 | Timeout value. |

Table 2-82. Returns for AmbaUSB API **AmbaUSBD\_Msc\_GetReadCacheMissTimeout()**.

#### Example:

```
AmbaUSBD_Msc_GetReadCacheMissTimeout (SCM_SLOT_SD);
```

#### See Also:

None



### 2.3.10 AmbaUSBD\_Msc\_SetReadCacheTaskInfo

#### API Syntax:

```
UINT32 AmbaUSBD_Msc_SetReadCacheTaskInfo (UDC_TASKINFO_s *info)
```

#### Function Description:

This function is used to set the information for creating MSC read cache task. MSC read cache task will prefetch the data for MSC continuous read so it can improve MSC read performance.

It should be called before **AmbaUSBD\_System\_ClassHook()**.

#### Parameters:

| Type             | Parameter   | Description   |
|------------------|-------------|---|
| UDC_TASKINFO_s * | <b>info</b> | [Input] Information for creating MSC read cache task. Please refer to Section 2.1.1.5 for more details. |

Table 2-83. Parameters for AmbaUSB API **AmbaUSBD\_Msc\_SetReadCacheTaskInfo()**.

#### Return:

| Return     | Description |
|------------|-------------|
| 0          | Success     |
| 0xFFFFFFFF | Failure     |

Table 2-84. Returns for AmbaUSB API **AmbaUSBD\_Msc\_SetReadCacheTaskInfo()**.

#### Example:

```
// set the stack size to 10KB, priority to 70, and AffinityMask to CORE
0 for the MSC read cache task.

UDC_TASKINFO_s ti;
ti.StackSize = 1024*10;
ti.Priority = 70;
ti.AffinityMask = 0x01;
AmbaUSBD_Msc_SetReadCacheTaskInfo (&ti);
```



See Also:

None

For PROTRULY Confidential Only



### 2.3.11 AmbaUSBD\_Msc\_GetReadCacheTaskInfo

#### API Syntax:

```
UINT32 AmbaUSBD_Msc_GetReadCacheTaskInfo (UDC_TASKINFO_s *info)
```

#### Function Description:

This function is used to get the information for creating MSC read cache task.

#### Parameters:

| Type             | Parameter | Description  |
|------------------|-----------|--|
| UDC_TASKINFO_s * | info      | [Output] Information for creating MSC read cache task. Please refer to Section 2.1.1.5 for more details. |

Table 2-85. Parameters for AmbaUSB API **AmbaUSBD\_Msc\_GetReadCacheTaskInfo()**.

#### Return:

| Return     | Description |
|------------|-------------|
| 0          | Success     |
| 0xFFFFFFFF | Failure     |

Table 2-86. Returns for AmbaUSB API **AmbaUSBD\_Msc\_GetReadCacheTaskInfo()**.

#### Example:

```
UDC_TASKINFO_s ti;  
AmbaUSBD_Msc_GetReadCacheTaskInfo (&ti);
```

#### See Also:

None



### 2.3.12 AmbaUSBD\_Msc\_SetCbwCallback

#### API Syntax:

```
UINT32 AmbaUSBD_Msc_SetCbwCallback (UDC_MSC_CBK_PROCESSCBW cb)
```

#### Function Description:

This command is used for App to control the CBWCB field of CBW in the USB mass storage class, where CBW is a command block wrapper sent from the host to the device's Bulk Out endpoint, and CBWCB is the command block data for the device to execute. The application layer calls **AmbaUSB\_Class\_Msc\_SetCbwCallback()** to register a callback function to do the vendor specific processing of the command block data. Please refer to the USB mass storage class specification for more details.

#### Parameters:

| Type                   | Parameter       | Description   |
|------------------------|-----------------|---|
| UDC_MSC_CBK_PROCESSCBW | pfnCallBackFunc | [Input] Callback function is called when MSC device receives a CBW. Please refer to Section 2.3.2.4 for definition. |

Table 2-87. Parameters for AmbaUSB API **AmbaUSBD\_Msc\_SetCbwCallback()**.

#### Return:

| Return     | Description |
|------------|-------------|
| 0          | Success     |
| 0xFFFFFFFF | Failure     |

Table 2-88. Returns for AmbaUSB API **AmbaUSBD\_Msc\_SetCbwCallback()**.

#### Example:

```
// The following example shows how to send our own page code data in  
// MODE_SENSE(6) commands.  
  
// It monitors page code in MODE_SENSE(6) and commands, if code is 0x30,  
// then send out
```



```
// our own page code data.

// It also monitors MODE_SELECT(10) command and then receives data sent from
HOST.

#define PAGE_CODE_AMBA_SPECIFIC 0x30
#define PAGE_CODE_AMBA_SPECIFIC_SIZE 31

static UINT8
mode_sense_page_code_amba_specific_data[PAGE_CODE_AMBA_SPECIFIC_SIZE] = {
    'A', 'M', 'B', 'A', 'R', 'E', 'L', 'L', 'A', ' ', 'M', 'S', 'C', ' ', 'S',
    'A', 'M', 'P', 'L', 'E', 0
};

UINT32 udc_msc_cbwcb_monitor(UINT8 *cbwcb, UINT32 cbwcb_length, UINT8
*status)
{
    switch (cbwcb[0]) {
        case 0x1A: // MODE SENSE(6)
        {
            UINT8 page_code = cbwcb[2] & 0x3F;
            AmbaPrint("SCSI command: MODE SENSE(6)");
            AmbaPrint("    page control: 0x%X", (cbwcb[2] >> 6));
            AmbaPrint("    page code : 0x%X", page_code);
            AmbaPrint("    subpage code: 0x%X", (cbwcb[3]) & 0x0FF);
            AmbaPrint("    alloc length: 0x%X", (cbwcb[4]) & 0x0FF);
            if (page_code == PAGE_CODE_AMBA_SPECIFIC) {
                *status = UDC_MSC_CSW_STS_GOOD;
                // send bulk-in data to Host
            }
        }
    }
}

AmbaUSBD_Msc_SendBulkInData(mode_sense_page_code_amba_specific_data,
    PAGE_CODE_AMBA_SPECIFIC_SIZE);

return UDC_MSC_CBW_PROCESS_DONE;
}
```



```
case 0x55: // MODE SELECT(10)
{
    UINT16 length = cbwcb[7] << 8;
    static UINT8 data[255];
    UINT32 byte_read = 0;
    UINT32 nRet = 0;

    length += cbwcb[8];

    AmbaPrint("SCSI command: MODE SELECT(10)");
    AmbaPrint("    length = 0x%X", length);
    if (length <= 255) {
        // receive bulk-out data from Host
        nRet = AmbaUSBD_Msc_RecvBulkOutData(data, length, &byte_read);
        if (byte_read == length) {
            AmbaPrint("        string = %s", &data[0]);
        } else {
            AmbaPrint("%s(): read %d bytes but need %d bytes.", __FUNCTION__,
                      byte_read, length);
        }
        // tell MSC class driver to ignore this command
        *status = UDC_MSC_CSW_STS_GOOD;
        return UDC_MSC_CBW_PROCESS_DONE;
    } else {
        AmbaPrint("%s(): length %d > 255.", __FUNCTION__, byte_read, length);
    }
    break;
}

case 0x00: // TEST UNIT READY
    // skip it.
    break;

default:
    AmbaPrint("Unknown SCSI command: 0x%X, len = %d", cbwcb[0],
              cbwcb_length);
    break;
```



```
}
```

```
*status = UDC_MSC_CSW_STS_GOOD;
```

```
return UDC_MSC_CBW_PROCESS_CONTINUE;
```

```
}
```

**See Also:**

None

For PROTRULY Confidential Only



### 2.3.13 AmbaUSBD\_Msc\_SendBulkInData

#### API Syntax:

```
UINT32 AmbaUSBD_Msc_SendBulkInData (UINT8 *data, UINT32 length)
```

#### Function Description:

This command is for the application layer to set and send the Bulk-In data in the callback function of the USB mass storage class. Please refer to [AmbaUSBD\\_Msc\\_SetCbwCallback\(\)](#) for the callback function usage.

#### Parameters:

| Type   | Parameter     | Description             |
|--------|---------------|-------------------------|
| UINT8  | <b>data</b>   | Bulk-in data to be sent |
| UINT32 | <b>length</b> | Data size to be sent    |

Table 2-89. Parameters for AmbaUSB API [AmbaUSBD\\_Msc\\_SendBulkInData\(\)](#).

#### Return:

| Return     | Description  |
|------------|--|
| 0          | Success  |
| 0x12       | The data length is larger than Endpoint buffer size. |
| 0x14       | Endpoint is not available.                           |
| 0xFFFFFFFF | Fail   |

Table 2-90. Returns for AmbaUSB API [AmbaUSBD\\_Msc\\_SendBulkInData\(\)](#).

#### Example:

None

#### See Also:

[AmbaUSBD\\_Msc\\_SetCbwCallback\(\)](#)



### 2.3.14 AmbaUSBD\_Msc\_RecvBulkOutData

#### API Syntax:

```
UINT32 AmbaUSBD_Msc_RecvBulkOutData (UINT8 *data, UINT32 length, UINT32  
*byte_read)
```

#### Function Description:

This command is for the application layer to receive the Bulk-Out data in the callback function of the USB mass storage class. Please refer to [AmbaUSBD\\_Msc\\_SetCbwCallback\(\)](#) for the callback function usage.

#### Parameters:

| Type     | Parameter        | Description                             |
|----------|------------------|---|
| UINT8    | <b>data</b>      | Buffer for receiving data from Bulk-Out |
| UINT32   | <b>length</b>    | The size of buffer                      |
| UINT32 * | <b>byte_read</b> | Actual received data size               |

Table 2-91. Parameters for AmbaUSB API **AmbaUSBD\_Msc\_RecvBulkOutData()**.

#### Return:

| Return     | Description  |
|------------|--|
| 0          | Success  |
| 0x12       | The data length is larger than Endpoint buffer size. |
| 0x14       | Endpoint is not available.                           |
| 0xFFFFFFFF | Fail   |

Table 2-92. Returns for AmbaUSB API **AmbaUSBD\_Msc\_RecvBulkOutData()**.

#### Example:

None

#### See Also:

[AmbaUSBD\\_Msc\\_SetCbwCallback\(\)](#)



### 2.3.15 AmbaUSBD\_Msc\_GetReadCacheTaskInfo

#### API Syntax:

```
UINT32 AmbaUSBD_Msc_GetReadCacheTaskInfo (UDC_TASKINFO_s *info)
```

#### Function Description:

This function is used to get the information for creating MSC read cache task.

#### Parameters:

| Type             | Parameter | Description  |
|------------------|-----------|--|
| UDC_TASKINFO_s * | info      | [Output] Information for creating MSC read cache task. Please refer to Section 2.1.1.5 for more details. |

Table 2-93. Parameters for AmbaUSB API **AmbaUSBD\_Msc\_GetReadCacheTaskInfo()**.

#### Return:

| Return     | Description |
|------------|-------------|
| 0          | Success     |
| 0xFFFFFFFF | Failure     |

Table 2-94. Returns for AmbaUSB API **AmbaUSBD\_Msc\_GetReadCacheTaskInfo()**.

#### Example:

```
UDC_TASKINFO_s ti;  
AmbaUSBD_Msc_GetReadCacheTaskInfo (&ti);
```

#### See Also:

None



### 2.3.16 AmbaUSBD\_Msc\_GetReadCacheTaskInfo

#### API Syntax:

```
UINT32 AmbaUSBD_Msc_GetReadCacheTaskInfo (UDC_TASKINFO_s *info)
```

#### Function Description:

This function is used to get the information for creating MSC read cache task.

#### Parameters:

| Type             | Parameter | Description  |
|------------------|-----------|--|
| UDC_TASKINFO_s * | info      | [Output] Information for creating MSC read cache task. Please refer to Section 2.1.1.5 for more details. |

Table 2-95. Parameters for AmbaUSB API **AmbaUSBD\_Msc\_GetReadCacheTaskInfo()**.

#### Return:

| Return     | Description |
|------------|-------------|
| 0          | Success     |
| 0xFFFFFFFF | Failure     |

Table 2-96. Returns for AmbaUSB API **AmbaUSBD\_Msc\_GetReadCacheTaskInfo()**.

#### Example:

```
UDC_TASKINFO_s ti;  
AmbaUSBD_Msc_GetReadCacheTaskInfo (&ti);
```

#### See Also:

None



### 2.3.17 AmbaUSBD\_Msc\_GetReadCacheTaskInfo

#### API Syntax:

```
UINT32 AmbaUSBD_Msc_GetReadCacheTaskInfo (UDC_TASKINFO_s *info)
```

#### Function Description:

This function is used to get the information for creating MSC read cache task.

#### Parameters:

| Type             | Parameter | Description  |
|------------------|-----------|--|
| UDC_TASKINFO_s * | info      | [Output] Information for creating MSC read cache task. Please refer to Section 2.1.1.5 for more details. |

Table 2-97. Parameters for AmbaUSB API **AmbaUSBD\_Msc\_GetReadCacheTaskInfo()**.

#### Return:

| Return     | Description |
|------------|-------------|
| 0          | Success     |
| 0xFFFFFFFF | Failure     |

Table 2-98. Returns for AmbaUSB API **AmbaUSBD\_Msc\_GetReadCacheTaskInfo()**.

#### Example:

```
UDC_TASKINFO_s ti;  
AmbaUSBD_Msc_GetReadCacheTaskInfo (&ti);
```

#### See Also:

None



## 2.4 MTP/PTP Class APIs

This section describes APIs for MTP/PTP Class. MTP is the super set of PTP, so MTP is used in this document to avoid confusion.

For PROTRULY Confidential Only



## 2.4.1 Data Structures

### 2.4.1.1 MTP\_EVENT\_s

| Type   | Parameter                       | Description   |
|--------|---------------------------------|---|
| UINT32 | <b>MTP_EVENT_CODE</b>           | [Input] MTP Event Code.   |
| UINT32 | <b>MTP_EVENT_SESSION_ID</b>     | [Input] MTP Session ID. It is used by MTP class driver so always set it to 0.     |
| UINT32 | <b>MTP_EVENT_TRANSACTION_ID</b> | [Input] MTP Transaction ID. It is used by MTP class driver so always set it to 0. |
| UINT32 | <b>MTP_EVENT_PARAMETER_1</b>    | [Input] MTP parameter 1   |
| UINT32 | <b>MTP_EVENT_PARAMETER_2</b>    | [Input] MTP parameter 2   |
| UINT32 | <b>MTP_EVENT_PARAMETER_3</b>    | [Input] MTP parameter 3   |

Table 2-99. Definition of **MTP\_EVENT\_s**.

### 2.4.1.2 MTP\_DEVICE\_INFO\_s

| Type   | Parameter             | Description  |
|--------|-----------------------|--|
| UINT32 | <b>StorageId</b>      | Storage ID. Default is 0x10001.  |
| UINT32 | <b>StorageType</b>    | MTP Storage Type (please see the MTP specification):<br><b>MTP_STC_UNDEFINED</b><br><b>MTP_STC_FIXED_ROM</b><br><b>MTP_STC_REMOVABLE_ROM</b><br><b>MTP_STC_FIXED_RAM</b><br><b>MTP_STC_REMOVABLE_RAM</b> |
| UINT32 | <b>FileSystemType</b> | MTP File System Type (please see the MTP specification):<br><b>MTP_FSTC_UNDEFINED</b><br><b>MTP_FSTC_GENERIC_FLAT</b><br><b>MTP_FSTC_GENERIC_HIERARCHICAL</b>  |



| Type             | Parameter                         | Description  |
|------------------|-----------------------------------|--|
|                  |                                   | <b>MTP_FSTC_DCF</b>  |
| UINT32           | <b>AccessCapability</b>           | MTP AccessCapability   |
| UINT32           | <b>MaxCapabilityLow</b>           | Low bytes of maximum storage capability                                  |
| UINT32           | <b>MaxCapabilityHigh</b>          | High bytes of maximum storage capability                                 |
| UINT32           | <b>FreeSpaceLow</b>               | Low bytes of free space  |
| UINT32           | <b>FreeSpaceHigh</b>              | High bytes of free space   |
| UINT32           | <b>FreeSpaceImage</b>             | Free space in objects  |
| UINT8*           | <b>DeviceInfoVendorName</b>       | Manufacturer Name string   |
| UINT8*           | <b>DeviceInfoProductName</b>      | Product Name string  |
| UINT8*           | <b>DeviceInfoSerialNo</b>         | Serial Number string   |
| UINT8*           | <b>DeviceInfoVersion</b>          | Version string   |
| UINT8*           | <b>VolumeDescription</b>          | Volume Description string  |
| UINT8*           | <b>VolumeLabel</b>                | Volume Label string  |
| UINT16*          | <b>DeviceSupportProp</b>          | Supported MTP Device Properties  |
| UINT16*          | <b>DeviceSupportCaptureFormat</b> | Supported MTP Capture Formats  |
| UINT16*          | <b>DeviceSupportImgFormat</b>     | Supported MTP Image Formats  |
| UINT16*          | <b>ObjectSupportProp</b>          | Supported MTP Object Properties  |
| UINT16*          | <b>OperationSupportList</b>       | Supported MTP Operations   |
| Function Pointer | <b>PropDescGet</b>                | It is called when the MTP <b>GetDevicePropDesc</b> command is received.  |
| Function Pointer | <b>PropValueGet</b>               | It is called when the MTP <b>GetDevicePropValue</b> command is received. |
| Function Pointer | <b>PropValueSet</b>               | It is called when the MTP <b>SetDevicePropDesc</b> command is received.  |
| Function Pointer | <b>StorageFormat</b>              | It is called when the MTP <b>FormatStore</b> command is received.        |
| Function Pointer | <b>ObjectDelete</b>               | It is called when the MTP <b>DeleteObject</b> command is received.       |
| Function Pointer | <b>DeviceReset</b>                | It is called when the MTP <b>ResetDevice</b> command is received.        |
| Function         | <b>StorageInfoGet</b>             | It is called when the MTP <b>GetStorageInfo</b>                          |



| Type             | Parameter                   | Description  |
|------------------|-----------------------------|--|
| Pointer          |                             | command is received.   |
| Function Pointer | <b>ObjectNumberGet</b>      | It is called when the MTP <b>GetNumObjects</b> command is received.      |
| Function Pointer | <b>ObjectHandlesGet</b>     | It is called when the MTP <b>GetObjectHandles</b> command is received.   |
| Function Pointer | <b>ObjectInfoGet</b>        | It is called when the MTP <b>GetObjectInfo</b> command is received.      |
| Function Pointer | <b>ObjectDataGet</b>        | It is called when the MTP <b>GetObject</b> command is received.          |
| Function Pointer | <b>ObjectInfoSend</b>       | It is called when the MTP <b>SendObjectInfo</b> command is received.     |
| Function Pointer | <b>ObjectDataSend</b>       | It is called when the MTP <b>SendObject</b> command is received.         |
| Function Pointer | <b>ObjectPropDescGet</b>    | It is called when the MTP <b>GetObjectPropDesc</b> command is received.  |
| Function Pointer | <b>ObjectPropValueGet</b>   | It is called when the MTP <b>GetObjectPropValue</b> command is received. |
| Function Pointer | <b>ObjectPropValueSet</b>   | It is called when the MTP <b>SetObjectPropValue</b> command is received. |
| Function Pointer | <b>ObjectPropListGet</b>    | It is called when the MTP <b>GetObjectPropList</b> command is received.  |
| Function Pointer | <b>ObjectReferenceGet</b>   | It is called when the MTP <b>GetObjectReference</b> command is received. |
| Function Pointer | <b>ObjectReferenceSet</b>   | It is called when the MTP <b>SetObjectReference</b> command is received. |
| Function Pointer | <b>ObjectCustomCommand</b>  | Callback function for "iTuner" commands                                  |
| Function Pointer | <b>ObjectCustomDataGet</b>  | Callback function for "iTuner" commands                                  |
| Function Pointer | <b>ObjectCustomDataSend</b> | Callback function for "iTuner" commands                                  |
| Function Pointer | <b>ObjectClearAll</b>       | It is called when the MTP driver wants to cleanup.                       |



| Type             | Parameter               | Description   |
|------------------|-------------------------|---|
| Function Pointer | <b>VendorCmdProcess</b> | It is called when a MTP vendor command is received. |
| UINT8*           | <b>RootPath</b>         | Root directory of this MTP device                   |

Table 2-100. Definition of **MTP\_DEVICE\_INFO\_s**.

#### 2.4.1.3 USB\_MTP\_VENDOR\_OP\_s

| Type               | Parameter           | Description   |
|--------------------|---------------------|---|
| UINT32             | <b>operation</b>    | <b>MTP_VENDOR_OP_SEND</b> - Send data to Host<br><b>MTP_VENDOR_OP_RECEIVE</b> - Receive data from Host<br><b>MTP_VENDOR_OP_RESPONSE</b> - Send MTP response to Host |
| UINT8 *            | <b>BufferPtr</b>    | The buffer to send/receive data from the MTP Host   |
| UINT32             | <b>length</b>       | The buffer size   |
| UINT32             | <b>ActualLength</b> | The actual sent/received size   |
| USB_MTP_RESPONSE_s | <b>response</b>     | MTP response information. It is only valid when the operation is <b>MTP_VENDOR_OP_RESPONSE</b> . Please refer to Section 2.4.1.4 for further information.           |

Table 2-101. Definition of **USB\_MTP\_VENDOR\_OP\_s**.

#### 2.4.1.4 USB\_MTP\_RESPONSE\_s

| Type   | Parameter           | Description                         |
|--------|---------------------|-------------------------------------|
| UINT32 | <b>ResponseCode</b> | MTP Response Code                   |
| UINT32 | <b>parmCnt</b>      | Parameter count in the MTP response |
| UINT32 | <b>parm[3]</b>      | 3 Parameters in the MTP response    |

Table 2-102. Definition of **USB\_MTP\_RESPONSE\_s**.



#### 2.4.1.5 MTP\_OBJECT\_s

| Type    | Parameter                         | Description  |
|---------|-----------------------------------|--|
| UINT32  | <b>ObjectStorageId</b>            | The storage ID for this object                           |
| UINT32  | <b>ObjectFormat</b>               | The object format code                                   |
| UINT32  | <b>ObjectProtectionStatus</b>     | The protection status                                    |
| UINT32  | <b>ObjectCompressedSize</b>       | The object size  |
| UINT32  | <b>ObjectThumbFormat</b>          | The format code of thumbnail for this object             |
| UINT32  | <b>ObjectThumbCompressedSize</b>  | The size of thumbnail for this object                    |
| Type    | Parameter                         | Description  |
| UINT32  | <b>ObjectThumbPixWidth</b>        | The width of thumbnail for this object                   |
| UINT32  | <b>ObjectThumbPixHeight</b>       | The height of thumbnail for this object                  |
| UINT32  | <b>ObjectImagePixWidth</b>        | The width of this object                                 |
| UINT32  | <b>ObjectImagePixHeight</b>       | The height of this object                                |
| UINT32  | <b>ObjectImageBitDepth</b>        | The image depth of this object                           |
| UINT32  | <b>ObjectParentObject</b>         | The parent object ID of this object                      |
| UINT32  | <b>ObjectAssociationType</b>      | The association type                                     |
| UINT32  | <b>ObjectAssociationDesc</b>      | The association descriptor                               |
| UINT32  | <b>ObjectSequenceNumber</b>       | The object sequence number                               |
| UINT8   | <b>ObjectFilename[256]</b>        | The file name of this object                             |
| UINT8   | <b>ObjectCaptureDate[64]</b>      | The capture date of this object                          |
| UINT8   | <b>ObjectModificationDate[64]</b> | The modification date of this object                     |
| UINT8   | <b>ObjectKeywords[256]</b>        | The keyword of this object                               |
| UINT32  | <b>ObjectState</b>                | The state of this object                                 |
| UINT32  | <b>ObjectOffset</b>               | The current offset of this object for reading or writing |
| UINT32  | <b>ObjectTransferStatus</b>       | The transfer status of this object                       |
| UINT32  | <b>ObjectHandleId</b>             | The handle ID of this object                             |
| UINT32  | <b>ObjectLength</b>               | Used internally, do not access it                        |
| UINT8 * | <b>ObjectBuffer</b>               | Used internally, do not access it                        |

Table 2-103. Definition of **MTP\_OBJECT\_s**.



#### 2.4.1.6 USB\_MTP\_CMD\_s

| Type   | Parameter            | Description                |
|--------|----------------------|----------------------------|
| UINT32 | <b>OpCode</b>        | [Input] MTP Operation Code |
| UINT32 | <b>TransactionID</b> | [Input] Transaction ID     |
| UINT32 | <b>parmCnt</b>       | [Input] Parameter count    |
| UINT32 | <b>parm[5]</b>       | [Input] Parameters         |

Table 2-104. Definition of **USB\_MTP\_CMD\_s**.



## 2.4.2 Callback functions

### 2.4.2.1 PropDescGet

#### API Syntax:

```
UINT32 PropDescGet (UINT32 DeviceProperty, UINT8 **DevicePropDataset, UINT32  
*DevicePropDsetLength)
```

#### Function Description:

This function is called when the MTP **GetDevicePropDesc** command is received.

#### Parameters:

| Type     | Parameter                   | Description  |
|----------|-----------------------------|--|
| UINT32   | <b>DeviceProperty</b>       | [Input] MTP Device Property Code                                   |
| UINT8 ** | <b>DevicePropDataset</b>    | [Output] The buffer containing the Device Property Descriptor      |
| UINT32 * | <b>DevicePropDsetLength</b> | [Output] The size of the Device Property Descriptor in desc buffer |

Table 2-105. Parameters for AmbaUSB API **PropDescGet()**.

#### Return:

| Return  | Description                      |
|---------|----------------------------------|
| 0       | Success                          |
| < 0x100 | Failure with Internal Error code |
| > 0x100 | Failure with MTP response code   |

Table 2-106. Returns for AmbaUSB API **PropDescGet()**.

#### Example:

None

#### See Also:

None



#### 2.4.2.2 PropValueGet

##### API Syntax:

```
UINT32 PropValueGet (UINT32 DeviceProperty, UINT8 **DevicePorpValue, UINT32  
*DevicePropValueLength)
```

##### Function Description:

This function is called when the MTP **GetDevicePropValue** command is received.

##### Parameters:

| Type     | Parameter             | Description   |
|----------|-----------------------|---|
| UINT32   | DeviceProperty        | [Input] MTP Device Property Code                                  |
| UINT8 ** | DevicePorpValue       | [Output] The buffer containing the Device Property Value          |
| UINT32 * | DevicePropValueLength | [Output] The size of the Device Property Value in the desc buffer |

Table 2-107. Parameters for AmbaUSB API **PropValueGet()**.

##### Return:

| Return  | Description                      |
|---------|----------------------------------|
| 0       | Success                          |
| < 0x100 | Failure with Internal Error code |
| > 0x100 | Fail with MTP response code      |

Table 2-108. Returns for AmbaUSB API **PropValueGet()**.

##### Example:

None

##### See Also:

None



### 2.4.2.3 PropValueSet

#### API Syntax:

```
UINT32 PropValueSet (UINT32 DeviceProperty, UINT8 *DevicePropValue, UINT32  
DevicePropValueLength);
```

#### Function Description:

This function is called when the MTP **SetDevicePropValue** command is received.

#### Parameters:

| Type    | Parameter                    | Description  |
|---------|------------------------------|--|
| UINT32  | <b>DeviceProperty</b>        | [Input] MTP Device Property Code                                 |
| UINT8 * | <b>DevicePropValue</b>       | [Input] The buffer containing the Device Property Value          |
| UINT32  | <b>DevicePropValueLength</b> | [Input] The size of the Device Property Value in the desc buffer |

Table 2-109. Parameters for AmbaUSB API **PropValueSet()**.

#### Return:

| Return  | Description                      |
|---------|----------------------------------|
| 0       | Success                          |
| < 0x100 | Failure with Internal Error code |
| > 0x100 | Failure with MTP response code   |

Table 2-110. Returns for AmbaUSB API **PropValueSet()**.

#### Example:

None

#### See Also:

None



#### 2.4.2.4 StorageFormat

##### API Syntax:

UINT32 **StorageFormat** (UINT32 StorageId)

##### Function Description:

This function is called when the MTP **FormatStore** command is received.

##### Parameters:

| Type   | Parameter        | Description           |
|--------|------------------|-----------------------|
| UINT32 | <b>StorageId</b> | [Input] MTP StorageID |

Table 2-111. Parameters for AmbaUSB API **StorageFormat()**.

##### Return:

| Return  | Description                      |
|---------|----------------------------------|
| 0       | Success                          |
| < 0x100 | Failure with Internal Error code |
| > 0x100 | Failure with MTP response code   |

Table 2-112. Returns for AmbaUSB API **StorageFormat()**.

##### Example:

None

##### See Also:

None



#### 2.4.2.5 ObjectDelete

##### API Syntax:

UINT32 **ObjectDelete** (UINT32 ObjectHandle)

##### Function Description:

This function is called when the MTP **DeleteObject** command is received.

##### Parameters:

| Type   | Parameter           | Description               |
|--------|---------------------|---------------------------|
| UINT32 | <b>ObjectHandle</b> | [Input] MTP Object Handle |

Table 2-113. Parameters for AmbaUSB API **ObjectDelete()**.

##### Return:

| Return  | Description                      |
|---------|----------------------------------|
| 0       | Success                          |
| < 0x100 | Failure with Internal Error code |
| > 0x100 | Failure with MTP response code   |

Table 2-114. Returns for AmbaUSB API **ObjectDelete()**.

##### Example:

None

##### See Also:

None



#### 2.4.2.6 DeviceReset

##### API Syntax:

```
UINT32 DeviceReset (void)
```

##### Function Description:

This function is called when the MTP **ResetDevice** command is received.

##### Parameters:

None

##### Return:

| Return  | Description                      |
|---------|----------------------------------|
| 0       | Success                          |
| < 0x100 | Failure with Internal Error code |
| > 0x100 | Failure with MTP response code   |

Table 2-115. Returns for AmbaUSB API **DeviceReset()**.

##### Example:

None

##### See Also:

None



#### 2.4.2.7 StorageInfoGet

##### API Syntax:

```
UINT32 StorageInfoGet (UINT32 StorageId, UINT32* MaxCapacityLow, UINT32*
MaxCapacityHigh, UINT32* FreeSpaceLow, UINT32* FreeSpaceHigh)
```

##### Function Description:

This function is called when MTP **GetDevicePropValue** command is received.

##### Parameters:

| Type     | Parameter              | Description   |
|----------|------------------------|---|
| UINT32   | <b>StorageId</b>       | [Input] MTP Storage ID  |
| UINT32 * | <b>MaxCapacityLow</b>  | [Output] The high bytes of maximum capacity of the Storage ID |
| UINT32 * | <b>MaxCapacityHigh</b> | [Output] The low bytes of maximum capacity of the Storage ID  |
| UINT32 * | <b>FreeSpaceLow</b>    | [Output] The high bytes of free space of the Storage ID       |
| UINT32 * | <b>FreeSpaceHigh</b>   | [Output] The low bytes of free space of the Storage ID        |

Table 2-116. Parameters for AmbaUSB API **StorageInfoGet()**.

##### Return:

| Return  | Description                      |
|---------|----------------------------------|
| 0       | Success                          |
| < 0x100 | Failure with Internal Error code |
| > 0x100 | Failure with MTP response code   |

Table 2-117. Returns for AmbaUSB API **StorageInfoGet()**.

##### Example:

None



**See Also:**

None

#### 2.4.2.8    [ObjectNumberGet](#)

**API Syntax:**

```
UINT32 ObjectNumberGet (UINT32 ObjectFormatCode, UINT32 ObjectAssociation,  
                       UINT32 *ObjectNumber)
```

**Function Description:**

This function is called when the MTP **GetNumObjects** command is received.

**Parameters:**

| Type     | Parameter                | Description                      |
|----------|--------------------------|----------------------------------|
| UINT32   | <b>ObjectFormatCode</b>  | [Input] MTP Object Format Code   |
| UINT32   | <b>ObjectAssociation</b> | [Input] The parent Object Handle |
| UINT32 * | <b>ObjectNumber</b>      | [Output] The number of objects   |

Table 2-118. Parameters for AmbaUSB API **ObjectNumberGet()**.

**Return:**

| Return  | Description                      |
|---------|----------------------------------|
| 0       | Success                          |
| < 0x100 | Failure with Internal Error code |
| > 0x100 | Failure with MTP response code   |

Table 2-119. Returns for AmbaUSB API **ObjectNumberGet()**.

**Example:**

None

**See Also:**

None



#### 2.4.2.9 ObjectHandlesGet

##### API Syntax:

```
UINT32 ObjectHandlesGet (UINT32 FormatCode, UINT32 Association, UINT32 *Handles,  
UINT32 MaxHandle, UINT32 *RequestNumber, USHORT *ResponseCode)
```

##### Function Description:

This function is called when the MTP **GetObjectHandles** command is received.

##### Parameters:

| Type     | Parameter            | Description   |
|----------|----------------------|---|
| UINT32   | <b>FormatCode</b>    | [Input] MTP Object Format Code  |
| UINT8 ** | <b>Association</b>   | [Input] The parent Object Handle  |
| UINT32 * | <b>Handles</b>       | [Input/Output] The buffer containing all Object Handles.  |
| UINT32   | <b>MaxHandle</b>     | [Input] The maximum Object Handle number in the buffer  |
| UINT32 * | <b>RequestNumber</b> | [Output] Request Object Handle number. If <b>RequestNumber</b> > <b>MaxHandle</b> , then the MTP driver will call it again with a larger buffer size that can contain all Object Handles. |
| USHORT * | <b>ResponseCode</b>  | [Output] MTP response code. It is used when the return value is not 0.  |

Table 2-120. Parameters for AmbaUSB API **ObjectHandlesGet()**.

##### Return:

| Return | Description                      |
|--------|----------------------------------|
| 0      | Success                          |
| -1     | Failure with Internal Error code |

Table 2-121. Returns for AmbaUSB API **ObjectHandlesGet()**.



**Example:**

None

**See Also:**

None

For PROTRULY Confidential Only



#### 2.4.2.10 ObjectInfoGet

##### API Syntax:

```
UINT32 ObjectInfoGet (UINT32 ObjectHandle, MTP_OBJECT_s **object)
```

##### Function Description:

This function is called when the MTP **GetObjectInfo** command is received.

##### Parameters:

| Type            | Parameter           | Description   |
|-----------------|---------------------|---|
| UINT32          | <b>ObjectHandle</b> | [Input] MTP Object Handle   |
| MTP_OBJECT_s ** | <b>object</b>       | [Output] The buffer containing the Object Information. Please refer to Section 2.4.1.5. |

Table 2-122. Parameters for AmbaUSB API **ObjectInfoGet()**.

##### Return:

| Return       | Description  |
|--------------|--|
| 0            | Success  |
| Respond Code | Please refer to Media Transfer Protocol revision 1.0 for definition. |

Table 2-123. Returns for AmbaUSB API **ObjectInfoGet()**.



**Example:**

```
static UINT32 MTP_ObjectInfoGet (UINT32 ObjectHandle, MTP_OBJECT_S
**object) {
    UINT        status;
    UINT32      HandleIndex;

    /* Check if ObjectHandle valid */
    status = MTP_ObjectHandlerCheck((UINT32)ObjectHandle, &HandleIndex);

    /* Get object information */

    MTP.UtilityGetObjectInfo(ObjectHandle, object);

    if (status == OK) {
        return(OK);
    } else {
        return(MTP_RC_INVALID_OBJECT_HANDLE);
    }
}
```

**See Also:**

None

#### 2.4.2.11 ObjectDataGet

**API Syntax:**

```
UINT32 ObjectDataGet (UINT32 ObjectHandle, UINT8 *ObjectBuffer, UINT32 ObjectOffset,
UINT32 ObjectLengthRequested, UINT32 *ObjectActualLength)
```

**Function Description:**

This function is called when the MTP **GetObject** or **GetPartialObject** command is received.

**Parameters:**

| Type    | Parameter           | Description                                    |
|---------|---------------------|--|
| UINT32  | <b>ObjectHandle</b> | [Input] MTP Object Handle                      |
| UINT8 * | <b>ObjectBuffer</b> | [Output] The buffer containing the object data |



|          |                              |  |
|----------|------------------------------|--|
| UINT32   | <b>ObjectOffset</b>          | [Input] The offset of the data to be sent          |
| UINT32   | <b>ObjectLengthRequested</b> | [Input] The requested data size                    |
| UINT32 * | <b>ObjectActualLength</b>    | [Output] The actual size of the data in the buffer |

Table 2-124. Parameters for AmbaUSB API **ObjectDataGet()**.

**Return:**

| Return  | Description                      |
|---------|----------------------------------|
| 0       | Success                          |
| < 0x100 | Failure with Internal Error code |
| > 0x100 | Failure with MTP response code   |

Table 2-125. Returns for AmbaUSB API **ObjectDataGet()**.

**Example:**

```
UINT32          status;
INT             StatusClose;
UINT32          HandleIndex;
UINT32          ReadLen;
char            FullFileName[ 64 ];

/* Check the object handle. It must be in the local array. */
status = MTP_ObjectHandlerCheck((UINT32)ObjectHandle, &HandleIndex);
if (status == OK) {
    /* We are either at the beginning of the transfer or continuing the
transfer.

    Check of the filex array handle exist already. */
    if (MtpClassInfo.FpArray == NULL) {
        /* File not yet opened for this object. Open the file. */
        MTP.UtilityGetFullFileName(ObjectHandle, FullFileName);
        MtpClassInfo.FpArray = AmbaFS_fopen((const char*)FullFileName,
"r");
        AmbaPrint("open %s", FullFileName);
        if(MtpClassInfo.FpArray == NULL) {
            AmbaPrint("Failed to open %s", FullFileName);
```



```
        return(MTP_RC_OBJECT_NOT_OPENED);

    }

}

/* Read from the file into the media buffer. */

ReadLen = AmbaFS_fread(ObjectBuffer,
                       1,
                       ObjectLengthRequested,
                       MtpClassInfo.FpArray);

*ObjectActualLength = ReadLen;

if (ReadLen != 0) {
    status = OK;
} else {
    status = MTP_RC_ACCESS_DENIED;
}

/* Check if we have read the entire file. */

if (AmbaFS_feof(MtpClassInfo.FpArray)) {
    /* This is the end of the transfer for the object. Close it. */
    StatusClose = AmbaFS_fclose(MtpClassInfo.FpArray);
    MtpClassInfo.FpArray = NULL;
    AmbaPrint("close file");
    if (StatusClose == 0 && status == OK) {
        return (OK);
    } else {
        StatusClose = MTP_RC_ACCESS_DENIED;
        /* If status is error. We return status. If StatusClose is error
we return StatusClose. */
        if(status != OK) {
            return(status);
        } else {
            return(StatusClose);
        }
    }
}
```



```
    return (OK);  
}  
  
return(MTP_RC_INVALID_OBJECT_HANDLE);
```

**See Also:**

None

For Confidential  
PROTRULY Only



#### 2.4.2.12 ObjectInfoSend

##### API Syntax:

```
UINT32 ObjectInfoSend (MTP_OBJECT_s *Object, UINT32 StorageId, UINT32  
ParentObjectHandle, UINT32 *ObjectHandle);
```

##### Function Description:

This function is called when the MTP **SendObjectInfo** command is received.

##### Parameters:

| Type           | Parameter                 | Description   |
|----------------|---------------------------|---|
| MTP_OBJECT_s * | <b>Object</b>             | [Input] MTP object information. Please refer to Section 2.4.1.5 for more details. |
| UINT32         | <b>StorageId</b>          | [Input] MTP Storage ID  |
| UINT32         | <b>ParentObjectHandle</b> | [Input] Parent Object Handle  |
| UINT32 *       | <b>ObjectHandle</b>       | [Output] Object Handle  |

Table 2-126. Parameters for AmbaUSB API **ObjectInfoSend()**.

##### Return:

| Return  | Description                      |
|---------|----------------------------------|
| 0       | Success                          |
| < 0x100 | Failure with Internal Error code |
| > 0x100 | Failure with MTP response code   |

Table 2-127. Returns for AmbaUSB API **ObjectInfoSend()**.

##### Example:

None

##### See Also:

None



#### 2.4.2.13 ObjectDataSend

##### API Syntax:

```
UINT32 ObjectDataSend (UINT32 ObjectHandle, UINT32 Phase, UINT8 *ObjectBuffer,  
UINT32 ObjectOffset, UINT32 ObjectLength);
```

##### Function Description:

This function is called when the MTP **SendObject** command is received.

##### Parameters:

| Type    | Parameter           | Description  |
|---------|---------------------|--|
| UINT32  | <b>ObjectHandle</b> | [Input] MTP Object Handle  |
| UINT32  | <b>Phase</b>        | [Input] Current transfer phase:<br>0 - On going. The Application should copy data<br>from the <b>ObjectBuffer</b> .<br>1 - Transfer complete. No more action required.<br>2 - Error during transfer. |
| UINT8 * | <b>ObjectBuffer</b> | [Input] Object data  |
| UINT32  | <b>ObjectOffset</b> | [Input] Object offset  |
| UINT32  | <b>ObjectLength</b> | [Input] Object length  |

Table 2-128. Parameters for AmbaUSB API **ObjectDataSend()**.

##### Return:

| Return  | Description                      |
|---------|----------------------------------|
| 0       | Success                          |
| < 0x100 | Failure with Internal Error code |
| > 0x100 | Failure with MTP response code   |

Table 2-129. Returns for AmbaUSB API **ObjectDataSend()**.

##### Example:

None



**See Also:**

None

#### 2.4.2.14 ObjectPropDescGet

**API Syntax:**

```
UINT32 ObjectPropDescGet (UINT32 ObjectProperty, UINT32 ObjectFormatCode, UINT8
    **ObjectPropDataset, UINT32 *ObjectPropDatasetLength);
```

**Function Description:**

This function is called when the MTP **GetObjectPropDesc** command is received.

**Parameters:**

| Type     | Parameter                      | Description   |
|----------|--------------------------------|---|
| UINT32   | <b>ObjectProperty</b>          | [Input] MTP Object Property                                       |
| UINT32   | <b>ObjectFormatCode</b>        | [Input] MTP Object Format Code                                    |
| UINT8 ** | <b>ObjectPropDataset</b>       | [Output] The buffer containing the Object Property Descriptor     |
| UINT32 * | <b>ObjectPropDatasetLength</b> | [Output] The size of the Object Property Descriptor in the buffer |

Table 2-130. Parameters for AmbaUSB API **ObjectPropDescGet()**.

**Return:**

| Return  | Description                      |
|---------|----------------------------------|
| 0       | Success                          |
| < 0x100 | Failure with Internal Error code |
| > 0x100 | Failure with MTP response code   |

Table 2-131. Returns for AmbaUSB API **ObjectPropDescGet()**.

**Example:**

None



**See Also:**

None

#### 2.4.2.15 ObjectPropValueGet

**API Syntax:**

```
UINT32 ObjectPropValueGet (UINT32 ObjectHandle, UINT32 ObjectProperty, UINT8  
**ObjectPropValue, UINT32 *ObjectPropValueLength);
```

**Function Description:**

This function is called when the MTP **GetObjectPropValue** command is received.

**Parameters:**

| Type     | Parameter                    | Description   |
|----------|------------------------------|---|
| UINT32   | <b>ObjectHandle</b>          | [Input] MTP Object Handle                                   |
| UINT32   | <b>ObjectProperty</b>        | [Input] MTP Object Property Code                            |
| UINT8 ** | <b>ObjectPropValue</b>       | [Input] The buffer containing the Object Property Value     |
| UINT32 * | <b>ObjectPropValueLength</b> | [Input] The size of the Object Property Value in the buffer |

Table 2-132. Parameters for AmbaUSB API **ObjectPropValueGet()**.

**Return:**

| Return  | Description                      |
|---------|----------------------------------|
| 0       | Success                          |
| < 0x100 | Failure with Internal Error code |
| > 0x100 | Failure with MTP response code   |

Table 2-133. Returns for AmbaUSB API **ObjectPropValueGet()**.

**Example:**

None

**See Also:**



None

#### 2.4.2.16 ObjectPropValueSet

##### API Syntax:

```
UINT32 ObjectPropValueSet (UINT32 ObjectHandle, UINT32 ObjectProperty, UINT8  
*ObjectPropValue, UINT32 ObjectPropValueLength)
```

##### Function Description:

This function is called when the MTP **GetObjectPropValue** command is received.

##### Parameters:

| Type    | Parameter                    | Description   |
|---------|------------------------------|---|
| UINT32  | <b>ObjectHandle</b>          | [Input] MTP Object Handle                                   |
| UINT32  | <b>ObjectProperty</b>        | [Input] MTP Object Property Code                            |
| UINT8 * | <b>ObjectPropValue</b>       | [Input] The buffer containing the Object Property Value     |
| UINT32  | <b>ObjectPropValueLength</b> | [Input] The size of the Object Property Value in the buffer |

Table 2-134. Parameters for AmbaUSB API **ObjectPropValueSet()**.

##### Return:

| Return  | Description                      |
|---------|----------------------------------|
| 0       | Success                          |
| < 0x100 | Failure with Internal Error code |
| > 0x100 | Failure with MTP response code   |

Table 2-135. Returns for AmbaUSB API **ObjectPropValueSet()**.

##### Example:

None

##### See Also:



None

#### 2.4.2.17 ObjectPropListGet

##### API Syntax:

```
UINT32 ObjectPropListGet (UINT32 *params, UINT8 **PropList, UINT32 *PropListLength)
```

##### Function Description:

This function is called when the MTP **GetObjectPropList** command is received.

##### Parameters:

| Type     | Parameter             | Description                               |
|----------|-----------------------|---|
| UINT32   | <b>params</b>         | [Input] MTP 5 parameters in command block |
| UINT8 ** | <b>PropList</b>       | [Output] MTP Object Property List         |
| UINT32 * | <b>PropListLength</b> | [Output] The length if PropList in bytes  |

Table 2-136. Parameters for AmbaUSB API **ObjectPropListGet()**.

##### Return:

| Return  | Description                      |
|---------|----------------------------------|
| 0       | Success                          |
| < 0x100 | Failure with Internal Error code |
| > 0x100 | Failure with MTP response code   |

Table 2-137. Returns for AmbaUSB API **ObjectPropListGet()**.

##### Example:

None

##### See Also:

None



#### 2.4.2.18 ObjectReferenceGet

##### API Syntax:

```
UINT32 ObjectReferenceGet (UINT32 ObjectHandle, UINT8 **ObjectReferenceArray,  
                          UINT32 *ObjectReferenceArrayLength)
```

##### Function Description:

This function is called when the MTP **GetObjectReference** command is received.

##### Parameters:

| Type     | Parameter                         | Description   |
|----------|-----------------------------------|---|
| UINT32   | <b>ObjectHandle</b>               | [Input] MTP Object Handle                                     |
| UINT8 ** | <b>ObjectReferenceArray</b>       | [Output] The buffer containing the Object Reference Array     |
| UINT32 * | <b>ObjectReferenceArrayLength</b> | [Output] The size of the Object Reference Array in the buffer |

Table 2-138. Parameters for AmbaUSB API **ObjectReferenceGet()**.

##### Return:

| Return  | Description                      |
|---------|----------------------------------|
| 0       | Success                          |
| < 0x100 | Failure with Internal Error code |
| > 0x100 | Failure with MTP response code   |

Table 2-139. Returns for AmbaUSB API **ObjectReferenceGet()**.

##### Example:

None

##### See Also:

None



#### 2.4.2.19 ObjectReferenceSet

##### API Syntax:

```
UINT32 ObjectReferenceSet (UINT32 ObjectHandle, UINT8 *ObjectReferenceArray,  
                          UINT32 ObjectReferenceArrayLength)
```

##### Function Description:

This function is called when the MTP **SetObjectReference** command is received.

##### Parameters:

| Type    | Parameter                  | Description  |
|---------|----------------------------|--|
| UINT32  | ObjectHandle               | [Input] MTP Object Handle                                    |
| UINT8 * | ObjectReferenceArray       | [Input] The buffer containing the Object Reference Array     |
| UINT32  | ObjectReferenceArrayLength | [Input] The size of the Object Reference Array in the buffer |

Table 2-140. Parameters for AmbaUSB API **ObjectReferenceSet()**.

##### Return:

| Return  | Description                      |
|---------|----------------------------------|
| 0       | Success                          |
| < 0x100 | Failure with Internal Error code |
| > 0x100 | Failure with MTP response code   |

Table 2-141. Returns for AmbaUSB API **ObjectReferenceSet()**.

##### Example:

None

##### See Also:

None



#### 2.4.2.20 ObjectClearAll

##### API Syntax:

```
void ObjectClearAll (void)
```

##### Function Description:

This function is called when the MTP class driver wants to clear all object information.

##### Parameters:

None

##### Return:

| Return | Description |
|--------|-------------|
| 0      | Success     |
| 0xFF   | Failure     |

Table 2-142. Returns for AmbaUSB API *ObjectClearAll()*.

##### Example:

None

##### See Also:

None



#### 2.4.2.21 ObjectCustomCommand

##### API Syntax:

```
UINT32 ObjectCustomCommand (UINT32 Parameter1, UINT32 Parameter2, UINT32  
Parameter3, UINT32 Parameter4, UINT32 Parameter5, UINT32 *DataLength, UINT32  
*Direction)
```

##### Function Description:

This function is called when it receives an iTuner command (0x9999). It is of no use in normal MTP applications.

##### Parameters:

| Type     | Parameter  | Description   |
|----------|------------|---|
| UINT32   | Parameter1 | [Input] MTP parameter 1 in command block  |
| UINT32   | Parameter2 | [Input] MTP parameter 2 in command block  |
| UINT32   | Parameter3 | [Input] MTP parameter 3 in command block  |
| UINT32   | Parameter4 | [Input] MTP parameter 4 in command block  |
| UINT32   | Parameter5 | [Input] MTP parameter 5 in command block  |
| UINT32 * | DataLength | [Output] The length of the next data transfer   |
| UINT32 * | Direction  | [Output] The direction of the next data transfer:<br><b>0</b> : Receive data from HOST (Bulk-Out)<br><b>1</b> : Send data to HOST (Bulk-In) |

Table 2-143. Parameters for AmbaUSB API **ObjectCustomCommand()**.

##### Return:

| Return | Description |
|--------|-------------|
| 0      | Success     |
| 0xFF   | Failure     |

Table 2-144. Returns for AmbaUSB API **ObjectCustomCommand()**.



**Example:**

None

**See Also:**

None

For PROTRULY Confidential Only



#### 2.4.2.22 ObjectCustomDataGet

##### API Syntax:

```
UINT32 ObjectCustomDataGet(UINT8 *ObjectBuffer, UINT32 ObjectOffset, UINT32  
ObjectLengthRequested, UINT32 *ObjectActualLength)
```

##### Function Description:

It is called by the MTP class driver if **Direction** is 1 and **DataLength** is not 0 after it is returned by the **ObjectCustomCommand** callback function. It cannot be used in normal MTP applications. This operation is equivalent to the USB "Bulk-In" data transfer.

##### Parameters:

| Type     | Parameter                    | Description  |
|----------|------------------------------|--|
| UINT8 *  | <b>ObjectBuffer</b>          | [Output] The buffer containing the data            |
| UINT32   | <b>ObjectOffset</b>          | [Input] The offset of the data to be sent          |
| UINT32   | <b>ObjectLengthRequested</b> | [Input] The size of the data requested             |
| UINT32 * | <b>ObjectActualLength</b>    | [Output] The actual size of the data in the buffer |

Table 2-145. Parameters for AmbaUSB API **ObjectCustomDataGet()**.

##### Return:

| Return     | Description |
|------------|-------------|
| 0          | Success     |
| 0xFFFFFFFF | Failure     |

Table 2-146. Returns for AmbaUSB API **ObjectCustomDataGet()**.

##### Example:

None

##### See Also:

None



#### 2.4.2.23 ObjectCustomDataSend

##### API Syntax:

```
UINT32 ObjectCustomDataSend (UINT8 *ObjectBuffer, UINT32 ObjectOffset, UINT32  
ObjectLength);
```

##### Function Description:

This function is called by the MTP class driver if **Direction** is 0 and **DataLength** is not 0 after it is returned by the **ObjectCustomCommand** callback function. It cannot be used in normal MTP applications. This operation is equivalent to an USB "Bulk-Out" data transfer.

##### Parameters:

| Type    | Parameter           | Description                                |
|---------|---------------------|--|
| UINT8 * | <b>ObjectBuffer</b> | [Input] The buffer containing the data     |
| UINT32  | <b>ObjectOffset</b> | [Input] The offset of the data             |
| UINT32  | <b>ObjectLength</b> | [Input] The size of the data in the buffer |

Table 2-147. Parameters for AmbaUSB API **ObjectCustomDataSend()**.

##### Return:

| Return     | Description |
|------------|-------------|
| 0          | Success     |
| 0xFFFFFFFF | Failure     |

Table 2-148. Returns for AmbaUSB API **ObjectCustomDataSend()**.

##### Example:

None

##### See Also:

None



#### 2.4.2.24 VendorCmdProcess

##### API Syntax:

```
UINT32 VendorCmdProcess (USB_MTP_CMD_s *VendorCmdInfo)
```

##### Function Description:

This function is called when a MTP vendor command is received. In this callback function, the application can call **AmbaUSBD\_Mtp\_VendorOperation()** for further processing.

##### Parameters:

| Type            | Parameter            | Description  |
|-----------------|----------------------|--|
| USB_MTP_CMD_s * | <b>VendorCmdInfo</b> | [Input] Vendor command information.<br>Please refer to Section 2.4.1.6 for more details. |

Table 2-149. Parameters for AmbaUSB API **VendorCmdProcess()**.

##### Return:

| Return | Description |
|--------|-------------|
| 0      | Success     |
| 0xFF   | Failure     |

Table 2-150. Returns for AmbaUSB API **VendorCmdProcess()**.

##### Example:

None

##### See Also:

None



### 2.4.3 AmbaUSBD\_Mtp\_AddEvent

#### API Syntax:

```
UINT32 AmbaUSBD_Mtp_AddEvent (MTP_EVENT_s* event)
```

#### Function Description:

This function informs the Host of a MTP event through an interrupt endpoint.

#### Parameters:

| Type          | Parameter | Description  |
|---------------|-----------|--|
| MTP_EVENT_s * | event     | [Input] The event information. Please refer to Section 2.4.1.1 for more details. |

Table 2-151. Parameters for AmbaUSB API **AmbaUSBD\_Mtp\_AddEvent()**.

#### Return:

| Return     | Description |
|------------|-------------|
| 0          | Success     |
| 0xFFFFFFFF | Failure     |

Table 2-152. Returns for AmbaUSB API **AmbaUSBD\_Mtp\_AddEvent()**.

#### Example:

```
MTP_EVENT_s event = { 0 } ;  
event.MTP_EVENT_CODE = MTP_EC_OBJECT_ADDED;  
/* ObjectHandle */  
event.MTP_EVENT_PARAMETER_1 = 0x0003;  
event.MTP_EVENT_PARAMETER_2 = 0;  
event.MTP_EVENT_PARAMETER_3 = 0;  
event.MTP_EVENT_SESSION_ID = 0;  
event.MTP_EVENT_TRANSACTION_ID= 0;  
AmbaUSBD_Mtp_AddEvent (&event);
```



See Also:

None

For PROTRULY Confidential Only



## 2.4.4 AmbaUSBD\_Mtp\_SetInfo

### API Syntax:

```
UINT32 AmbaUSBD_Mtp_SetInfo (MTP_DEVICE_INFO_s *DeviceInfo)
```

### Function Description:

This function helps to setup the MTP class system information. It must be called before **AmbaUSBD\_System\_ClassHook()**.

### Parameters:

| Type                | Parameter  | Description   |
|---------------------|------------|---|
| MTP_DEVICE_INFO_s * | DeviceInfo | [Input] MTP device information. Please refer to Section 2.4.1.2 for more details. |

Table 2-153. Parameters for AmbaUSB API **AmbaUSBD\_Mtp\_SetInfo()**.

### Return:

| Return     | Description |
|------------|-------------|
| 0          | Success     |
| 0xFFFFFFFF | Failure     |

Table 2-154. Returns for AmbaUSB API **AmbaUSBD\_Mtp\_SetInfo()**.

### Example:

```
/* PIMA MTP names */

UINT8 MtpDeviceInfoVendorName[] = "Ambarella";
UINT8 MtpDeviceInfoProductName[] = "A9 MTP Device";
UINT8 MtpDeviceInfoSerialNo[] = "0.0.0.1";
UINT8 MtpDeviceInfoVersion[] = "V1.0";

/* PIMA MTP storage names. */
UINT8 MtpVolumeDescription[] = "A9 MTP Client Disk Volume";
UINT8 MtpVolumeLabel[] = "A9 MTP Client SD slot";
```



```
#define DEVICE_PROP_DATE_TIME_DATASET_LENGTH 39
UINT8 DevicePropDateTimeDataset[] = {
/* Device prop code : Date/Time. */
0x11, 0x50, /* Prop code */
0xff, 0xff, /* String */
0x01, /* GET/SET */
0x00, /* Default value : empty string. */
0x10, /* Current value : length of the unicode string. */
0x31, 0x00, 0x39, 0x00, 0x38, 0x00, 0x30, 0x00, /* YYYY */
0x30, 0x00, 0x31, 0x00, /* MM */
0x30, 0x00, 0x31, 0x00, /* DD */
0x54, 0x00, /* T */
0x30, 0x00, 0x30, 0x00, /* HH */
0x30, 0x00, 0x30, 0x00, /* MM */
0x30, 0x00, 0x30, 0x00, /* SS */
0x00, 0x00, /* Unicode terminator. */
0x00 /* Form Flag : None. */
};

#define DEVICE_PROP_SYNCHRONIZATION_PARTNER_DATASET_LENGTH 8
UINT8 DevicePropSynchronizationPartnerDataset[] = {
/* Device prop code : Synchronization Partner. */
0x01, 0xD4, /* Prop code */
0xff, 0xff, /* String */
0x01, /* GET/SET */
0x00, /* Default value : empty string. */
0x00, /* Current value : empty string. */
0x00 /* Form Flag : None. */
};

#define DEVICE_PROP_DEVICE_FRIENDLY_NAME_DATASET_LENGTH 64
UINT8 DevicePropDeviceFriendlyNameDataset[] = {
/* Device prop code : Device Friendly Name. */
0x02, 0xD4, /* Prop code */
```



```
0xff, 0xff, /* String */
0x01, /* GET/SET */
0x0E, /* Default value. Length of Unicode string. */
0x41, 0x00, 0x6D, 0x00, 0x62, 0x00, 0x61, 0x00, /* Unicode string. */
0x20, 0x00, 0x4D, 0x00, 0x54, 0x00, 0x50, 0x00,
0x20, 0x00, 0x44, 0x00, 0x53, 0x00, 0x43, 0x00,
0x20, 0x00,
0x00, 0x00, /* Unicode terminator. */
0x0E, /* Current value. Length of Unicode string. */
0x41, 0x00, 0x6D, 0x00, 0x62, 0x00, 0x61, 0x00,
0x20, 0x00, 0x4D, 0x00, 0x54, 0x00, 0x50, 0x00,
0x20, 0x00, 0x44, 0x00, 0x53, 0x00, 0x43, 0x00, /* Unicode terminator.
*/
0x20, 0x00,
0x00, 0x00, /* Unicode terminator. */
0x00 /* Form Flag : None. */
};

MTP_DEVICE_INFO_s MtpDeviceInfo = {0};
MtpDeviceInfo.DeviceInfoVendorName = MtpDeviceInfoVendorName;
MtpDeviceInfo.DeviceInfoProductName = MtpDeviceInfoProductName;
MtpDeviceInfo.DeviceInfoSerialNo = MtpDeviceInfoSerialNo;
MtpDeviceInfo.DeviceInfoVersion = MtpDeviceInfoVersion;
MtpDeviceInfo.VolumeDescription = MtpVolumeDescription;
MtpDeviceInfo.VolumeLabel = MtpVolumeLabel;
MtpDeviceInfo.DevicePropDataTimeDataset = DevicePropDataTimeDataset;
MtpDeviceInfo.DevicePropDataTimeLength =
DEVICE_PROP_DATE_TIME_DATASET_LENGTH;
MtpDeviceInfo.DevicePropSynchronizationPartnerDataset =
DevicePropSynchronizationPartnerDataset;
MtpDeviceInfo.DevicePropSynchronizationPartnerLength =
DEVICE_PROP_SYNCHRONIZATION_PARTNER_DATASET_LENGTH;
MtpDeviceInfo.DevicePropDeviceFriendlyNameDataset =
DevicePropDeviceFriendlyNameDataset;
```



```
MtpDeviceInfo.DevicePropDeviceFriendlyNameLength =  
DEVICE_PROP_DEVICE_FRIENDLY_NAME_DATASET_LENGTH;  
  
strncpy( (CHAR*)&MtpDeviceInfo.RootPath, "d:\\\", sizeof("d:\\\")-1);  
AmbaUSBD_Mtp_SetInfo (&MtpDeviceInfo);
```

**See Also:**

None

For PROTRULY Confidential Only



## 2.4.5 AmbaUSBD\_Mtp\_VendorOperation

### API Syntax:

```
UINT32 AmbaUSBD_Mtp_VendorOperation (USB_MTP_VENDOR_OP_s  
*VendorOperation)
```

### Function Description:

This utility function is used for processing vendor commands.

### Parameters:

| Type                | Parameter       | Description  |
|---------------------|-----------------|--|
| USB_MTP_VENDOR_OP_s | VendorOperation | [Input] Information for the next vendor operation. Please refer to Section 2.4.1.3 for more details. |

Table 2-155. Parameters for AmbaUSB API **AmbaUSBD\_Mtp\_VendorOperation()**.

### Return:

| Return     | Description |
|------------|-------------|
| 0          | Success     |
| 0xFFFFFFFF | Failure     |

Table 2-156. Returns for AmbaUSB API **AmbaUSBD\_Mtp\_VendorOperation()**.

### Example:

None

### See Also:

None



## 2.4.6 AmbaUSBD\_Mtp\_EnableDebug

### API Syntax:

```
UINT32 AmbaUSBD_Mtp_EnableDebug (void)
```

### Function Description:

This utility function is used to enable the MTP debug information. If the MTP debug

information is enabled, then MTP commands and responses will be printed to UART.

Please do not call this function in normal application flow.

### Parameters:

None

### Return:

| Return     | Description |
|------------|-------------|
| 0          | Success     |
| 0xFFFFFFFF | Failure     |

Table 2-157. Returns for AmbaUSB API **AmbaUSBD\_Mtp\_EnableDebug()**.

### Example:

None

### See Also:

None



## 2.4.7 AmbaUSBD\_Mtp\_DisableDebug

### API Syntax:

```
UINT32 AmbaUSBD_Mtp_DisableDebug (void)
```

### Function Description:

This utility function is used to disable the MTP debug information.

### Parameters:

None

### Return:

| Return     | Description |
|------------|-------------|
| 0          | Success     |
| 0xFFFFFFFF | Failure     |

Table 2-158. Returns for AmbaUSB API `AmbaUSBD_Mtp_DisableDebug()`.

### Example:

None

### See Also:

[AmbaUSBD\\_Mtp\\_EnableDebug\(\)](#)



## 2.4.8 AmbaUSBD\_Mtp\_IsDebugEnabled

### API Syntax:

```
UINT32 AmbaUSBD_Mtp_IsDebugEnabled (void)
```

### Function Description:

This utility function is used to get if MTP debug information is enabled or not.

### Parameters:

None

### Return:

| Return | Description                        |
|--------|------------------------------------|
| 0      | MTP debug information is disabled. |
| 1      | MTP debug information is enabled.  |

Table 2-159. Returns for AmbaUSB API *AmbaUSBD\_Mtp\_IsDebugEnabled()*.

### Example:

None

### See Also:

[AmbaUSBD\\_Mtp\\_EnableDebug\(\)](#)  
[AmbaUSBD\\_Mtp\\_DisableDebug\(\)](#)



## 2.4.9 AmbaUSBD\_Mtp\_SetReadCacheTaskInfo

### API Syntax:

```
UINT32 AmbaUSBD_Mtp_SetReadCacheTaskInfo (UDC_TASKINFO_s *info)
```

### Function Description:

This function is used to set the information for creating MTP read cache task. MTP read cache task will prefetch data for MTP continuous read so it can improve MTP read performance.

It should be called before **AmbaUSBD\_System\_ClassHook()**.

### Parameters:

| Type             | Parameter   | Description   |
|------------------|-------------|---|
| UDC_TASKINFO_s * | <b>info</b> | [Input] Task Information. Please refer to Section 2.1.1.5 for more details. |

Table 2-160. Parameters for AmbaUSB API **AmbaUSBD\_Mtp\_SetReadCacheTaskInfo()**.

### Return:

| Return     | Description |
|------------|-------------|
| 0          | Success     |
| 0xFFFFFFFF | Failure     |

Table 2-161. Returns for AmbaUSB API **AmbaUSBD\_Mtp\_SetReadCacheTaskInfo()**.

### Example:

```
// set the stack size to 10KB, priority to 70, and AffinityMask to CORE
0 for the MTP read cache task.

UDC_TASKINFO_s ti;
ti.StackSize = 1024*10;
ti.Priority = 70;
ti.AffinityMask = 0x01;
AmbaUSBD_Mtp_SetReadCacheTaskInfo (&ti);
```



See Also:

None

For PROTRULY Confidential Only



## 2.4.10 AmbaUSBD\_Mtp\_GetReadCacheTaskInfo

### API Syntax:

```
UINT32 AmbaUSBD_Mtp_GetReadCacheTaskInfo (UDC_TASKINFO_s *info)
```

### Function Description:

This function is used to get the information for creating MTP read cache task.

### Parameters:

| Type             | Parameter | Description  |
|------------------|-----------|--|
| UDC_TASKINFO_s * | info      | [Output] Task Information. Please refer to Section 2.1.1.5 for more details. |

Table 2-162. Parameters for AmbaUSB API **AmbaUSBD\_Mtp\_GetReadCacheTaskInfo()**.

### Return:

| Return     | Description |
|------------|-------------|
| 0          | Success     |
| 0xFFFFFFFF | Failure     |

Table 2-163. Returns for AmbaUSB API **AmbaUSBD\_Mtp\_GetReadCacheTaskInfo()**.

### Example:

```
UDC_TASKINFO_s ti;  
AmbaUSBD_Mtp_GetReadCacheTaskInfo (&ti);
```

### See Also:

None



## 2.4.11 AmbaUSBD\_Mtp\_SetSupportedEvents

### API Syntax:

```
UINT32 AmbaUSBD_Mtp_SetSupportedEvents (UINT16 *list)
```

### Function Description:

This function is used to set supported event list. It must be called before `AmbaUSBD_System_ClassHook()`.

### Parameters:

| Type     | Parameter | Description   |
|----------|-----------|---|
| UINT16 * | list      | [Input] Supported MTP event list. It must end with 0. |

Table 2-164. Parameters for AmbaUSB API `AmbaUSBD_Mtp_SetSupportedEvents()`.

### Return:

| Return     | Description |
|------------|-------------|
| 0          | Success     |
| 0xFFFFFFFF | Failure     |

Table 2-165. Returns for AmbaUSB API `AmbaUSBD_Mtp_SetSupportedEvents()`.

### Example:

```
static UINT16 _supported_events[] = {  
    MTP_EC_CANCEL_TRANSACTION,  
    MTP_EC_OBJECT_ADDED,  
    MTP_EC_OBJECT_REMOVED,  
    MTP_EC_STORE_ADDED,  
    MTP_EC_STORE_REMOVED,  
    MTP_EC_DEVICE_PROP_CHANGED,  
    MTP_EC_OBJECT_INFO_CHANGED,  
    MTP_EC_DEVICE_INFO_CHANGED,  
    MTP_EC_REQUEST_OBJECT_TRANSFER,
```



```
MTP_EC_STORE_FULL,  
MTP_EC_DEVICE_RESET,  
MTP_EC_STORAGE_INFO_CHANGED,  
MTP_EC_UNREPORTED_STATUS,  
MTP_EC_VENDOR_AMBA_TEST,  
0  
};  
  
// set supported event list. If not set, default event list is used.  
AmbaUSBD_Mtp_SetSupportedEvents (_supported_events);
```

**See Also:**

None



## 2.5 Pictbridge Class APIs

This section describes APIs for Pictbridge Class.

For PROTRULY Confidential Only



## 2.5.1 Data Structures and Defines

### 2.5.1.1 PICT\_JOB\_OP\_s

| Type             | Parameter             | Description   |
|------------------|-----------------------|---|
| Function Pointer | <b>ReadObjectData</b> | Callback function to read image data during transfer process. Please refer to Section 2.5.2.1 for more details. |
| Function Pointer | <b>JobEndReason</b>   | Callback function for getting the Job End Reason. Please refer to Section 2.5.2.2 for more details.             |

Table 2-166. Definition of **PICT\_JOB\_OP\_s**.

### 2.5.1.2 PICT\_JOB\_INFO\_s

| Type   | Parameter                   | Description  |
|--------|-----------------------------|--|
| UINT32 | <b>JobinfoQuality</b>       | Pictbridge Qualities Codes. Please refer to Section 2.5.1.3 for more details.      |
| UINT32 | <b>JobinfoPapertype</b>     | Pictbridge PaperTypes Codes. Please refer to Section 2.5.1.4 for more details.     |
| UINT32 | <b>JobinfoPapersize</b>     | Pictbridge PaperSizes Codes. Please refer to Section 2.5.1.5 for more details.     |
| UINT32 | <b>JobinfoFiletype</b>      | Pictbridge FileTypes Codes. Please refer to Section 2.5.1.6 for more details.      |
| UINT32 | <b>JobinfoDateprint</b>     | Pictbridge DatePrints Codes. Please refer to Section 2.5.1.7 for more details.     |
| UINT32 | <b>JobinfoFilenameprint</b> | Pictbridge FileNamePrints Codes. Please refer to Section 2.5.1.8 for more details. |
| UINT32 | <b>JobinfoImageoptimize</b> | Pictbridge ImageOptimizes Codes. Please refer to Section 2.5.1.9 for more details. |
| UINT32 | <b>JobinfoLayout</b>        | Pictbridge Layouts Codes. Please refer to  |



| Type    | Parameter               | Description   |
|---------|-------------------------|---|
|         |                         | Section 2.5.1.10 for more details.  |
| UINT32  | <b>JobinfoFixedSize</b> | Pictbridge FixedSizes Codes. Please refer to Section 2.5.1.11 for more details. |
| UINT32  | <b>JobinfoCropping</b>  | Pictbridge Croppings Codes. Please refer to Section 2.5.1.12 for more details.  |
| UINT32  | <b>ObjectFormat</b>     | PIMA Object Format Codes  |
| UINT32  | <b>ObjectSize</b>       | Object size   |
| UINT8 * | <b>FileName</b>         | Object file name  |
| UINT8 * | <b>date</b>             | Object date string  |
| UINT8 * | <b>buff</b>             | Not used  |

Table 2-167. Definition of **PICT\_JOB\_INFO\_s**.

#### 2.5.1.3 JobinfoQuality

| Type                         | Field      |
|------------------------------|------------|
| PICTBRIDGE_QUALITIES_DEFAULT | 0x50000000 |
| PICTBRIDGE_QUALITIES_NORMAL  | 0x50010000 |
| PICTBRIDGE_QUALITIES_DRAFT   | 0x50020000 |
| PICTBRIDGE_QUALITIES_FINE    | 0x50030000 |

Table 2-168. Definition of **JobinfoQuality**.

#### 2.5.1.4 JobinfoPapertype

| Type                              | Field      |
|-----------------------------------|------------|
| PICTBRIDGE_PAPER_TYPES_DEFAULT    | 0x52000000 |
| PICTBRIDGE_PAPER_TYPES_PLAIN      | 0x52010000 |
| PICTBRIDGE_PAPER_TYPES_PHOTO      | 0x52020000 |
| PICTBRIDGE_PAPER_TYPES_FAST_PHOTO | 0x52030000 |

Table 2-169. Definition of **JobinfoPapertype**.



#### 2.5.1.5 JobinfoPapersize

| Type                                   | Field      |
|--|------------|
| PICTBRIDGE_PAPER_SIZES_DEFAULT         | 0x51000000 |
| PICTBRIDGE_PAPER_SIZES_L               | 0x51010000 |
| PICTBRIDGE_PAPER_SIZES_2L              | 0x51020000 |
| PICTBRIDGE_PAPER_SIZES_HAGAKI_POSTCARD | 0x51030000 |
| PICTBRIDGE_PAPER_SIZES_CARD_SIZE       | 0x51040000 |
| PICTBRIDGE_PAPER_SIZES_100X150         | 0x51050000 |
| PICTBRIDGE_PAPER_SIZES_4IX6I           | 0x51060000 |
| PICTBRIDGE_PAPER_SIZES_8IX10I          | 0x51070000 |
| PICTBRIDGE_PAPER_SIZES_LETTER          | 0x51080000 |
| PICTBRIDGE_PAPER_SIZES_11IX17I         | 0x510A0000 |

Table 2-170. Definition of **JobinfoPapersize**.

#### 2.5.1.6 JobinfoFiletype

| Type                             | Field      |
|----------------------------------|------------|
| PICTBRIDGE_FILE_TYPES_DEFAULT    | 0x53000000 |
| PICTBRIDGE_FILE_TYPES_EXIF_JPEG  | 0x53010000 |
| PICTBRIDGE_FILE_TYPES_OTHER_EXIF | 0x53020000 |
| PICTBRIDGE_FILE_TYPES_JPEG       | 0x53030000 |
| PICTBRIDGE_FILE_TYPES_TIFF_EP    | 0x53040000 |
| PICTBRIDGE_FILE_TYPES_FLASHPIX   | 0x53050000 |
| PICTBRIDGE_FILE_TYPES_BMP        | 0x53060000 |
| PICTBRIDGE_FILE_TYPES_CIFF       | 0x53070000 |
| PICTBRIDGE_FILE_TYPES_GIF        | 0x53080000 |
| PICTBRIDGE_FILE_TYPES_JFIF       | 0x53090000 |
| PICTBRIDGE_FILE_TYPES_PCD        | 0x530A0000 |
| PICTBRIDGE_FILE_TYPES_PICT       | 0x530B0000 |
| PICTBRIDGE_FILE_TYPES_PNG        | 0x530C0000 |



| Type                              | Field      |
|-----------------------------------|------------|
| PICTBRIDGE_FILE_TYPES_TIFF        | 0x530D0000 |
| PICTBRIDGE_FILE_TYPES_TIFF_IT     | 0x530E0000 |
| PICTBRIDGE_FILE_TYPES_JP2         | 0x530F0000 |
| PICTBRIDGE_FILE_TYPES_JPX         | 0x53110000 |
| PICTBRIDGE_FILE_TYPES_UNDEFINED   | 0x53120000 |
| PICTBRIDGE_FILE_TYPES_ASSOCIATION | 0x53130000 |
| PICTBRIDGE_FILE_TYPES_SCRIPT      | 0x53140000 |
| PICTBRIDGE_FILE_TYPES_EXECUTABLE  | 0x53150000 |
| PICTBRIDGE_FILE_TYPES_TEXT        | 0x53160000 |
| PICTBRIDGE_FILE_TYPES_HTML        | 0x53170000 |
| PICTBRIDGE_FILE_TYPES_DPOF        | 0x53180000 |
| PICTBRIDGE_FILE_TYPES_AIFF        | 0x53190000 |
| PICTBRIDGE_FILE_TYPES_WAV         | 0x531A0000 |
| PICTBRIDGE_FILE_TYPES_MP3         | 0x531B0000 |
| PICTBRIDGE_FILE_TYPES_AVI         | 0x531C0000 |
| PICTBRIDGE_FILE_TYPES_MPEG        | 0x531D0000 |
| PICTBRIDGE_FILE_TYPES ASF         | 0x531E0000 |

Table 2-171. Definition of **JobinfoFiletype**.

#### 2.5.1.7 JobinfoDateprint

| Type                           | Field      |
|--------------------------------|------------|
| PICTBRIDGE_DATE_PRINTS_DEFAULT | 0x54000000 |
| PICTBRIDGE_DATE_PRINTS_OFF     | 0x54010000 |
| PICTBRIDGE_DATE_PRINTS_ON      | 0x54020000 |

Table 2-172. Definition of **JobinfoDateprint**.

#### 2.5.1.8 JobinfoFilenameprint

| Type | Field |
|------|-------|
|      |       |



|                                |            |
|--------------------------------|------------|
| PICTBRIDGE_DATE_PRINTS_DEFAULT | 0x54000000 |
| PICTBRIDGE_DATE_PRINTS_OFF     | 0x54010000 |
| PICTBRIDGE_DATE_PRINTS_ON      | 0x54020000 |

Table 2-173. Definition of **JobinfoFilenameprint**.

#### 2.5.1.9 JobinfoImageoptimize

| Type                               | Field      |
|------------------------------------|------------|
| PICTBRIDGE_IMAGE_OPTIMIZES_DEFAULT | 0x56000000 |
| PICTBRIDGE_IMAGE_OPTIMIZES_OFF     | 0x56010000 |
| PICTBRIDGE_IMAGE_OPTIMIZES_ON      | 0x56020000 |

Table 2-174. Definition of **JobinfoImageoptimize**.

#### 2.5.1.10 JobinfoLayout

| Type                               | Field      |
|------------------------------------|------------|
| PICTBRIDGE_LAYOUTS_DEFAULT         | 0x57000000 |
| PICTBRIDGE_LAYOUTS_1_UP_BORDER     | 0x57010000 |
| PICTBRIDGE_LAYOUTS_1_UP_LAYOUT     | 0x57020000 |
| PICTBRIDGE_LAYOUTS_2_UP_LAYOUT     | 0x57030000 |
| PICTBRIDGE_LAYOUTS_3_UP_LAYOUT     | 0x57040000 |
| PICTBRIDGE_LAYOUTS_4_UP_LAYOUT     | 0x57050000 |
| PICTBRIDGE_LAYOUTS_5_UP_LAYOUT     | 0x57060000 |
| PICTBRIDGE_LAYOUTS_6_UP_LAYOUT     | 0x57070000 |
| PICTBRIDGE_LAYOUTS_7_UP_LAYOUT     | 0x57080000 |
| PICTBRIDGE_LAYOUTS_8_UP_LAYOUT     | 0x57090000 |
| PICTBRIDGE_LAYOUTS_INDEX_PRINT     | 0x57FE0000 |
| PICTBRIDGE_LAYOUTS_1_UP_BORDERLESS | 0x57FF0000 |

Table 2-175. Definition of **JobinfoLayout**.



### 2.5.1.11 JobinfoFixedsize

| Type                              | Field      |
|-----------------------------------|------------|
| PICTBRIDGE_FIXED_SIZE_DEFAULT     | 0x58000000 |
| PICTBRIDGE_FIXED_SIZE_2IX3I       | 0x58010000 |
| PICTBRIDGE_FIXED_SIZE_35IX5I      | 0x58020000 |
| PICTBRIDGE_FIXED_SIZE_4IX6I       | 0x58030000 |
| PICTBRIDGE_FIXED_SIZE_5IX7I       | 0x58040000 |
| PICTBRIDGE_FIXED_SIZE_8IX10I      | 0x58050000 |
| PICTBRIDGE_FIXED_SIZE_254MMX178MM | 0x58060000 |
| PICTBRIDGE_FIXED_SIZE_110MMX74MM  | 0x58070000 |
| PICTBRIDGE_FIXED_SIZE_6CMX8CM     | 0x58080000 |
| PICTBRIDGE_FIXED_SIZE_7CMX10CM    | 0x58090000 |
| PICTBRIDGE_FIXED_SIZE_9CMX13CM    | 0x580A0000 |
| PICTBRIDGE_FIXED_SIZE_10CMX13CM   | 0x580B0000 |
| PICTBRIDGE_FIXED_SIZE_15CMX21CM   | 0x580C0000 |
| PICTBRIDGE_FIXED_SIZE_18CMX24CM   | 0x580D0000 |
| PICTBRIDGE_FIXED_SIZE_A4          | 0x580E0000 |
| PICTBRIDGE_FIXED_SIZE_LETTER      | 0x580F0000 |

Table 2-176. Definition of **JobinfoFixedsize**.

### 2.5.1.12 JobinfoCropping

| Type                         | Field      |
|------------------------------|------------|
| PICTBRIDGE_CROPPINGS_DEFAULT | 0x59000000 |
| PICTBRIDGE_CROPPINGS_OFF     | 0x59010000 |
| PICTBRIDGE_CROPPINGS_ON      | 0x59020000 |

Table 2-177. Definition of **JobinfoCropping**.



## 2.5.2 Callback functions

### 2.5.2.1 ReadObjectData

#### API Syntax:

```
UINT32 ReadObjectData (UINT8 *FileName, UINT8 *ObjectBuffer, UINT32 ObjectOffset,  
UINT32 ObjectLength, UINT32 *ActualLength)
```

#### Function Description:

This function is called when the Pictbridge driver wants to read data.

#### Parameters:

| Type     | Parameter           | Description                             |
|----------|---------------------|---|
| UINT8 *  | <b>FileName</b>     | [Input] The target file name            |
| UINT8 *  | <b>ObjectBuffer</b> | [Output] The buffer containing the data |
| UINT32   | <b>ObjectOffset</b> | [Input] The offset of the file          |
| UINT32   | <b>ObjectLength</b> | [Input] The buffer size                 |
| UINT32 * | <b>ActualLength</b> | [Output] The actual data size           |

Table 2-178. Parameters for AmbaUSB API **ReadObjectData()**.

#### Return:

| Return     | Description |
|------------|-------------|
| 0          | Success     |
| 0xFFFFFFFF | Failure     |

Table 2-179. Returns for AmbaUSB API **ReadObjectData()**.



**Example:**

```
static UINT32 PictBridge_ReadObjectData(UINT8 *FileName, UINT8
*ObjectBuffer, UINT32 ObjectOffset, UINT32 ObjectLength, UINT32
*ActualLength) {
    UINT32     ReadLen = 0;
    UINT32     status = 0;
    int         StatusClose = 0;
    if (PictBridgeFileInfo.FpArray == NULL) {
        PictBridgeFileInfo.FpArray = AmbaFS_fopen((const char*)FileName,
        "r");
        AmbaPrint("open %s",FileName);
        if(PictBridgeFileInfo.FpArray== NULL) {
            AmbaPrint("Failed to open %s",FileName);
            return(0xFF);
        }
    }
    AmbaFS_fseek(PictBridgeFileInfo.FpArray, ObjectOffset,
    AMBA_FS_SEEK_START);
    ReadLen = AmbaFS_fread(ObjectBuffer, 1, ObjectLength,
    PictBridgeFileInfo.FpArray);

    if (ReadLen != 0) {
        status = OK;
    } else {
        status = 0xFF;
    }

    *ActualLength = ReadLen;
    if (AmbaFS_feof(PictBridgeFileInfo.FpArray)) {
        StatusClose = AmbaFS_fclose(PictBridgeFileInfo.FpArray);
        PictBridgeFileInfo.FpArray = NULL;
        AmbaPrint("close file");
        if (status == OK && StatusClose == 0) {
            return (OK);
        }
    }
}
```



```
    } else {
        AmbaPrint("Close file fail");
        return 0xFF;
    }
}
else {
    return (status);
}
};
```

**See Also:**

None

For PROTRULY Only



### 2.5.2.2 JobEndReason

#### API Syntax:

```
UINT32 JobEndReason (UINT32 reason)
```

#### Function Description:

This function is called when Pictbridge driver wants to inform the reason as to why the job ended.

#### Parameters:

| Type   | Parameter | Description                            |
|--------|-----------|--|
| UINT32 | reason    | [Input] Pictbridge Job End Reason Code |

Table 2-180. Parameters for AmbaUSB API **JobEndReason()**.

#### Return:

| Return     | Description |
|------------|-------------|
| 0          | Success     |
| 0xFFFFFFFF | Failure     |

Table 2-181. Returns for AmbaUSB API **JobEndReason()**.

#### Example:

```
static UINT32 PictBridge_JobEndReason(UINT32 reason) {
    int      StatusClose = 0;
    UINT32   status = 0;
    switch (reason) {
        case PICTBRIDGE_JOB_END_REASON_END_NORMAL:
        case PICTBRIDGE_JOB_END_REASON_ABORT_1:
        case PICTBRIDGE_JOB_END_REASON_ABORT_2:
        case PICTBRIDGE_JOB_END_REASON_ABORT_3:
            if (PictBridgeFileInfo.FpArray != NULL) {
                StatusClose = AmbaFS_fclose(PictBridgeFileInfo.FpArray);
            }
    }
}
```



```
PictBridgeFileInfo.FpArray = NULL;
```

```
    AmbaPrint("close file");
```

```
    if (status == OK && StatusClose == 0) {
```

```
        return (OK);
```

```
    } else {
```

```
        AmbaPrint("Close file fail");
```

```
        return 0xFF;
```

```
    }
```

```
}
```

```
break;
```

```
case PICTBRIDGE_JOB_END_REASON_NOT_END:
```

```
    /* Job still not complete */
```

```
    break;
```

```
default:
```

```
    /* Wrong Job End Reason */
```

```
    break;
```

```
}
```

```
return (status);
```

```
}
```

**See Also:**

None



### 2.5.3 AmbaUSBD\_Pictbridge\_SetInfo

#### API Syntax:

```
UINT32 AmbaUSBD_Pictbridge_SetInfo (UINT8 *vendor, UINT8 *product, UINT8 *sn)
```

#### Function Description:

This function sets up the PictBridge class device information.

#### Parameters:

| Type    | Parameter      | Description                  |
|---------|----------------|------------------------------|
| UINT8 * | <b>vendor</b>  | [Input] Vendor string        |
| UINT8 * | <b>product</b> | [Input] Product string       |
| UINT8 * | <b>sn</b>      | [Input] Serial Number string |

Table 2-182. Parameters for AmbaUSB API **AmbaUSBD\_Pictbridge\_SetInfo()**.

#### Return:

| Return     | Description |
|------------|-------------|
| 0          | Success     |
| 0xFFFFFFFF | Failure     |

Table 2-183. Returns for AmbaUSB API **AmbaUSBD\_Pictbridge\_SetInfo()**.

#### Example:

```
UINT8 AmbaPictVendorName[ ] = "Ambarella" ;  
  
UINT8 AmbaPictProduceName[ ] = "A9_DSC_platform" ;  
  
UINT8 AmbaPictSerialNo[ ] = "6668828" ;  
  
INT32 rval = 0 ;  
  
  
rval = AmbaUSBD_Pictbridge_SetInfo (AmbaPictVendorName,  
AmbaPictProduceName, AmbaPictSerialNo) ;  
  
if (rval != OK) {
```



```
AmbaPrint( "AmbeUSB_Class_Pictbridge_SetInfo fail, status = 0x%x" ,  
rval);  
  
}  
  
rval = AmbeUSB_Class_Pictbridge_RegisterCb(&PictBridgeJobOp);  
  
return rval;
```

**See Also:**

None

For PROTRULY Confidential Only



## 2.5.4 AmbaUSBD\_Pictbridge\_RegisterCb

### API Syntax:

```
UINT32 AmbaUSBD_Pictbridge_RegisterCb (PICT_JOB_OP_s *cb)
```

### Function Description:

This function is used to register callback functions for Pictbridge operations.

### Parameters:

| Type            | Parameter | Description  |
|-----------------|-----------|--|
| PICT_JOB_OP_s * | cb        | Information on callback functions. Please refer to Section 2.5.1.1 for more details. |

Table 2-184. Parameters for AmbaUSB API **AmbaUSBD\_Pictbridge\_RegisterCb()**.

### Return:

| Return     | Description |
|------------|-------------|
| 0          | Success     |
| 0xFFFFFFFF | Failure     |

Table 2-185. Returns for AmbaUSB API **AmbaUSBD\_Pictbridge\_RegisterCb()**.

### Example:

```
PICT_JOB_OP_s    PictBridgeJobOp = {  
    PictBridge_ReadObjectData,  
    PictBridge_JobEndReason  
};  
  
UINT32 AmbaUSB_PictBridge_Init(void) {  
    INT32 rval = 0;  
    rval = AmbaUSBD_Pictbridge_SetInfo(AmbaPictVendorName,  
    AmbaPictProduceName, AmbaPictSerialNo);  
    if (rval != OK) {  
        AmbaPrint("AmbaUSB_Class_Pictbridge_SetInfo fail, status = 0x%x",  
        rval);  
    }  
}
```



```
    }  
  
    rval = AmbaUSBD_Pictbridge_RegisterCb (&PictBridgeJob0p);  
    return rval;  
}
```

**See Also:**

None

For PROTRULY Confidential Only



## 2.5.5 AmbaUSBD\_Pictbridge\_CheckCapability

### API Syntax:

```
UINT32 AmbaUSBD_Pictbridge_CheckCapability (UINT32 capability)
```

### Function Description:

This function is used to issue Pictbridge "check capability" events.

### Parameters:

| Type   | Parameter  | Description   |
|--------|------------|---|
| UINT32 | capability | Could be one of following values:<br><b>PICTBRIDGE_CAPABILITY_QUALITIES</b><br><b>PICTBRIDGE_CAPABILITY_PAPER_SIZES</b><br><b>PICTBRIDGE_CAPABILITY_PAPER_TYPES</b><br><b>PICTBRIDGE_CAPABILITY_PAPER_TYPES_SIZE</b><br><b>PICTBRIDGE_CAPABILITY_FILE_TYPES</b><br><b>PICTBRIDGE_CAPABILITY_DATE_PRINTS</b><br><b>PICTBRIDGE_CAPABILITY_FILE_NAME_PRINTS</b><br><b>PICTBRIDGE_CAPABILITY_IMAGE_OPTIMIZES</b><br><b>PICTBRIDGE_CAPABILITY_LAYOUTS</b><br><b>PICTBRIDGE_CAPABILITY_LAYOUTS_SIZE</b><br><b>PICTBRIDGE_CAPABILITY_FIXED_SIZES</b><br><b>PICTBRIDGE_CAPABILITY_CROPPINGS</b> |

Table 2-186. Parameters for AmbaUSB API **AmbaUSBD\_Pictbridge\_CheckCapability()**.

### Return:

| Return | Description |
|--------|-------------|
| 0      | Success     |
| 0xff   | Failure     |

Table 2-187. Returns for AmbaUSB API **AmbaUSBD\_Pictbridge\_CheckCapability()**.



**Example:**

```
status = AmbaUSBD_Pictbridge_CheckCapability  
(PICTBRIDGE_CAPABILITY_QUALITIES);  
  
status = AmbaUSBD_Pictbridge_CheckCapability  
(PICTBRIDGE_CAPABILITY_PAPER_SIZES);  
  
status = AmbaUSBD_Pictbridge_CheckCapability  
(PICTBRIDGE_CAPABILITY_FILE_TYPES);  
  
status = AmbaUSBD_Pictbridge_CheckCapability  
(PICTBRIDGE_CAPABILITY_DATE_PRINTS);  
  
status = AmbaUSBD_Pictbridge_CheckCapability  
(PICTBRIDGE_CAPABILITY_FILE_NAME_PRINTS);  
  
status = AmbaUSBD_Pictbridge_CheckCapability  
(PICTBRIDGE_CAPABILITY_IMAGE_OPTIMIZES);  
  
status = AmbaUSBD_Pictbridge_CheckCapability  
(PICTBRIDGE_CAPABILITY_LAYOUTS);  
  
status = AmbaUSBD_Pictbridge_CheckCapability  
(PICTBRIDGE_CAPABILITY_FIXED_SIZES);  
  
status = AmbaUSBD_Pictbridge_CheckCapability  
(PICTBRIDGE_CAPABILITY_CROPPINGS);
```

**See Also:**

None



## 2.5.6 AmbaUSBD\_Pictbridge\_ConfigJob

### API Syntax:

```
UINT32 AmbaUSBD_Pictbridge_ConfigJob (PICT_JOB_INFO_s *config)
```

### Function Description:

This function is used to setup and start the Pictbridge job.

### Parameters:

| Type              | Parameter | Description   |
|-------------------|-----------|---|
| PICT_JOB_INFO_s * | config    | [Input] Information of the Pictbridge job.<br>Please refer to Section 2.5.1.2 for more details. |

Table 2-188. Parameters for AmbaUSB API **AmbaUSBD\_Pictbridge\_ConfigJob()**.

### Return:

| Return | Description |
|--------|-------------|
| 0      | Success     |
| -1     | Failure     |

Table 2-189. Returns for AmbaUSB API **AmbaUSBD\_Pictbridge\_ConfigJob()**.

### Example:

```
INT32 status = 0;  
  
PICT_JOB_INFO_s *pconfig = &config;  
  
pconfig->FileName = PictBridgeFileInfo.FileName;  
pconfig->date = PictBridgeFileInfo.Date;  
pconfig->JobinfoQuality = PICTBRIDGE_QUALITIES_DEFAULT;  
pconfig->JobinfoPapertype = PICTBRIDGE_PAPER_TYPES_DEFAULT;  
pconfig->JobinfoPapersize = PICTBRIDGE_PAPER_SIZES_DEFAULT;  
pconfig->JobinfoFiletype = PICTBRIDGE_FILE_TYPES_DEFAULT;  
pconfig->JobinfoDateprint = PICTBRIDGE_DATE_PRINTS_DEFAULT;  
pconfig->JobinfoFilenameprint = PICTBRIDGE_FILE_NAME_PRINTS_DEFAULT;  
pconfig->JobinfoImageoptimize = PICTBRIDGE_IMAGE_OPTIMIZES_DEFAULT;
```



```
pconfig->JobinfoLayout = PICTBRIDGE_LAYOUTS_DEFAULT;  
pconfig->JobinfoFixedSize = PICTBRIDGE_FIXED_SIZE_DEFAULT;  
pconfig->JobinfoCropping = PICTBRIDGE_CROPPINGS_DEFAULT;  
pconfig->ObjectFormat = CLASS_PIMA_OFC_EXIF_JPEG;  
pconfig->ObjectSize = PictBridgeFileInfo.FileLength  
status = AmbaUSBD_Pictbridge_ConfigJob (pconfig);
```

**See Also:**

None



## 2.5.7 AmbaUSBD\_Pictbridge\_IsPrinterServiceReady

### API Syntax:

```
UINT32 AmbaUSBD_Pictbridge_IsPrinterServiceReady (void)
```

### Function Description:

This function is used to check if the printer service is ready or not.

### Parameters:

None

### Return:

| Return | Description                   |
|--------|-------------------------------|
| 0      | Printer service is ready.     |
| -1     | Printer service is not ready. |

Table 2-190. Returns for AmbaUSB API `AmbaUSBD_Pictbridge_IsPrinterServiceReady()`.

### Example:

```
/* Wait until configure_print_service is done */
while(!AmbaUSBD_Pictbridge_IsPrinterServiceReady()) {
    tx_thread_sleep(100);
    if (count-- < 0) {
        break;
    }
}
```

### See Also:

None



## 2.5.8 AmbaUSBD\_Pictbridge\_AbortJob

### API Syntax:

```
UINT32 AmbaUSBD_Pictbridge_AbortJob (void)
```

### Function Description:

This function is used to request the printer to stop the print job.

### Parameters:

None

### Return:

| Return | Description |
|--------|-------------|
| 0      | Success     |
| -1     | Failure     |

Table 2-191. Returns for AmbaUSB API `AmbaUSBD_Pictbridge_AbortJob()`.

### Example:

None

### See Also:

None



## 2.5.9 AmbaUSBD\_Pictbridge\_ContinueJob

### API Syntax:

```
UINT32 AmbaUSBD_Pictbridge_ContinueJob (void)
```

### Function Description:

This function is used to request the printer to resume its previous printing job.

### Parameters:

None

### Return:

| Return | Description |
|--------|-------------|
| 0      | Success     |
| -1     | Failure     |

Table 2-192. Returns for AmbaUSB API `AmbaUSBD_Pictbridge_ContinueJob()`.

### Example:

None

### See Also:

None



## 2.5.10 AmbaUSBD\_Pictbridge\_SetMainTaskInfo

### API Syntax:

```
UINT32 AmbaUSBD_Pictbridge_SetMainTaskInfo (UDC_TASKINFO_s *info)
```

### Function Description:

This function is used to set the information for creating Pictbridge main task. It should be called before **AmbaUSBD\_System\_ClassHook()**.

### Parameters:

| Type             | Parameter | Description   |
|------------------|-----------|---|
| UDC_TASKINFO_s * | info      | [Input] Task information. Please refer to Section 2.1.1.5 for more details. |

Table 2-193. Parameters for AmbaUSB API **AmbaUSBD\_Pictbridge\_SetMainTaskInfo()**.

### Return:

| Return     | Description |
|------------|-------------|
| 0          | Success     |
| 0xFFFFFFFF | Failure     |

Table 2-194. Returns for AmbaUSB API **AmbaUSBD\_Pictbridge\_SetMainTaskInfo()**.

### Example:

```
// set the stack size to 10KB, priority to 70, and AffinityMask to CORE
0.

UDC_TASKINFO_s ti;
ti.StackSize = 1024*10;
ti.Priority = 70;
ti.AffinityMask = 0x01;
AmbaUSBD_Pictbridge_SetMainTaskInfo (&ti);
```

### See Also:

None



## 2.5.11 AmbaUSBD\_Pictbridge\_GetMainTaskInfo

### API Syntax:

```
UINT32 AmbaUSBD_Pictbridge_GetMainTaskInfo (UDC_TASKINFO_s *info)
```

### Function Description:

This function is used to get the information for creating Pictbridge main task.

### Parameters:

| Type             | Parameter | Description  |
|------------------|-----------|--|
| UDC_TASKINFO_s * | info      | [Output] Task information. Please refer to Section 2.1.1.5 for more details. |

Table 2-195. Parameters for AmbaUSB API **AmbaUSBD\_Pictbridge\_GetMainTaskInfo()**.

### Return:

| Return     | Description |
|------------|-------------|
| 0          | Success     |
| 0xFFFFFFFF | Failure     |

Table 2-196. Returns for AmbaUSB API **AmbaUSBD\_Pictbridge\_GetMainTaskInfo()**.

### Example:

```
UDC_TASKINFO_s ti;  
AmbaUSBD_Pictbridge_GetMainTaskInfo (&ti);
```

### See Also:

None



## 2.6 CDC ACM Class APIs

The USB CDC ACM class is also called as the “Serial over USB”. The full name of CDC ACM is “Communication Device Class with Abstract Control Model”. Note that two categories are supported; one is CDC ACM with single instance and the other is CDC ACM with multiple instances. An application can just use former category if it just wants to have one CDC-ACM instance.

For PROTRULY Confidential Only



## 2.6.1 AmbaUSBD\_CDC\_ACM\_TerminalOpen

### API Syntax:

```
UINT32 AmbaUSBD_CDC_ACM_TerminalOpen (void)
```

### Function Description:

This function checks if the terminal is opened by the Host.

### Parameters:

None

### Return:

| Return | Description                            |
|--------|--|
| 0      | The terminal is NOT opened by the Host |
| 1      | The terminal is opened by the Host     |

Table 2-197. Returns for AmbaUSB API **AmbaUSBD\_CDC\_ACM\_TerminalOpen()**.

### Example:

```
while (UsbCdcParam.TaskRunning == 1) {
    if (AmbaUSBD_CDC_ACM_TerminalOpen () == 1) {
        ret = AmbaUSBD_CDC_ACM_Read(RcvBuf,
                                      CDC_ACM_RCV_BUFSIZE, &bytes_recv);
        if (ret != 0 && ret != UER_REQUEST_EMPTY) {
            break;
        } else {
            if (ret != UER_REQUEST_EMPTY) {
                CDC_ACM_WriteRingBuf(AmbshBuf, CDC_ACM_AMBSH_BUFSIZE,
                                     &(UsbCdcParam.AmbshHead),
                                     &(UsbCdcParam.AmbshTail), RcvBuf,
                                     bytes_recv);
            }
        }
    }
}
```



```
    } else {  
        AmbaKAL_TaskSleep(100);  
    }  
}
```

**See Also:**

None

For PROTRULY Confidential Only



## 2.6.2 AmbaUSBD\_CDC\_ACM\_Write

### API Syntax:

```
UINT32 AmbaUSBD_CDC_ACM_Write (UINT8 *buffer, UINT32 RequestLength, UINT32  
*ActualLength)
```

### Function Description:

This function requests the USB CDC ACM class driver to send data to the USB host.

### Parameters:

| Type     | Parameter            | Description                                   |
|----------|----------------------|---|
| UINT8 *  | <b>buffer</b>        | [Input] The buffer to write                   |
| UINT32   | <b>RequestLength</b> | [Input] The data size for write               |
| UINT32 * | <b>ActualLength</b>  | [Output] The actual data size written to Host |

Table 2-198. Parameters for AmbaUSB API **AmbaUSBD\_CDC\_ACM\_Write()**.

### Return:

| Return | Description |
|--------|-------------|
| 0      | Success     |
| -1     | Failure     |

Table 2-199. Returns for AmbaUSB API **AmbaUSBD\_CDC\_ACM\_Write()**.

### Example:

```
for (i = 0; i < (size / PerSize); i++) {  
    if ((head + PerSize) > CDC_ACM_LOG_BUFSIZE) {  
        UINT32 part_size = CDC_ACM_LOG_BUFSIZE - head;  
        memcpy(SendBuf, LogBuf + head, part_size);  
        memcpy(SendBuf + part_size, LogBuf, PerSize - part_size);  
        head = PerSize - part_size;  
    } else {  
        memcpy(SendBuf, LogBuf + head, PerSize);  
        head += PerSize;
```



{

```
    if (head > CDC_ACM_LOG_BUFSIZE) {  
        AmbaPrint("[Error] head(%d) should not greater than mem_size(%d).",  
head, CDC_ACM_LOG_BUFSIZE);  
        head = 0;  
    }  
    ret = AmbaUSBD_CDC_ACM_Write (SendBuf, PerSize, &ActualLength)  
}
```

**See Also:**

None



### 2.6.3 AmbaUSBD\_CDC\_ACM\_Read

#### API Syntax:

```
UINT32 AmbaUSBD_CDC_ACM_Read (UINT8 *buffer, UINT32 RequestLength, UINT32  
*ActualLength)
```

#### Function Description:

This function requests the USB CDC ACM class driver to receive data from the USB host.

#### Parameters:

| Type     | Parameter            | Description  |
|----------|----------------------|--|
| UINT8 *  | <b>buffer</b>        | [Input] The buffer to read                           |
| UINT32   | <b>RequestLength</b> | [Input] The data size to read                        |
| UINT32 * | <b>ActualLength</b>  | [Output] The actual data size received from the Host |

Table 2-200. Parameters for AmbaUSB API **AmbaUSBD\_CDC\_ACM\_Read()**.

#### Return:

| Return | Description |
|--------|-------------|
| 0      | Success     |
| -1     | Failure     |

Table 2-201. Returns for AmbaUSB API **AmbaUSBD\_CDC\_ACM\_Read()**.

#### Example:

```
while (UsbCdcParam.TaskRunning == 1) {  
    if (AmbaUSBD_CDC_ACM_TerminalOpen() == 1) {  
        ret = AmbaUSBD_CDC_ACM_Read (RcvBuf,  
                                      CDC_ACM_RCV_BUFSIZE, &bytes_recv);  
        if (ret != 0 && ret != UER_REQUEST_EMPTY) {  
            break;  
        } else {  
            if (ret != UER_REQUEST_EMPTY) {
```



```
CDC_ACM_WriteRingBuf(AmbshBuf, CDC_ACM_AMBSH_BUFSIZE,  
    &(UsbCdcParam.AmbshHead), &(UsbCdcParam.AmbshTail),  
    RcvBuf, bytes_recv);  
}  
}  
}  
} else {  
    AmbaKAL_TaskSleep(100);  
}  
}
```

**See Also:**

None



## 2.6.4 AmbaUSBD\_CDC\_ACM\_Multi\_TerminalOpen

### API Syntax:

```
UINT32 AmbaUSBD_CDC_ACM_Multi_TerminalOpen (UINT32 instance_id)
```

### Function Description:

This function checks if a terminal is opened by the Host.

### Parameters:

| Type   | Parameter   | Description  |
|--------|-------------|--|
| UINT32 | instance_id | [Input] The instance ID of the terminal.<br>0 - 1st instance<br>1 - 2nd instance |

Table 2-202. Parameters for AmbaUSB API **AmbaUSBD\_CDC\_ACM\_Multi\_TerminalOpen()**.

### Return:

| Return | Description                            |
|--------|--|
| 0      | The terminal is NOT opened by the Host |
| 1      | The terminal is opened by the Host     |

Table 2-203. Returns for AmbaUSB API **AmbaUSBD\_CDC\_ACM\_Multi\_TerminalOpen()**.

### Example:

```
while (UsbCdcParam.TaskRunning == 1) {  
    if (AmbaUSBD_CDC_ACM_Multi_TerminalOpen (0) == 1) {  
        ret = AmbaUSBD_CDC_ACM_Multi_Read(0, RcvBuf,  
                                           CDC_ACM_RCV_BUFSIZE, &bytes_recv);  
        if (ret != 0 && ret != UER_REQUEST_EMPTY) {  
            break;  
        } else {  
            if (ret != UER_REQUEST_EMPTY) {  
                // Process received data  
            }  
        }  
    }  
}
```



```
CDC_ACM_WriteRingBuf(AmbshBuf, CDC_ACM_AMBSH_BUFSIZE,  
    &(UsbCdcParam.AmbshHead), &(UsbCdcParam.AmbshTail),  
    RcvBuf, bytes_recv);  
}  
}  
}  
} else {  
    AmbaKAL_TaskSleep(100);  
}  
}
```

**See Also:**

None

For PROTRULY Confidential Only



## 2.6.5 AmbaUSBD\_CDC\_ACM\_Multi\_Write

### API Syntax:

```
UINT32 AmbaUSBD_CDC_ACM_Multi_Write (UINT32 instance_id, UINT8 *buffer, UINT32 RequestLength, UINT32 *ActualLength)
```

### Function Description:

This function requests the USB CDC ACM class driver to send data to the USB host through one terminal.

### Parameters:

| Type     | Parameter            | Description  |
|----------|----------------------|--|
| UINT32   | <b>instance_id</b>   | [Input] The instance ID of the terminal.<br>0 - 1st instance<br>1 - 2nd instance |
| UINT8 *  | <b>buffer</b>        | [Input] The buffer to write  |
| UINT32   | <b>RequestLength</b> | [Input] The data size for write  |
| UINT32 * | <b>ActualLength</b>  | [Output] The actual data size written to Host                                    |

Table 2-204. Parameters for AmbaUSB API **AmbaUSBD\_CDC\_ACM\_Multi\_Write()**.

### Return:

| Return | Description |
|--------|-------------|
| 0      | Success     |
| -1     | Failure     |

Table 2-205. Returns for AmbaUSB API **AmbaUSBD\_CDC\_ACM\_Multi\_Write()**.

### Example:

```
for (i = 0; i < (size / PerSize); i++) {  
    if ((head + PerSize) > CDC_ACM_LOG_BUFSIZE) {  
        UINT32 part_size = CDC_ACM_LOG_BUFSIZE - head;  
        memcpy(SendBuf, LogBuf + head, part_size);  
        memcpy(SendBuf + part_size, LogBuf, PerSize - part_size);  
    }  
}
```



```
head = PerSize - part_size;  
} else {  
    memcpy(SendBuf, LogBuf + head, PerSize);  
    head += PerSize;  
}  
  
if (head > CDC_ACM_LOG_BUFSIZE) {  
    AmbaPrint("[Error] head(%d) should not greater than mem_size(%d).",  
head, CDC_ACM_LOG_BUFSIZE);  
    head = 0;  
}  
ret = AmbaUSBD_CDC_ACM_Multi_Write (0, SendBuf, PerSize,  
&ActualLength)  
}
```

**See Also:**

None



## 2.6.6 AmbaUSBD\_CDC\_ACM\_Multi\_Read

### API Syntax:

```
UINT32 AmbaUSBD_CDC_ACM_Multi_Read (UINT8 *buffer, UINT32 RequestLength,  
UINT32 *ActualLength)
```

### Function Description:

This function requests the USB CDC ACM class driver to receive data from the USB host through one terminal.

### Parameters:

| Type     | Parameter            | Description  |
|----------|----------------------|--|
| UINT32   | <b>instance_id</b>   | [Input] The instance ID of the terminal.<br>0: 1st instance<br>1: 2nd instance |
| UINT8 *  | <b>buffer</b>        | [Input] The buffer to read   |
| UINT32   | <b>RequestLength</b> | [Input] The data size to read  |
| UINT32 * | <b>ActualLength</b>  | [Output] The actual data size received from the Host                           |

Table 2-206. Parameters for AmbaUSB API **AmbaUSBD\_CDC\_ACM\_Multi\_Read()**.

### Return:

| Return | Description |
|--------|-------------|
| 0      | Success     |
| -1     | Failure     |

Table 2-207. Returns for AmbaUSB API **AmbaUSBD\_CDC\_ACM\_Multi\_Read()**.

### Example:

```
while (UsbCdcParam.TaskRunning == 1) {  
    if (AmbaUSBD_CDC_ACM_Multi_TerminalOpen(0) == 1) {  
        ret = AmbaUSBD_CDC_ACM_Multi_Read (0, RcvBuf,  
                                         CDC_ACM_RCV_BUFSIZE, &bytes_recv);
```



```
if (ret != 0 && ret != UER_REQUEST_EMPTY) {  
    break;  
} else {  
    if (ret != UER_REQUEST_EMPTY) {  
        CDC_ACM_WriteRingBuf(AmbshBuf, CDC_ACM_AMBSH_BUFSIZE,  
            &(UsbCdcParam.AmbshHead), &(UsbCdcParam.AmbshTail),  
            RcvBuf, bytes_recv);  
    }  
}  
}  
}  
} else {  
    AmbaKAL_TaskSleep(100);  
}  
}  
}
```

**See Also:**

None



## 2.7 Simple Class APIs

Simple Class is a proprietary class defined by Ambarella. Currently, USB simple class supports 4 endpoints (2 bulk-in, 2 bulk-out) transfer. User should use predefined endpoint address with specific type as below:

- Bulk-in : 0x82
- Bulk-out : 0x01
- Bulk-in : 0x84
- Bulk-out: 0x03



## 2.7.1 AmbaUSBD\_Simple\_Init

### API Syntax:

```
UINT32 AmbaUSBD_Simple_Init (void)
```

### Function Description:

Pre-initialization of simple class before class is hooked.

#### Parameters:

None

#### Return:

None

#### Example:

```
AmbaUSBD_Simple_Init ();  
AmbaUSBD_System_ClassHook(UDC_CLASS_SIMPLE);
```

#### See Also:

None



## 2.7.2 AmbaUSBD\_Simple\_Read

### API Syntax:

```
UINT32 AmbaUSBD_Simple_Read (UINT8 *buffer, UINT32 RequestLength, UINT32  
*ActualLength, UINT32 EndpointAddress, UINT32 Timeout);
```

### Function Description:

The function tells the USB simple class driver to receive data from the USB host.

### Parameters:

| Type     | Parameter              | Description  |
|----------|------------------------|--|
| UINT8 *  | <b>buffer</b>          | [Input] The buffer for read  |
| UINT32   | <b>RequestLength</b>   | [Input] The data size for read   |
| UINT32 * | <b>ActualLength</b>    | [Output] The actual data size read from Host   |
| UINT32   | <b>EndpointAddress</b> | [Input] Used endpoint  |
| UINT32   | <b>Timeout</b>         | [Input] Timeout for transfer. 0 means “No Wait”.<br>0xFFFFFFFF means “Wait Forever”. |

Table 2-208. Parameters for AmbaUSB API **AmbaUSBD\_Simple\_Read()**.

### Return:

| Return | Description |
|--------|-------------|
| 0      | Success     |
| 0xFF   | Fail        |

Table 2-209. Returns for AmbaUSB API **AmbaUSBD\_Simple\_Read()**.

### Example:

```
UINT8 buff[10];  
  
UINT32 bytes_transferred;  
  
UINT32 nRet;  
  
  
nRet = AmbaUSBD_Simple_Read (buff, 10, &bytes_transferred, 0x01);
```



See Also:

None

For PROTRULY Confidential Only



### 2.7.3 AmbaUSBD\_Simple\_Write

#### API Syntax:

```
UINT32 AmbaUSBD_Simple_Write (UINT8 *buffer, UINT32 RequestLength, UINT32  
*ActualLength, UINT32 EndpointAddress, UINT32 Timeout)
```

#### Function Description:

The function tells the USB simple class driver to send the data to the USB host.

#### Parameters:

| Type     | Parameter              | Description  |
|----------|------------------------|--|
| UINT8 *  | <b>buffer</b>          | [Input] The buffer for write   |
| UINT32   | <b>RequestLength</b>   | [Input] The data size for write  |
| UINT32 * | <b>ActualLength</b>    | [Output] The actual data size written to the Host                                    |
| UINT32   | <b>EndpointAddress</b> | [Input] Used endpoint  |
| UINT32   | <b>Timeout</b>         | [Input] Timeout for transfer. 0 means “No Wait”.<br>0xFFFFFFFF means “Wait Forever”. |

Table 2-210. Parameters for AmbaUSB API **AmbaUSBD\_Simple\_Write()**.

#### Return:

| Return | Description |
|--------|-------------|
| 0      | Success     |
| 0xFF   | Fail        |

Table 2-211. Returns for AmbaUSB API **AmbaUSBD\_Simple\_Write()**.



**Example:**

```
UINT8 buff[10];  
UINT32 bytes_transferred;  
UINT32 nRet;  
buff[0] = 0x20;  
buff[0] = 0x22;  
buff[0] = 0x28;  
nRet = AmbaUSBD_Simple_Write (ptr, 3, &bytes_transferred, 0x82);
```

**See Also:**

None



## 2.8 UVC Class APIs

UVC stands for "USB Video Class". Currently, only MJPEG video format and bulk transfer are supported.

Note that UVC bulk transfer follows UVC specification but has compatible issues with different Host operating systems. This is because most of UVC devices support isochronous transfer and UVC specification does not provide clear operating steps for UVC bulk transfer, each Host operating system implements its own steps for UVC bulk transfer devices.



## 2.8.1 Data Structures and Defines

### 2.8.1.1 USBD\_UVC\_EVENT\_CALLBACK\_s

| Type             | Parameter          | Description   |
|------------------|--------------------|---|
| Function Pointer | <b>EncodeStart</b> | Callback function called when Host wants to read video data. Please refer to Section 2.8.2.1 for more details.                              |
| Function Pointer | <b>EncodeStop</b>  | Callback function called when Host wants to stop reading video data. Please refer to Section 2.8.2.2 for more details.                      |
| Function Pointer | <b>PuSet</b>       | Callback function called when Host wants to set a value for a control of processing unit. Please refer to Section 2.8.2.3 for more details. |
| Function Pointer | <b>ItSet</b>       | Callback function called when Host wants to set a value for a control of input terminal. Please refer to Section 2.8.2.4 for more details.  |

Table 2-212. Definition of **USBD\_UVC\_EVENT\_CALLBACK\_s**.

### 2.8.1.2 USBD\_UVC\_HOST\_CONFIG\_s

| Type   | Parameter           | Description                     |
|--------|---------------------|---------------------------------|
| UINT32 | <b>Width</b>        | Video width set by Host         |
| UINT32 | <b>Height</b>       | Video height set by Host        |
| UINT32 | <b>MjpegQuality</b> | MJPEG quality level set by Host |
| UINT32 | <b>Framerate</b>    | Video framerate set by Host     |

Table 2-213. Definition of **USBD\_UVC\_HOST\_CONFIG\_s**.



### 2.8.1.3 USBD\_UVC\_PU\_CONTROL\_SETTING\_s

| Type   | Parameter           | Description  |
|--------|---------------------|--|
| UINT32 | <b>ControlIndex</b> | The index of this control in processing unit. Below are the possible values:<br>UVC_PU_CONTROL_UNDEFINED<br>UVC_PU_BACKLIGHT_COMPENSATION_CONTROL<br>UVC_PU_BRIGHTNESS_CONTROL<br>UVC_PU_CONTRAST_CONTROL<br>UVC_PU_GAIN_CONTROL<br>UVC_PU_POWER_LINE_FREQUENCY_CONTROL<br>UVC_PU_HUE_CONTROL<br>UVC_PU_SATURATION_CONTROL<br>UVC_PU_SHARPNESS_CONTROL<br>UVC_PU_GAMMA_CONTROL<br>UVC_PU_WHITE_BALANCE_TEMPERATURE_CONTROL<br>UVC_PU_WHITE_BALANCE_TEMPERATURE_AUTO_CONTROL<br>UVC_PU_WHITE_BALANCE_COMPONENT_CONTROL<br>UVC_PU_WHITE_BALANCE_COMPONENT_AUTO_CONTROL<br>UVC_PU_DIGITAL_MULTIPLIER_CONTROL<br>UVC_PU_DIGITAL_MULTIPLIER_LIMIT_CONTROL<br>UVC_PU_HUE_AUTO_CONTROL<br>UVC_PU_ANALOG_VIDEO_STANDARD_CONTROL<br>UVC_PU_ANALOG_LOCK_STATUS_CONTROL |
| UINT32 | <b>MaximumValue</b> | Maximum value of this control in processing unit   |
| UINT32 | <b>MinimumValue</b> | Minimum value of this control in processing unit   |
| UINT32 | <b>DefaultValue</b> | Default value of this control in processing unit   |
| UINT32 | <b>CurrentValue</b> | Current value of this control in processing unit   |

Table 2-214. Definition of **USBD\_UVC\_PU\_CONTROL\_SETTING\_s**.



## 2.8.2 Callback functions

### 2.8.2.1 EncodeStart

#### API Syntax:

```
UINT32 EncodeStart (void)
```

#### Function Description:

This function is called when the Host wants to read video data. Applications should send video data afterwards.

#### Parameters:

None

#### Return:

| Return     | Description |
|------------|-------------|
| 0          | Success     |
| 0xFFFFFFFF | Fail        |

Table 2-215 Returns for AmbaUSB API *EncodeStart()*.

#### Example:

None

#### See Also:

None



### 2.8.2.2 EncodeStop

#### API Syntax:

```
UINT32 EncodeStop (void)
```

#### Function Description:

This function is called when the Host wants to stop reading video data. Applications should not send video data afterwards.

#### Parameters:

None

#### Return:

| Return     | Description |
|------------|-------------|
| 0          | Success     |
| 0xFFFFFFFF | Fail        |

Table 2-216 Table 2-217 Returns for AmbaUSB API *EncodeStop()*.

#### Example:

None

#### See Also:

None



### 2.8.2.3 PuSet

#### API Syntax:

UINT32 **PuSet** (UINT32 index)

#### Function Description:

This function is called when the Host wants to set a value for a control of processing unit.



**Parameters:**

| Type   | Parameter    | Description   |
|--------|--------------|---|
| UINT32 | <b>index</b> | The index of this control in processing unit. Below is possible values:<br>UVC_PU_CONTROL_UNDEFINED<br>UVC_PU_BACKLIGHT_COMPENSATION_CONTROL<br>UVC_PU_BRIGHTNESS_CONTROL<br>UVC_PU_CONTRAST_CONTROL<br>UVC_PU_GAIN_CONTROL<br>UVC_PU_POWER_LINE_FREQUENCY_CONTROL<br>UVC_PU_HUE_CONTROL<br>UVC_PU_SATURATION_CONTROL<br>UVC_PU_SHARPNESS_CONTROL<br>UVC_PU_GAMMA_CONTROL<br>UVC_PU_WHITE_BALANCE_TEMPERATURE_CONTROL<br>UVC_PU_WHITE_BALANCE_TEMPERATURE_AUTO_CONTROL<br>UVC_PU_WHITE_BALANCE_COMPONENT_CONTROL<br>UVC_PU_WHITE_BALANCE_COMPONENT_AUTO_CONTROL<br>UVC_PU_DIGITAL_MULTIPLIER_CONTROL<br>UVC_PU_DIGITAL_MULTIPLIER_LIMIT_CONTROL<br>UVC_PU_HUE_AUTO_CONTROL<br>UVC_PU_ANALOG_VIDEO_STANDARD_CONTROL<br>UVC_PU_ANALOG_LOCK_STATUS_CONTROL |

Table 2-218 Parameters for AmbaUSB API **PuSet()**.

**Return:**

| Return     | Description |
|------------|-------------|
| 0          | Success     |
| 0xFFFFFFFF | Fail        |

Table 2-219 Returns for AmbaUSB API **PuSet()**.



**Example:**

```
static UINT32 uvc_dpu_set(UINT32 index)
{
    USBD_UVC_PU_CONTROL_SETTING_s setting;

    switch(index) {
        case UVC_PU_BRIGHTNESS_CONTROL:
            AmbaUSBD_UVC_GetBrightnessSetting(&setting);
            AmbaPrint("[UVCD] Host set Brightness to %d [%d ~ %d]",
                      setting.CurrentValue,
                      setting.MinimumValue,
                      setting.MaximumValue);
            break;
        case UVC_PU_CONTRAST_CONTROL:
            AmbaUSBD_UVC_GetContrastSetting(&setting);
            AmbaPrint("[UVCD] Host set Contrast to %d [%d ~ %d]",
                      setting.CurrentValue,
                      setting.MinimumValue,
                      setting.MaximumValue);
            break;
        case UVC_PU_HUE_CONTROL:
            AmbaUSBD_UVC_GetHueSetting(&setting);
            AmbaPrint("[UVCD] Host set Hue to %d [%d ~ %d]",
                      setting.CurrentValue,
                      setting.MinimumValue,
                      setting.MaximumValue);
            break;
        case UVC_PU_SATURATION_CONTROL:
            AmbaUSBD_UVC_GetSaturationSetting(&setting);
            AmbaPrint("[UVCD] Host set Saturation to %d [%d ~ %d]",
                      setting.CurrentValue,
                      setting.MinimumValue,
                      setting.MaximumValue);
            break;
    }
}
```



```
case UVC_PU_SHARPNESS_CONTROL:  
    AmbaUSBD_UVC_GetSharpnessSetting(&setting);  
    AmbaPrint("[UVCD] Host set Sharpness to %d [%d ~ %d]",  
             setting.CurrentValue,  
             setting.MinimumValue,  
             setting.MaximumValue);  
    break;  
  
case UVC_PU_POWER_LINE_FREQUENCY_CONTROL:  
    AmbaUSBD_UVC_GetPowerLineSetting(&setting);  
    AmbaPrint("[UVCD] Host set PowerLine to %d [%d ~ %d]",  
             setting.CurrentValue,  
             setting.MinimumValue,  
             setting.MaximumValue);  
    break;  
  
default:  
    AmbaPrint("[UVCD] Host wants to PU control, index %d", index);  
    break;  
}  
return TX_SUCCESS;  
}
```

**See Also:**

None



#### 2.8.2.4 ItSet

##### API Syntax:

UINT32 **ItSet** (UINT32 index)

##### Function Description:

This function is called when the Host wants to set a value for a control of input terminal.

##### Parameters:

| Type   | Parameter | Description   |
|--------|-----------|---|
| UINT32 | index     | The index of this control in input terminal. Below is possible values:<br>UVC_CT_CONTROL_UNDEFINED<br>UVC_CT_SCANNING_MODE_CONTROL<br>UVC_CT_AE_MODE_CONTROL<br>UVC_CT_AE_PRIORITY_CONTROL<br>UVC_CT_EXPOSURE_TIME_ABSOLUTE_CONTROL<br>UVC_CT_EXPOSURE_TIME_RELATIVE_CONTROL<br>UVC_CT_FOCUS_ABSOLUTE_CONTROL<br>UVC_CT_FOCUS_RELATIVE_CONTROL<br>UVC_CT_FOCUS_AUTO_CONTROL<br>UVC_CT_IRIS_ABSOLUTE_CONTROL<br>UVC_CT_IRIS_RELATIVE_CONTROL<br>UVC_CT_ZOOM_ABSOLUTE_CONTROL<br>UVC_CT_ZOOM_RELATIVE_CONTROL<br>UVC_CT_PANTILT_ABSOLUTE_CONTROL<br>UVC_CT_PANTILT_RELATIVE_CONTROL<br>UVC_CT_ROLL_ABSOLUTE_CONTROL<br>UVC_CT_ROLL_RELATIVE_CONTROL<br>UVC_CT_PRIVACY_CONTROL |



Table 2-220 Returns for AmbaUSB API *ItSet()*.

**Return:**

| Return     | Description |
|------------|-------------|
| 0          | Success     |
| 0xFFFFFFFF | Fail        |

Table 2-221 Returns for AmbaUSB API *ItSet()*.

**Example:**

None

**See Also:**

None



### 2.8.3 AmbaUSBD\_UVC\_BulkSend

#### API Syntax:

```
UINT32 AmbaUSBD_UVC_BulkSend (UINT8 *Buffer, UINT32 Size, UINT32 Timeout)
```

#### Function Description:

This function is used to send data to Host through UVC bulk-in endpoint.

#### Parameters:

| Type   | Parameter | Description                       |
|--------|-----------|-----------------------------------|
| UINT8  | Buffer    | [Input] The buffer to send        |
| UINT32 | Size      | [Input] The data size for sending |
| UINT32 | Timeout   | [Input] Timeout for this API      |

Table 2-222. Parameters for AmbaUSB API **AmbaUSBD\_UVC\_BulkSend()**.

#### Return:

| Return     | Description |
|------------|-------------|
| 0          | Success     |
| 0xFFFFFFFF | Fail        |

Table 2-223. Returns for AmbaUSB API **AmbaUSBD\_UVC\_BulkSend()**.

#### Example:

None

#### See Also:

None



## 2.8.4 AmbaUSBD\_UVC\_BulkSendEx

### API Syntax:

```
UINT32 AmbaUSBD_UVC_BulkSendEx (UINT8 *Buffer, UINT32 Size, UINT32 Timeout,  
UINT32 NoCopy)
```

### Function Description:

This function is used to send data to Host through UVC bulk-in endpoint.

### Parameters:

| Type   | Parameter | Description   |
|--------|-----------|---|
| UINT8  | Buffer    | [Input] The buffer to send  |
| UINT32 | Size      | [Input] The data size for send  |
| UINT32 | Timeout   | [Input] Timeout for this API  |
| UINT32 | NoCopy    | [Input] Indicate whether the data in buffer should be used directly<br>0: The data in buffer is copied before sending.<br>1: The data in buffer is used directly. |

Table 2-224. Parameters for AmbaUSB API **AmbaUSBD\_UVC\_BulkSendEx()**.

### Return:

| Return | Description |
|--------|-------------|
| 0      | Success     |
| others | Fail        |

Table 2-225. Returns for AmbaUSB API **AmbaUSBD\_UVC\_BulkSendEx()**.

### Example:

None

### See Also:

[AmbaUSBD\\_UVC\\_BulkSend\(\)](#)



## 2.8.5 AmbaUSBD\_UVC\_IsoSend

### API Syntax:

```
UINT32 AmbaUSBD_UVC_IsoSend (UINT8 *Buffer, UINT32 Size, UINT32 Timeout,  
UINT32 NoCopy)
```

### Function Description:

This function is used to send data to Host through UVC isochronous-in endpoint.

### Parameters:

| Type   | Parameter | Description  |
|--------|-----------|--|
| UINT8  | Buffer    | [Input] The buffer to send   |
| UINT32 | Size      | [Input] The data size for sending  |
| UINT32 | Timeout   | [Input] Timeout for this API   |
| UINT32 | NoCopy    | [Input] Indicate whether the data in buffer should be used directly<br>0: The data in buffer is copied before send<br>1: The data in buffer is used directly |

Table 2-226. Parameters for AmbaUSB API **AmbaUSBD\_UVC\_IsoSend()**.

### Return:

| Return | Description |
|--------|-------------|
| 0      | Success     |
| others | Fail        |

Table 2-227. Returns for AmbaUSB API **AmbaUSBD\_UVC\_IsoSend()**.

### Example:

None

### See Also:

None



## 2.8.6 AmbaUSBD\_UVC\_GetHostConfig

### API Syntax:

```
UINT32 AmbaUSBD_UVC_GetHostConfig (USBD_UVC_HOST_CONFIG_s *config)
```

### Function Description:

This function is used to get current configurations set by Host.

### Parameters:

| Type                     | Parameter | Description  |
|--------------------------|-----------|--|
| USBD_UVC_HOST_CONFIG_s * | config    | Information on Host configurations.<br>Please refer to Section 2.8.1.2 for more details. |

Table 2-228. Parameters for AmbaUSB API **AmbaUSBD\_UVC\_GetHostConfig()**.

### Return:

| Return     | Description |
|------------|-------------|
| 0          | Success     |
| 0xFFFFFFFF | Fail        |

Table 2-229. Returns for AmbaUSB API **AmbaUSBD\_UVC\_GetHostConfig()**.

### Example:

None

### See Also:

None



## 2.8.7 AmbaUSBD\_UVC\_GetInputTerminalAttribute

### API Syntax:

```
UINT32 AmbaUSBD_UVC_GetInputTerminalAttribute (void)
```

### Function Description:

This function is used to get current attributes of the UVC Input Terminal.

### Parameters:

None

### Return:

| Return | Description   |
|--------|---|
| UINT32 | The bmControls filed in the Input Terminal Descriptor |

Table 2-230. Returns for AmbaUSB API **AmbaUSBD\_UVC\_GetInputTerminalAttribute()**.

### Example:

None

### See Also:

None



## 2.8.8 AmbaUSBD\_UVC\_GetProcessingUnitAttribute

### API Syntax:

```
UINT32 AmbaUSBD_UVC_GetProcessingUnitAttribute (void)
```

### Function Description:

This function is used to get the current attributes of the UVC Processing Unit.

#### Parameters:

None

#### Return:

| Return | Description  |
|--------|--|
| UINT32 | The bmControls filed in the Processing Unit Descriptor |

Table 2-231. Returns for AmbaUSB API **AmbaUSBD\_UVC\_GetProcessingUnitAttribute()**.

#### Example:

None

#### See Also:

None



## 2.8.9 AmbaUSBD\_UVC\_SetInputTerminalAttribute

### API Syntax:

```
UINT32 AmbaUSBD_UVC_SetInputTerminalAttribute (UINT32 attribute)
```

### Function Description:

This function is used to set attributes for UVC Input Terminal.

### Parameters:

| Type   | Parameter | Description   |
|--------|-----------|---|
| UINT32 | attribute | [Input] The bmControls filed in Input Terminal Descriptor |

Table 2-232. Parameters for AmbaUSB API **AmbaUSBD\_UVC\_SetInputTerminalAttribute()**.

### Return:

| Return     | Description |
|------------|-------------|
| 0          | Success     |
| 0xFFFFFFFF | Fail        |

Table 2-233. Returns for AmbaUSB API **AmbaUSBD\_UVC\_SetInputTerminalAttribute()**.

### Example:

None

### See Also:

None



## 2.8.10 AmbaUSBD\_UVC\_SetProcessingUnitAttribute

### API Syntax:

```
UINT32 AmbaUSBD_UVC_SetProcessingUnitAttribute (UINT32 attribute)
```

### Function Description:

This function is used to set attributes for UVC Processing Unit.

### Parameters:

| Type   | Parameter | Description  |
|--------|-----------|--|
| UINT32 | attribute | [Input] The bmControls filed in Processing Unit Descriptor |

Table 2-234. Parameters for AmbaUSB API **AmbaUSBD\_UVC\_SetProcessingUnitAttribute()**.

### Return:

| Return     | Description |
|------------|-------------|
| 0          | Success     |
| 0xFFFFFFFF | Fail        |

Table 2-235. Returns for AmbaUSB API **AmbaUSBD\_UVC\_SetProcessingUnitAttribute()**.

### Example:

None

### See Also:

None



## 2.8.11 AmbaUSBD\_UVC\_RegisterCallback

### API Syntax:

```
UINT32 AmbaUSBD_UVC_RegisterCallback (USBD_UVC_EVENT_CALLBACK_s *cbs)
```

### Function Description:

This function is used to register callback functions for UVC events.

### Parameters:

| Type                        | Parameter | Description   |
|-----------------------------|-----------|---|
| USBD_UVC_EVENT_CALLBACK_s * | cbs       | Information on callback functions.<br>Please refer to Section 2.8.1.1 for more details. |

Table 2-236. Parameters for AmbaUSB API **AmbaUSBD\_UVC\_RegisterCallback()**.

### Return:

| Return     | Description |
|------------|-------------|
| 0          | Success     |
| 0xFFFFFFFF | Fail        |

Table 2-237. Returns for AmbaUSB API **AmbaUSBD\_UVC\_RegisterCallback()**.

### Example:

```
static UINT32 uvc_encode_start(void)
{
    AmbaPrint( "[UVC] Host wants to Start encode!" );
    return TX_SUCCESS;
}

static UINT32 uvc_encode_stop(void)
{
    AmbaPrint( "[UVC] Host wants to Stop encode!" );
    return TX_SUCCESS;
}
```



```
}
```

```
USBD_UVC_EVENT_CALLBACK_S cbs;
```

```
cbs.EncodeStart = uvc_encode_start;
```

```
cbs.EncodeStop = uvc_encode_stop;
```

```
AmbaUSBD_UVC_RegisterCallback (&cbs);
```

**See Also:**

None

For Confidential  
PROTRULY Only



## 2.8.12 AmbaUSBD\_UVC\_GetBrightnessSetting

### API Syntax:

```
UINT32 AmbaUSBD_UVC_GetBrightnessSetting  
(USBD_UVC_PU_CONTROL_SETTING_s *s)
```

### Function Description:

This function is used to get current configurations on brightness control in processing unit.

### Parameters:

| Type                               | Parameter | Description   |
|------------------------------------|-----------|---|
| USBD_UVC_PU_CONTROL_SETTING_s<br>* | s         | Information on control configurations in the processing unit. Please refer to Section 2.8.1.3 for more details. |

Table 2-238. Parameters for AmbaUSB API **AmbaUSBD\_UVC\_GetBrightnessSetting()**.

### Return:

| Return | Description |
|--------|-------------|
| 0      | Success     |
| others | Fail        |

Table 2-239. Returns for AmbaUSB API **AmbaUSBD\_UVC\_GetBrightnessSetting()**.

### Example:

None

### See Also:

None



## 2.8.13 AmbaUSBD\_UVC\_SetBrightnessSetting

### API Syntax:

```
UINT32 AmbaUSBD_UVC_SetBrightnessSetting  
(USBD_UVC_PU_CONTROL_SETTING_s *s)
```

### Function Description:

This function is used to set configurations on brightness control in processing unit. It should be called before **AmbaUSBD\_System\_ClassHook()**.

### Parameters:

| Type                               | Parameter | Description   |
|------------------------------------|-----------|---|
| USBD_UVC_PU_CONTROL_SETTING_s<br>* | s         | Information on control configurations in the processing unit. Please refer to Section 2.8.1.3 for more details. |

Table 2-240. Parameters for AmbaUSB API **AmbaUSBD\_UVC\_SetBrightnessSetting()**.

### Return:

| Return | Description |
|--------|-------------|
| 0      | Success     |
| others | Fail        |

Table 2-241. Returns for AmbaUSB API **AmbaUSBD\_UVC\_SetBrightnessSetting()**.

### Example:

None

### See Also:

None



## 2.8.14 AmbaUSBD\_UVC\_GetSaturationSetting

### API Syntax:

```
UINT32 AmbaUSBD_UVC_GetSaturationSetting  
(USBD_UVC_PU_CONTROL_SETTING_s *s)
```

### Function Description:

This function is used to get current configurations on saturation control in processing unit.

### Parameters:

| Type                               | Parameter | Description   |
|------------------------------------|-----------|---|
| USBD_UVC_PU_CONTROL_SETTING_s<br>* | s         | Information on control configurations in the processing unit. Please refer to Section 2.8.1.3 for more details. |

Table 2-242. Parameters for AmbaUSB API **AmbaUSBD\_UVC\_GetSaturationSetting()**.

### Return:

| Return | Description |
|--------|-------------|
| 0      | Success     |
| others | Fail        |

Table 2-243. Returns for AmbaUSB API **AmbaUSBD\_UVC\_GetSaturationSetting()**.

### Example:

None

### See Also:

None



## 2.8.15 AmbaUSBD\_UVC\_SetSaturationSetting

### API Syntax:

```
UINT32 AmbaUSBD_UVC_SetSaturationSetting  
(USBD_UVC_PU_CONTROL_SETTING_s *s)
```

### Function Description:

This function is used to set configurations on saturation control in processing unit. It should be called before **AmbaUSBD\_System\_ClassHook()**.

### Parameters:

| Type                               | Parameter | Description  |
|------------------------------------|-----------|--|
| USBD_UVC_PU_CONTROL_SETTING_s<br>* | s         | Information on control configurations in processing unit.<br>Please refer to Section 2.8.1.3 for more details. |

Table 2-244. Parameters for AmbaUSB API **AmbaUSBD\_UVC\_SetSaturationSetting()**.

### Return:

| Return | Description |
|--------|-------------|
| 0      | Success     |
| others | Fail        |

Table 2-245. Returns for AmbaUSB API **AmbaUSBD\_UVC\_SetSaturationSetting()**.

### Example:

None

### See Also:

None



## 2.8.16 AmbaUSBD\_UVC\_GetContrastSetting

### API Syntax:

```
UINT32 AmbaUSBD_UVC_GetContrastSetting  
(USBD_UVC_PU_CONTROL_SETTING_s *s)
```

### Function Description:

This function is used to get the current configurations on contrast control in processing unit.

### Parameters:

| Type                               | Parameter | Description  |
|------------------------------------|-----------|--|
| USBD_UVC_PU_CONTROL_SETTING_s<br>* | s         | Information on control configurations in processing unit.<br>Please refer to Section 2.8.1.3 for more details. |

Table 2-246. Parameters for AmbaUSB API **AmbaUSBD\_UVC\_GetContrastSetting()**.

### Return:

| Return | Description |
|--------|-------------|
| 0      | Success     |
| others | Fail        |

Table 2-247. Returns for AmbaUSB API **AmbaUSBD\_UVC\_GetContrastSetting()**.

### Example:

None

### See Also:

None



## 2.8.17 AmbaUSBD\_UVC\_SetContrastSetting

### API Syntax:

```
UINT32 AmbaUSBD_UVC_SetContrastSetting  
(USBD_UVC_PU_CONTROL_SETTING_s *s)
```

### Function Description:

This function is used to set configurations on contrast control in processing unit. It should be called before **AmbaUSBD\_System\_ClassHook()**.

### Parameters:

| Type                               | Parameter | Description  |
|------------------------------------|-----------|--|
| USBD_UVC_PU_CONTROL_SETTING_s<br>* | s         | Information on control configurations in processing unit.<br>Please refer to Section 2.8.1.3 for more details. |

Table 2-248. Parameters for AmbaUSB API **AmbaUSBD\_UVC\_SetContrastSetting()**.

### Return:

| Return | Description |
|--------|-------------|
| 0      | Success     |
| others | Fail        |

Table 2-249. Returns for AmbaUSB API **AmbaUSBD\_UVC\_SetContrastSetting()**.

### Example:

None

### See Also:

None



## 2.8.18 AmbaUSBD\_UVC\_GetHueSetting

### API Syntax:

```
UINT32 AmbaUSBD_UVC_GetHueSetting (USBD_UVC_PU_CONTROL_SETTING_s *s)
```

### Function Description:

This function is used to get current configurations on hue control in the processing unit.

### Parameters:

| Type                               | Parameter | Description   |
|------------------------------------|-----------|---|
| USBD_UVC_PU_CONTROL_SETTING_s<br>* | s         | Information on control configurations in the processing unit. Please refer to Section 2.8.1.3 for more details. |

Table 2-250. Parameters for AmbaUSB API **AmbaUSBD\_UVC\_GetHueSetting()**.

### Return:

| Return | Description |
|--------|-------------|
| 0      | Success     |
| others | Fail        |

Table 2-251. Returns for AmbaUSB API **AmbaUSBD\_UVC\_GetHueSetting()**.

### Example:

None

### See Also:

None



## 2.8.19 AmbaUSBD\_UVC\_SetHueSetting

### API Syntax:

```
UINT32 AmbaUSBD_UVC_SetHueSetting (USBD_UVC_PU_CONTROL_SETTING_s *s)
```

### Function Description:

This function is used to set configurations on hue control in the processing unit. It should be called before `AmbaUSBD_System_ClassHook()`.

### Parameters:

| Type                               | Parameter | Description  |
|------------------------------------|-----------|--|
| USBD_UVC_PU_CONTROL_SETTING_s<br>* | s         | Information on control configurations in processing unit.<br>Please refer to Section 2.8.1.3 for more details. |

Table 2-252. Parameters for AmbaUSB API `AmbaUSBD_UVC_SetHueSetting()`.

### Return:

| Return | Description |
|--------|-------------|
| 0      | Success     |
| others | Fail        |

Table 2-253. Returns for AmbaUSB API `AmbaUSBD_UVC_SetHueSetting()`.

### Example:

None

### See Also:

None



## 2.8.20 AmbaUSBD\_UVC\_GetSharpnessSetting

### API Syntax:

```
UINT32 AmbaUSBD_UVC_GetSharpnessSetting  
(USBD_UVC_PU_CONTROL_SETTING_s *s)
```

### Function Description:

This function is used to get current configurations on sharpness control in the processing unit.

### Parameters:

| Type                            | Parameter | Description   |
|---------------------------------|-----------|---|
| USBD_UVC_PU_CONTROL_SETTING_s * | s         | Information on control configurations in the processing unit. Please refer to Section 2.8.1.3 for more details. |

Table 2-254. Parameters for AmbaUSB API **AmbaUSBD\_UVC\_GetSharpnessSetting()**.

### Return:

| Return | Description |
|--------|-------------|
| 0      | Success     |
| others | Fail        |

Table 2-255. Returns for AmbaUSB API **AmbaUSBD\_UVC\_GetSharpnessSetting()**.

### Example:

None

### See Also:

None



## 2.8.21 AmbaUSBD\_UVC\_SetSharpnessSetting

### API Syntax:

```
UINT32 AmbaUSBD_UVC_SetSharpnessSetting  
(USBD_UVC_PU_CONTROL_SETTING_s *s)
```

### Function Description:

This function is used to set configurations on sharpness control in the processing unit. It should be called before **AmbaUSBD\_System\_ClassHook()**.

### Parameters:

| Type                               | Parameter | Description   |
|------------------------------------|-----------|---|
| USBD_UVC_PU_CONTROL_SETTING_s<br>* | s         | Information on control configurations in processing unit. Please refer to Section 2.8.1.3 for more details. |

Table 2-256. Parameters for AmbaUSB API **AmbaUSBD\_UVC\_SetSharpnessSetting()**.

### Return:

| Return | Description |
|--------|-------------|
| 0      | Success     |
| others | Fail        |

Table 2-257. Returns for AmbaUSB API **AmbaUSBD\_UVC\_SetSharpnessSetting()**.

### Example:

None

### See Also:

None



## 2.8.22 AmbaUSBD\_UVC\_GetPowerLineSetting

### API Syntax:

```
UINT32 AmbaUSBD_UVC_GetPowerLineSetting  
(USBD_UVC_PU_CONTROL_SETTING_s *s)
```

### Function Description:

This function is used to get current configurations on power line frequency control in the processing unit.

### Parameters:

| Type                               | Parameter | Description   |
|------------------------------------|-----------|---|
| USBD_UVC_PU_CONTROL_SETTING_s<br>* | s         | Information on control configurations in the processing unit. Please refer to Section 2.8.1.3 for more details. |

Table 2-258. Parameters for AmbaUSB API **AmbaUSBD\_UVC\_GetPowerLineSetting()**.

### Return:

| Return | Description |
|--------|-------------|
| 0      | Success     |
| others | Fail        |

Table 2-259. Returns for AmbaUSB API **AmbaUSBD\_UVC\_GetPowerLineSetting()**.

### Example:

None

### See Also:

None



## 2.8.23 AmbaUSBD\_UVC\_SetPowerLineSetting

### API Syntax:

```
UINT32 AmbaUSBD_UVC_SetPowerLineSetting  
(USBD_UVC_PU_CONTROL_SETTING_s *s)
```

### Function Description:

This function is used to set configurations on power line frequency control in the processing unit. It should be called before **AmbaUSBD\_System\_ClassHook()**.

### Parameters:

| Type                               | Parameter | Description   |
|------------------------------------|-----------|---|
| USBD_UVC_PU_CONTROL_SETTING_s<br>* | s         | Information on control configurations in the processing unit. Please refer to Section 2.8.1.3 for more details. |

Table 2-260. Parameters for AmbaUSB API **AmbaUSBD\_UVC\_SetPowerLineSetting()**.

### Return:

| Return | Description |
|--------|-------------|
| 0      | Success     |
| others | Fail        |

Table 2-261. Returns for AmbaUSB API **AmbaUSBD\_UVC\_SetPowerLineSetting()**.

### Example:

None

### See Also:

None



## 2.9 Stream Class APIs

Stream Class is a proprietary class defined by Ambarella. It has two interfaces: one is interface 0, and the other is interface 1. Each interface has 2 endpoints: one is bulk-in endpoint, and the other is bulk-out endpoint.

Interface 1 is only available when Host selects interface 1 by "set interface" standard command.

Interface 1 is also called "Native Interface".

For PROTRULY Confidential Only



## 2.9.1 AmbaUSBD\_Stream\_Read

### API Syntax:

```
UINT32 AmbaUSBD_Stream_Read (UINT8 *buffer, UINT32 RequestLength, UINT32  
*ActualLength, ULONG timeout);
```

### Function Description:

This function informs the USB stream class driver to receive data from USB host through interface 0, bulk-out endpoint.

### Parameters:

| Type     | Parameter            | Description                                   |
|----------|----------------------|---|
| UINT8 *  | <b>buffer</b>        | [Input] The buffer for read                   |
| UINT32   | <b>RequestLength</b> | [Input] The data size for read                |
| UINT32 * | <b>ActualLength</b>  | [Output] The actual data size read from Host. |
| ULONG    | <b>timeout</b>       | [Input] Timeout value                         |

Table 2-262. Parameters for AmbaUSB API **AmbaUSBD\_Stream\_Read()**.

### Return:

| Return | Description |
|--------|-------------|
| 0      | Success     |
| 0xFF   | Fail        |

Table 2-263. Returns for AmbaUSB API **AmbaUSBD\_Stream\_Read()**.

### Example:

```
UINT8 buff[10];  
  
UINT32 bytes_transferred;  
  
UINT32 nRet;  
  
nRet = AmbaUSBD_Stream_Read (buff, 10, &bytes_transferred, 500);
```



See Also:

None

For PROTRULY Confidential Only



## 2.9.2 AmbaUSBD\_Stream\_Write

### API Syntax:

```
UINT32 AmbaUSBD_Stream_Write (UINT8 *buffer, UINT32 RequestLength, UINT32  
*ActualLength, ULONG timeout)
```

### Function Description:

This function informs USB stream class driver to send data to the USB host through interface 0, bulk-in endpoint.

### Parameters:

| Type     | Parameter            | Description                                   |
|----------|----------------------|---|
| UINT8 *  | <b>buffer</b>        | [Input] The buffer for write                  |
| UINT32   | <b>RequestLength</b> | [Input] The data size for write               |
| UINT32 * | <b>ActualLength</b>  | [Output] The actual data size written to Host |
| ULONG    | <b>timeout</b>       | [Input] Timeout value                         |

Table 2-264. Parameters for AmbaUSB API **AmbaUSBD\_Stream\_Write()**.

### Return:

| Return | Description |
|--------|-------------|
| 0      | Success     |
| 0xFF   | Fail        |

Table 2-265. Returns for AmbaUSB API **AmbaUSBD\_Stream\_Write()**.

### Example:

```
UINT8 buff[10];  
  
UINT32 bytes_transferred;  
  
UINT32 nRet;  
  
buff[0] = 0x20;  
buff[0] = 0x22;  
buff[0] = 0x28;  
  
nRet = AmbaUSBD_Stream_Write (ptr, 3, &bytes_transferred, 500);
```



See Also:

None

For PROTRULY Confidential Only



### 2.9.3 AmbaUSBD\_Stream\_NativeRead

#### API Syntax:

```
UINT32 AmbaUSBD_Stream_NativeRead (UINT8 *buffer, UINT32 RequestLength,  
UINT32 *ActualLength, ULONG timeout);
```

#### Function Description:

This function informs the USB stream class driver to receive data from the USB host through interface 1, bulk-out endpoint.

#### Parameters:

| Type     | Parameter            | Description                                  |
|----------|----------------------|--|
| UINT8 *  | <b>buffer</b>        | [Input] The buffer for read                  |
| UINT32   | <b>RequestLength</b> | [Input] The data size for read               |
| UINT32 * | <b>ActualLength</b>  | [Output] The actual data size read from Host |
| ULONG    | <b>timeout</b>       | [Input] Timeout value                        |

Table 2-266. Parameters for AmbaUSB API **AmbaUSBD\_Stream\_NativeRead()**.

#### Return:

| Return | Description |
|--------|-------------|
| 0      | Success     |
| 0xFF   | Fail        |

Table 2-267. Returns for AmbaUSB API **AmbaUSBD\_Stream\_NativeRead()**.

#### Example:

```
UINT8 buff[10];  
  
UINT32 bytes_transferred;  
  
UINT32 nRet;  
  
nRet = AmbaUSBD_Stream_NativeRead (buff, 10, &bytes_transferred, 500);
```



See Also:

None

For PROTRULY Confidential Only



## 2.9.4 AmbaUSBD\_Stream\_NativeWrite

### API Syntax:

```
UINT32 AmbaUSBD_Stream_NativeWrite (UINT8 *buffer, UINT32 RequestLength,  
UINT32 *ActualLength, ULONG timeout)
```

### Function Description:

This function informs the USB stream class driver to send data to the USB host through interface 1, bulk-in endpoint.

### Parameters:

| Type     | Parameter            | Description                                   |
|----------|----------------------|---|
| UINT8 *  | <b>buffer</b>        | [Input] The buffer for write                  |
| UINT32   | <b>RequestLength</b> | [Input] The data size for write               |
| UINT32 * | <b>ActualLength</b>  | [Output] The actual data size written to Host |
| ULONG    | <b>timeout</b>       | [Input] Timeout value                         |

Table 2-268. Parameters for AmbaUSB API **AmbaUSBD\_Stream\_NativeWrite()**.

### Return:

| Return | Description |
|--------|-------------|
| 0      | Success     |
| 0xFF   | Fail        |

Table 2-269. Returns for AmbaUSB API **AmbaUSBD\_Stream\_NativeWrite()**.

### Example:

```
UINT8 buff[10];  
  
UINT32 bytes_transferred;  
  
UINT32 nRet;  
  
buff[0] = 0x20;  
buff[0] = 0x22;  
buff[0] = 0x28;  
  
nRet = AmbaUSBD_Stream_NativeWrite (ptr, 3, &bytes_transferred, 500);
```



See Also:

None

For PROTRULY Confidential Only



### 3. USB Host APIs

This chapter describes APIs for the USB host mode.

For PROTRULY Confidential Only



### 3.1 System and Flow Control APIs

This section describes APIs for system and flow control.

For PROTRULY Confidential Only



### 3.1.1 Data Structures

#### 3.1.1.1 USB\_HOST\_SYSTEM\_INIT\_s

| Type    | Parameter                  | Description  |
|---------|----------------------------|--|
| UINT8 * | <b>MemPoolPtr</b>          | [Input] Memory for USBX Host driver stack                      |
| UINT32  | <b>TotalMemSize</b>        | [Input] Total memory size for USBX host system stack           |
| UINT32  | <b>EnumThreadStackSize</b> | [Input] Memory size for the USBX host enumeration thread stack |
| UINT32  | <b>EnumThreadPriority</b>  | [Input] Priority for USBX host enumeration thread stack        |
| UINT32  | <b>HcdThreadStackSize</b>  | [Input] Memory size for USBX host controller thread stack      |
| UINT32  | <b>HcdThreadPriority</b>   | [Input] Priority for the USBX host controller thread stack     |

Table 3-1 Definition of **USB\_HOST\_SYSTEM\_INIT\_s**.

#### 3.1.1.2 USB\_HOST\_CLASS\_INIT\_s

| Type        | Parameter                 | Description  |
|-------------|---------------------------|--|
| UHC_CLASS_e | <b>classID</b>            | [Input] USB Host Class ID:<br><b>UHC_CLASS_NONE</b> - No USB class<br><b>UHC_CLASS_HID</b> - HID class<br><b>UHC_CLASS_STORAGE</b> - Storage class |
| UINT32      | <b>ClassTaskStackSize</b> | [Input] Memory size for host class thread stack  |
| UINT32      | <b>ClassTaskPriority</b>  | [Input] Priority for host class thread stack   |

Table 3-2 Definition of **USB\_HOST\_CLASS\_INIT\_s**.



### 3.1.1.3 **USB0\_PORT\_OWNER\_e**

| Type              | Parameter      | Description   |
|-------------------|----------------|---|
| USB0_PORT_OWNER_e | <b>classID</b> | USB0 owner select:<br><b>UDC_OWN_PORT</b> – Device owns<br>USBPHY0<br><b>UHC_OWN_PORT</b> – Host owns USBPHY0 |

Table 3-3 *Definition of **USB0\_PORT\_OWNER\_e**.*

### 3.1.1.4 **USB\_HOST\_STORAGE\_INFO\_s**

| Type   | Parameter         | Description  |
|--------|-------------------|--|
| UINT32 | <b>present</b>    | [Input] Checks if the storage media is present<br>0: Not present<br>1: Present |
| INT32  | <b>format</b>     | [Input] File system format of storage media                                    |
| UINT32 | <b>lun</b>        | [Input] Logic unit number  |
| UINT32 | <b>SectorSize</b> | [Input] Sector size in byte  |
| UINT32 | <b>lba</b>        | [Input] Logic block addressing   |
| UINT32 | <b>wp</b>         | [Input] Write protection<br>0: Non-write protection<br>1: Write protection     |

Table 3-4 *Definition of **USB\_HOST\_STORAGE\_INFO\_s**.*



### 3.1.2 AmbaUSB\_HostSystemSetup

#### API Syntax:

```
UINT32 AmbaUSB_HostSystemSetup (USB_HOST_SYSTEM_INIT_s *config)
```

#### Function Description:

This function initializes the USBX host system and the driver stack.

#### Parameters:

| Type                     | Parameters | Description  |
|--------------------------|------------|--|
| USB_HOST_SYSTEM_INIT_s * | config     | Define the USB host system Initial parameters. Please refer to Section 3.1.1.1 for more details. The priority of the thread of host controller and enumeration should be no lower than the class thread. |

Table 3-5 Parameters for AmbaUSB API **AmbaUSB\_HostSystemSetup()**.

#### Return:

| Return | Description |
|--------|-------------|
| 0      | Success     |
| 0xFF   | Failure     |

Table 3-6 Returns for AmbaUSB API **AmbaUSB\_HostSystemSetup()**.

#### Example:

```
USB_HOST_SYSTEM_INIT_s Sysconfig = {0};  
  
/* Initialization of USB Host Class */  
Sysconfig.EnumThreadStackSize = USB_HOST_ENUM_THREAD_STACK_SIZE;  
Sysconfig.EnumThreadPriority = USB_HOST_ENUM_THREAD_PRIORITY;  
Sysconfig.HcdThreadStackSize = USB_HOST_HCD_THREAD_STACK_SIZE;  
Sysconfig.HcdThreadPriority = USB_HOST_HCD_THREAD_PRIORITY;  
AmbaUSB_HostSystemSetup (&Sysconfig);
```



See Also:

None

For PROTRULY Confidential Only



### 3.1.3 AmbaUSBH\_System\_ClassHook

#### API Syntax:

```
UINT32 AmbaUSBH_System_ClassHook (USB_HOST_CLASS_INIT_s *config)
```

#### Function Description:

This function is used for initialization of the USBX host class client software.

#### Parameters:

| Type                    | Parameters | Description  |
|-------------------------|------------|--|
| USB_HOST_CLASS_INIT_s * | config     | Define USB host class Initial parameters.<br>Please refer to Section 3.1.1.2 for more details. Note that the priority of the class thread should not be higher than the host controller and enumeration. |

Table 3-7 Parameters for AmbaUSB API **AmbaUSBH\_System\_ClassHook()**.

#### Return:

| Return | Description |
|--------|-------------|
| 0      | Success     |
| 0xFF   | Failure     |

Table 3-8 Returns for AmbaUSB API **AmbaUSBH\_System\_ClassHook()**.

#### Example:

```
USB_HOST_CLASS_INIT_s Classconfig = {UHC_CLASS_NONE, 0, 0};  
  
Classconfig.classID = UHC_CLASS_STORAGE;  
  
Classconfig.ClassTaskPriority =50;  
  
Classconfig.ClassTaskStackSize =  
    AmbaUSB_Host_ClassCtrl[UsbHostClass].TaskStackSize;  
  
AmbaUSBH_System_ClassHook (&Classconfig);
```

#### See Also:

None



### 3.1.4 AmbaUSBH\_System\_SetPhy0Owner

#### API Syntax:

```
void AmbaUSBH_System_SetPhy0Owner (USB0_PORT_OWNER_e owner)
```

#### Function Description:

USBPHY0 can only route to either the host or device controller. Application can use this API to switch the owner of USBPHY0. Note that the related hardware setup should be done before switching the owner.

Note that not every Ambarella chip supports multiple USB PHY.

Below is the A12 block diagram:

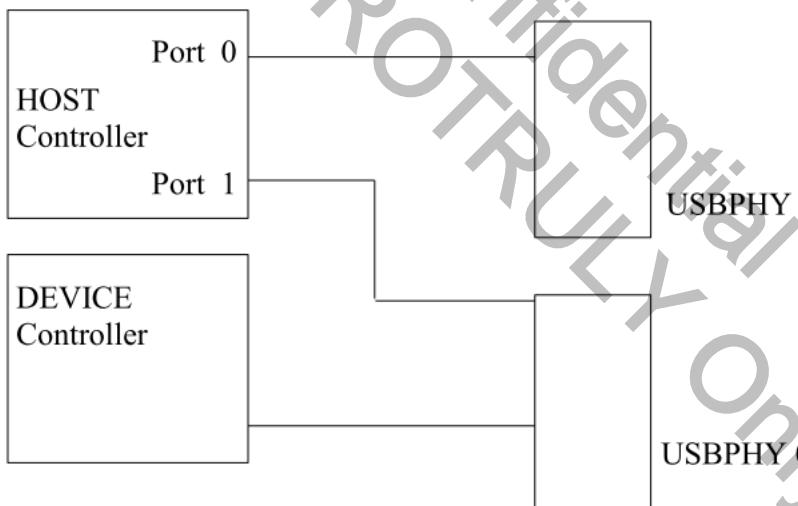


Figure 3-1 A12 Block Diagram.



Below is the A9 block diagram:

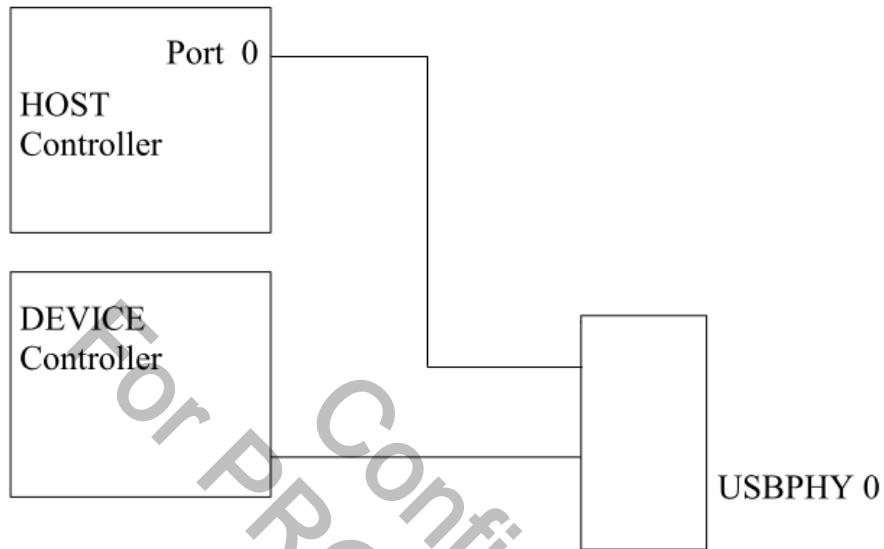


Figure 3-2 A9 Block Diagram.

**Parameters:**

| Type              | Parameters | Description  |
|-------------------|------------|--|
| USB0_PORT_OWNER_e | owner      | Define the owner of USBPHY0. Please refer to Section 3.1.1.3 for more details. |

Table 3-9 Parameters for AmbaUSB API **AmbaUSBH\_System\_SetPhy0Owner()**.

**Return:**

None

**Example:**

```
switch (owner) {  
    case UDC_OWN_PORT:  
        AmbaUSBH_System_SetPhy0Owner (owner);  
        break;  
    case UHC_OWN_PORT:  
        AmbaUSBH_System_SetPhy0Owner (owner);  
}
```



```
        break;  
  
    default:  
        AmbaPrint("Wrong owner = 0x%x", owner);  
        break;  
    }  
  
See Also:  
None
```

For PROTRULY Confidential Only



### 3.1.5 AmbaUSBH\_System\_SetUsbOwner

#### API Syntax:

```
void AmbaUSBH_System_SetUsbOwner (USB IRQ OWNER_e owner, int update)
```

#### Function Description:

This function is used to set the USB Host IRQ owner. Because there might be more than one operating system running on one Ambarella chip and only one of them can access USB at the same time, the application must call this function to assign USB owner to one of these operating systems.

Note that this function can only set the USB host IRQ owner to RTOS. To set the USB host IRQ owner to others, please refer to corresponding API documents.

#### Parameters:

| Type                 | Parameter     | Description  |
|----------------------|---------------|--|
| USB_HOST_IRQ_OWNER_e | <b>owner</b>  | [Input]<br>USB_HOST_IRQ_RTOS - IRQ owner is RTOS (ThreadX.)                                |
| int                  | <b>update</b> | [Input]<br>0: Disable IRQ after owner is switched<br>1: Enable IRQ after owner is switched |

Table 3-10. Parameters for AmbaUSB API **AmbaUSBH\_System\_SetUsbOwner()**.

#### Return:

None

#### Example:

None

#### See Also:

None



### 3.1.6 AmbaUSBH\_System\_ClassUnHook

#### API Syntax:

```
UINT32 AmbaUSBH_System_ClassUnHook (UHC_CLASS_e ClassID)
```

#### Function Description:

This function is used for release of the USBX host class client software.

#### Parameters:

| Type        | Parameters | Description  |
|-------------|------------|--|
| UHC_CLASS_e | classID    | [Input] USB Host Class ID:<br><b>UHC_CLASS_NONE</b> - No USB class<br><b>UHC_CLASS_HID</b> - HID class<br><b>UHC_CLASS_STORAGE</b> - Storage class |

Table 3-11 Parameters for AmbaUSB API **AmbaUSBH\_System\_ClassUnHook()**.

#### Return:

| Return | Description |
|--------|-------------|
| 0      | Success     |
| others | Failure     |

Table 3-12 Returns for AmbaUSB API **AmbaUSBH\_System\_ClassUnHook()**.

#### Example:

```
USB_HOST_CLASS_INIT_s Classconfig = {UHC_CLASS_NONE, 0, 0};  
Classconfig.classID = UHC_CLASS_STORAGE;  
Classconfig.ClassTaskPriority =50;  
Classconfig.ClassTaskStackSize =  
    AmbaUSB_Host_ClassCtrl[UsbHostClass].TaskStackSize;  
AmbaUSBH_System_ClassUnHook (&Classconfig);
```

#### See Also:

None



### 3.1.7 AmbaUSBH\_System\_SetMemoryPool

#### API Syntax:

```
UINT32 AmbaUSBH_System_SetMemoryPool (TX_BYTE_POOL *cached_pool,  
TX_BYTE_POOL *noncached_pool)
```

#### Function Description:

This function is used to set the memory pool for USB Host system. The USB Host system will allocate memory from the memory pool when running.

It should be called before calling `AmbaUSB_HostSystemSetup()`.

#### Parameters:

| Type           | Parameter             | Description                   |
|----------------|-----------------------|-------------------------------|
| TX_BYTE_POOL * | <b>cached_pool</b>    | [Input] cached memory pool    |
| TX_BYTE_POOL * | <b>noncached_pool</b> | [Input] noncached memory pool |

Table 3-13. Returns for AmbaUSB API `AmbaUSBH_System_SetMemoryPool()`.

#### Return:

| Return | Description |
|--------|-------------|
| 0      | Success     |
| others | Failure     |

Table 3-14. Returns for AmbaUSB API `AmbaUSBH_System_SetMemoryPool()`.

#### Example:

None

#### See Also:

None



### 3.1.8 AmbaUSBH\_System\_SetIsoSlopDelay

#### API Syntax:

```
void AmbaUSBH_System_SetIsoSlopDelay (UINT32 slop)
```

#### Function Description:

The application calls this API to change the slop value which is used to decide the delay times to start the isochronous transfer. For example, if the application starts a transfer request at time n ms, the USB hardware starts the transfer at (n+slop) ms. The application adjusts this value depending on the transfer length and the system performance. The slop value is between 10 and 1023.

#### Parameters:

| Type   | Parameters | Description |
|--------|------------|-------------|
| UINT32 | slop       | Slop value. |

Table 3-15 Parameters for AmbaUSB API *AmbaUSBH\_System\_SetIsoSlopDelay()*.

#### Return:

None

#### Example:

None

#### See Also :

None



## 3.2 Storage Class APIs

This section describes the APIs for Mass Storage Class.

For PROTRULY Confidential Only



### 3.2.1 AmbaUSBH\_Storage\_SetSlotInfo

#### API Syntax:

```
void AmbaUSBH_Storage_SetSlotInfo (UINT32 slot)
```

#### Function Description:

This function assigns the slot for the USB host storage media. The application must call this API before invoking the **AmbaUSBH\_ClassHook()**.

#### Parameters:

| Type   | Parameters | Description  |                         |        |
|--------|------------|--|-------------------------|--------|
| UINT32 | slot       | Currently application must set it as SCM_SLOT_USB. |                         |        |
|        |            | SCM_SLOT_FL0                                       | 0 /* NAND Flash:        | (A) */ |
|        |            | SCM_SLOT_FL1                                       | 1 /* NAND Flash:        | (B) */ |
|        |            | SCM_SLOT_SD0                                       | 2 /* SD/MMC controller: | (C) */ |
|        |            | SCM_SLOT_SD1                                       | 3 /* SD/MMC controller: | (D) */ |
|        |            | SCM_SLOT_RD  | 4 /* Ramdisk:           | (E) */ |
|        |            | SCM_SLOT_IDX                                       | 5 /* Media Partition:   | (F) */ |
|        |            | SCM_SLOT_PRF                                       | 6 /* Media Partition:   | (G) */ |
|        |            | SCM_SLOT_CAL                                       | 7 /* Media Partition:   | (H) */ |
|        |            | SCM_SLOT_USB                                       | 8 /* USB HOST:          | (I) */ |
|        |            | SCM_SLOT_SDIO                                      | 11 /* SD/SDIO:          | (L) */ |
|        |            | SCM_SLOT_RF1                                       | 24 /* DSP-ROMFS:        | (Y) */ |
|        |            | SCM_SLOT_RF2                                       | 25 /* ROM-ROMFS:        | (Z) */ |

Table 3-16 Parameters for AmbaUSB API **AmbaUSBH\_Storage\_SetSlotInfo()**.

#### Return:

None



**Example:**

```
USB_HOST_SYSTEM_INIT_s Sysconfig = {0};  
USB_HOST_CLASS_INIT_s Classconfig = {UHC_CLASS_NONE, 0, 0};  
  
/* Initialization of USB Host Class */  
Sysconfig.EnumThreadStackSize = USB_HOST_ENUM_THREAD_STACK_SIZE;  
Sysconfig.EnumThreadPriority = USB_HOST_ENUM_THREAD_PRIORITY;  
Sysconfig.HcdThreadStackSize = USB_HOST_HCD_THREAD_STACK_SIZE;  
Sysconfig.HcdThreadPriority = USB_HOST_HCD_THREAD_PRIORITY;  
AmbaUSB_HostSystemSetup(&Sysconfig);  
  
/* Hook the class client software */  
if (UsbHostClass == AMBA_USB_HOST_CLASS_STORAGE) {  
    AmbaUSBH_Storage_SetSlotInfo(SCM_SLOT_USB);  
}  
  
Classconfig.classID = UHC_CLASS_STORAGE;  
Classconfig.ClassTaskPriority = 50;  
Classconfig.ClassTaskStackSize =  
    AmbaUSB_Host_ClassCtrl[UsbHostClass].TaskStackSize;  
AmbaUSBH_ClassHook(&Classconfig);
```

**See Also:**

None



### 3.2.2 AmbaUSBH\_Storage\_GetStatus

#### API Syntax:

```
USB_HOST_STORAGE_INFO_s* AmbaUSBH_Storage_GetStatus (int UsbMedia)
```

#### Function Description:

The application calls this API to get the current storage media information.

#### Parameters:

| Type | Parameters | Description   |
|------|------------|---|
| int  | UsbMedia   | Currently only one USB storage media is supported. The application must set it as 0 to get the storage media. |

Table 3-17 Parameters for AmbaUSB API **AmbaUSBH\_Storage\_GetStatus()**.

#### Return:

| Return                   | Parameters | Description  |
|--------------------------|------------|--|
| USB_HOST_STORAGE_INFO_s* | UsbMedia   | [Output] Information of current USB storage media. Please refer to Section 3.1.1.4 for more details. |

Table 3-18 Returns for AmbaUSB API **AmbaUSBH\_Storage\_GetStatus()**.



**Example:**

```
USB_HOST_STORAGE_INFO_s *pUsbInfo;

pUsbInfo = AmbaUSBH_Storage_GetStatus ( 0 );
if (pUsbInfo == NULL) {
    return -1;
}
pStatus->CardPresent      = pUsbInfo->present;
pStatus->Format           = pUsbInfo->format;;
pStatus->SecSize          = pUsbInfo->SectorSize;
pStatus->WriteProtect     = pUsbInfo->wp;
```

**See Also :**

None



### 3.2.3 AmbaUSBH\_Storage\_Read

#### API Syntax:

```
int AmbaUSBH_Storage_Read (int ID, UINT8 *pBuf, UINT32 Sec, UINT32 Secs)
```

#### Function Description:

The application calls this API to read the storage media data.

#### Parameters:

| Type   | Parameters | Description  |
|--------|------------|--|
| int    | UsbMedia   | [Input] Currently only one USB storage media is supported. Application must set it as 0 for the storage media. |
| UINT8  | *pBuf      | [Input] The buffer for reading data from the USB media   |
| UINT32 | Sec        | [Input] The data offset and the unit is "sector".  |
| UINT32 | Secs       | [Input] The data size and the unit is "sector".  |

Table 3-19 Parameters for AmbaUSB API **AmbaUSBH\_Storage\_Read()**.

#### Return:

| Return     | Description |
|------------|-------------|
| 0          | Success     |
| 0xFFFFFFFF | Failure     |

Table 3-20 Returns for AmbaUSB API **AmbaUSBH\_Storage\_Read()**.

#### Example:

```
AmbaUSBH_Storage_Read (0, 0xc0100000, 0xe100, 8);
```

#### See Also:

None



### 3.2.4 AmbaUSBH\_Storage\_Write

#### API Syntax:

```
int AmbaUSBH_Storage_Write (int ID, UINT8 *pBuf, UINT32 Sec, UINT32 Secs)
```

#### Function Description:

The application calls this API to write storage media data.

#### Parameters:

| Type   | Parameters | Description  |
|--------|------------|--|
| int    | UsbMedia   | [Input] Currently only one USB storage media is supported. The application must set it as 0 for the storage media. |
| UINT8  | *pBuf      | [Input] The buffer containing data for writing to the USB media.   |
| UINT32 | Sec        | [Input] The data offset and the unit is "sector".  |
| UINT32 | Secs       | [Input] The data size and the unit is "sector".  |

Table 3-21 Parameters for AmbaUSB API **AmbaUSBH\_Storage\_Write()**.

#### Return:

| Return     | Description |
|------------|-------------|
| 0          | Success     |
| 0xFFFFFFFF | Failure     |

Table 3-22 Returns for AmbaUSB API **AmbaUSBH\_Storage\_Write()**.

#### Example:

```
AmbaUSBH_Storage_Write (0, 0xc0100000, 0xe100, 8);
```

#### See Also:

None



### 3.2.5 AmbaUSBH\_Msc\_SetMainTaskInfo

#### API Syntax:

```
UINT32 AmbaUSBH_Msc_SetMainTaskInfo (UHC_TASKINFO_s *info)
```

#### Function Description:

This function is used to set the information for creating MSC main task. It should be called before **AmbaUSBH\_System\_ClassHook()**.

#### Parameters:

| Type             | Parameter | Description   |
|------------------|-----------|---|
| UHC_TASKINFO_s * | info      | [Input] Task information. Please refer to Section 2.1.1.6 for more details. |

Table 3-23. Parameters for AmbaUSB API **AmbaUSBH\_Msc\_SetMainTaskInfo()**.

#### Return:

| Return     | Description |
|------------|-------------|
| 0          | Success     |
| 0xFFFFFFFF | Failure     |

Table 3-24. Returns for AmbaUSB API **AmbaUSBH\_Msc\_SetMainTaskInfo()**.

#### Example:

```
// set the stack size to 10KB, priority to 70, and AffinityMask to CORE
0.

UHC_TASKINFO_s ti;
ti.StackSize = 1024*10;
ti.Priority = 70;
ti.AffinityMask = 0x01;
AmbaUSBH_Msc_SetMainTaskInfo (&ti);
```

#### See Also:

None



### 3.2.6 AmbaUSBH\_Msc\_GetMainTaskInfo

#### API Syntax:

```
UINT32 AmbaUSBH_Msc_GetMainTaskInfo (UHC_TASKINFO_s *info)
```

#### Function Description:

This function is used to get the information for creating MSC main task.

#### Parameters:

| Type             | Parameter | Description  |
|------------------|-----------|--|
| UHC_TASKINFO_s * | info      | [Output] Task information. Please refer to Section 2.1.1.6 for more details. |

Table 3-25. Parameters for AmbaUSB API **AmbaUSBH\_Msc\_GetMainTaskInfo()**.

#### Return:

| Return     | Description |
|------------|-------------|
| 0          | Success     |
| 0xFFFFFFFF | Failure     |

Table 3-26. Returns for AmbaUSB API **AmbaUSBH\_Msc\_GetMainTaskInfo()**.

#### Example:

```
UHC_TASKINFO_s ti;  
AmbaUSBH_Msc_GetMainTaskInfo (&ti);
```

#### See Also:

None



### 3.3 MTP Class APIs

This section describes the APIs for MTP Class.

For PROTRULY Confidential Only



### 3.3.1 Data Structures

#### 3.3.1.1 USBH\_MTP\_SESSION

| Type                     | Parameter             | Description  |
|--------------------------|-----------------------|--|
| ULONG                    | <b>magic</b>          | Magic number   |
| ULONG                    | <b>id</b>             | MTP Session ID   |
| ULONG                    | <b>state</b>          | State of this session  |
| UX_HOST_CL<br>ASS_PIMA * | <b>pima_instance</b>  | MTP class instance   |
| ULONG                    | <b>nb_storage_ids</b> | Number of Storage IDs  |
| ULONG                    | <b>nb_objects</b>     | Number of objects in this session  |
| Function<br>Pointer      | <b>event_callback</b> | Callback function for MTP events sent by the device. Please refer to Section 3.4.2.3 for more details. |

Table 3-27 Definition of **USBH\_MTP\_SESSION**.

#### 3.3.1.2 USBH\_MTP\_DEVICE

This data structure maps DeviceInfo Dataset defined in MTP specification.

| Type  | Parameter                         | Description                                  |
|-------|-----------------------------------|--|
| ULONG | <b>standard_version</b>           | Standard Version                             |
| ULONG | <b>vendor_extension_id</b>        | MTP Vendor Extension ID                      |
| ULONG | <b>vendor_extension_version</b>   | MTP Version                                  |
| UCHAR | <b>vendor_extension_desc[256]</b> | MTP Extension string. It is an ASCII string. |
| ULONG | <b>functional_mode</b>            | MTP Functional Mode                          |
| UCHAR | <b>operations_supported[256]</b>  | Supported Operations                         |
| UCHAR | <b>events_supported[256]</b>      | Supported Events                             |
| UCHAR | <b>properties_supported[256]</b>  | Supported Properties                         |



| UCHAR | <b>capture_formats[256]</b> | Supported Capture Formats                     |
|-------|-----------------------------|---|
| UCHAR | <b>image_formats[256]</b>   | Supported Image Formats                       |
| UCHAR | <b>manufacturer[256]</b>    | Manufacturer string. It is an ASCII string.   |
| UCHAR | <b>model[256]</b>           | Model string. It is an ASCII string.          |
| UCHAR | <b>version[256]</b>         | Device Version string. It is an ASCII string. |
| Type  | Parameter                   | Description                                   |
| UCHAR | <b>serial_number[256]</b>   | Serial Number string. It is an ASCII string.  |

Table 3-28 Definition of **USBH\_MTP\_DEVICE**.

### 3.3.1.3 USBH\_MTP\_STORAGE

This data structure maps StorageInfo Dataset defined in MTP specification.

| Type  | Parameter                    | Description                                 |
|-------|------------------------------|---|
| ULONG | <b>type</b>                  | Storage type                                |
| ULONG | <b>file_system_type</b>      | FileSystem type                             |
| ULONG | <b>access_capability</b>     | Access capability                           |
| ULONG | <b>max_capacity_low</b>      | Maximum space in bytes                      |
| ULONG | <b>max_capacity_high</b>     |   |
| ULONG | <b>free_space_bytes_low</b>  | Free space in bytes                         |
| ULONG | <b>free_space_bytes_high</b> |   |
| ULONG | <b>free_space_images</b>     | Free space in objects                       |
| UCHAR | <b>description[256]</b>      | Storage description. It is an ASCII string. |
| UCHAR | <b>volume_label[256]</b>     | Volume label. It is an ASCII string.        |

Table 3-29 Definition of **USBH\_MTP\_STORAGE**.

### 3.3.1.4 USBH\_MTP\_OBJECT

This data structure maps ObjectInfo Dataset defined in MTP specification.

| Type  | Parameter         | Description |
|-------|-------------------|-------------|
| ULONG | <b>storage_id</b> | Storage ID  |



| Type    | Parameter                     | Description                                |
|---------|-------------------------------|--|
| ULONG   | <b>format</b>                 | Format code                                |
| ULONG   | <b>protection_status</b>      | Protection status                          |
| ULONG   | <b>compressed_size</b>        | Object size                                |
| ULONG   | <b>thumb_format</b>           | Thumb format code                          |
| ULONG   | <b>thumb_compressed_size</b>  | Thumbnail size                             |
| ULONG   | <b>thumb_pix_width</b>        | Thumbnail width                            |
| ULONG   | <b>thumb_pix_height</b>       | Thumbnail height                           |
| ULONG   | <b>image_pix_width</b>        | Image object width                         |
| ULONG   | <b>image_pix_height</b>       | Image object height                        |
| ULONG   | <b>image_bit_depth</b>        | Image object depth                         |
| Type    | Parameter                     | Description                                |
| ULONG   | <b>parent_object</b>          | Parent object handle                       |
| ULONG   | <b>association_type</b>       | Association type                           |
| ULONG   | <b>association_desc</b>       | Association description                    |
| ULONG   | <b>sequence_number</b>        | Sequence number                            |
| UCHAR   | <b>filename[256]</b>          | Filename. It is an ASCII string.           |
| UCHAR   | <b>capture_date[256]</b>      | Image capture date. It is an ASCII string. |
| UCHAR   | <b>modification_date[256]</b> | Modification date. It is an ASCII string.  |
| UCHAR   | <b>keywords[256]</b>          | Keyword. It is an ASCII string.            |
| ULONG   | <b>state</b>                  | For internal use only                      |
| ULONG   | <b>offset</b>                 | For internal use only                      |
| ULONG   | <b>transfer_status</b>        | For internal use only                      |
| ULONG   | <b>handle_id</b>              | For internal use only                      |
| ULONG   | <b>length</b>                 | For internal use only                      |
| UCHAR * | <b>buffer</b>                 | For internal use only                      |

Table 3-30 Definition of **USBH\_MTP\_OBJECT**.

### 3.3.1.5 USBH\_MTP\_COMMAND

| Type  | Parameter             | Description          |
|-------|-----------------------|----------------------|
| ULONG | <b>nb_parameters</b>  | Number of parameters |
| ULONG | <b>operation_code</b> | MTP operation code   |



| Type  | Parameter     | Description                         |
|-------|---------------|-------------------------------------|
| ULONG | parameter_1   | Parameter 1 for command or response |
| ULONG | parameter_2   | Parameter 2 for command or response |
| ULONG | parameter_3   | Parameter 3 for command or response |
| ULONG | parameter_4   | Parameter 4 for command or response |
| ULONG | parameter_5   | Parameter 5 for command or response |
| ULONG | response_code | MTP response code                   |

Table 3-31 Definition of **USBH\_MTP\_COMMAND**.

### 3.3.1.6 USBH\_MTP\_VENDOR\_PAYLOAD

| Type  | Parameter | Description                      |
|-------|-----------|----------------------------------|
| ULONG | size      | Data size for the vendor command |
| ULONG | offset    | Current data offset              |

Table 3-32 Definition of **USBH\_MTP\_VENDOR\_PAYLOAD**.

### 3.3.1.7 USBH\_MTP\_EVENT

| Type                          | Parameter      | Description             |
|-------------------------------|----------------|-------------------------|
| USBH_MTP_SESSION *            | session        | MTP session information |
| UX_HOST_CLASS_PIMA_ST<br>RUCT | instance       | MTP class instance      |
| ULONG                         | code           | Event code              |
| ULONG                         | session_id     | Session ID              |
| ULONG                         | transaction_id | Transaction ID          |
| ULONG                         | parameter_1    | Parameter 1             |
| ULONG                         | parameter_2    | Parameter 2             |
| ULONG                         | parameter_3    | Parameter 3             |

Table 3-33 Definition of **USBH\_MTP\_EVENT**.



### 3.3.1.8 **USB\_HOST\_MTP\_CB\_s**

| Type             | Parameter          | Description  |
|------------------|--------------------|--|
| Function pointer | <b>MediaInsert</b> | Callback function for media insertion. Please refer to Section 3.4.2.1 for more details. |
| Function pointer | <b>MediaRemove</b> | Callback function for media removal. Please refer to Section 3.4.2.2 for more details.   |

Table 3-34 Definition of **USB\_HOST\_MTP\_CB\_s**.



### 3.3.2 Callback functions

#### 3.3.2.1 MediaInsert

##### API Syntax:

void **MedianInsert** (void)

##### Function Description:

This function is called when a MTP device is inserted and recognized.

##### Parameters:

None

##### Return:

None

##### Example:

None

##### See Also:

None

#### 3.3.2.2 MediaRemove

##### API Syntax:

void **MediaRemove** (void)

##### Function Description:

This function is called when a MTP device is removed.



**Parameters:**

None

**Return:**

None

**Example:**

None

**See Also:**

None



### 3.3.2.3 EventCallback

#### API Syntax:

```
void EventCallback (UX_HOST_CLASS_PIMA_EVENT *event)
```

#### Function Description:

This function is called when a MTP event is received from the device.

#### Parameters:

| Type             | Parameters | Description  |
|------------------|------------|--|
| USBH_MTP_EVENT * | event      | [Input] MTP event information. Please refer to Section 3.4.1.7 for more details. |

#### Return:

None

#### Example:

None

#### See Also:

[AmaUSBH\\_Mtp\\_SessionOpen\(\)](#)



### 3.3.3 AmbaUSBH\_Mtp\_SessionOpen

#### API Syntax:

```
UINT32 AmbaUSBH_Mtp_SessionOpen (USBH_MTP_SESSION *Session)
```

#### Function Description:

This function issues an **OpenSession** command to the MTP device.

#### Parameters:

| Type               | Parameters | Description  |
|--------------------|------------|--|
| USBH_MTP_SESSION * | Session    | [Output] MTP session information. Please refer to Section 3.4.1.1 for more details. Applications should keep it and pass it to other MTP APIs. |

Table 3-35 Parameters for AmbaUSB API **AmbaUSBH\_Mtp\_SessionOpen()**.

#### Return:

| Return | Description |
|--------|-------------|
| 0      | Success     |
| others | Failure     |

Table 3-36 Returns for AmbaUSB API **AmbaUSBH\_Mtp\_SessionOpen()**.

#### Example:

```
static void AppUsbh_MtpTestEventCallback(USBH_MTP_EVENT *event)
{
    AmbaPrint("Event 0x%x Happen!", event->code);
    AmbaPrint("session id = 0x%x", event->session_id);
    AmbaPrint("transaction id = 0x%x", event->transaction_id);
    AmbaPrint("p1 = 0x%x", event->parameter_1);
    AmbaPrint("p2 = 0x%x", event->parameter_2);
    AmbaPrint("p3 = 0x%x", event->parameter_3);
    return;
}
```



```
UINT32 rval = 0;  
  
static USBH_MTP_SESSION AmbaUsbhMtpSession;  
  
memset(&AmbaUsbhMtpSession, 0, sizeof(USBH_MTP_SESSION));  
AmbaUsbhMtpSession.event_callback = AppUsbh_MtpTestEventCallback;  
rval = AmbaUSBH_Mtp_SessionOpen (&AmbaUsbhMtpSession);  
AmбаPrint("session id = 0x%x", AmbaUsbhMtpSession.id);  
AmбаPrint("magic = 0x%x", AmbaUsbhMtpSession.magic);  
AmбаPrint("state = 0x%x", AmbaUsbhMtpSession.state);  
AmбаPrint("storage_ids = 0x%x", AmbaUsbhMtpSession.nb_storage_ids);  
AmбаPrint("number of objects = %d", AmbaUsbhMtpSession.nb_objects);
```

**See Also:**

None



### 3.3.4 AmbaUSBH\_Mtp\_SessionClose

#### API Syntax:

```
UINT32 AmbaUSBH_Mtp_SessionClose (USBH_MTP_SESSION *Session)
```

#### Function Description:

This function issues a **CloseSession** command to the MTP device.

#### Parameters:

| Type               | Parameters | Description   |
|--------------------|------------|---|
| USBH_MTP_SESSION * | Session    | [Input] MTP session information. Please refer to Section 3.4.1.1 for more details. Applications should not use it afterwards. |

Table 3-37 Parameters for AmbaUSB API **AmbaUSBH\_Mtp\_SessionClose()**.

#### Return:

| Return | Description |
|--------|-------------|
| 0      | Success     |
| others | Failure     |

Table 3-38 Returns for AmbaUSB API **AmbaUSBH\_Mtp\_SessionClose()**.

#### Example:

```
UINT32 rval = 0;  
rval = AmbaUSBH_Mtp_SessionClose(&AmbarraUsbhMtpSession);  
return rval;
```

#### See Also:

None



### 3.3.5 AmbaUSBH\_Mtp\_RegisterCallback

#### API Syntax:

```
UINT32 AmbaUSBH_Mtp_RegisterCallback (USB_HOST_MTP_CB_s *cb)
```

#### Function Description:

This function is used to register callback functions for the various MTP host events.

#### Parameters:

| Type                | Parameters | Description  |
|---------------------|------------|--|
| USB_HOST_MTP_CB_s * | cb         | [Input] MTP host callback functions. Please refer to Section 3.4.1.8 for more details. |

Table 3-39 Parameters for AmbaUSB API **AmbaUSBH\_Mtp\_RegisterCallback()**.

#### Return:

| Return | Description |
|--------|-------------|
| 0      | Success     |
| others | Failure     |

Table 3-40 Returns for AmbaUSB API **AmbaUSBH\_Mtp\_RegisterCallback()**.

#### Example:

```
static void AppUsbh_MtpMediaInsert(void)
{
    AmbaUsbMtpInsert = TRUE;
    AmbaPrint("Media Insert");
}

static void AppUsbh_MtpMediaRemove(void)
{
    AmbaUsbMtpInsert = FALSE;
    AmbaPrint("Media Remove");
}

static USB_HOST_MTP_CB_S AmbaUsbhMtpCb;
```



```
AmbaUsbhMtpCb.MediaInsert = AppUsbh_MtpMediaInsert;  
AmbaUsbhMtpCb.MediaRemove = AppUsbh_MtpMediaRemove;  
AmbaUSBH_Mtp_RegisterCallback(&AmbaUsbhMtpCb);
```

**See Also:**

None

For PROTRULY Confidential Only



### 3.3.6 AmbaUSBH\_Mtp\_DeviceInfoGet

#### API Syntax:

```
UINT32 AmbaUSBH_Mtp_DeviceInfoGet (USBH_MTP_DEVICE *Device)
```

#### Function Description:

This function issues a **GetDeviceInfo** command to the MTP device.

#### Parameters:

| Type              | Parameters | Description  |
|-------------------|------------|--|
| USBH_MTP_DEVICE * | session    | [Output] MTP device information. Please refer to Section 3.4.1.2 for more details. |

Table 3-41 Parameters for AmbaUSB API **AmbaUSBH\_Mtp\_DeviceInfoGet()**.

#### Return:

| Return | Description |
|--------|-------------|
| 0      | Success     |
| others | Failure     |

Table 3-42 Returns for AmbaUSB API **AmbaUSBH\_Mtp\_DeviceInfoGet()**.

#### Example:

```
UINT32 rval = 0, i = 0;  
  
static USBH_MTP_DEVICE AmbaUsbhMtpDevice;  
  
memset(&AmbaUsbhMtpDevice, 0, sizeof(USBH_MTP_DEVICE));  
rval = AmbaUSBH_Mtp_DeviceInfoGet (&AmbaUsbhMtpDevice);  
if (rval == 0) {  
    AmbaPrint("MTP Test : Device Info Get");  
    AmbaPrint("device standard version = 0x%x",  
    AmbaUsbhMtpDevice.standard_version);  
    AmbaPrint("device vendor extension id = 0x%x",  
    AmbaUsbhMtpDevice.vendor_extension_id);
```



```
AmbaPrint("device vendor extension version = 0x%x",
          AmbaUsbhMtpDevice.vendor_extension_version);
AmbaPrint("device vendor extension descriptor = %s",
          AmbaUsbhMtpDevice.vendor_extension_desc);
AmbaPrint("device functional mode = 0x%x",
          AmbaUsbhMtpDevice.functional_mode);
AmbaPrint("device operation support");
for (i=0; i<(UX_HOST_CLASS_PIMA_ARRAY_MAX_LENGTH/2); i++) {
    UINT16 TmpValue;
    TmpValue = *((UINT16*)AmbaUsbhMtpDevice.operations_supported + i);
    if (!TmpValue) {
        break;
    }
    AmbaPrint("0x%x", TmpValue);
}
AmbaPrint("device event support");
for (i=0; i<(UX_HOST_CLASS_PIMA_ARRAY_MAX_LENGTH/2); i++) {
    UINT16 TmpValue;
    TmpValue = *((UINT16*)AmbaUsbhMtpDevice.events_supported + i);
    if (!TmpValue) {
        break;
    }
    AmbaPrint("0x%x", TmpValue);
}
AmbaPrint("device properties support");
for (i=0; i<(UX_HOST_CLASS_PIMA_ARRAY_MAX_LENGTH/2); i++) {
    UINT16 TmpValue;
    TmpValue = *((UINT16*)AmbaUsbhMtpDevice.properties_supported + i);
    if (!TmpValue) {
        break;
    }
    AmbaPrint("0x%x", TmpValue);
}
AmbaPrint("device capture format");
```



```
for (i=0; i<(UX_HOST_CLASS_PIMA_ARRAY_MAX_LENGTH/2); i++) {  
    UINT16 TmpValue;  
    TmpValue = *((UINT16*)AmbarraUsbhMtpDevice.capture_formats + i);  
    if (!TmpValue) {  
        break;  
    }  
    AmbaPrint("0x%x", TmpValue);  
}  
  
AmbaPrint("device image format");  
for (i=0; i<(UX_HOST_CLASS_PIMA_ARRAY_MAX_LENGTH/2); i++) {  
    UINT16 TmpValue;  
    TmpValue = *((UINT16*)AmbarraUsbhMtpDevice.image_formats + i);  
    if (!TmpValue) {  
        break;  
    }  
    AmbaPrint("0x%x", TmpValue);  
}  
  
AmbaPrint("device manufacturer = %s", AmbarraUsbhMtpDevice.manufacturer);  
AmbaPrint("device model = %s", AmbarraUsbhMtpDevice.model);  
AmbaPrint("device version = %s", AmbarraUsbhMtpDevice.version);  
AmbaPrint("device serial number = %s",  
AmbarraUsbhMtpDevice.serial_number);  
}
```

#### See Also:

None



### 3.3.7 AmbaUSBH\_Mtp\_StorageIdsGet

#### API Syntax:

```
UINT32 AmbaUSBH_Mtp_StorageIdsGet (USBH_MTP_SESSION *Session, UINT32  
*IdsArray, UINT32 ArrayLength)
```

#### Function Description:

This function issues a **GetStorageIDs** command to the MTP device.

#### Parameters:

| Type               | Parameters         | Description  |
|--------------------|--------------------|--|
| USBH_MTP_SESSION * | <b>Session</b>     | [Input] MTP session information. Please refer to Section 3.4.1.1 for more details. |
| UINT32 *           | <b>IdsArray</b>    | [Output] Buffer contains MTP Storage IDs.  |
| UINT32             | <b>ArrayLength</b> | [Input] The buffer length  |

Table 3-43 Parameters for AmbaUSB API **AmbaUSBH\_Mtp\_StorageIdsGet()**.

#### Return:

| Return | Description |
|--------|-------------|
| 0      | Success     |
| others | Failure     |

Table 3-44 Returns for AmbaUSB API **AmbaUSBH\_Mtp\_StorageIdsGet()**.

#### Example:

```
#define MAX_STORAGE_IDS_NUMBER      5  
  
static AMBA_MEM_CTRL_s UsbhMtpTempMem = {0};  
  
UINT32 rval = 0, i = 0;  
UINT32 *ptr = 0;  
  
  
rval = AmbaKAL_MemAllocate(&AmbaBytePool_Cached,  
                           &UsbhMtpTempMem,  
                           MAX_STORAGE_IDS_NUMBER* sizeof(UINT32), 32);
```



```
if (rval != OK) {  
    AmbaPrint("[Error] %s(): can't allocate %d bytes memory for USBX, 0x%x.",  
    __func__,  
    MAX_STORAGE_IDS_NUMBER*sizeof(UINT32), rval);  
    return rval;  
}  
  
memset(UsbhMtpTempMem.pMemAlignedBase, 0,  
MAX_STORAGE_IDS_NUMBER*sizeof(UINT32));  
ptr = (UINT32*)UsbhMtpTempMem.pMemAlignedBase;  
rval = AmbaUSBH_Mtp_StorageIdsGet (&AmbaUsbhMtpSession,  
    ptr,  
    MAX_STORAGE_IDS_NUMBER*sizeof(UINT32));  
while(ptr[i] != 0) {  
    AmbaPrint("Storage [%d] ID = 0x%x", i, ptr[i]);  
    if ((++i) == MAX_STORAGE_IDS_NUMBER) {  
        break;  
    }  
}
```

**See Also:**

None



### 3.3.8 AmbaUSBH\_Mtp\_StorageInfoGet

#### API Syntax:

```
UINT32 AmbaUSBH_Mtp_StorageInfoGet (USBH_MTP_SESSION *Session, UINT32  
StorageId, USBH_MTP_STORAGE *Storage)
```

#### Function Description:

This function issues a **GetStorageInfo** command to the MTP device.

#### Parameters:

| Type               | Parameters       | Description   |
|--------------------|------------------|---|
| USBH_MTP_SESSION * | <b>Session</b>   | [Input] MTP session information. Please refer to Section 3.4.1.1 for more details.  |
| UINT32             | <b>StorageId</b> | [Input] Storage ID  |
| USBH_MTP_STORAGE * | <b>Storage</b>   | [Output] MTP storage information. Please refer to Section 3.4.1.3 for more details. |

Table 3-45 Parameters for AmbaUSB API **AmbaUSBH\_Mtp\_StorageInfoGet()**.

#### Return:

| Return | Description |
|--------|-------------|
| 0      | Success     |
| others | Failure     |

Table 3-46 Returns for AmbaUSB API **AmbaUSBH\_Mtp\_StorageInfoGet()**.

#### Example:

```
static USBH_MTP_STORAGE AmbaUsbhMtpStorage;  
  
UINT32 rval = 0;  
  
rval = AmbaUSBH_Mtp_StorageInfoGet (&AmbaUsbhMtpSession, StorageId,  
&AmbaUsbhMtpStorage);  
  
AmбаPrint("Storage Type = 0x%x", AmbaUsbhMtpStorage.type);  
AmбаPrint("Storage File System Type = 0x%x",  
AmbaUsbhMtpStorage.file_system_type);
```



```
AmбаPrint("Storage ACCESS CAPABILITY = 0x%x",  
AmбаUsbhMtpStorage.access_capability);  
  
AmбаPrint("Storage MAX CAPACITY LOW = 0x%x",  
AmбаUsbhMtpStorage.max_capacity_low);  
  
AmбаPrint("Storage MAX CAPACITY HIGH = 0x%x",  
AmбаUsbhMtpStorage.max_capacity_high);  
  
AmбаPrint("Storage FREE SPACE BYTES LOW = 0x%x",  
AmбаUsbhMtpStorage.free_space_bytes_low);  
  
AmбаPrint("Storage FREE SPACE BYTES HIGH = 0x%x",  
AmбаUsbhMtpStorage.free_space_bytes_high);  
  
AmбаPrint("Storage FREE SPACE IMAGES = 0x%x",  
AmбаUsbhMtpStorage.free_space_images);  
  
AmбаPrint("Storage Type = %s", AmбаUsbhMtpStorage.description);  
AmбаPrint("Storage label = %s", AmбаUsbhMtpStorage.volume_label);
```

**See Also:**

None



### 3.3.9 AmbaUSBH\_Mtp\_ObjectNumberGet

#### API Syntax:

```
UINT32 AmbaUSBH_Mtp_ObjectNumberGet (USBH_MTP_SESSION *Session, UINT32  
StorageId, UINT32 FormatCode, UINT32 Association)
```

#### Function Description:

This function issues a **GetNumObjects** command to the MTP device.

#### Parameters:

| Type               | Parameters  | Description   |
|--------------------|-------------|---|
| USBH_MTP_SESSION * | Session     | [Input/Output] MTP session information. Please refer to Section 3.4.1.1 for more details. Note that <b>nb_objects</b> field will contain the result if this function returns success. |
| UINT32             | StorageId   | [Input] Storage ID  |
| UINT32             | FormatCode  | [Input] Format code   |
| UINT32             | Association | [Input] Association   |

Table 3-47 Parameters for AmbaUSB API **AmbaUSBH\_Mtp\_ObjectNumberGet()**.

#### Return:

| Return | Description |
|--------|-------------|
| 0      | Success     |
| others | Failure     |

Table 3-48 Returns for AmbaUSB API **AmbaUSBH\_Mtp\_ObjectNumberGet()**.



**Example:**

```
UINT32 rval = 0;  
  
rval = AmbaUSBH_Mtp_ObjectNumberGet (&AmbeUsbhMtpSession,  
                                      StorageId,  
                                      FormatCode,  
                                      Association);  
  
AmbaPrint("number of objects = %d",  
          AmbeUsbhMtpSession.nb_objects);  
  
return rval;
```

**See Also:**

None



### 3.3.10 AmbaUSBH\_Mtp\_ObjectHandlesGet

#### API Syntax:

```
UINT32 AmbaUSBH_Mtp_ObjectHandlesGet (USBH_MTP_SESSION *Session, UINT32  
*ObjectHandlesArray, UINT32 ObjectHandlesLength, UINT32 StorageId, UINT32  
FormatCode, UINT32 ObjectHandleAssociation)
```

#### Function Description:

This function issues a **GetObjectHandles** command to the MTP device.

#### Parameters:

| Type                | Parameters              | Description  |
|---------------------|-------------------------|--|
| USBH_MTP_SESS ION * | Session                 | [Input] MTP session information. Please refer to Section 3.4.1.1 for more details. |
| UINT32 *            | ObjectHandlesArray      | [Output] Buffer contains Object Handles. Applications must allocate it.            |
| UINT32              | ObjectHandlesLength     | [Input] Buffer length  |
| UINT32              | StorageId               | [Input] Storage ID   |
| UINT32              | FormatCode              | [Input] Format code  |
| UINT32              | ObjectHandleAssociation | [Input] Association code   |

Table 3-49 Parameters for AmbaUSB API **AmbaUSBH\_Mtp\_ObjectHandlesGet()**.

#### Return:

| Return | Description |
|--------|-------------|
| 0      | Success     |
| others | Failure     |

Table 3-50 Returns for AmbaUSB API **AmbaUSBH\_Mtp\_ObjectHandlesGet()**.



**Example:**

```
#define MAX_HANDLES_NUMBER           255
static AMBA_MEM_CTRL_s UsbhMtpTempMem = {0};
UINT32 rval = 0, i = 0;
UINT32 *ptr = 0;

rval = AmbaKAL_MemAllocate(&AmбаBytePool_Cached,
                           &UsbhMtpTempMem,
                           MAX_HANDLES_NUMBER*sizeof(UINT32), 32);

if (rval != OK) {
    AmbaPrint("[Error] %s(): can't allocate %d bytes memory for USBX, 0x%X.",
              __func__,
              MAX_HANDLES_NUMBER*sizeof(UINT32), rval);
    return rval;
}

memset(UsbhMtpTempMem.pMemAlignedBase, 0,
MAX_HANDLES_NUMBER*sizeof(UINT32));
ptr = (UINT32*)UsbhMtpTempMem.pMemAlignedBase;

// This is a demo. Force to get 5 handles.
// Applications should get object numbers from device by
AmbaUSBH_Mtp_ObjectNumberGet().

AmbaUsbhMtpSession.nb_objects = MAX_HANDLES_NUMBER;
rval = AmbaUSBH_Mtp_ObjectHandlesGet (&AmbaUsbhMtpSession,
                                      ptr,
                                      MAX_HANDLES_NUMBER*sizeof(UINT32),
                                      StorageId,
                                      FormatCode,
                                      association);

AmbaPrint("number of objects = %d",
          AmbaUsbhMtpSession.nb_objects);
while(ptr[i] != 0) {
    AmbaPrint("Handle [%d] = 0x%x", i, ptr[i]);
```



```
if ((++i) == MAX_HANDLES_NUMBER) {  
    break;  
}  
}
```

**See Also:**

None

For PROTRULY Only  
Confidential



### 3.3.11 AmbaUSBH\_Mtp\_ObjectInfoGet

#### API Syntax:

```
UINT32 AmбаUSBH_Mtp_ObjectInfoGet(USBH_MTP_SESSION *Session, UINT32  
ObjectHandle, USBH_MTP_OBJECT *Object)
```

#### Function Description:

This function issues a **GetObjectInfo** command to the MTP device.

#### Parameters:

| Type               | Parameters   | Description  |
|--------------------|--------------|--|
| USBH_MTP_SESSION * | Session      | [Input] MTP session information. Please refer to Section 3.4.1.1 for more details. |
| UINT32             | ObjectHandle | [Input] Object handle  |
| USBH_MTP_OBJECT *  | Object       | [Output] MTP object information. Please refer to Section 3.4.1.4 for more details. |

Table 3-51 Parameters for AmbaUSB API **AmбаUSBH\_Mtp\_ObjectInfoGet()**.

#### Return:

| Return | Description |
|--------|-------------|
| 0      | Success     |
| others | Failure     |

Table 3-52 Returns for AmbaUSB API **AmбаUSBH\_Mtp\_ObjectInfoGet()**.

#### Example:

```
UINT32 rval = 0;  
  
USBH_MTP_OBJECT AmbaUsbhMtpObject;  
  
memset(&AmbaUsbhMtpObject, 0, sizeof(USBH_MTP_OBJECT));  
  
rval = AmбаUSBH_Mtp_ObjectInfoGet (&AmbaUsbhMtpSession,  
                                ObjectHandle,  
                                &AmbaUsbhMtpObject);
```



```
AmбаPrint("Object Info: Storage Id = 0x%x",
AmбаUsbhMtpObject.storage_id);

AmбаPrint("Object Info: format = 0x%x", AmбаUsbhMtpObject.format);

AmбаPrint("Object Info: protection = 0x%x",
AmбаUsbhMtpObject.protection_satus);

AmбаPrint("Object Info: compressed size = %d",
AmбаUsbhMtpObject.compressed_size);

AmбаPrint("Object Info: thumb format = 0x%x",
AmбаUsbhMtpObject.thumb_format);

AmбаPrint("Object Info: thumb compressed size = %d",
AmбаUsbhMtpObject.thumb_compressed_size);

AmбаPrint("Object Info: thumb pix width = %d",
AmбаUsbhMtpObject.thumb_pix_width);

AmбаPrint("Object Info: thumb pix height = %d",
AmбаUsbhMtpObject.thumb_pix_height);

AmбаPrint("Object Info: image pix width = %d",
AmбаUsbhMtpObject.image_pix_width);

AmбаPrint("Object Info: image pix height = %d",
AmбаUsbhMtpObject.image_pix_height);

AmбаPrint("Object Info: image bit depth = %d",
AmбаUsbhMtpObject.image_bit_depth);

AmбаPrint("Object Info: parent object = 0x%x",
AmбаUsbhMtpObject.parent_object);

AmбаPrint("Object Info: association type = 0x%x",
AmбаUsbhMtpObject.association_type);

AmбаPrint("Object Info: association desc = 0x%x",
AmбаUsbhMtpObject.association_desc);

AmбаPrint("Object Info: sequence num = 0x%x",
AmбаUsbhMtpObject.sequence_number);

AmбаPrint("Object Info: file name = %s", AmбаUsbhMtpObject.filename);

AmбаPrint("Object Info: capture date = %s",
AmбаUsbhMtpObject.capture_date);

AmбаPrint("Object Info: modification date = %s",
AmбаUsbhMtpObject.modification_date);
```



```
AmбаPrint("Object Info: keywords = %s", AmбаUsbhMtpObject.keywords);
AmбаPrint("Object Info: object state = 0x%x", AmбаUsbhMtpObject.state);
AmбаPrint("Object Info: object offset = 0x%x", AmбаUsbhMtpObject.offset);
AmбаPrint("Object Info: object transfer status = 0x%x",
AmбаUsbhMtpObject.transfer_status);
AmбаPrint("Object Info: object Handle ID = 0x%x",
AmбаUsbhMtpObject.handle_id);
AmбаPrint("Object Info: object length = %d", AmбаUsbhMtpObject.length);
AmбаPrint("Object Info: object buf = 0x%x", AmбаUsbhMtpObject.buffer);
```

**See Also:**

None



### 3.3.12 AmbaUSBH\_Mtp\_ObjectGet

#### API Syntax:

```
UINT32 AmbaUSBH_Mtp_ObjectGet (USBH_MTP_SESSION *Session, UINT32  
ObjectHandle, USBH_MTP_OBJECT *Object, UINT8 *ObjectBuf, UINT32 ObjectBufLen,  
UINT32 *ObjectActualLen)
```

#### Function Description:

This function issues a **GetObject** command to the MTP device.

#### Parameters:

| Type               | Parameters      | Description  |
|--------------------|-----------------|--|
| USBH_MTP_SESSION * | Session         | [Input] MTP session information. Please refer to Section 3.4.1.1 for more details. |
| UINT32             | ObjectHandle    | [Input] Object handle.   |
| USBH_MTP_OBJECT *  | Object          | [Input] MTP object information. Please refer to Section 3.4.1.4 for more details.  |
| UINT8 *            | ObjectBuf       | [Output] Buffer contains object data. Applications should allocate memory for it.  |
| UINT32             | ObjectBufLen    | [Input] Buffer length  |
| UINT32 *           | ObjectActualLen | [Output] Received data length  |

Table 3-53 Parameters for AmbaUSB API **AmbaUSBH\_Mtp\_ObjectGet()**.

#### Return:

| Return | Description |
|--------|-------------|
| 0      | Success     |
| others | Failure     |

Table 3-54 Returns for AmbaUSB API **AmbaUSBH\_Mtp\_ObjectGet()**.



**Example:**

```
#define MAX_OBJECT_BUF_SIZE      16*1024
static AMBA_MEM_CTRL_s UsbhMtpTempMem = {0};
UINT32 rval = 0;
UINT8 *ptr = 0;
UINT32 actual_length = 0;
UINT32 object_offset = 0;
UINT32 request_length = 0;
UINT32 total_length = 0;
AMBA_FS_FILE *pFile = NULL;
char path[64];
UINT64 size = 0;
UINT32 file_length;

// object information was get earlier by AmbaUSBH_Mtp_ObjectInfoGet().

// Create system buffer
rval = AmbaKAL_MemAllocate(&AmbaBytePool_Cached,
                           &UsbhMtpTempMem,
                           MAX_OBJECT_BUF_SIZE*sizeof(UINT8),
                           32);
if (rval != OK) {
    AmbaPrint("[Error] %s(): can't allocate %d bytes memory for USBX, 0x%X.",
              __func__, MAX_OBJECT_BUF_SIZE*sizeof(UINT8), rval);
    return rval;
}

memset(UsbhMtpTempMem.pMemAlignedBase, 0,
MAX_OBJECT_BUF_SIZE*sizeof(UINT8));
ptr = (UINT8*)UsbhMtpTempMem.pMemAlignedBase;

// Change object state to opened
AmbaUsbhMtpObject.state = UX_HOST_CLASS_PIMA_OBJECT_STATE_OPENED;
total_length = AmbaUsbhMtpObject.compressed_size;
file_length = AmbaUsbhMtpObject.compressed_size;
```



```
// save object data into c:\  
sprintf(path, "c:\\%s", AmbaUsbhMtpObject.filename);  
pFile = AmbaFS_fopen(path, "w");  
  
// Get object data until data length is matched the compressed size of object  
info  
while (1) {  
    // Decide request length  
    if (total_length >= MAX_OBJECT_BUF_SIZE) {  
        request_length = MAX_OBJECT_BUF_SIZE;  
    } else {  
        request_length = total_length;  
    }  
  
    // Get object data  
    rval = AmbaUSBH_Mtp_ObjectGet (&AmbaUsbhMtpSession,  
                                    ObjectHandle,  
                                    &AmbaUsbhMtpObject,  
                                    ptr,  
                                    request_length,  
                                    &actual_length);  
    if (rval != OK) {  
        AmbaPrint("%s(): Failed to get object data, error: 0x%x", __func__,  
rval);  
        break;  
    }  
  
    // Store data to SD card  
    size = AmbaFS_fwrite(ptr, 1, actual_length, pFile);  
    if (size != actual_length) {  
        AmbaPrint("Failed to write file");  
        break;  
    }  
}
```



```
object_offset += actual_length;  
total_length -= actual_length;  
  
AmбаPrint("-----");  
AmбаPrint("object offset = %d", object_offset);  
AmбаPrint("total_length = %d", total_length);  
  
// Check if length we received matches compressed size  
if (object_offset == file_length) {  
    AmбаPrint("%s(): completed", __func__);  
    break;  
} else if (object_offset > file_length) {  
    AmбаPrint("%s(): error", __func__);  
    break;  
}  
}  
  
// Change object state to closed  
AmбаUsbhMtpObject.state = UX_HOST_CLASS_PIMA_OBJECT_STATE_CLOSED;  
AmбаFS_fclose(pFile);
```

**See Also:**

None



### 3.3.13 AmbaUSBH\_Mtp\_ObjectInfoSend

#### API Syntax:

```
UINT32 AmbaUSBH_Mtp_ObjectInfoSend (USBH_MTP_SESSION *Session, UINT32  
ObjectHandle, USBH_MTP_OBJECT *Object)
```

#### Function Description:

This function issues a **SendObjectInfo** command to the MTP device.

#### Parameters:

| Type               | Parameters     | Description  |
|--------------------|----------------|--|
| USBH_MTP_SESSION * | Session        | [Input] MTP session information. Please refer to Section 3.4.1.1 for more details. |
| UINT32             | ParentObjectId | [Input] The parent object handle of the new object                                 |
| USBH_MTP_OBJECT *  | Object         | [Input] MTP object information. Please refer to Section 3.4.1.4 for more details.  |

Table 3-55 Parameters for AmbaUSB API **AmbaUSBH\_Mtp\_ObjectInfoSend()**.

#### Return:

| Return | Description |
|--------|-------------|
| 0      | Success     |
| others | Failure     |

Table 3-56 Returns for AmbaUSB API **AmbaUSBH\_Mtp\_ObjectInfoSend()**.



**Example:**

```
UINT32 rval = 0;

static USBH_MTP_OBJECT     AmbaUsbhMtpObject;
// setup object information here
//...
rval = AmbaUSBH_Mtp_ObjectInfoSend (&AmbaUsbhMtpSession,
                                     StorageID,
                                     ParentObjectId,
                                     &AmbaUsbhMtpObject);

if (rval != OK) {
    AmbaPrint("%s(): error = 0x%x", __func__, rval);
} else {
    AmbaPrint("%s(): completed", __func__);
}
```

**See Also:**

None



### 3.3.14 AmbaUSBH\_Mtp\_ObjectSend

#### API Syntax:

```
UINT32 AmbaUSBH_Mtp_ObjectSend (USBH_MTP_SESSION *Session,  
USBH_MTP_OBJECT *Object, UINT8 *ObjectBuf, UINT32 ObjectBufLen)
```

#### Function Description:

This function issues a **SendObject** command to the MTP device.

#### Parameters:

| Type               | Parameters   | Description  |
|--------------------|--------------|--|
| USBH_MTP_SESSION * | Session      | [Input] MTP session information. Please refer to Section 3.4.1.1 for more details. |
| USBH_MTP_OBJECT *  | Object       | [Input] MTP object information. Please refer to Section 3.4.1.4 for more details.  |
| UINT8 *            | ObjectBuf    | [Input] Buffer contains object data  |
| UINT32             | ObjectBufLen | [Input] Buffer length  |

Table 3-57 Parameters for AmbaUSB API **AmbaUSBH\_Mtp\_ObjectSend()**.

#### Return:

| Return | Description |
|--------|-------------|
| 0      | Success     |
| others | Failure     |

Table 3-58 Returns for AmbaUSB API **AmbaUSBH\_Mtp\_ObjectSend()**.



**Example:**

```
UINT32 rval = 0;
UINT8 *ptr = 0;
UINT32 request_length = 0;
UINT32 total_length = 0;
AMBA_FS_FILE *pFile = NULL;
char path[64];
UINT64 size = 0;

// Create system buffer
rval = AmbaKAL_MemAllocate(&AmbarBytePool_Cached,
                           &UsbhMtpTempMem,
                           MAX_OBJECT_BUF_SIZE*sizeof(UINT8),
                           32);
if (rval != OK) {
    AmbaPrint("[Error] %s(): can't allocate %d bytes memory for USBX, 0x%X.",
              __func__, MAX_OBJECT_BUF_SIZE*sizeof(UINT8), rval);
    return rval;
}

memset(UsbhMtpTempMem.pMemAlignedBase, 0,
       MAX_OBJECT_BUF_SIZE*sizeof(UINT8));
ptr = (UINT8*)UsbhMtpTempMem.pMemAlignedBase;
// Change object state to opened
AmbaUsbhMtpObject.state = UX_HOST_CLASS_PIMA_OBJECT_STATE_OPENED;
total_length = AmbaUsbhMtpObject.compressed_size;
AmbaPrint("total_length = %d", total_length);
// read data from c:\

sprintf(path, "c:\\%s", AmbaUsbhMtpObject.filename);
pFile = AmbaFS_fopen(path, "r");
if (pFile == NULL) {
    AmbaPrint("%s(): Failed to open %s", __func__, path);
}
```



```
while (1) {  
    // Decide request length  
    if (total_length >= MAX_OBJECT_BUF_SIZE) {  
        request_length = MAX_OBJECT_BUF_SIZE;  
    } else {  
        request_length = total_length;  
    }  
  
    // Read file  
    size = AmbaFS_fread(ptr, 1, request_length, pFile);  
    if (size != request_length) {  
        AmbaPrint("%s(): Read file error", __func__);  
        break;  
    }  
  
    // Send object data  
    rval = AmbaUSBH_Mtp_ObjectSend (&AmbaUsbhMtpSession,  
                                    &AmbaUsbhMtpObject,  
                                    ptr,  
                                    request_length);  
  
    if (rval != OK) {  
        AmbaPrint("%s(): Failed to send object data, error: 0x%x", __func__,  
rval);  
        break;  
    }  
  
    total_length -= request_length;  
    AmbaPrint("-----");  
    AmbaPrint("total length = %d", total_length);  
    AmbaPrint("request_length = %d", request_length);  
    if (total_length == 0) {  
        AmbaPrint("%s(): completed", __func__);  
        break;  
    }  
}
```



```
    } else if (total_length < 0) {  
        AmbaPrint("%s(): error", __func__);  
        break;  
    }  
}  
  
// Change object state to closed  
AmbaUsbhMtpObject.state = UX_HOST_CLASS_PIMA_OBJECT_STATE_CLOSED;  
AmbaFS_fclose(pFile);
```

**See Also:**

None

For PROTRULY Only



### 3.3.15 AmbaUSBH\_Mtp\_ObjectCopy

#### API Syntax:

```
UINT32 AmbaUSBH_Mtp_ObjectCopy (USBH_MTP_SESSION *session, UINT32  
ObjectHandle, UINT32 StorageId, UINT32 ParentObjectId)
```

#### Function Description:

This function issues a **CopyObject** command to the MTP device.

#### Parameters:

| Type               | Parameters     | Description  |
|--------------------|----------------|--|
| USBH_MTP_SESSION * | Session        | [Input] MTP session information. Please refer to Section 3.4.1.1 for more details. |
| UINT32             | ObjectHandle   | [Input] Object handle to copy  |
| UINT32             | StorageId      | [Input] Storage ID for the new object handle                                       |
| UINT32             | ParentObjectId | [Input] Parent object handle for the new object handle                             |

Table 3-59 Parameters for AmbaUSB API **AmbaUSBH\_Mtp\_ObjectCopy()**.

#### Return:

| Return | Description |
|--------|-------------|
| 0      | Success     |
| others | Failure     |

Table 3-60 Returns for AmbaUSB API **AmbaUSBH\_Mtp\_ObjectCopy()**.

#### Example:

None

#### See Also:

None



### 3.3.16 AmbaUSBH\_Mtp\_ObjectMove

#### API Syntax:

```
UINT32 AmbaUSBH_Mtp_ObjectMove (USBH_MTP_SESSION *Session, UINT32  
ObjectHandle, UINT32 StorageId, UINT32 ParentObjectId)
```

#### Function Description:

This function issues a **MoveObject** command to the MTP device.

#### Parameters:

| Type               | Parameters     | Description  |
|--------------------|----------------|--|
| USBH_MTP_SESSION * | Session        | [Input] MTP session information. Please refer to Section 3.4.1.1 for more details. |
| UINT32             | ObjectHandle   | [Input] Object Handle to move  |
| UINT32             | StorageId      | [Input] Storage ID for the moved object handle                                     |
| UINT32             | ParentObjectId | [Input] Parent object handle for the moved object handle                           |

Table 3-61 Parameters for AmbaUSB API **AmbaUSBH\_Mtp\_ObjectMove()**.

#### Return:

| Return | Description |
|--------|-------------|
| 0      | Success     |
| others | Failure     |

Table 3-62 Returns for AmbaUSB API **AmbaUSBH\_Mtp\_ObjectMove()**.

#### Example:

None

#### See Also:

None



### 3.3.17 AmbaUSBH\_Mtp\_ObjectDelete

#### API Syntax:

```
UINT32 AmbaUSBH_Mtp_ObjectDelete (USBH_MTP_SESSION *Session, UINT32  
ObjectHandle, UINT32 FormatCode)
```

#### Function Description:

This function issues a **DeleteObject** command to the MTP device.

#### Parameters:

| Type               | Parameters   | Description  |
|--------------------|--------------|--|
| USBH_MTP_SESSION * | Session      | [Input] MTP session information. Please refer to Section 3.4.1.1 for more details. |
| UINT32             | ObjectHandle | [Input] Object handle to delete  |
| UINT32             | FormatCode   | [Input] Format code  |

Table 3-63 Parameters for AmbaUSB API **AmbaUSBH\_Mtp\_ObjectDelete()**.

#### Return:

| Return | Description |
|--------|-------------|
| 0      | Success     |
| others | Failure     |

Table 3-64 Returns for AmbaUSB API **AmbaUSBH\_Mtp\_ObjectDelete()**.

#### Example:

None

#### See Also:

None



### 3.3.18 AmbaUSBH\_Mtp\_ObjectTransferAbort

#### API Syntax:

```
UINT32 AmbaUSBH_Mtp_ObjectTransferAbort (USBH_MTP_SESSION *Session,  
                                         UINT32 ObjectHandle, USBH_MTP_OBJECT *Object)
```

#### Function Description:

This function will abort the current MTP command.

#### Parameters:

| Type               | Parameters   | Description  |
|--------------------|--------------|--|
| USBH_MTP_SESSION * | Session      | [Input] MTP session information. Please refer to Section 3.4.1.1 for more details. |
| UINT32             | ObjectHandle | [Input] Object Handle  |
| USBH_MTP_OBJECT *  | Object       | [Input] MTP object information. Please refer to Section 3.4.1.4 for more details.  |

Table 3-65 Parameters for AmbaUSB API **AmbaUSBH\_Mtp\_ObjectTransferAbort()**.

#### Return:

| Return | Description |
|--------|-------------|
| 0      | Success     |
| others | Failure     |

Table 3-66 Returns for AmbaUSB API **AmbaUSBH\_Mtp\_ObjectTransferAbort()**.

#### Example:

None

#### See Also:

None



### 3.3.19 AmbaUSBH\_Mtp\_ThumbGet

#### API Syntax:

```
UINT32 AmbaUSBH_Mtp_ThumbGet (USBH_MTP_SESSION *Session, UINT32  
ObjectHandle, UINT32 FormatCode, USBH_MTP_OBJECT *Object, UINT8 *ObjectBuf,  
UINT32 ObjectBufLen, UINT32 *ObjectActualLen)
```

#### Function Description:

This function issues a **GetThumb** command to the MTP device.

#### Parameters:

| Type               | Parameters      | Description   |
|--------------------|-----------------|---|
| USBH_MTP_SESSION * | Session         | [Input] MTP session information. Please refer to Section 3.4.1.1 for more details.    |
| UINT32             | ObjectHandle    | [Input] Object handle   |
| UINT32             | FormatCode      | [Input] Format code   |
| USBH_MTP_OBJECT *  | Object          | [Input] MTP object information. Please refer to Section 3.4.1.4 for more details.     |
| UINT8 *            | ObjectBuf       | [Output] Buffer containing thumb data.<br>Applications should allocate memory for it. |
| UINT32             | ObjectBufLen    | [Input] Buffer length   |
| UINT32             | ObjectActualLen | [Output] Received data length   |

Table 3-67 Parameters for AmbaUSB API **AmbaUSBH\_Mtp\_ThumbGet()**.

#### Return:

| Return | Description |
|--------|-------------|
| 0      | Success     |
| others | Failure     |

Table 3-68 Returns for AmbaUSB API **AmbaUSBH\_Mtp\_ThumbGet()**.



**Example:**

```
#define MAX_OBJECT_BUF_SIZE      16*1024
static AMBA_MEM_CTRL_s UsbhMtpTempMem = {0};
UINT32 rval = 0;
UINT8 *ptr = 0;
UINT32 actual_length = 0;
UINT32 thumb_offset = 0;
UINT32 request_length = 0;
UINT32 total_length = 0;
AMBA_FS_FILE *pFile = NULL;
char path[64];
UINT64 size = 0;
UINT32 thumb_size;

// Create system buffer
rval = AmbaKAL_MemAllocate(&AmbaBytePool_Cached,
                           &UsbhMtpTempMem,
                           MAX_OBJECT_BUF_SIZE*sizeof(UINT8), 32);
if (rval != OK) {
    AmbaPrint("[Error] %s(): can't allocate %d bytes memory for USBX, 0x%X.",
              __func__,
              MAX_OBJECT_BUF_SIZE*sizeof(UINT8),
              rval);
    return rval;
}

memset(UsbhMtpTempMem.pMemAlignedBase, 0,
MAX_OBJECT_BUF_SIZE*sizeof(UINT8));
ptr = (UINT8*)UsbhMtpTempMem.pMemAlignedBase;

// object information was get earlier by AmbaUSBH_Mtp_ObjectInfoGet().
AmbaUsbhMtpObject.state = UX_HOST_CLASS_PIMA_OBJECT_STATE_OPENED;

total_length = AmbaUsbhMtpObject.thumb_compressed_size;
```



```
thumb_size = total_length;
// the thumbnail will be saved into c:\

sprintf(path, "c:\\THM_%s", AmbaUsbhMtpObject.filename);
pFile = AmbaFS_fopen(path, "w");

while (1) {
    // Decide request length
    if (total_length >= MAX_OBJECT_BUF_SIZE) {
        request_length = MAX_OBJECT_BUF_SIZE;
    } else {
        request_length = total_length;
    }

    rval = AmbaUSBH_Mtp_ThumbGet (&AmbaUsbhMtpSession,
                                  ObjectHandle,
                                  FormatCode,
                                  &AmbaUsbhMtpObject,
                                  ptr,
                                  request_length,
                                  &actual_length);

    if (rval != OK) {
        AmbaPrint("%s(): Failed to get object data, error: 0x%x", __func__,
rval);
        break;
    }
    // Store data to SD card
    size = AmbaFS_fwrite(ptr, 1, actual_length, pFile);
    if (size != actual_length) {
        AmbaPrint("Failed to write file");
        break;
    }

    thumb_offset += actual_length;
```



```
total_length -= actual_length;  
AmбаPrint("-----");  
AmбаPrint("thumb_offset = %d", thumb_offset);  
AmбаPrint("total_length = %d", total_length);  
  
// Check if length we received matches compressed size  
if (thumb_offset == thumb_size) {  
    AmбаPrint("%s(): completed", __func__);  
    break;  
} else if (thumb_offset > thumb_size) {  
    AmбаPrint("%s(): error", __func__);  
    break;  
}  
}  
  
// change state to closed  
AmбаUsbhMtpObject.state = UX_HOST_CLASS_PIMA_OBJECT_STATE_CLOSED;  
AmбаFS_fclose(pFile);
```

**See Also:**

None



### 3.3.20 AmbaUSBH\_Mtp\_DeviceReset

#### API Syntax:

```
UINT32 AmbaUSBH_Mtp_DeviceReset (void)
```

#### Function Description:

This function issues a **ResetDevice** command to the MTP device.

#### Parameters:

None

#### Return:

| Return | Description |
|--------|-------------|
| 0      | Success     |
| others | Failure     |

Table 3-69 Returns for AmbaUSB API *AmbaUSBH\_Mtp\_DeviceReset()*.

#### Example:

None

#### See Also:

None



### 3.3.21 AmbaUSBH\_Mtp\_VendorCommandSend

#### API Syntax:

```
UINT32 AmbaUSBH_Mtp_VendorCommandSend (USBH_MTP_SESSION *Session,  
USBH_MTP_COMMAND * Command, USBH_MTP_VENDOR_PAYLOAD *VendorPayload,  
UINT8 *Buffer, UINT32 Length)
```

#### Function Description:

This function is used to send a **vendor specific** command to the MTP device. Note that this function will also send the data to the MTP device if VendorPayload.size is not 0.

#### Parameters:

| Type                      | Parameters    | Description   |
|---------------------------|---------------|---|
| USBH_MTP_SESSION *        | Session       | [Input] MTP session information. Please refer to Section 3.4.1.1 for more details.  |
| USBH_MTP_COMMAND *        | Command       | [Input] MTP command information. Please refer to Section 3.4.1.5 for more details.  |
| USBH_MTP_VENDOR_PAYLOAD * | VendorPayload | [Input] Data payload information. Please refer to Section 3.4.1.6 for more details. |
| UINT8 *                   | Buffer        | [Input] Buffer contains data.   |
| UINT32                    | Length        | [Input] Buffer length   |

Table 3-70 Parameters for AmbaUSB API **AmbaUSBH\_Mtp\_VendorCommandSend()**.

#### Return:

| Return | Description |
|--------|-------------|
| 0      | Success     |
| others | Failure     |

Table 3-71 Returns for AmbaUSB API **AmbaUSBH\_Mtp\_VendorCommandSend()**.



**Example:**

```
#define MAX_VENDOR_PAYLOAD_SIZE      16*1024
#define VENDOR_DATA_SIZE           6
USBH_MTP_COMMAND command = {0};
USBH_MTP_VENDOR_PAYLOAD vendor_payload = {0};
UINT8 *ptr = 0;
UINT32 request_length = 0;
UINT32 rval = 0;
UINT32 actual_length = 0;
UINT32 total_length = 0;
UINT32 receive_length = 0;
int i = 0;

// Create system buffer
rval = AmbaKAL_MemAllocate(&AmbarBytePool_Cached,
                           &UsbhMtpTempMem,
                           MAX_VENDOR_PAYLOAD_SIZE*sizeof(UINT8),
                           32);
if (rval != OK) {
    AmbaPrint("[Error] %s(): can't allocate %d bytes memory for USBX, 0x%X.",
              __func__, MAX_VENDOR_PAYLOAD_SIZE*sizeof(UINT8), rval);
    return rval;
}

memset(UsbhMtpTempMem.pMemAlignedBase, 0,
MAX_OBJECT_BUF_SIZE*sizeof(UINT8));
ptr = (UINT8*)UsbhMtpTempMem.pMemAlignedBase;
// setup vendor command
// Ambarelle vendor command: set date time.
command.operation_code = UX_HOST_CLASS_PIMA_OC_AMBA_SET_DATETIME;
command.nb_parameters = param0;
command.parameter_1 = param1;
command.parameter_2 = param2;
command.parameter_3 = param3;
```



```
command.parameter_4 = param4;
command.parameter_5 = param5;
// send 6 bytes data in data phase.
vendor_payload.size = VENDOR_DATA_SIZE;

total_length = VENDOR_DATA_SIZE;
// Decide request length
if (total_length >= MAX_VENDOR_PAYLOAD_SIZE) {
    request_length = MAX_VENDOR_PAYLOAD_SIZE;
} else {
    request_length = VENDOR_DATA_SIZE;
}
// setup 6 bytes data.
for (i = 0; i<VENDOR_DATA_SIZE; i++) {
    ptr[i] = i;
}

while(1) {
    AmbaUSBH_Mtp_VendorCommandSend (&AmbaUsbhMtpSession,
                                    &command,
                                    &vendor_payload,
                                    ptr,
                                    request_length);

    total_length -= request_length;
    if (total_length == 0) {
        AmbaPrint("%s(): completed", __func__);
        break;
    } else if (total_length < 0) {
        AmbaPrint("%s(): error", __func__);
        break;
    }
}
```



See Also:

None

For PROTRULY Confidential Only



### 3.3.22 AmbaUSBH\_Mtp\_VendorCommandGet

#### API Syntax:

```
UINT32 AmbaUSBH_Mtp_VendorCommandGet (USBH_MTP_SESSION *Session,  
USBH_MTP_COMMAND * Command, USBH_MTP_VENDOR_PAYLOAD *VendorPayload,  
UINT8 *Buffer, UINT32 Length, UINT32 *ActualLength)
```

#### Function Description:

This function is used to send a **vendor specific** command with data from the MTP device.

Note that this function will also receive data from the MTP device if VendorPayload.size is not 0.

#### Parameters:

| Type                      | Parameters     | Description   |
|---------------------------|----------------|---|
| USBH_MTP_SESSION *        | Session        | [Input] MTP session information. Please refer to Section 3.4.1.1 for more details.  |
| USBH_MTP_COMMAND *        | Command        | [Input] MTP command information. Please refer to Section 3.4.1.5 for more details.  |
| USBH_MTP_VENDOR_PAYLOAD * | *VendorPayload | [Input] Data payload information. Please refer to Section 3.4.1.6 for more details. |
| UINT8 *                   | Buffer         | [Output] Buffer contains data. Applications should allocate memory for it.          |
| UINT32                    | Length         | [Input] Buffer length   |
| UINT32 *                  | ActualLength   | [Input] Received data length  |

Table 3-72 Parameters for AmbaUSB API **AmbaUSBH\_Mtp\_VendorCommandGet()**.

#### Return:

| Return | Description |
|--------|-------------|
| 0      | Success     |
| others | Failure     |

Table 3-73 Returns for AmbaUSB API **AmbaUSBH\_Mtp\_VendorCommandGet()**.



**Example:**

```
#define MAX_VENDOR_PAYLOAD_SIZE      16*1024
#define VENDOR_DATA_SIZE           6
USBH_MTP_COMMAND command = {0};
USBH_MTP_VENDOR_PAYLOAD vendor_payload = {0};
UINT8 *ptr = 0;
UINT32 request_length = 0;
UINT32 rval = 0;
UINT32 actual_length = 0;
UINT32 total_length = 0;
UINT32 receive_length = 0;
int i = 0;

// Create system buffer
rval = AmbaKAL_MemAllocate(&AmbaBytePool_Cached,
                           &UsbhMtpTempMem,
                           MAX_VENDOR_PAYLOAD_SIZE*sizeof(UINT8),
                           32);
if (rval != OK) {
    AmbaPrint("[Error] %s(): can't allocate %d bytes memory for USBX, 0x%X.",
              __func__, MAX_VENDOR_PAYLOAD_SIZE*sizeof(UINT8), rval);
    return rval;
}

memset(UsbhMtpTempMem.pMemAlignedBase, 0,
MAX_OBJECT_BUF_SIZE*sizeof(UINT8));
ptr = (UINT8*)UsbhMtpTempMem.pMemAlignedBase;
// setup vendor command
// Ambarelle vendor command: get date time.
command.operation_code = UX_HOST_CLASS_PIMA_OC_AMBA_GET_DATETIME;
command.nb_parameters = param0;
command.parameter_1 = param1;
command.parameter_2 = param2;
command.parameter_3 = param3;
```



```
command.parameter_4 = param4;
command.parameter_5 = param5;
// receive 6 bytes data in data phase.
vendor_payload.size = VENDOR_DATA_SIZE;
total_length = VENDOR_DATA_SIZE;
// Decide request length
if (total_length >= MAX_VENDOR_PAYLOAD_SIZE) {
    request_length = MAX_VENDOR_PAYLOAD_SIZE;
} else {
    request_length = VENDOR_DATA_SIZE;
}
while (1) {
    AmbaUSBH_Mtp_VendorCommandGet (&AmbaUsbhMtpSession,
                                    &command,
                                    &vendor_payload,
                                    ptr,
                                    request_length,
                                    &actual_length);
    total_length -= actual_length;
    receive_length += actual_length;
    if (receive_length == vendor_payload.size) {
        AmbaPrint("%s(): completed", __func__);
        break;
    } else if (receive_length > vendor_payload.size) {
        AmbaPrint("%s(): error", __func__);
        break;
    }
}

// print received data
for (i = 0; i<VENDOR_DATA_SIZE; i++) {
    AmbaPrint("ptr[%d] = %d", i, ptr[i]);
}
```



See Also:

None

For PROTRULY Confidential Only



### 3.3.23 AmbaUSBH\_Mtp\_GetResponseCode

#### API Syntax:

```
UINT32 AmbaUSBH_Mtp_GetResponseCode (USBH_MTP_COMMAND* response)
```

#### Function Description:

This function gets the response code for the current MTP command.

#### Parameters:

| Type              | Parameters | Description  |
|-------------------|------------|--|
| USBH_MTP_COMMAND* | response   | [Output] MTP command/response information. Please refer to Section 3.4.1.5 for more details. |

Table 3-74 Parameters for AmbaUSB API **AmbaUSBH\_Mtp\_GetResponseCode()**.

#### Return:

| Return | Description |
|--------|-------------|
| 0      | Success     |
| others | Failure     |

Table 3-75 Returns for AmbaUSB API **AmbaUSBH\_Mtp\_GetResponseCode()**.

#### Example:

```
USBH_MTP_COMMAND AmbaUsbhMtpResponse;  
  
AmbaUSBH_Mtp_GetResponseCode (&AmbaUsbhMtpResponse);  
  
AmбаPrint("===== PTP RSP BLK CMD =====");  
AmбаPrint("RSP code: 0x%x", AmbaUsbhMtpResponse.response_code);  
AmбаPrint("Param1 : 0x%x", AmbaUsbhMtpResponse.parameter_1);  
AmбаPrint("Param2 : 0x%x", AmbaUsbhMtpResponse.parameter_2);  
AmбаPrint("Param3 : 0x%x", AmbaUsbhMtpResponse.parameter_3);  
AmбаPrint("Param4 : 0x%x", AmbaUsbhMtpResponse.parameter_4);  
AmбаPrint("Param5 : 0x%x", AmbaUsbhMtpResponse.parameter_5);  
AmбаPrint("===== PTP RSP BLK END =====");
```



See Also:

None

For PROTRULY Confidential Only



### 3.3.24 AmbaUSBH\_Mtp\_SetResponseCode

#### API Syntax:

```
UINT32 AmbaUSBH_Mtp_SetResponseCode (USBH_MTP_COMMAND* response)
```

#### Function Description:

This function sets the response code for next MTP command. Usually, applications use this function to clear the response code of the previous MTP command.

#### Parameters:

| Type              | Parameters | Description  |
|-------------------|------------|--|
| USBH_MTP_COMMAND* | response   | [Input] MTP command/response information.<br>Please refer to Section 3.4.1.5 for more details. |

Table 3-76 Parameters for AmbaUSB API **AmbaUSBH\_Mtp\_SetResponseCode()**.

#### Return:

| Return | Description |
|--------|-------------|
| 0      | Success     |
| others | Failure     |

Table 3-77 Returns for AmbaUSB API **AmbaUSBH\_Mtp\_SetResponseCode()**.

#### Example:

None

#### See Also:

None



### 3.4 Video Class (UVC) APIs

This section describes the APIs for the Video Class. Starting from document v1.7, the USB host supports up to three UVC devices at the same time. Each UVC API before v1.7 had extension APIs for multiple devices. Ambarella suggests using these extension APIs instead of the original APIs, as the orginal APIs could be deprecated in future.

For PROTRULY Confidential Only



### 3.4.1 Data Structures

#### 3.4.1.1 UVCH\_DEVICE\_INFO

| Type               | Parameter             | Description   |
|--------------------|-----------------------|---|
| ULONG              | <b>size</b>           | [Input] The size of this data structure   |
| ULONG              | <b>it_id</b>          | [Output] Input Terminal ID of the UVC device  |
| ULONG              | <b>xu_id</b>          | [Output] eXtension Unit ID of the UVC device  |
| ULONG              | <b>pu_id</b>          | [Output] Processing Unit ID of the UVC device   |
| ULONG              | <b>format_count</b>   | [Output] The supported UVC format number of the UVC device  |
| UVCH_FORMAT_INFO * | <b>format</b>         | [Output] The supported UVC format of the UVC device. Please refer to Section 3.4.1.2 for more details.          |
| ULONG              | <b>eu_id</b>          | [Output] The encoding unit ID supported by UVC v1.5 and later.  |
| ULONG              | <b>uvc_ver</b>        | [Output] The UVC version number.<br>0x00 before UVC 1.5<br>0x01 UVC 1.5   |
| ULONG              | <b>device_id</b>      | [Output] The device ID assigned by the USB stack via the callback function (Section 3.4.1.14) by the USB stack. |
| USHORT             | <b>vendor_id</b>      | [Output] The vendor ID of the connected device.   |
| USHORT             | <b>product_id</b>     | [Output] The product ID of the connected device.  |
| ULONG              | <b>cur_altsetting</b> | [output] The current alternate setting of connected devices.  |

Table 3-78 Definition of **UVCH\_DEVICE\_INFO**.



### 3.4.1.2 UVCH\_FORMAT\_INFO

| Type              | Parameter          | Description   |
|-------------------|--------------------|---|
| ULONG             | <b>subtype</b>     | [Output] Video Class-Specific VS Interface Descriptor Subtypes. Please refer to Section 3.4.1.4 for more details. |
| ULONG             | <b>index</b>       | [Output] Index of this format   |
| ULONG             | <b>frame_count</b> | [Output] The supported UVC frame number of this format  |
| UVCH_FRAME_INFO * | <b>frame</b>       | [Output] The supported UVC frame of this format. Please refer to Section 3.4.1.3 for more details.                |

Table 3-79 Definition of **UVCH\_FORMAT\_INFO**.

### 3.4.1.3 UVCH\_FRAME\_INFO

| Type    | Parameter                     | Description  |
|---------|-------------------------------|--|
| ULONG   | <b>index</b>                  | [Output] Index of this frame   |
| ULONG   | <b>width</b>                  | [Output] Width of this frame   |
| ULONG   | <b>height</b>                 | [Output] Height of this frame  |
| ULONG   | <b>min_bitrate</b>            | [Output] Minimal bitrate   |
| ULONG   | <b>max_bitrate</b>            | [Output] Maximum bitrate   |
| ULONG   | <b>max_buffer_size</b>        | [Output] Maximum buffer size for each video frame  |
| ULONG   | <b>default_frame_interval</b> | [Output] Default frame interval.<br>10000000 means 1 second, i.e., 1 FPS.  |
| ULONG   | <b>interval_type</b>          | [Output] Interval type.<br>0: Continue frame interval<br>Others: Discrete frame interval                         |
| ULONG * | <b>interval</b>               | [Output] For continued frame interval, interval[0] is the minimum interval, interval[1] is the maximum interval, |



| Type | Parameter | Description  |
|------|-----------|--|
|      |           | interval[2] is steps.<br>The discrete frame interval contains all the intervals. |

Table 3-80 Definition of **UVCH\_FRAME\_INFO**.

#### 3.4.1.4 Video Class-Specific VS Interface Descriptor Subtypes

| Type                                    | Field |
|---|-------|
| UVC_DESC_SUBTYPE_VS_FORMAT_UNCOMPRESSED | 0x04  |
| UVC_DESC_SUBTYPE_VS_FRAME_UNCOMPRESSED  | 0x05  |
| UVC_DESC_SUBTYPE_VS_FORMAT_MJPEG        | 0x06  |
| UVC_DESC_SUBTYPE_VS_FRAME_MJPEG         | 0x07  |
| UVC_DESC_SUBTYPE_VS_FORMAT_MPEG2TS      | 0x0A  |
| UVC_DESC_SUBTYPE_VS_FORMAT_DV           | 0x0C  |
| UVC_DESC_SUBTYPE_VS_FORMAT_FRAME_BASED  | 0x10  |
| UVC_DESC_SUBTYPE_VS_FRAME_FRAME_BASED   | 0x11  |
| UVC_DESC_SUBTYPE_VS_FORMAT_STREAM_BASED | 0x12  |

Table 3-81 Definition of **Video Class-Specific VS Interface Descriptor Subtypes**.

#### 3.4.1.5 UVCH\_CONTROL\_ITEM

| Type    | Parameter           | Description   |
|---------|---------------------|---|
| UINT8   | <b>selector</b>     | [Output] Index of this control  |
| UINT8   | <b>is_supported</b> | [Output] Indicates if this control is supported<br>0: Not supported.<br>1: Supported. |
| UINT8   | <b>size</b>         | [Output] The value size of this control   |
| UINT8   | <b>rsvd</b>         | [Output] Reserved   |
| UINT32  | <b>buffer_size</b>  | [Input] The buffer size for values  |
| UINT8 * | <b>max</b>          | [Output] Buffer contains maximum value of this control. Application should allocate   |



| Type    | Parameter | Description  |
|---------|-----------|--|
|         |           | memory for it.   |
| UINT8 * | min       | [Output] Buffer contains minimum value of this control. Application should allocate memory for it. |
| UINT8 * | def       | [Output] Buffer contains default value of this control. Application should allocate memory for it. |
| UINT8 * | cur       | [Output] Buffer contains current value of this control. Application should allocate memory for it. |

Table 3-82 Definition of **UVCH\_CONTROL\_ITEM**.

#### 3.4.1.6 UVCH\_CTRL\_INFO\_s

| Type   | Parameter | Description                           |
|--------|-----------|---------------------------------------|
| UINT32 | max       | [Output] maximum value of UVC control |
| UINT32 | min       | [Output] minimum value of UVC control |
| UINT32 | cur       | [Output] current value of UVC control |

Table 3-83 Definition of **UVCH\_CTRL\_INFO\_s**.

#### 3.4.1.7 Camera Terminal Control Selectors

| Type                                  | Field |
|---------------------------------------|-------|
| UVC_CT_CONTROL_UNDEFINED              | 0x00  |
| UVC_CT_SCANNING_MODE_CONTROL          | 0x01  |
| UVC_CT_AE_MODE_CONTROL                | 0x02  |
| UVC_CT_AE_PRIORITY_CONTROL            | 0x03  |
| UVC_CT_EXPOSURE_TIME_ABSOLUTE_CONTROL | 0x04  |
| UVC_CT_EXPOSURE_TIME_RELATIVE_CONTROL | 0x05  |
| UVC_CT_FOCUS_ABSOLUTE_CONTROL         | 0x06  |
| UVC_CT_FOCUS_RELATIVE_CONTROL         | 0x07  |
| UVC_CT_FOCUS_AUTO_CONTROL             | 0x08  |



| Type                            | Field |
|---------------------------------|-------|
| UVC_CT_IRIS_ABSOLUTE_CONTROL    | 0x09  |
| UVC_CT_IRIS_RELATIVE_CONTROL    | 0x0A  |
| UVC_CT_ZOOM_ABSOLUTE_CONTROL    | 0x0B  |
| UVC_CT_ZOOM_RELATIVE_CONTROL    | 0x0C  |
| UVC_CT_PANTILT_ABSOLUTE_CONTROL | 0x0D  |
| UVC_CT_PANTILT_RELATIVE_CONTROL | 0x0E  |
| UVC_CT_ROLL_ABSOLUTE_CONTROL    | 0x0F  |
| UVC_CT_ROLL_RELATIVE_CONTROL    | 0x10  |
| UVC_CT_PRIVACY_CONTROL          | 0x11  |

Table 3-84 Definition of **Camera Terminal Control Selectors**.

#### 3.4.1.8 Processing Unit Control Selectors

| Type  | Field |
|---|-------|
| UVC_PU_CONTROL_UNDEFINED                      | 0x00  |
| UVC_PU_BACKLIGHT_COMPENSATION_CONTROL         | 0x01  |
| UVC_PU_BRIGHTNESS_CONTROL                     | 0x02  |
| UVC_PU_CONTRAST_CONTROL                       | 0x03  |
| UVC_PU_GAIN_CONTROL                           | 0x04  |
| UVC_PU_POWER_LINE_FREQUENCY_CONTROL           | 0x05  |
| UVC_PU_HUE_CONTROL                            | 0x06  |
| UVC_PU_SATURATION_CONTROL                     | 0x07  |
| UVC_PU_SHARPNESS_CONTROL                      | 0x08  |
| UVC_PU_GAMMA_CONTROL                          | 0x09  |
| UVC_PU_WHITE_BALANCE_TEMPERATURE_CONTROL      | 0x0A  |
| UVC_PU_WHITE_BALANCE_TEMPERATURE_AUTO_CONTROL | 0x0B  |
| UVC_PU_WHITE_BALANCE_COMPONENT_CONTROL        | 0x0C  |
| UVC_PU_WHITE_BALANCE_COMPONENT_AUTO_CONTROL   | 0x0D  |
| UVC_PU_DIGITAL_MULTIPLIER_CONTROL             | 0x0E  |
| UVC_PU_DIGITAL_MULTIPLIER_LIMIT_CONTROL       | 0x0F  |
| UVC_PU_HUE_AUTO_CONTROL                       | 0x10  |
| UVC_PU_ANALOG_VIDEO_STANDARD_CONTROL          | 0x11  |



| Type                              | Field |
|-----------------------------------|-------|
| UVC_PU_ANALOG_LOCK_STATUS_CONTROL | 0x12  |

Table 3-85 Definition of **Processing Unit Control Selectors**.

For Confidential PROTRULY Only



### 3.4.1.9 Processing Unit Power Line Frequency Control

| Type             | Field |
|------------------|-------|
| UVC_PLF_DISABLED | 0     |
| UVC_PLF_50_HZ    | 1     |
| UVC_PLF_60_HZ    | 2     |

Table 3-86 Definition of **Processing Unit Power Line Frequency Control**.

### 3.4.1.10 Processing Unit Analog Video Standard Control

| Type                 | Field |
|----------------------|-------|
| UVC_AVN_NONE         | 0     |
| UVC_AVN_NTSC_525_60  | 1     |
| UVC_AVN_PAL_625_50   | 2     |
| UVC_AVN_SECAM_625_50 | 3     |
| UVC_AVN_NSC_625_50   | 4     |
| UVC_AVN_PAL_525_60   | 5     |

Table 3-87 Definition of **Processing Unit Analog Video Standard Control**.

### 3.4.1.11 Processing Unit Video Lock Status Control

| Type             | Field |
|------------------|-------|
| UVC_AVL_LOCKED   | 0     |
| UVC_AVL_UNLOCKED | 1     |

Table 3-88 Definition of **Processing Unit Video Lock Status Control**.

### 3.4.1.12 UVCX\_VIDEO\_CONFIG\_S

| Type  | Parameter              | Description                       |
|-------|------------------------|-----------------------------------|
| ULONG | <b>dwFrameInterval</b> | [Output] Frame interval in 100ns. |
| ULONG | <b>dwBitRate</b>       | [Output] Average bit per second.  |



| Type   | Parameter                  | Description  |
|--------|----------------------------|--|
| USHORT | <b>bmHints</b>             | [Output] Advises what configuration parameter(s) should be maintained.<br>0x0001: Resolution (wHeight and wWidth)<br>0x0002: Profile (wProfile)<br>0x0004: Rate Control Mode<br>(bRateControlMode)<br>0x0008: Usage Type (bUsageType)<br>0x0010: Slice Mode (wSliceMode)<br>0x0020: Slice Unit (wSliceUnits)<br>0x0040: MVC View (bView)<br>0x0080: Temporal<br>(bTemporalScaleMode)<br>0x0100: SNR (bSNRScaleMode)<br>0x0200: Spatial (bSpatialScaleMode)<br>0x0400: Spatial Layer Ratio<br>(bSpatialLayerRatio)<br>0x0800: Frame interval<br>(dwFrameInterval)<br>0x1000: Leaky Bucket Size<br>(wLeakyBucketSize)<br>0x2000: Bit Rate (dwBitRate)<br>0x4000: Entropy CABAC<br>(bEntropyCABAC)<br>0x8000: I FramePeriod (wlFramePeriod) |
| USHORT | <b>wConfigurationIndex</b> | [Output] Configuration index, an increasing number from 1 to max wConfigurationIndex that increments for each subsequent GET_CUR.  |
| USHORT | <b>wWidth</b>              | [Output] Encoder input image width in pixels.  |
| USHORT | <b>wHeight</b>             | [Output] Encoder input image height in   |



| Type   | Parameter          | Description  |
|--------|--------------------|--|
|        |                    | pixels.  |
| USHORT | <b>wSliceUnits</b> | [Output] The parameter defines the units of the wSliceMode.<br>wSliceMode=0x0000: wSliceUnits ignored<br>wSliceMode=0x0001: wSliceUnits in bits/slice<br>wSliceMode=0x0002: wSliceUnits in MBs/slice<br>wSliceMode=0x0003: wSliceUnits in slices/frame   |
| USHORT | <b>wSliceMode</b>  | [Output]<br>0x0000 -> no multiple slices<br>0x0001 -> multiple slices - bits/slice,<br>0x0002 -> multiple slices-MBs/slice,<br>0x0003 -> number of slices per frame<br>0x0004-0xFFFF = Reserved  |
| USHORT | <b>wProfile</b>    | [Output] profile_idc as defined in H.264 specification.<br>Profiles<br>(Bits 8-15)<br>0x4200 -> Baseline Profile<br>0x4D00 -> Main Profile<br>0x6400 -> High Profile<br>0x5300 -> Scalable Baseline Profile<br>0x5600 -> Scalable High Profile<br>0x7600 -> Multiview High Profile<br>0x8000 -> Stereo High Profile<br>Constrained flags<br>(Bits 0-7)<br>0x0080 -> constraint_set0_flag<br>0x0040 -> constraint_set1_flag |



| Type   | Parameter                | Description  |
|--------|--------------------------|--|
|        |                          | 0x0020 -> constraint_set2_flag<br>0x0010 -> constraint_set3_flag<br>0x0008 -> constraint_set4_flag<br>0x0004 -> constraint_set5_flag<br>0x0002 -> Reserved<br>0x0001 -> Reserved                                   |
| USHORT | wIFramePeriod            | [Output] The time between IDR frames in milliseconds.<br>0x0000= No periodicity requirements for IDR frames.   |
| USHORT | wEstimatedVideoDelay     | [Output] Estimated time between the end of exposure and the presentation on the USB interface, in milliseconds.  |
| USHORT | wEstimatedMaxConfigDelay | [Output] Estimated maximum time to change configuration modes, in milliseconds.  |
| UCHAR  | bUsageType               | [Output] Encoder Configuration based on the host configured usage type.<br>0x00: Reserved<br>0x01: Real-time (video conf)<br>0x02: Broadcast<br>0x03: Storage<br>0x04-0x0F: UCCONFIG MODES<br>0x10-0xFF = Reserved |
| UCHAR  | bRateControlMode         | [Output]<br>Bits 0-3 Modes:<br>0x00: Reserved<br>0x01: CBR<br>0x02: VBR<br>0x03: Constant QP<br>Bits 4-7 Flags:<br>0x10: fixed_frame_rate_flag   |



| Type  | Parameter                 | Description  |
|-------|---------------------------|--|
|       |                           | 0x20: Reserved set to zero<br>0x40: Reserved set to zero<br>0x80: Reserved set to zero   |
| UCHAR | <b>bTemporalScaleMode</b> | [Output]<br>0x00: No Temporal enhancement layer<br>0x01- 0x07: Number of Temporal enhancement layers<br>0x08-0xFF = Reserved<br>Note: Constrained by bUsageType.   |
| UCHAR | <b>bSpatialScaleMode</b>  | [Output]<br>0x00: No Spatial Enhancement Layer<br>0x01-0x08: Number of Spatial enhancement layers<br>0x09-0xFF = Reserved<br>Note: Constrained by bUsageType.  |
| UCHAR | <b>bSNRScaleMode</b>      | [Output]<br>0x00: No SNR Enhancement Layer<br>0x01: Reserved<br>0x02: CGS_NonRewrite_TwoLayer<br>0x03: CGS_NonRewrite_ThreeLayer<br>0x04: CGS_Rewrite_TwoLayer<br>0x05: CGS_Rewrite_ThreeLayer<br>0x06: MGS_TwoLayer<br>0x07-0xFF = Reserved<br>Note: Constrained by bUsageType. |
| UCHAR | <b>bStreamMuxOption</b>   | [Output] Auxiliary stream control<br>Bit 0: Enable/Disable auxiliary stream<br>0: Auxiliary stream disabled. Bits 1-7 ignored.<br>1: Auxiliary stream enabled.<br>PROBE/COMMIT fields apply to streams indicated by bits 1-7.  |



| Type  | Parameter                  | Description  |
|-------|----------------------------|--|
|       |                            | <p>Bit 1: Embed H.264 auxiliary stream.<br/>Bit 2: Embed YUY2 auxiliary stream.<br/>Bit 3: Embed NV12 auxiliary stream.<br/>Bit 4-5: Reserved<br/>Bit 6: MJPEG payload used as a container.<br/>Bit 7: Reserved<br/>Note: For SET_CUR operation, only one auxiliary stream bit shall be set.</p> |
| UCHAR | <b>bStreamFormat</b>       | <p>[Output]<br/>0x00 – Output data in Byte stream format (H.264 Annex- B)<br/>0x01 – Output data in NAL stream format<br/>0x02-0xFF = Reserved</p>   |
| UCHAR | <b>bEntropyCABAC</b>       | <p>[Output]<br/>0x00=CAVLC<br/>0x01=CABAC<br/>0x02-0xFF = Reserved</p>   |
| UCHAR | <b>bTimestamp</b>          | <p>[Output]<br/>0x00=picture timing SEI disabled<br/>0x01=picture timing SEI enabled<br/>0x02-0xFF = Reserved</p>  |
| UCHAR | <b>bNumOfReorderFrames</b> | <p>[Output] Number of B frames between the reference frames.</p>   |
| UCHAR | <b>bPreviewFlipped</b>     | <p>[Output]<br/>0x00 = No Change<br/>0x01 = Horizontal Flipped Image for non-H.264 streams.<br/>0x02-0xFF = Reserved</p>   |
| UCHAR | <b>bView</b>               | <p>[Output] Number of additional MVC Views.<br/>0x00: none</p>   |



| Type   | Parameter                 | Description  |
|--------|---------------------------|--|
| UCHAR  | <b>bReserved1</b>         | Reserved - set to zero   |
| UCHAR  | <b>bReserved2</b>         | Reserved - set to zero   |
| UCHAR  | <b>bStreamID</b>          | [Output]<br>0x00-0x06 = Simulcast stream index<br>0x07-0xFF = Reserved   |
| USHORT | <b>bSpatialLayerRatio</b> | [Output] Specifies the ratio between each spatial layer.<br>The high nibble is defined for the integer part and low nibble is for the fractional part. It is represented in fixed point.<br>Example:<br>For 1.5 ratio bSpatialLayerRatio = 0x18<br>For 2.0 ratio bSpatialLayerRatio = 0x20 |

Table 3-89 Definition of **UVCX\_VIDEO\_CONFIG\_S**.

#### 3.4.1.13 USB\_HOST\_UVC\_CB\_s

| Type             | Parameter          | Description  |
|------------------|--------------------|--|
| Function Pointer | <b>MediaInsert</b> | [Input] callback function of the device insert event. See Section 3.4.2.2 for details. |
| Function Pointer | <b>MediaRemove</b> | [Input] callback function of the device remove event. See Section 3.4.2.3 for details. |

Table 3-90 Definition of **USB\_HOST\_UVC\_CB\_s**.

#### 3.4.1.14 USB\_HOST\_UVC\_MULTI\_CB\_s

| Type             | Parameter          | Description  |
|------------------|--------------------|--|
| Function Pointer | <b>MediaInsert</b> | [Input] callback function of the device insert event. See Section 3.4.2.4 for details. |



| Type             | Parameter          | Description  |
|------------------|--------------------|--|
| Function Pointer | <b>MediaRemove</b> | [Input] callback function of the device remove event. See Section 3.4.2.5 for details. |

Table 3-91 *Definition of **USB\_HOST\_UVC\_MULTI\_CB\_s**.*

#### 3.4.1.15 **UVCH\_ALTERNATE\_SETTING\_s**

| Type   | Parameter               | Description  |
|--------|-------------------------|--|
| UINT32 | <b>AlternateSetting</b> | [Output] alternate setting number.   |
| UINT32 | <b>MaxPacketSize</b>    | [Output] maximum packet size in bytes of this alternate setting.               |
| UINT32 | <b>BandWidth</b>        | [Output] maximum packet bandwidth in micro-seconds for this alternate setting. |

Table 3-92 *Definition of **UVCH\_ALTERNATE\_SETTING\_s**.*

#### 3.4.1.16 **UX\_EHCI\_ISO\_REQUEST**

| Type   | Parameter      | Description                             |
|--------|----------------|---|
| UCHAR* | buf            | Buffer pointer to store received packet |
| ULONG  | status         | Transfer result                         |
| USHORT | request_length | Request length                          |

Table 3-93 *Definition of **UX\_EHCI\_ISO\_REQUEST**.*

| Return Value | Description           |
|--------------|-----------------------|
| 0            | UX_SUCCESS            |
| 0x22         | UX_TRANSFER_NO_ANSWER |
| 0x23         | UX_TRANSFER_ERROR     |

Table 3-94 *Return value of **status** in data structure **UX\_EHCI\_ISO\_REQUEST**.*



### 3.4.2 Callback Functions

#### 3.4.2.1 complete\_func

##### API Syntax:

```
void complete_func (UX_EHCI_ISO_REQUEST *request)
```

##### Function Description:

This function is called when a transfer request is complete.

##### Parameters:

Please refer to Section 3.4.1.16 for Parameter **request**.

##### Return:

None

##### Example:

None

##### See Also:

None

#### 3.4.2.2 USB\_HOST\_UVC\_CB\_s - MediaInsert

##### API Syntax:

```
void MediaInsert (void)
```

**Function Description:**

This function is called when a UVC device is inserted and recognized. This function is registered by ***AmbaUSBH\_Uvc\_RegisterCallback()*** (Section 3.4.15) via the data structure ***USB\_HOST\_UVC\_CB\_s*** (Section 3.4.1.13).

**Parameters:**

None

**Return:**

None

**Example:**

None

**See Also:**

None

### 3.4.2.3    **USB\_HOST\_UVC\_CB\_s - MediaRemove**

**API Syntax:**

```
void MediaRemove (void)
```

**Function Description:**

This function is called when a UVC device is removed. This function is registered by ***AmbaUSBH\_Uvc\_RegisterCallback()*** (Section 3.4.15) via the data structure ***USB\_HOST\_UVC\_CB\_s*** (Section 3.4.1.13).

**Parameters:**

None

**Return:**

None



**Example:**

None

**See Also:**

None

#### 3.4.2.4 **USB\_HOST\_UVC\_MULTI\_CB\_s – MedialInsert**

**API Syntax:**

```
void MedialInsert(UINT32 id)
```

**Function Description:**

This function is called when a UVC device is inserted and recognized. This function is registered by *AmbaUSBH\_Uvc\_RegisterCallbackEx()* (Section 3.4.79) via the data structure **USB\_HOST\_UVC\_MULTI\_CB\_s** (Section 3.4.1.14).

**Parameters:**

| Type   | Parameters | Description                            |
|--------|------------|--|
| UINT32 | id         | [Output] The event of the [id] device. |

Table 3-95 Parameters for AmbaUSB API **MedialInsert()**.

**Return:**

None

**Example:**

None

**See Also:**

None



### 3.4.2.5 USB\_HOST\_UVC\_MULTI\_CB\_s – MediaRemove

#### API Syntax:

```
void MediaRemove (UINT32 id)
```

#### Function Description:

This function is called when a UVC device is removed. This function is registered by `AmbaUSBH_Uvc_RegisterCallbackEx()` (Section 3.4.79) via the data structure `USB_HOST_UVC_MULTI_CB_s` (Section 3.4.1.14).

#### Parameters:

| Type   | Parameters | Description                            |
|--------|------------|--|
| UINT32 | id         | [Output] The event of the [id] device. |

Table 3-96 Parameters for AmbaUSB API `MediaRemove()`.

#### Return:

None

#### Example:

None

#### See Also:

None



### 3.4.3 AmbaUSBH\_Uvc\_GetDeviceInfo

#### API Syntax:

```
UINT32 AmbaUSBH_Uvc_GetDeviceInfo (UVCH_DEVICE_INFO *info)
```

#### Function Description:

The application calls this API to get the UVC device information.

#### Parameters:

| Type               | Parameters | Description  |
|--------------------|------------|--|
| UVCH_DEVICE_INFO * | info       | [Output] UVC device information. Please refer to Section 3.4.1.1 for more details. |

Table 3-97 Parameters for AmbaUSB API **AmbaUSBH\_Uvc\_GetDeviceInfo ()**.

#### Return:

| Return     | Description |
|------------|-------------|
| 0          | Success     |
| 0xFFFFFFFF | Failure     |

Table 3-98 Returns for AmbaUSB API **AmbaUSBH\_Uvc\_GetDeviceInfo ()**.

#### Example:

```
static char *uvch_get_format_subtype_string(ULONG subtype)
{
    switch (subtype) {
        case UVC_DESC_SUBTYPE_VS_FORMAT_UNCOMPRESSED:
            return "UNCOMPRESSED";
        case UVC_DESC_SUBTYPE_VS_FORMAT_MJPEG:
            return "MJPEG";
        default:
            return "Unknown";
    }
}
```



```
static void uvch_print_frame_info(UVCH_FRAME_INFO *frame_info)
{
    ULONG interval, fps;

    if (frame_info == NULL) {
        return;
    }

    AmbaPrint("      Index: %d", frame_info->index);
    AmbaPrint("      Resolution: %dx%d", frame_info->width,
frame_info->height);

    interval = frame_info->default_frame_interval;
    if (interval != 0) {
        fps = 10000000 / interval;
    }

    AmbaPrint("      Default Frame Interval: 0x%X(%d FPS)", interval, fps);
    AmbaPrint("      Interval Type: %d", frame_info->interval_type);
    if ((frame_info->interval_type == 0) && (frame_info->interval != NULL))
    {
        // continuous interval
        AmbaPrint("          Min: 0x%X", frame_info->interval[0]);
        AmbaPrint("          Max: 0x%X", frame_info->interval[1]);
        AmbaPrint("          Step: 0x%X", frame_info->interval[2]);
    } else if (frame_info->interval != NULL) {
        // discrete interval
        ULONG idx;
        for (idx = 0; idx < frame_info->interval_type; idx++) {
            interval = frame_info->interval[idx];
            if (interval) {
                fps = 10000000/interval;
            } else {

```



```
    fps = 0;
}
AmбаPrint("      Interval: 0x%X(%d FPS)", interval, fps);
}
}

static void uvch_print_device_info(UVCH_DEVICE_INFO *dev_info)
{
ULONG format_idx, frame_idx;

AmбаPrint("UVC device info:");
AmбаPrint("  PU ID: %d", dev_info->pu_id);
AmбаPrint("  IT ID: %d", dev_info->it_id);
AmбаPrint("  XU ID: %d", dev_info->xu_id);
AmбаPrint("  Format Number: %d", dev_info->format_count);

for (format_idx = 0; format_idx < dev_info->format_count; format_idx++)
{
    UVCH_FORMAT_INFO *format_info = &dev_info->format[format_idx];
    if (format_info != NULL) {
        AmбаPrint("  Format %d:", format_idx);
        AmбаPrint("    Index: %d", format_info->index);
        AmбаPrint("    SubType: %d(%s)", format_info->subtype,
uvch_get_format_subtype_string(format_info->subtype));
        AmбаPrint("    Frame Number: %d", format_info->frame_count);
        for (frame_idx = 0; frame_idx < format_info->frame_count; frame_idx++)
{
            UVCH_FRAME_INFO *frame_info = &format_info->frame[frame_idx];
            AmбаPrint("      Frame %d:", frame_idx);
            uvch_print_frame_info(frame_info);
        }
    }
}

}

For Confidential Only
```



```
}
```

```
void AppUvch_GetDeviceInfo(void)
{
    UINT32 nRet = OK;
    UVCH_DEVICE_INFO dev_info;
    dev_info.size = sizeof(UVCH_DEVICE_INFO);
    nRet = AmbaUSBH_Uvc_GetDeviceInfo (&dev_info);
    if (nRet == OK) {
        uvch_print_device_info(&dev_info);
    } else {
        AmbaPrint("[UVCH] can't get device info, code 0x%X", nRet);
    }
}
```

See Also:

[AmbaUSBH\\_Uvc\\_GetDeviceInfoEx\(\)](#)



### 3.4.4 AmbaUSBH\_Uvc\_GetItControlInfo

#### API Syntax:

```
void AmbaUSBH_Uvc_GetItControlInfo (UVCH_CONTROL_ITEM *item,  
                                     UVC_CAMERA_TERMINAL_CONTROL_SELECTOR selector)
```

#### Function Description:

The application calls this API to get the Input Terminal control information for the UVC device.

#### Parameters:

| Type                                 | Parameters | Description   |
|--------------------------------------|------------|---|
| UVCH_CONTROL_ITEM *                  | item       | [Output] UVC control information. Please refer to Section 3.4.1.5 for more details. |
| UVC_CAMERA_TERMINAL_CONTROL_SELECTOR | selector   | [Input] Index for the control. Please refer to Section 3.4.1.7 for more details.    |

Table 3-99 Parameters for AmbaUSB API **AmbaUSBH\_Uvc\_GetItControlInfo()**.

#### Return:

| Return     | Description |
|------------|-------------|
| 0          | Success     |
| 0xFFFFFFFF | Failure     |

Table 3-100 Returns for AmbaUSB API **AmbaUSBH\_Uvc\_GetItControlInfo()**.

#### Example:

```
static UINT16 u8_to_u16(UINT8 *ptr)  
{  
    UINT16 val;  
    val = ptr[0];  
    val += (ptr[1] << 8);  
    return val;  
}
```



```
static UINT32 u8_to_u32(UINT8 *ptr)
{
    UINT32 val;
    val = ptr[0];
    val += (ptr[1] << 8);
    val += (ptr[2] << 16);
    val += (ptr[3] << 24);
    return val;
}

static INT16 u8_to_s16(UINT8 *ptr)
{
    return (INT16)u8_to_u16(ptr);
}

static INT32 u8_to_s32(UINT8 *ptr)
{
    return (INT32)u8_to_u32(ptr);
}

static void uvch_print_control_item(UVCH_CONTROL_ITEM *item, char *name)
{
    if (item->is_supported == 0) {
        AmbaPrint("%s: Not Supported", name);
        return;
    }

    switch (item->size) {
        case 1:
            AmbaPrint("%s: { %d - %d, def %d, cur %d} ",
                      name,
                      item->min[0],
                      item->max[0],
                      item->def[0],

```



```
        item->cur[0]

    );
    break;
case 2:
    AmbaPrint("%s: { %d - %d, def %d, cur %d} ",
              name,
              u8_to_s16(item->min),
              u8_to_s16(item->max),
              u8_to_s16(item->def),
              u8_to_s16(item->cur)
            );
    break;
case 4:
    AmbaPrint("%s: { %d - %d, def %d, cur %d} ",
              name,
              u8_to_s32(item->min),
              u8_to_s32(item->max),
              u8_to_s32(item->def),
              u8_to_s32(item->cur)
            );
    break;
default:
    AmbaPrint("%s: size %d unsupported", name, item->size);
    break;
}
}

static void uvch_print_it_control(UVCH_CONTROL_ITEM *item,
                                  UVC_CAMERA_TERMINAL_CONTROL_SELECTOR
selector)
{
    UINT32 status = AmbaUSBH_Uvc_GetItControlInfo (item, selector);
    if (status != OK) {
        AmbaPrint("Can't get UVC IT control %d, code 0x%X",
                  selector, status);
    }
}
```



```
    return;
}

switch (selector) {
    case UVC_CT_AE_MODE_CONTROL:
        uvch_print_control_item(item, "AE Mode");
        break;
    case UVC_CT_SCANNING_MODE_CONTROL:
        uvch_print_control_item(item, "Scanning Mode");
        break;
    default:
        AmbaPrint("Unknown IT control: %d", selector);
}
}

// main function
// allocate memory for control values.
UVCH_CONTROL_ITEM item;
UINT8 max_buffer[20];
UINT8 min_buffer[20];
UINT8 def_buffer[20];
UINT8 cur_buffer[20];
item.max = max_buffer;
item.min = min_buffer;
item.def = def_buffer;
item.cur = cur_buffer;
item.buffer_size = 20;
// get and print AE Mode control
uvch_print_it_control(&item, UVC_CT_AE_MODE_CONTROL);
// get and print Scanning Mode control
uvch_print_it_control(&item, UVC_CT_SCANNING_MODE_CONTROL);
```

**See Also:**

[AmbeUSBH\\_Uvc\\_GetItControlInfoEx\(\)](#)



### 3.4.5 AmbaUSBH\_Uvc\_GetPuControlInfo

#### API Syntax:

```
void AmbaUSBH_Uvc_GetPuControlInfo (UVCH_CONTROL_ITEM *item,  
                                     UVC_PROCESSING_UNIT_CONTROL_SELECTOR selector)
```

#### Function Description:

The application calls this API to get Processing Unit control information for the UVC device.

#### Parameters:

| Type                                 | Parameters | Description   |
|--------------------------------------|------------|---|
| UVCH_CONTROL_ITEM *                  | item       | [Output] UVC control information. Please refer to Section 3.4.1.5 for more details. |
| UVC_PROCESSING_UNIT_CONTROL_SELECTOR | selector   | [Input] Index for the control. Please refer to Section 3.4.1.8 for more details.    |

Table 3-101 Parameters for AmbaUSB API **AmbaUSBH\_Uvc\_GetPuControlInfo()**.

#### Return:

| Return     | Description |
|------------|-------------|
| 0          | Success     |
| 0xFFFFFFFF | Failure     |

Table 3-102 Returns for AmbaUSB API **AmbaUSBH\_Uvc\_GetPuControlInfo()**.

#### Example:

```
static UINT16 u8_to_u16(UINT8 *ptr)  
{  
    UINT16 val;  
    val = ptr[0];  
    val += (ptr[1] << 8);  
    return val;  
}
```



```
static UINT32 u8_to_u32(UINT8 *ptr)
{
    UINT32 val;
    val = ptr[0];
    val += (ptr[1] << 8);
    val += (ptr[2] << 16);
    val += (ptr[3] << 24);
    return val;
}

static INT16 u8_to_s16(UINT8 *ptr)
{
    return (INT16)u8_to_u16(ptr);
}

static INT32 u8_to_s32(UINT8 *ptr)
{
    return (INT32)u8_to_u32(ptr);
}

static void uvch_print_control_item(UVCH_CONTROL_ITEM *item, char *name)
{
    if (item->is_supported == 0) {
        AmbaPrint("%s: Not Supported", name);
        return;
    }

    switch (item->size) {
        case 1:
            AmbaPrint("%s: { %d - %d, def %d, cur %d} ",
                      name,
                      item->min[0],
                      item->max[0],
                      item->def[0],

```



```
    item->cur[0]

);

break;

case 2:

AmbaPrint("%s: { %d - %d, def %d, cur %d}",
           name,
           u8_to_s16(item->min),
           u8_to_s16(item->max),
           u8_to_s16(item->def),
           u8_to_s16(item->cur)
);

break;

case 4:

AmbaPrint("%s: { %d - %d, def %d, cur %d}",
           name,
           u8_to_s32(item->min),
           u8_to_s32(item->max),
           u8_to_s32(item->def),
           u8_to_s32(item->cur)
);

break;

default:

AmbaPrint("%s: size %d unsupported", name, item->size);
break;
}

}

static void uvch_print_pu_control(UVCH_CONTROL_ITEM *item,
                                  UVC_PROCESSING_UNIT_CONTROL_SELECTOR
selector)

{
    UINT32 status = AmbaUSBH_Uvc_GetPuControlInfo (item, selector);

    if (status != OK) {

        AmbaPrint("Can't get UVC PU control %d, code 0x%X", selector, status);
    }
}
```



```
    return;  
}  
  
switch (selector) {  
    case UVC_PU_BRIGHTNESS_CONTROL:  
        uvch_print_control_item(item, "Brightness");  
        break;  
    case UVC_PU_CONTRAST_CONTROL:  
        uvch_print_control_item(item, "Contrast");  
        break;  
    case UVC_PU_HUE_CONTROL:  
        uvch_print_control_item(item, "Hue");  
        break;  
    case UVC_PU_SATURATION_CONTROL:  
        uvch_print_control_item(item, "Saturation");  
        break;  
    case UVC_PU_SHARPNESS_CONTROL:  
        uvch_print_control_item(item, "Sharpness");  
        break;  
    case UVC_PU_GAMMA_CONTROL:  
        uvch_print_control_item(item, "Gamma");  
        break;  
    case UVC_PU_HUE_AUTO_CONTROL:  
        uvch_print_control_item(item, "Hue Auto");  
        break;  
    default:  
        AmbaPrint("Unknown PU control: %d", selector);  
}  
}  
  
// main function  
// allocate memory for control values.  
UVCH_CONTROL_ITEM item;  
UINT8 max_buffer[20];
```



```
UINT8 min_buffer[20];
UINT8 def_buffer[20];
UINT8 cur_buffer[20];
item.max = max_buffer;
item.min = min_buffer;
item.def = def_buffer;
item.cur = cur_buffer;
item.buffer_size = 20;
// get and print Brightness control
uvch_print_pu_control(&item, UVC_PU_BRIGHTNESS_CONTROL);
// get and print Contrast control
uvch_print_pu_control(&item, UVC_PU_CONTRAST_CONTROL);
// get and print Saturation control
uvch_print_pu_control(&item, UVC_PU_SATURATION_CONTROL);
// get and print Hue control
uvch_print_pu_control(&item, UVC_PU_HUE_CONTROL);
// get and print Sharpness control
uvch_print_pu_control(&item, UVC_PU_SHARPNESS_CONTROL);
```

**See Also:**

[\*\*AmbaUSBH\\_Uvc\\_GetPuControlInfoEx\(\)\*\*](#)



### 3.4.6 **AmbaUSBH\_Uvc\_PrintDeviceValues**

#### API Syntax:

```
void AmbaUSBH_Uvc_PrintDeviceValues (void)
```

#### Function Description:

The application calls this API to print the values retrieved by the UVC device.

#### Parameters:

None

#### Return:

None

#### Example:

None

#### See Also :

*AmbaUSBH\_Uvc\_PrintDeviceValuesEx()*



### 3.4.7 **AmbaUSBH\_Uvc\_ProbeAndCommit**

#### API Syntax:

```
void AmbaUSBH_Uvc_ProbeAndCommit (ULONG formatIndex,  
                                     ULONG frameIndex,  
                                     ULONG fps)
```

#### Function Description:

The application calls this API to probe and commit the stream format.

#### Parameters:

| Type  | Parameters  | Description                        |
|-------|-------------|------------------------------------|
| ULONG | formatIndex | [Input] Format index               |
| ULONG | frameIndex  | [Input] Frame index                |
| ULONG | fps         | [Input] Preferred frame per second |

Table 3-103 Parameters for AmbaUSB API **AmbaUSBH\_Uvc\_ProbeAndCommit()**.

#### Return:

None

#### Example:

None

#### See Also :

[\*\*AmbaUSBH\\_Uvc\\_ProbeAndCommitEx\(\)\*\*](#)



### 3.4.8 **AmbaUSBH\_Uvc\_GetCurrentProbeSetting**

#### API Syntax:

```
void AmbaUSBH_Uvc_GetCurrentProbeSetting(UINT32 *formatIndex,  
                                         UINT32 *frameIndex,  
                                         UINT32 *fps)
```

#### Function Description:

The application calls this API to get the probe values.

#### Parameters:

| Type    | Parameters  | Description                         |
|---------|-------------|-------------------------------------|
| UINT32* | formatIndex | [Output] Format index               |
| UINT32* | frameIndex  | [Output] Frame index                |
| UINT32* | fps         | [Output] Preferred frame per second |

Table 3-104 Parameters for AmbaUSB API **AmbaUSBH\_Uvc\_GetCurrentProbeSetting()**.

#### Return:

None

#### Example:

None

#### See Also :

[\*\*AmbaUSBH\\_Uvc\\_GetCurrentProbeSettingEx\(\)\*\*](#)



### 3.4.9 AmbaUSBH\_Uvc\_SetCurValues

#### API Syntax:

```
void AmbaUSBH_Uvc_SetCurValues (ULONG unit, ULONG selector, ULONG value)
```

#### Function Description:

The application calls this API to set the current information for a unit/terminal.

#### Parameters:

| Type  | Parameters | Description                               |
|-------|------------|---|
| ULONG | unit       | [Input] Unit/terminal ID                  |
| ULONG | selector   | [Input] Selector ID for the unit/terminal |
| ULONG | value      | [Input] Value for the selector            |

Table 3-105 Parameters for AmbaUSB API **AmbaUSBH\_Uvc\_SetCurValues()**.

#### Return:

None

#### Example:

#### See Also :

**AmbaUSBH\_Uvc\_SetCurValuesEx( )**



### 3.4.10 AmbaUSBH\_Uvc\_StreamingStart

#### API Syntax:

```
void AmbaUSBH_Uvc_StreamingStart (void)
```

#### Function Description:

The application calls this API to start video streaming.

#### Parameters:

None

#### Return:

None

#### Example:

None

#### See Also :

*AmbaUSBH\_Uvc\_StreamingStartEx()*



### 3.4.11 AmbaUSBH\_Uvc\_StreamingStop

#### API Syntax:

```
void AmbaUSBH_Uvc_StreamingStop (void)
```

#### Function Description:

The application calls this API to stop video streaming.

#### Parameters:

None

#### Return:

None

#### Example:

None

#### See Also :

*AmbaUSBH\_Uvc\_StreamingStopEx( )*



### 3.4.12 AmbaUSBH\_Uvc\_Read

Deprecated

For PROTRULY Confidential Only



### 3.4.13 AmbaUSBH\_Uvc\_ReadBlock

#### API Syntax:

```
UINT AmbaUSBH_Uvc_ReadBlock (UX_EHCI_ISO_REQUEST *request,  
                           UINT32 PacketNum,  
                           void (*complete_func)(UX_EHCI_ISO_REQUEST *request))
```

#### Function Description:

The application calls this API to receive video streaming.

#### Parameters:

| Type                 | Parameters    | Description   |
|----------------------|---------------|---|
| UX_EHCI_ISO_REQUEST* | request       | [Input] Pointer of request array. See Section 3.4.1.16 for details. |
| UINT32               | PacketNum     | [Input] Packet number of request array                              |
| Callback function    | complete_func | [Input] See Section 3.4.2.1   |

Table 3-106 Parameters for AmbaUSB API **AmbaUSBH\_Uvc\_ReadBlock()**

#### Return:

| Return | Description            |
|--------|------------------------|
| 0      | Success                |
| 0x12   | Memory Insufficient    |
| 0x13   | No TD available        |
| 0x1C   | Parameters error       |
| 0x25   | Transfer not ready     |
| 0x41   | No bandwidth available |
| 0x53   | No streaming endpoint  |
| 0x54   | Function not support   |
| 0x5B   | Unknown class instance |
| 0x5E   | No alternate setting   |



| Return | Description |
|--------|-------------|
| 0xFF   | Error       |

Table 3-107 Returns for API *AmbaUSBH\_Uvc\_ReadBlock()*.

**Example:**

```
#define UVCH_MAX_PACKET_NUM 2000  
  
#define UVCH_MAX_PACKET_SIZE 3072  
  
#define UVCH_REQUEST_ARRAY_SIZE  
UVCH_MAX_PACKET_NUM*sizeof(UX_EHCI_ISO_REQUEST)  
  
#define UVCH_READ_BUFF_SIZE UVCH_MAX_PACKET_SIZE* UVCH_MAX_PACKET_NUM  
  
// Allocate the data buffers and request buffers.  
  
AmbaKAL_MemAllocate(&AmбаBytePool_Cached, &uvch_read_buffer,  
UVCH_READ_BUFF_SIZE, 32);  
  
AmbaKAL_MemAllocate(&AmбаBytePool_Cached, &uvch_request,  
UVCH_REQUEST_ARRAY_SIZE, 32);  
  
// Fill the request array with responded data.  
  
UX_EHCI_ISO_REQUEST *TmpRequest = uvch_request. pMemAlignedBase;  
UINT8 * ReadBuf = uvch_read_buffer. pMemAlignedBase;  
for (i=0; i< UVCH_MAX_PACKET_NUM; i++) {  
  
    TmpRequest->buf = (UCHAR*)(ReadBuf + UVCH_MAX_PACKET_SIZE * i);  
  
    TmpRequest->request_length = UVCH_MAX_PACKET_SIZE;  
  
    TmpRequest++;  
}
```

**See Also :**

***AmbaUSBH\_Uvc\_ReadBlockEx()***



### 3.4.14 AmbaUSBH\_Uvc\_ReadAppend

#### API Syntax:

```
UINT AmbaUSBH_Uvc_ReadAppend (UX_EHCI_ISO_REQUEST *request, UINT32  
PacketNum, void (*complete_func)(UX_EHCI_ISO_REQUEST *request))
```

#### Function Description:

The application calls this API to receive video streaming successively and USB stack would make sure the received data between every transfer request are continuous. USB stack also makes the necessary delay which may block the task.

#### Parameters:

See *AmbaUSBH\_Uvc\_ReadBlock()*.

#### Return:

See *AmbaUSBH\_Uvc\_ReadBlock()*.

#### Example:

See *AmbaUSBH\_Uvc\_ReadBlock()*.

#### See Also:

See *AmbaUSBH\_Uvc\_ReadBlock()*.

See *AmbaUSBH\_Uvc\_ReadAppendEx()*



### 3.4.15 AmbaUSBH\_Uvc\_GetMaxPacketSize

#### API Syntax:

```
UINT AmbaUSBH_Uvc_GetMaxPacketSize (void)
```

#### Function Description:

The application calls this API to get the current maximum packet size.

#### Parameters:

None

#### Return:

| Return | Description             |
|--------|-------------------------|
| UINT   | The maximum packet size |

Table 3-108 Returns of API *AmbaUSBH\_Uvc\_GetMaxPacketSize()*.

#### Example:

None

#### See Also :

*AmbaUSBH\_Uvc\_GetMaxPacketSizeEx()*



### 3.4.16 AmbaUSBH\_Uvc\_RegisterCallback

#### API Syntax:

```
UINT32 AmbaUSBH_Uvc_RegisterCallback (USB_HOST_UVC_CB_s *cb)
```

#### Function Description:

The application calls this API to register the callback for the camera insertion/removal events.

#### Parameters:

| Type               | Parameters | Description  |
|--------------------|------------|--|
| USB_HOST_UVC_CB_s* | cb         | Pointer to callback functions. See Section 3.4.1.13 for details. |

Table 3-109 Parameters for AmbaUSB API *AmbaUSBH\_Uvc\_RegisterCallback()*.

#### Return:

| Return Value | Description        |
|--------------|--------------------|
| 0            | UX_SUCCESS         |
| 0x1c         | UX_PARAMETER_ERROR |

Table 3-110 Returns for API *AmbaUSBH\_Uvc\_RegisterCallback()*.

#### Example:

None

#### See Also :

*AmbaUSBH\_Uvc\_RegisterCallbackEx()*



### 3.4.17 **AmbaUSBH\_Uvc\_GetBacklightCtrlInfo**

#### API Syntax:

```
UINT32 AmbaUSBH_Uvc_GetBacklightCtrlInfo (UVCH_CTRL_INFO_s *info);
```

#### Function Description:

The application calls this API to get the backlight control value.

#### Parameters:

| Type              | Parameters | Description   |
|-------------------|------------|---|
| UVCH_CTRL_INFO_s* | info       | [Output] Pointer to control information. See Section 3.4.1.6 for details. |

Table 3-111 Parameters for AmbaUSB API **AmbaUSBH\_Uvc\_GetBacklightCtrlInfo()**.

#### Return:

| Return Value | Description             |
|--------------|-------------------------|
| 0            | UX_SUCCESS              |
| 0xFFFFFFFF   | Operation not supported |

Table 3-112 Returns for API **AmbaUSBH\_Uvc\_GetCurBacklightCtrl()**.

#### Example:

```
UVCH_CTRL_INFO_s info = {0};  
AmbaUSBH_Uvc_GetBacklightCtrlInfo(&info);
```

#### See Also :

[\*\*AmbaUSBH\\_Uvc\\_GetBacklightCtrlInfoEx\(\)\*\*](#)



### 3.4.18 AmbaUSBH\_Uvc\_SetBacklightCtrl

#### API Syntax:

```
UINT32 AmbaUSBH_Uvc_SetBacklightCtrl (UINT32 ctrl)
```

#### Function Description:

The application calls this API to set the backlight control value, which is in the range defined by `AmbaUSBH_Uvc_GetBacklightCtrlInfo()`.

#### Parameters:

| Type   | Parameters | Description             |
|--------|------------|-------------------------|
| UINT32 | ctrl       | Backlight control value |

Table 3-113 Parameters for AmbaUSB API `AmbaUSBH_Uvc_SetBacklightCtrl()`.

#### Return:

| Return Value | Description            |
|--------------|------------------------|
| 0            | UX_SUCCESS             |
| 0x5F         | UX_NO_DEVICE_CONNECTED |

Table 3-114 Returns for API `AmbaUSBH_Uvc_SetBacklightCtrl()`.

#### Example:

None

#### See Also :

[`AmbaUSBH\_Uvc\_SetBacklightCtrlEx\(\)`](#)



### 3.4.19 **AmbaUSBH\_Uvc\_GetBrightnessCtrlInfo**

#### API Syntax:

```
UINT32 AmbaUSBH_Uvc_GetBrightnessCtrlInfo (UVCH_CTRL_INFO_s *info)
```

#### Function Description:

The application calls this API to get the brightness control value.

#### Parameters:

| Type              | Parameters | Description   |
|-------------------|------------|---|
| UVCH_CTRL_INFO_s* | info       | [Output] Pointer to control information. See Section 3.4.1.6 for details. |

Table 3-115 Parameters for AmbaUSB API **AmbaUSBH\_Uvc\_GetBrightnessCtrlInfo()**.

#### Return:

| Return Value | Description             |
|--------------|-------------------------|
| 0            | UX_SUCCESS              |
| 0xFFFFFFFF   | Operation not supported |

Table 3-116 Returns for API **AmbaUSBH\_Uvc\_GetBrightnessCtrlInfo()**.

#### Example:

```
UVCH_CTRL_INFO_s info = {0};  
AmbaUSBH_Uvc_GetBrightnessCtrlInfo(&info);
```

#### See Also :

[\*\*AmbaUSBH\\_Uvc\\_GetBrightnessCtrlInfoEx\(\)\*\*](#)



### 3.4.20 AmbaUSBH\_Uvc\_SetBrightnessCtrl

#### API Syntax:

```
UINT32 AmbaUSBH_Uvc_SetBrightnessCtrl (UINT32 ctrl)
```

#### Function Description:

The application calls this API to set the brightness control value, which is in the range defined by `AmbaUSBH_Uvc_GetBrightnessCtrlInfo()`.

#### Parameters:

| Type   | Parameters | Description              |
|--------|------------|--------------------------|
| UINT32 | ctrl       | Brightness control value |

Table 3-117 Parameters for AmbaUSB API `AmbaUSBH_Uvc_SetBrightnessCtrl()`.

#### Return:

| Return Value | Description            |
|--------------|------------------------|
| 0            | UX_SUCCESS             |
| 0x5F         | UX_NO_DEVICE_CONNECTED |

Table 3-118 Returns for API `AmbaUSBH_Uvc_SetBrightnessCtrl()`.

#### Example:

None

#### See Also :

`AmbaUSBH_Uvc_SetBrightnessCtrlEx()`



### 3.4.21 **AmbaUSBH\_Uvc\_GetContrastCtrlInfo**

#### API Syntax:

```
UINT32 AmbaUSBH_Uvc_GetContrastCtrlInfo (UVCH_CTRL_INFO_s *info)
```

#### Function Description:

The application calls this API to get the contrast control value.

#### Parameters:

| Type              | Parameters | Description   |
|-------------------|------------|---|
| UVCH_CTRL_INFO_s* | info       | [Output] Pointer to control information. See Section 3.4.1.6 for details. |

Table 3-119 Parameters for AmbaUSB API **AmbaUSBH\_Uvc\_GetContrastCtrlInfo()**.

#### Return:

| Return Value | Description             |
|--------------|-------------------------|
| 0            | UX_SUCCESS              |
| 0xFFFFFFFF   | Operation not supported |

Table 3-120 Returns for API **AmbaUSBH\_Uvc\_GetContrastCtrlInfo()**.

#### Example:

```
UVCH_CTRL_INFO_s info = { 0 };  
AmbaUSBH_Uvc_GetContrastCtrlInfo(&info);
```

#### See Also :

**AmbaUSBH\_Uvc\_GetContrastCtrlInfoEx()**



### 3.4.22 AmbaUSBH\_Uvc\_SetContrastCtrl

#### API Syntax:

```
UINT32 AmbaUSBH_Uvc_SetContrastCtrl (UINT32 ctrl)
```

#### Function Description:

The application calls this API to set the contrast control value, which is in the range defined by `AmbaUSBH_Uvc_GetContrastCtrlInfo()`.

#### Parameters:

| Type   | Parameters | Description            |
|--------|------------|------------------------|
| UINT32 | ctrl       | Contrast control value |

Table 3-121 Parameters for AmbaUSB API `AmbaUSBH_Uvc_SetContrastCtrl()`.

#### Return:

| Return Value | Description            |
|--------------|------------------------|
| 0            | UX_SUCCESS             |
| 0x5F         | UX_NO_DEVICE_CONNECTED |

Table 3-122 Returns for API `AmbaUSBH_Uvc_SetContrastCtrl()`.

#### Example:

None

#### See Also :

`AmbaUSBH_Uvc_SetContrastCtrlEx()`



### 3.4.23 AmbaUSBH\_Uvc\_GetHueCtrlInfo

#### API Syntax:

```
UINT32 AmbaUSBH_Uvc_GetHueCtrlInfo (UVCH_CTRL_INFO_s *info)
```

#### Function Description:

The application calls this API to get the Hue control value.

#### Parameters:

| Type              | Parameters | Description   |
|-------------------|------------|---|
| UVCH_CTRL_INFO_s* | info       | [Output] Pointer to control information. See Section 3.4.1.6 for details. |

Table 3-123 Parameters for AmbaUSB API *AmbaUSBH\_Uvc\_GetHueCtrlInfo()*.

#### Return:

| Return Value | Description             |
|--------------|-------------------------|
| 0            | UX_SUCCESS              |
| 0xFFFFFFFF   | Operation not supported |

Table 3-124 Returns for API *AmbaUSBH\_Uvc\_GetHueCtrlInfo()*.

#### Example:

```
UVCH_CTRL_INFO_s info = {0};  
AmbaUSBH_Uvc_GetHueCtrlInfo(&info);
```

#### See Also :

*AmbaUSBH\_Uvc\_GetHueCtrlInfoEx()*



### 3.4.24 AmbaUSBH\_Uvc\_SetHueCtrl

#### API Syntax:

```
UINT32 AmbaUSBH_Uvc_SetHueCtrl (UINT32 ctrl)
```

#### Function Description:

The application calls this API to set the hue control value, which is in the range defined by [AmbaUSBH\\_Uvc\\_GetHueCtrlInfo\(\)](#).

#### Parameters:

| Type   | Parameters | Description       |
|--------|------------|-------------------|
| UINT32 | ctrl       | Hue control value |

Table 3-125 Parameters for AmbaUSB API [AmbaUSBH\\_Uvc\\_SetHueCtrl\(\)](#).

#### Return:

| Return Value | Description            |
|--------------|------------------------|
| 0            | UX_SUCCESS             |
| 0x5F         | UX_NO_DEVICE_CONNECTED |

Table 3-126 Returns for API [AmbaUSBH\\_Uvc\\_SetHueCtrl\(\)](#).

#### Example:

None

#### See Also :

[AmbaUSBH\\_Uvc\\_SetHueCtrlEx\(\)](#)



### 3.4.25 AmbaUSBH\_Uvc\_GetSaturationCtrlInfo

#### API Syntax:

```
UINT32 AmbaUSBH_Uvc_GetSaturationCtrlInfo (UVCH_CTRL_INFO_s *info)
```

#### Function Description:

The application calls this API to get the saturation control value.

#### Parameters:

| Type              | Parameters | Description   |
|-------------------|------------|---|
| UVCH_CTRL_INFO_s* | info       | [Output] Pointer to control information. See Section 3.4.1.6 for details. |

Table 3-127 Parameters for AmbaUSB API **AmbaUSBH\_Uvc\_GetSaturationCtrlInfo()**.

#### Return:

| Return Value | Description             |
|--------------|-------------------------|
| 0            | UX_SUCCESS              |
| 0xFFFFFFFF   | Operation not supported |

Table 3-128 Returns for API **AmbaUSBH\_Uvc\_GetSaturationCtrlInfo()**.

#### Example:

```
UVCH_CTRL_INFO_s info = {0};  
AmbaUSBH_Uvc_GetSaturationCtrlInfo(&info);
```

#### See Also :

[\*\*AmbaUSBH\\_Uvc\\_GetSaturationCtrlInfoEx\(\)\*\*](#)



### 3.4.26 AmbaUSBH\_Uvc\_SetSaturationCtrl

#### API Syntax:

```
UINT32 AmbaUSBH_Uvc_SetSaturationCtrl (UINT32 ctrl)
```

#### Function Description:

The application calls this API to set the saturation control value, which is in the range defined by [AmbaUSBH\\_Uvc\\_GetSaturationCtrlInfo\(\)](#).

#### Parameters:

| Type   | Parameters | Description              |
|--------|------------|--------------------------|
| UINT32 | ctrl       | Saturation control value |

Table 3-129 Parameters for AmbaUSB API [AmbaUSBH\\_Uvc\\_SetSaturationCtrl\(\)](#).

#### Return:

| Return Value | Description            |
|--------------|------------------------|
| 0            | UX_SUCCESS             |
| 0x5F         | UX_NO_DEVICE_CONNECTED |

Table 3-130 Returns for API [AmbaUSBH\\_Uvc\\_SetSaturationCtrl\(\)](#).

#### Example:

None

#### See Also :

[AmbaUSBH\\_Uvc\\_SetSaturationCtrl\(\)](#)



### 3.4.27 **AmbaUSBH\_Uvc\_GetSharpnessCtrlInfo**

#### API Syntax:

```
UINT32 AmbaUSBH_Uvc_GetSharpnessCtrlInfo (UVCH_CTRL_INFO_s *info)
```

#### Function Description:

The application calls this API to get the sharpness control value.

#### Parameters:

| Type              | Parameters | Description   |
|-------------------|------------|---|
| UVCH_CTRL_INFO_s* | info       | [Output] Pointer to control information. See Section 3.4.1.6 for details. |

Table 3-131 Parameters for AmbaUSB API **AmbaUSBH\_Uvc\_GetSharpnessCtrlInfo()**.

#### Return:

| Return Value | Description             |
|--------------|-------------------------|
| 0            | UX_SUCCESS              |
| 0xFFFFFFFF   | Operation not supported |

Table 3-132 Returns for API **AmbaUSBH\_Uvc\_GetSharpnessCtrlInfo()**.

#### Example:

```
UVCH_CTRL_INFO_s info = {0};  
AmbaUSBH_Uvc_GetSharpnessCtrlInfo(&info);
```

#### See Also :

**AmbaUSBH\_Uvc\_GetSharpnessCtrlInfo()**



### 3.4.28 AmbaUSBH\_Uvc\_SetSharpnessCtrl

#### API Syntax:

```
UINT32 AmbaUSBH_Uvc_SetSharpnessCtrl (UINT32 ctrl)
```

#### Function Description:

The application calls this API to set the sharpness control value, which is in the range defined by `AmbaUSBH_Uvc_GetSharpnessCtrlInfo()`.

#### Parameters:

| Type   | Parameters | Description             |
|--------|------------|-------------------------|
| UINT32 | ctrl       | Sharpness control value |

Table 3-133 Parameters for AmbaUSB API `AmbaUSBH_Uvc_SetSharpnessCtrl()`.

#### Return:

| Return Value | Description            |
|--------------|------------------------|
| 0            | UX_SUCCESS             |
| 0x5F         | UX_NO_DEVICE_CONNECTED |

Table 3-134 Returns for API `AmbaUSBH_Uvc_SetSharpnessCtrl()`.

#### Example:

None

#### See Also :

[`AmbaUSBH\_Uvc\_SetSharpnessCtrl\(\)`](#)



### 3.4.29 AmbaUSBH\_Uvc\_GetGammaCtrlInfo

#### API Syntax:

```
UINT32 AmbaUSBH_Uvc_GetGammaCtrlInfo (UVCH_CTRL_INFO_s *info)
```

#### Function Description:

The application calls this API to get the Gamma control value.

#### Parameters:

| Type              | Parameters | Description   |
|-------------------|------------|---|
| UVCH_CTRL_INFO_s* | info       | [Output] Pointer to control information. See Section 3.4.1.6 for details. |

Table 3-135 Parameters for AmbaUSB API *AmbaUSBH\_Uvc\_GetGammaCtrlInfo()*.

#### Return:

| Return Value | Description             |
|--------------|-------------------------|
| 0            | UX_SUCCESS              |
| 0xFFFFFFFF   | Operation not supported |

Table 3-136 Returns for API *AmbaUSBH\_Uvc\_GetGammaCtrlInfo()*.

#### Example:

```
UVCH_CTRL_INFO_s info = { 0 };  
AmbaUSBH_Uvc_GetGammaCtrlInfo(&info);
```

#### See Also :

*AmbaUSBH\_Uvc\_GetGammaCtrlInfo()*



### 3.4.30 AmbaUSBH\_Uvc\_SetGammaCtrl

#### API Syntax:

```
UINT32 AmbaUSBH_Uvc_SetGammaCtrl (UINT32 ctrl)
```

#### Function Description:

The application calls this API to set the gamma control value, which is in the range defined by [AmbaUSBH\\_Uvc\\_GetGammaCtrlInfo\(\)](#).

#### Parameters:

| Type   | Parameters | Description         |
|--------|------------|---------------------|
| UINT32 | ctrl       | Gamma control value |

Table 3-137 Parameters for AmbaUSB API [AmbaUSBH\\_Uvc\\_SetGammaCtrl\(\)](#).

#### Return:

| Return Value | Description            |
|--------------|------------------------|
| 0            | UX_SUCCESS             |
| 0x5F         | UX_NO_DEVICE_CONNECTED |

Table 3-138 Returns for API [AmbaUSBH\\_Uvc\\_SetGammaCtrl\(\)](#).

#### Example:

None

#### See Also :

[AmbaUSBH\\_Uvc\\_SetGammaCtrl\(\)](#)



### 3.4.31 **AmbaUSBH\_Uvc\_GetWBTemperatureCtrlInfo**

#### API Syntax:

```
UINT32 AmbaUSBH_Uvc_GetWBTemperatureCtrlInfo (UVCH_CTRL_INFO_s *info)
```

#### Function Description:

The application calls this API to get the white balance temperature control value.

#### Parameters:

| Type              | Parameters | Description   |
|-------------------|------------|---|
| UVCH_CTRL_INFO_s* | info       | [Output] Pointer to control information. See Section 3.4.1.6 for details. |

Table 3-139 Parameters for AmbaUSB API **AmbaUSBH\_Uvc\_GetWBTemperatureCtrlInfo()**.

#### Return:

| Return Value | Description             |
|--------------|-------------------------|
| 0            | UX_SUCCESS              |
| 0xFFFFFFFF   | Operation not supported |

Table 3-140 Returns for API **AmbaUSBH\_Uvc\_GetWBTemperatureCtrlInfo()**.

#### Example:

```
UVCH_CTRL_INFO_s info = {0};  
AmbaUSBH_Uvc_GetWBTemperatureCtrlInfo(&info);
```

#### See Also :

[\*\*AmbaUSBH\\_Uvc\\_GetWBTemperatureCtrlInfo\(\)\*\*](#)



### 3.4.32 AmbaUSBH\_Uvc\_SetWBTemperatureCtrl

#### API Syntax:

```
UINT32 AmbaUSBH_Uvc_SetWBTemperatureCtrl (UINT32 ctrl)
```

#### Function Description:

The application calls this API to set the white balance temperature control value, which is in the range defined by `AmbaUSBH_Uvc_GetWBTemperatureCtrlInfo()`.

#### Parameters:

| Type   | Parameters | Description                             |
|--------|------------|---|
| UINT32 | ctrl       | White balance temperature control value |

Table 3-141 Parameters for AmbaUSB API `AmbaUSBH_Uvc_SetWBTemperatureCtrl()`.

#### Return:

| Return Value | Description            |
|--------------|------------------------|
| 0            | UX_SUCCESS             |
| 0x5F         | UX_NO_DEVICE_CONNECTED |

Table 3-142 Returns for API `AmbaUSBH_Uvc_SetWBTemperatureCtrl()`.

#### Example:

None

#### See Also :

[`AmbaUSBH\_Uvc\_SetWBTemperatureCtrl\(\)`](#)



### 3.4.33 AmbaUSBH\_Uvc\_GetWBComponentCtrlInfo

#### API Syntax:

```
UINT32 AmbaUSBH_Uvc_GetWBComponentCtrlInfo (UVCH_CTRL_INFO_s *InfoBlue,  
                                            UVCH_CTRL_INFO_s *InfoRed)
```

#### Function Description:

The application calls this API to get the white balance component control value.

#### Parameters:

| Type              | Parameters | Description   |
|-------------------|------------|---|
| UVCH_CTRL_INFO_s* | InfoBlue   | [Output] Pointer to control information. See Section 3.4.1.6 for details. |
| UVCH_CTRL_INFO_s* | InfoRed    | [Output] Pointer to control information. See Section 3.4.1.6 for details. |

Table 3-143 Parameters for AmbaUSB API **AmbaUSBH\_Uvc\_GetWBComponentCtrlInfo()**.

#### Return:

| Return Value | Description             |
|--------------|-------------------------|
| 0            | SUCCESS                 |
| 0xFFFFFFFF   | Operation not supported |

Table 3-144 Returns for API **AmbaUSBH\_Uvc\_GetWBComponentCtrlInfo()**.

#### Example:

```
UVCH_CTRL_INFO_s InfoBlue = { 0 } ;  
UVCH_CTRL_INFO_s InfoRed = { 0 } ;  
AmbaUSBH_Uvc_GetWBComponentCtrlInfo(&InfoBlue, &InfoRed) ;
```

#### See Also:

**AmbaUSBH\_Uvc\_GetWBComponentCtrlInfo()**



### 3.4.34 AmbaUSBH\_Uvc\_SetWBComponentCtrl

#### API Syntax:

```
UINT32 AmbaUSBH_Uvc_SetWBComponentCtrl(UINT16 blue, UINT16 red)
```

#### Function Description:

The application calls this API to set the white balance component control value, which is in the range defined by *AmbaUSBH\_Uvc\_GetWBComponentCtrlInfo*.

#### Parameters:

| Type   | Parameters | Description          |
|--------|------------|----------------------|
| UINT16 | blue       | Blue component value |
| UINT16 | red        | Red component value  |

Table 3-145 Parameters for AmbaUSB API *AmbaUSBH\_Uvc\_SetWBComponentCtrl()*.

#### Return:

| Return Value | Description            |
|--------------|------------------------|
| 0            | UX_SUCCESS             |
| 0x5F         | UX_NO_DEVICE_CONNECTED |

Table 3-146 Returns for API *AmbaUSBH\_Uvc\_SetWBComponentCtrl()*.

#### Example:

None

#### See Also :

*AmbaUSBH\_Uvc\_SetWBComponentCtrl()*



### 3.4.35 AmbaUSBH\_Uvc\_GetGainCtrlInfo

#### API Syntax:

```
UINT32 AmbaUSBH_Uvc_GetGainCtrlInfo (UVCH_CTRL_INFO_s *info)
```

#### Function Description:

The application calls this API to get the gain control value.

#### Parameters:

| Type              | Parameters | Description   |
|-------------------|------------|---|
| UVCH_CTRL_INFO_s* | info       | [Output] Pointer to control information. See Section 3.4.1.6 for details. |

Table 3-147 Parameters for AmbaUSB API *AmbaUSBH\_Uvc\_GetGainCtrlInfo()*.

#### Return:

| Return Value | Description             |
|--------------|-------------------------|
| 0            | UX_SUCCESS              |
| 0xFFFFFFFF   | Operation not supported |

Table 3-148 Returns for API *AmbaUSBH\_Uvc\_GetGainCtrlInfo()*.

#### Example:

```
UVCH_CTRL_INFO_s info = {0};  
AmbaUSBH_Uvc_GetGainCtrlInfo(&info);
```

#### See Also :

*AmbaUSBH\_Uvc\_GetGainCtrlInfo()*



### 3.4.36 AmbaUSBH\_Uvc\_SetGainCtrl

#### API Syntax:

```
UINT32 AmbaUSBH_Uvc_SetGainCtrl (UINT32 ctrl)
```

#### Function Description:

The application calls this API to set the gain control value, which is in the range defined by `AmbaUSBH_Uvc_GetGainCtrlInfo()`.

#### Parameters:

| Type   | Parameters | Description        |
|--------|------------|--------------------|
| UINT32 | ctrl       | Gain control value |

Table 3-149 Parameters for AmbaUSB API `AmbaUSBH_Uvc_SetGainCtrl()`.

#### Return:

| Return Value | Description            |
|--------------|------------------------|
| 0            | UX_SUCCESS             |
| 0x5F         | UX_NO_DEVICE_CONNECTED |

Table 3-150 Returns for API `AmbaUSBH_Uvc_SetGainCtrl()`.

#### Example:

None

#### See Also :

`AmbaUSBH_Uvc_SetGainCtrlEx()`



### 3.4.37 AmbaUSBH\_Uvc\_GetCurLineFrequencyCtrl

#### API Syntax:

```
UINT32 AmbaUSBH_Uvc_GetCurLineFrequencyCtrl (void)
```

#### Function Description:

The application calls this API to get the current power line frequency control value.

#### Parameters:

None

#### Return:

| Return Value | Description             |
|--------------|-------------------------|
| X            | See Section 3.4.1.9     |
| 0xFFFFFFFF   | Operation not supported |

Table 3-151 Returns for API *AmbaUSBH\_Uvc\_GetCurLineFrequencyCtrl()*.

#### Example:

None

#### See Also :

*AmbaUSBH\_Uvc\_GetCurLineFrequencyCtrlEx()*



### 3.4.38 AmbaUSBH\_Uvc\_SetLineFrequencyCtrl

#### API Syntax:

```
UINT32 AmbaUSBH_Uvc_SetLineFrequencyCtrl (UVC_POWER_LINE_FREQUENCY_e  
ctrl)
```

#### Function Description:

The application calls this API to set the power line frequency control value.

#### Parameters:

| Type                       | Parameters | Description  |
|----------------------------|------------|--|
| UVC_POWER_LINE_FREQUENCY_e | ctrl       | Power line frequency control value. See Section 3.4.1.9 for details. |

Table 3-152 Parameters for AmbaUSB API *AmbaUSBH\_Uvc\_SetLineFrequencyCtrl()*.

#### Return:

| Return Value | Description            |
|--------------|------------------------|
| 0            | UX_SUCCESS             |
| 0x5F         | UX_NO_DEVICE_CONNECTED |

Table 3-153 Returns for API *AmbaUSBH\_Uvc\_SetLineFrequencyCtrl()*.

#### Example:

None

#### See Also :

[\*AmbaUSBH\\_Uvc\\_SetLineFrequencyCtrlEx\(\)\*](#)



### 3.4.39 **AmbaUSBH\_Uvc\_GetHueAutoCtrlInfo**

#### API Syntax:

```
UINT32 AmbaUSBH_Uvc_GetHueAutoCtrlInfo (UVCH_CTRL_INFO_s *info)
```

#### Function Description:

The application calls this API to get the Hue Auto control value.

#### Parameters:

| Type              | Parameters | Description   |
|-------------------|------------|---|
| UVCH_CTRL_INFO_s* | info       | [Output] Pointer to control information. See Section 3.4.1.6 for details. |

Table 3-154 Parameters for AmbaUSB API **AmbaUSBH\_Uvc\_GetHueAutoCtrlInfo()**.

#### Return:

| Return Value | Description             |
|--------------|-------------------------|
| 0            | UX_SUCCESS              |
| 0xFFFFFFFF   | Operation not supported |

Table 3-155 Returns for API **AmbaUSBH\_Uvc\_GetHueAutoCtrlInfo()**.

#### Example:

```
UVCH_CTRL_INFO_s info = {0};  
AmbaUSBH_Uvc_GetHueAutoCtrlInfo(&info);
```

#### See Also :

**AmbaUSBH\_Uvc\_GetHueAutoCtrlInfoEx()**



### 3.4.40 **AmbaUSBH\_Uvc\_SetHueAutoCtrl**

#### API Syntax:

```
UINT32 AmbaUSBH_Uvc_SetHueAutoCtrl (UINT32 ctrl)
```

#### Function Description:

The application calls this API to set the Hue Auto control value, which is in the range defined by **AmbaUSBH\_Uvc\_GetHueAutoCtrlInfo()**.

#### Parameters:

| Type   | Parameters | Description            |
|--------|------------|------------------------|
| UINT32 | ctrl       | Hue Auto control value |

Table 3-156 Parameters for AmbaUSB API **AmbaUSBH\_Uvc\_SetHueAutoCtrl()**.

#### Return:

| Return Value | Description            |
|--------------|------------------------|
| 0            | UX_SUCCESS             |
| 0x5F         | UX_NO_DEVICE_CONNECTED |

Table 3-157 Returns for API **AmbaUSBH\_Uvc\_SetHueAutoCtrl()**.

#### Example:

None

#### See Also :

***AmbaUSBH\_Uvc\_SetHueAutoCtrlEx()***



### 3.4.41 **AmbaUSBH\_Uvc\_GetWBTempAutoCtrlInfo**

#### API Syntax:

```
UINT32 AmbaUSBH_Uvc_GetWBTempAutoCtrlInfo (UVCH_CTRL_INFO_s *info)
```

#### Function Description:

The application calls this API to get the white balance temperature auto control value.

#### Parameters:

| Type              | Parameters | Description   |
|-------------------|------------|---|
| UVCH_CTRL_INFO_s* | info       | [Output] Pointer to control information. See Section 3.4.1.6 for details. |

Table 3-158 Parameters for AmbaUSB API **AmbaUSBH\_Uvc\_GetWBTempAutoCtrlInfo()**.

#### Return:

| Return Value | Description             |
|--------------|-------------------------|
| 0            | UX_SUCCESS              |
| 0xFFFFFFFF   | Operation not supported |

Table 3-159 Returns for API **AmbaUSBH\_Uvc\_GetWBTempAutoCtrlInfo()**.

#### Example:

```
UVCH_CTRL_INFO_s info = {0};  
AmbaUSBH_Uvc_GetWBTempAutoCtrlInfo(&info);
```

#### See Also :

[\*\*AmbaUSBH\\_Uvc\\_GetWBTempAutoCtrlInfoEx\(\)\*\*](#)



### 3.4.42 AmbaUSBH\_Uvc\_SetWBTempAutoCtrl

#### API Syntax:

```
UINT32 AmbaUSBH_Uvc_SetWBTempAutoCtrl (UINT32 ctrl)
```

#### Function Description:

The application calls this API to set the white balance temperature auto control value, which is in the range defined by *AmbaUSBH\_Uvc\_GetWBTempAutoCtrlInfo*.

#### Parameters:

| Type   | Parameters | Description                                  |
|--------|------------|--|
| UINT32 | ctrl       | White balance temperature auto control value |

Table 3-160 Parameters for AmbaUSB API *AmbaUSBH\_Uvc\_SetWBTempAutoCtrl()*.

#### Return:

| Return Value | Description            |
|--------------|------------------------|
| 0            | UX_SUCCESS             |
| 0x5F         | UX_NO_DEVICE_CONNECTED |

Table 3-161 Returns for API *AmbaUSBH\_Uvc\_SetWBTempAutoCtrl()*.

#### Example:

None

#### See Also :

*AmbaUSBH\_Uvc\_SetWBTempAutoCtrlEx()*



### 3.4.43 **AmbaUSBH\_Uvc\_GetWBCompAutoCtrlInfo**

#### API Syntax:

```
UINT32 AmbaUSBH_Uvc_GetWBCompAutoCtrlInfo (UVCH_CTRL_INFO_s *info)
```

#### Function Description:

The application calls this API to get the white balance component auto control value.

#### Parameters:

| Type              | Parameters | Description   |
|-------------------|------------|---|
| UVCH_CTRL_INFO_s* | info       | [Output] Pointer to control information. See Section 3.4.1.6 for details. |

Table 3-162 Parameters for AmbaUSB API **AmbaUSBH\_Uvc\_GetWBCompAutoCtrlInfo()**.

#### Return:

| Return Value | Description             |
|--------------|-------------------------|
| 0            | UX_SUCCESS              |
| 0xFFFFFFFF   | Operation not supported |

Table 3-163 Returns for API **AmbaUSBH\_Uvc\_GetWBCompAutoCtrlInfo()**.

#### Example:

```
UVCH_CTRL_INFO_s info = {0};  
AmbaUSBH_Uvc_GetWBCompAutoCtrlInfo(&info);
```

#### See Also :

[\*\*AmbaUSBH\\_Uvc\\_GetWBCompAutoCtrlInfoEx\(\)\*\*](#)



### 3.4.44 AmbaUSBH\_Uvc\_SetWBCompAutoCtrl

#### API Syntax:

```
UINT32 AmbaUSBH_Uvc_SetWBCompAutoCtrl (UINT32 ctrl)
```

#### Function Description:

The application calls this API to set the white balance component auto control value, which is in the range defined by *AmbaUSBH\_Uvc\_GetWBCompAutoCtrlInfo*.

#### Parameters:

| Type   | Parameters | Description                                |
|--------|------------|--|
| UINT32 | ctrl       | White balance component auto control value |

Table 3-164 Parameters for AmbaUSB API *AmbaUSBH\_Uvc\_SetWBCompAutoCtrl()*.

#### Return:

| Return Value | Description            |
|--------------|------------------------|
| 0            | UX_SUCCESS             |
| 0x5F         | UX_NO_DEVICE_CONNECTED |

Table 3-165 Returns for API *AmbaUSBH\_Uvc\_SetWBCompAutoCtrl()*.

#### Example:

None

#### See Also :

*AmbaUSBH\_Uvc\_SetWBCompAutoCtrlEx()*



### 3.4.45 AmbaUSBH\_Uvc\_GetDigitalMultiplierCtrlInfo

#### API Syntax:

```
UINT32 AmbaUSBH_Uvc_GetDigitalMultiplierCtrlInfo (UVCH_CTRL_INFO_s *info)
```

#### Function Description:

The application calls this API to get the digital multiplier control value.

#### Parameters:

| Type              | Parameters | Description   |
|-------------------|------------|---|
| UVCH_CTRL_INFO_s* | info       | [Output] Pointer to control information. See Section 3.4.1.6 for details. |

Table 3-166 Parameters for AmbaUSB API *AmbaUSBH\_Uvc\_GetDigitalMultiplierCtrlInfo()*.

#### Return:

| Return Value | Description             |
|--------------|-------------------------|
| 0            | UX_SUCCESS              |
| 0xFFFFFFFF   | Operation not supported |

Table 3-167 Returns for API *AmbaUSBH\_Uvc\_GetDigitalMultiplierCtrlInfo()*.

#### Example:

```
UVCH_CTRL_INFO_s info = {0};  
AmbaUSBH_Uvc_GetDigitalMultiplierCtrlInfo(&info);
```

#### See Also :

[\*AmbaUSBH\\_Uvc\\_GetDigitalMultiplierCtrlInfoEx\(\)\*](#)



### 3.4.46 AmbaUSBH\_Uvc\_SetDigitalMultiplierCtrl

#### API Syntax:

```
UINT32 AmbaUSBH_Uvc_SetDigitalMultiplierCtrl (UINT32 ctrl)
```

#### Function Description:

The application calls this API to set the digital multiplier control value, which is in the range defined by *AmbaUSBH\_Uvc\_GetDigitalMultiplierCtrlInfo*.

#### Parameters:

| Type   | Parameters | Description                      |
|--------|------------|----------------------------------|
| UINT32 | ctrl       | Digital multiplier control value |

Table 3-168 Parameters for AmbaUSB API *AmbaUSBH\_Uvc\_SetDigitalMultiplierCtrl()*.

#### Return:

| Return Value | Description            |
|--------------|------------------------|
| 0            | UX_SUCCESS             |
| 0x5F         | UX_NO_DEVICE_CONNECTED |

Table 3-169 Returns for API *AmbaUSBH\_Uvc\_SetDigitalMultiplierCtrl()*.

#### Example:

None

#### See Also :

*AmbaUSBH\_Uvc\_SetDigitalMultiplierCtrlEx()*



### 3.4.47 AmbaUSBH\_Uvc\_GetDigitalMultiplierLimitInfo

#### API Syntax:

```
UINT32 AmbaUSBH_Uvc_GetDigitalMultiplierLimitInfo(UVCH_CTRL_INFO_s *info)
```

#### Function Description:

The application calls this API to get the digital multiplier control limit.

#### Parameters:

| Type              | Parameters | Description   |
|-------------------|------------|---|
| UVCH_CTRL_INFO_s* | info       | [Output] Pointer to control information. See Section 3.4.1.6 for details. |

Table 3-170 Parameters for AmbaUSB API **AmbaUSBH\_Uvc\_GetDigitalMultiplierLimitInfo()**.

#### Return:

| Return Value | Description             |
|--------------|-------------------------|
| 0            | UX_SUCCESS              |
| 0xFFFFFFFF   | Operation not supported |

Table 3-171 Returns for API **AmbaUSBH\_Uvc\_GetDigitalMultiplierLimitInfo()**.

#### Example:

```
UVCH_CTRL_INFO_s info = {0};  
AmbaUSBH_Uvc_GetDigitalMultiplierLimitInfo(&info);
```

#### See Also :

[\*\*AmbaUSBH\\_Uvc\\_GetDigitalMultiplierLimitInfoEx\(\)\*\*](#)



### 3.4.48 **AmbaUSBH\_Uvc\_SetDigitalMultiplierLimit**

#### API Syntax:

```
UINT32 AmbaUSBH_Uvc_SetDigitalMultiplierLimit (UINT32 ctrl)
```

#### Function Description:

The application calls this API to set the digital multiplier control limit, which is in the range defined by *AmbaUSBH\_Uvc\_GetDigitalMultiplierLimitInfo*.

#### Parameters:

| Type   | Parameters | Description                      |
|--------|------------|----------------------------------|
| UINT32 | ctrl       | Digital multiplier control value |

Table 3-172 Parameters for AmbaUSB API *AmbaUSBH\_Uvc\_SetDigitalMultiplierLimit()*.

#### Return:

| Return Value | Description            |
|--------------|------------------------|
| 0            | UX_SUCCESS             |
| 0x5F         | UX_NO_DEVICE_CONNECTED |

Table 3-173 Returns for API *AmbaUSBH\_Uvc\_SetDigitalMultiplierLimit()*.

#### Example:

None

#### See Also :

*AmbaUSBH\_Uvc\_SetDigitalMultiplierLimitEx()*



### 3.4.49 AmbaUSBH\_Uvc\_GetCurAnalogVideoStandard

#### API Syntax:

```
UINT32 AmbaUSBH_Uvc_GetCurAnalogVideoStandard (void)
```

#### Function Description:

The application calls this API to get the current analog video standard value.

#### Parameters:

None

#### Return:

| Return Value | Description             |
|--------------|-------------------------|
| X            | See Section 3.4.1.10.   |
| 0xFFFFFFFF   | Operation not supported |

Table 3-174 Returns for API *AmbaUSBH\_Uvc\_GetCurAnalogVideoStandard()*.

#### Example:

None

#### See Also :

*AmbaUSBH\_Uvc\_GetCurAnalogVideoStandardEx()*



### 3.4.50 AmbaUSBH\_Uvc\_SetAnalogVideoStandard

#### API Syntax:

```
UINT32 AmbaUSBH_Uvc_SetAnalogVideoStandard  
(UVC_ANALOG_VIDEO_STANDARD_e ctrl)
```

#### Function Description:

The application calls this API to set the analog video standard value.

#### Parameters:

| Type                        | Parameters | Description  |
|-----------------------------|------------|--|
| UVC_ANALOG_VIDEO_STANDARD_e | ctrl       | Analog video standard control value. See Section 3.4.1.10 for details. |

Table 3-175 Parameters for AmbaUSB API *AmbaUSBH\_Uvc\_SetAnalogVideoStandard()*.

#### Return:

| Return Value | Description            |
|--------------|------------------------|
| 0            | UX_SUCCESS             |
| 0x5F         | UX_NO_DEVICE_CONNECTED |

Table 3-176 Returns for API *AmbaUSBH\_Uvc\_SetAnalogVideoStandard()*.

#### Example:

None

#### See Also :

*AmbaUSBH\_Uvc\_SetAnalogVideoStandardEx()*



### 3.4.51 AmbaUSBH\_Uvc\_GetCurAnalogLockStatus

#### API Syntax:

```
UINT32 AmbaUSBH_Uvc_GetCurAnalogLockStatus (void)
```

#### Function Description:

The application calls this API to get the current analog video lock status.

#### Parameters:

None

#### Return:

| Return Value | Description             |
|--------------|-------------------------|
| X            | See Section 3.4.1.11.   |
| 0xFFFFFFFF   | Operation not supported |

Table 3-177 Returns for API *AmbaUSBH\_Uvc\_GetCurAnalogLockStatus ()*.

#### Example:

None

#### See Also :

*AmbaUSBH\_Uvc\_GetCurAnalogLockStatusEx()*



### 3.4.52 AmbaUSBH\_Uvc\_SetAnalogLockStatus

#### API Syntax:

```
UINT32 AmbaUSBH_Uvc_SetAnalogLockStatus (UVC_ANALOG_VIDEO_LOCK_e  
status)
```

#### Function Description:

The application calls this API to set the analog video lock status value.

#### Parameters:

| Type                        | Parameters | Description  |
|-----------------------------|------------|--|
| UVC_ANALOG_VIDEO_L<br>OCK_e | status     | Analog video lock status value. See Section<br>3.4.1.11 for details. |

Table 3-178 Parameters for AmbaUSB API **AmbaUSBH\_Uvc\_SetAnalogLockStatus()**.

#### Return:

| Return Value | Description            |
|--------------|------------------------|
| 0            | UX_SUCCESS             |
| 0x5F         | UX_NO_DEVICE_CONNECTED |

Table 3-179 Returns for API **AmbaUSBH\_Uvc\_SetAnalogLockStatus()**.

#### Example:

None

#### See Also :

[\*\*AmbaUSBH\\_Uvc\\_SetAnalogLockStatusEx\(\)\*\*](#)



### 3.4.53 AmbaUSBH\_Uvc\_H264Enable

#### API Syntax:

```
UINT32 AmbaUSBH_Uvc_H264Enable (UINT32 enable)
```

#### Function Description:

The application calls this API to enable/disable mux h.264 data stream. When enabled, H.264 stream data would be muxed in the app4 segment of the received MJPEG. This API must be called before **AmbaUSBH\_Uvc\_ProbeAndCommit()**.

#### Parameters:

| Type   | Parameters | Description  |
|--------|------------|--|
| UINT32 | Enable     | 0: Disable mux h.264 data stream.<br>1: Enable mux h.264 data stream |

Table 3-180 Parameters for AmbaUSB API **AmbaUSBH\_Uvc\_H264Enable()**.

#### Return:

| Return Value | Description                    |
|--------------|--------------------------------|
| 0            | UX_SUCCESS                     |
| 0x5B         | UX_HOST_CLASS_INSTANCE_UNKNOWN |
| 0xFF         | Mux H.264 not supported        |

Table 3-181 Returns for API **AmbaUSBH\_Uvc\_H264Enable()**.

#### Example:

None

#### See Also :

**AmbaUSBH\_Uvc\_H264EnableEx()**



### 3.4.54 AmbaUSBH\_Uvc\_IsH264Supported

#### API Syntax:

```
UINT32 AmbaUSBH_Uvc_IsH264Supported(void)
```

#### Function Description:

The application calls **AmbaUSBH\_Uvc\_IsH264Supported()** or **AmbaUSBH\_Uvc\_H264Support()** to know if the connected device supports H.264 data stream.

#### Parameters:

None

#### Return:

| Return Value | Description   |
|--------------|---------------|
| 0            | Not supported |
| 1            | Supported     |

Table 3-182 Returns for API **AmbaUSBH\_Uvc\_H264Support()**.

#### Example:

None

#### See Also :

**AmbaUSBH\_Uvc\_IsH264SupportedEx()**



### 3.4.55 AmbaUSBH\_Uvc\_H264GetVideoConfigInfo

#### API Syntax:

```
void AmbaUSBH_Uvc_H264GetVideoConfigInfo(UVCX_VIDEO_CONFIG_S *ConfigMin,  
                                         UVCX_VIDEO_CONFIG_S *ConfigMax,  
                                         UVCX_VIDEO_CONFIG_S *ConfigCur)
```

#### Function Description:

The application calls this API to get the video configuration of the H.264 extension unit.

#### Parameters:

| Type                 | Parameters | Description  |
|----------------------|------------|--|
| UVCX_VIDEO_CONFIG_S* | ConfigMin  | Minimum video configuration of H264 format.<br>See Section 3.4.1.12 for details. |
| UVCX_VIDEO_CONFIG_S* | ConfigMax  | Maximum video configuration of H264 format.<br>See Section 3.4.1.12 for details. |
| UVCX_VIDEO_CONFIG_S* | ConfigCur  | Current video configuration of H264 format.<br>See Section 3.4.1.12 for details. |

Table 3-183 Parameters for AmbaUSB API **AmbaUSBH\_Uvc\_H264GetVideoConfigInfo()**.

#### Return:

None



**Example:**

```
UVCX_VIDEO_CONFIG_S VideoConfig = {0};  
UVCX_VIDEO_CONFIG_S VideoConfigMin = {0};  
UVCX_VIDEO_CONFIG_S VideoConfigMax = {0};  
  
AmbaUSBH_Uvc_H264GetVideoConfigInfo(&VideoConfigMin,  
                                         &VideoConfigMax,  
                                         &VideoConfig);  
  
AmbaPrint("Bit Rate %d ~ %d, cur %d", VideoConfigMin.dwBitRate,  
          VideoConfigMax.dwBitRate, VideoConfig.dwBitRate);
```

**See Also:**

*AmbaUSBH\_Uvc\_H264GetVideoConfigInfoEx()*



### 3.4.56 AmbaUSBH\_Uvc\_H264SetVideoConfig

#### API Syntax:

```
void AmbaUSBH_Uvc_H264SetVideoConfig(UVCX_VIDEO_CONFIG_S *config)
```

#### Function Description:

The application calls this API to set the video configuration of the H.264 extension unit. The configuration values should be in the range defined by

**AmbaUSBH\_Uvc\_H264GetVideoConfigInfo**. The configuration values do not take effect immediately until the application issues **AmbaUSBH\_Uvc\_ProbeAndCommit()**. Note that “wWidth”, “wHeight”, “dwFrameInterval” and “bStreamMuxOption” are set by the USB stack and the application does not need to configure them.

#### Parameters:

| Type                 | Parameters | Description                       |
|----------------------|------------|-----------------------------------|
| UVCX_VIDEO_CONFIG_S* | config     | See Section 3.4.1.12 for details. |

Table 3-184 Parameters for AmbaUSB API **AmbaUSBH\_Uvc\_H264SetVideoConfig()**.

#### Return:

None

#### Example:

None

#### See Also :

**AmbaUSBH\_Uvc\_H264SetVideoConfigEx()**



### 3.4.57 AmbaUSBH\_Uvc\_H264RequestIdr

#### API Syntax:

```
UINT32 AmbaUSBH_Uvc_H264RequestIdr (void)
```

#### Function Description:

The application calls this API for requesting the next frame as an IDR frame with new SPS and PPS.

#### Parameters:

None

#### Return:

| Return Value | Description             |
|--------------|-------------------------|
| 0            | SUCCESS                 |
| 0x12         | UX_MEMORY_INSUFFICIENT  |
| 0xFF         | Operation not supported |
| 0x5F         | UX_NO_DEVICE_CONNECTED  |

Table 3-185 Returns for API *AmbaUSBH\_Uvc\_H264RequestIdr()*.

#### Example:

None

#### See Also :

*AmbaUSBH\_Uvc\_H264RequestIdrEx()*



### 3.4.58 AmbaUSBH\_Uvc\_H264GetRateCtrlModelInfo

#### API Syntax:

```
UINT32 AmbaUSBH_Uvc_H264GetRateCtrlModelInfo (UVCH_CTRL_INFO_s *info)
```

#### Function Description:

The application calls this API to get the current rate control mode.

#### Parameters:

| Type              | Parameters | Description   |
|-------------------|------------|---|
| UVCH_CTRL_INFO_s* | info       | [Output] Pointer to control information. See Section 3.4.1.6 for details.<br>Bits 0-3 Modes<br>0x00: Reserved<br>0x01: CBR<br>0x02: VBR<br>0x03: Constant QP<br>Bits 4-7 Flags<br>0x10: Fixed_frame_rate_flag |

Table 3-186 Parameters for AmbaUSB API **AmbaUSBH\_Uvc\_H264GetRateCtrlModelInfo()**.

#### Return:

| Return Value | Description            |
|--------------|------------------------|
| 0            | UX_SUCCESS             |
| 0x5F         | UX_NO_DEVICE_CONNECTED |

Table 3-187 Returns for API **AmbaUSBH\_Uvc\_H264GetRateCtrlModelInfo()**.

#### Example:

```
UVCH_CTRL_INFO_s info = {0};  
AmbaUSBH_Uvc_H264GetRateCtrlModelInfo(&info);
```



See Also :

*AmbaUSBH\_Uvc\_H264GetRateCtrlModeInfoEx()*

For PROTRULY Confidential Only



### 3.4.59 AmbaUSBH\_Uvc\_H264SetRateCtrlMode

#### API Syntax:

```
UINT32 AmbaUSBH_Uvc_H264SetRateCtrlMode (UINT32 mode)
```

#### Function Description:

The application calls this API to set the rate control mode, which is in the range defined by `AmbaUSBH_Uvc_H264GetRateCtrlModeInfo()`.

#### Parameters:

| Type   | Parameters | Description  |
|--------|------------|--|
| UINT32 | mode       | Bits 0-3 Modes:<br>0x00: Reserved<br>0x01: CBR<br>0x02: VBR<br>0x03: Constant QP<br>Bits 4-7 Flags:<br>0x10: fixed_frame_rate_flag |

Table 3-188 Parameters for AmbaUSB API `AmbaUSBH_Uvc_H264SetRateCtrlMode()`.

#### Return:

| Return Value | Description             |
|--------------|-------------------------|
| 0            | UX_SUCCESS              |
| 0x5F         | UX_NO_DEVICE_CONNECTED  |
| 0xFF         | Operation not supported |

Table 3-189 Returns for API `AmbaUSBH_Uvc_H264SetRateCtrlMode()`.

#### Example:

None

#### See Also :

`AmbaUSBH_Uvc_H264SetRateCtrlModeEx()`



### 3.4.60 AmbaUSBH\_Uvc\_H264GetTemporalScaleModelInfo

#### API Syntax:

```
UINT32 AmbaUSBH_Uvc_H264GetTemporalScaleModelInfo (UVCH_CTRL_INFO_s  
*info)
```

#### Function Description:

The application calls this API to get the temporal scale mode.

#### Parameters:

| Type              | Parameters | Description  |
|-------------------|------------|--|
| UVCH_CTRL_INFO_s* | info       | [Output] Pointer to control information. See Section 3.4.1.6 for details.<br>0x00: No Temporal Enhancement Layer<br>0x01-0x07: Number of Temporal Enhancement Layers |

Table 3-190 Parameters for AmbaUSB API

**AmbaUSBH\_Uvc\_H264GetTemporalScaleModelInfo()**.

#### Return:

| Return Value | Description            |
|--------------|------------------------|
| 0            | UX_SUCCESS             |
| 0x5F         | UX_NO_DEVICE_CONNECTED |

Table 3-191 Returns for API **AmbaUSBH\_Uvc\_H264GetTemporalScaleModelInfo()**.

#### Example:

```
UVCH_CTRL_INFO_s info = { 0 };  
AmbaUSBH_Uvc_H264GetTemporalScaleModelInfo(&info);
```

#### See Also :

**AmbaUSBH\_Uvc\_H264GetTemporalScaleModelInfoEx()**



### 3.4.61 AmbaUSBH\_Uvc\_H264SetTemporalScaleMode

#### API Syntax:

```
UINT32 AmbaUSBH_Uvc_H264SetTemporalScaleMode (UINT32 mode)
```

#### Function Description:

The application calls this API to set the temporal scale control mode, which is in the range defined by [AmbaUSBH\\_Uvc\\_H264GetTemporalScaleModeInfo\(\)](#).

#### Parameters:

| Type   | Parameters | Description  |
|--------|------------|--|
| UINT32 | mode       | 0x00: No Temporal Enhancement Layer<br>0x01- 0x07: Number of Temporal Enhancement Layers |

Table 3-192 Parameters for AmbaUSB API [AmbaUSBH\\_Uvc\\_H264SetTemporalScaleMode \(\)](#).

#### Return:

| Return Value | Description             |
|--------------|-------------------------|
| 0            | UX_SUCCESS              |
| 0x5F         | UX_NO_DEVICE_CONNECTED  |
| 0xFF         | Operation not supported |

Table 3-193 Returns for API [AmbaUSBH\\_Uvc\\_H264SetTemporalScaleMode \(\)](#).

#### Example:

None

#### See Also :

[AmbaUSBH\\_Uvc\\_H264SetTemporalScaleModeEx\(\)](#)



### 3.4.62 AmbaUSBH\_Uvc\_H264GetSpatialScaleModelInfo

#### API Syntax:

```
UINT32 AmbaUSBH_Uvc_H264GetSpatialScaleModelInfo (UVCH_CTRL_INFO_s *info)
```

#### Function Description:

The application calls this API to get the spatial scale mode.

#### Parameters:

| Type              | Parameters | Description  |
|-------------------|------------|--|
| UVCH_CTRL_INFO_s* | info       | [Output] Pointer to control information. See Section 3.4.1.6 for details.<br>0x00: No Spatial Enhancement Layer<br>0x01-0x08: Number of Spatial Enhancement Layers |

Table 3-194 Parameters for AmbaUSB API **AmbaUSBH\_Uvc\_H264GetSpatialScaleModelInfo()**.

#### Return:

| Return Value | Description            |
|--------------|------------------------|
| 0            | UX_SUCCESS             |
| 0x5F         | UX_NO_DEVICE_CONNECTED |

Table 3-195 Returns for API **AmbaUSBH\_Uvc\_H264GetSpatialScaleModelInfo()**.

#### Example:

```
UVCH_CTRL_INFO_s info = { 0 };  
AmbaUSBH_Uvc_H264GetSpatialScaleModelInfo(&info);
```

#### See Also :

**AmbaUSBH\_Uvc\_H264GetSpatialScaleModelInfoEx()**



### 3.4.63 AmbaUSBH\_Uvc\_H264SetSpatialScaleMode

#### API Syntax:

```
UINT32 AmbaUSBH_Uvc_H264SetSpatialScaleMode (UINT32 mode)
```

#### Function Description:

The application calls this API to set the spatial scale control mode, which is in the range defined by **AmbaUSBH\_Uvc\_H264GetSpatialScaleModeInfo()**.

#### Parameters:

| Type   | Parameters | Description   |
|--------|------------|---|
| UINT32 | mode       | 0x00: No Spatial Enhancement Layer<br>0x01-0x08: Number of Spatial Enhancement Layers |

Table 3-196 Parameters for AmbaUSB API **AmbaUSBH\_Uvc\_H264SetSpatialScaleMode ()**.

#### Return:

| Return Value | Description             |
|--------------|-------------------------|
| 0            | UX_SUCCESS              |
| 0x5F         | UX_NO_DEVICE_CONNECTED  |
| 0xFF         | Operation not supported |

Table 3-197 Returns for API **AmbaUSBH\_Uvc\_H264SetSpatialScaleMode ()**.

#### Example:

None

#### See Also :

[\*\*AmbaUSBH\\_Uvc\\_H264SetSpatialScaleModeEx\(\)\*\*](#)



### 3.4.64 AmbaUSBH\_Uvc\_H264GetFrameRateConfigInfo

#### API Syntax:

```
UINT32 AmbaUSBH_Uvc_H264GetFrameRateConfigInfo (UVCH_CTRL_INFO_s *info)
```

#### Function Description:

The application calls this API to get the frame rate configuration.

#### Parameters:

| Type              | Parameters | Description   |
|-------------------|------------|---|
| UVCH_CTRL_INFO_s* | info       | [Output] Pointer to control information. See Section 3.4.1.6 for details.<br>In 100 ns frame intervals. |

Table 3-198 Parameters for AmbaUSB API **AmbaUSBH\_Uvc\_H264GetFrameRateConfigInfo()**.

#### Return:

| Return Value | Description            |
|--------------|------------------------|
| 0            | UX_SUCCESS             |
| 0x5F         | UX_NO_DEVICE_CONNECTED |

Table 3-199 Returns for API **AmbaUSBH\_Uvc\_H264GetFrameRateConfigInfo()**.

#### Example:

```
UVCH_CTRL_INFO_s info = { 0 };  
AmbaUSBH_Uvc_H264GetFrameRateConfigInfo(&info);
```

#### See Also:

**AmbaUSBH\_Uvc\_H264GetFrameRateConfigInfoEx()**



### 3.4.65 AmbaUSBH\_Uvc\_H264SetFrameRateConfig

#### API Syntax:

```
UINT32 AmbaUSBH_Uvc_H264SetFrameRateConfig (UINT32 FrameRate)
```

#### Function Description:

The application calls this API to set the frame rate configuration, which is in the range defined by `AmbaUSBH_Uvc_H264GetFrameRateConfigInfo()`.

#### Parameters:

| Type   | Parameters | Description               |
|--------|------------|---------------------------|
| UINT32 | FrameRate  | In 100 ns frame intervals |

Table 3-200 Parameters for AmbaUSB API `AmbaUSBH_Uvc_H264SetFrameRateConfig ()`.

#### Return:

| Return Value | Description             |
|--------------|-------------------------|
| 0            | UX_SUCCESS              |
| 0x5F         | UX_NO_DEVICE_CONNECTED  |
| 0xFF         | Operation not supported |

Table 3-201 Returns for API `AmbaUSBH_Uvc_H264SetFrameRateConfig ()`.

#### Example:

None

#### See Also :

`AmbaUSBH_Uvc_H264SetFrameRateConfigEx()`



### 3.4.66 AmbaUSBH\_Uvc\_GetDeviceInfoEx

#### API Syntax:

```
UINT32 AmbaUSBH_Uvc_GetDeviceInfoEx(UINT32 id, UVCH_DEVICE_INFO *info)
```

#### Function Description:

The extension function of **AmbaUSBH\_Uvc\_GetDeviceInfo()**. The application calls this API to get the specified UVC device information by the *id* parameter.

#### Parameters:

| Type               | Parameters | Description   |
|--------------------|------------|---|
| UINT32             | id         | [Input] The id of UVC device, which is assigned by the device insert/remove callback functions. See Section 3.4.1.14 for details. |
| UVCH_DEVICE_INFO * | info       | [Output] UVC device information. Please refer to Section 3.4.1.1 for more details.  |

Table 3-202 Parameters for AmbaUSB API **AmbaUSBH\_Uvc\_GetDeviceInfoEx ()**.

#### Return:

| Return Value | Description            |
|--------------|------------------------|
| 0            | UX_SUCCESS             |
| 0x19         | UX_MEMORY_CORRUPTED    |
| 0x1C         | UX_PARAMETER_ERROR     |
| 0x5F         | UX_NO_DEVICE_CONNECTED |

Table 3-203 Returns for API **AmbaUSBH\_Uvc\_GetDeviceInfoEx ()**.

#### Example:

```
UVCH_CTRL_INFO_s info = {0};  
AmbaUSBH_Uvc_GetDeviceInfoEx(1, &info);
```

#### See Also :



*AmbaUSBH\_Uvc\_GetDeviceInfo( )*

For PROTRULY Confidential Only



### 3.4.67 AmbaUSBH\_Uvc\_GetActiveDeviceEx

#### API Syntax:

```
UINT32 AmbaUSBH_Uvc_GetActiveDeviceEx(UINT8* IdArray, UINT32* length)
```

#### Function Description:

The application calls this API to obtain all active device ID.

#### Parameters:

| Type    | Parameters | Description                                  |
|---------|------------|--|
| UINT8*  | IdArray    | [Output] IdArray store the active device ID. |
| UINT32* | length     | [Input/ Out] The active device number.       |

Table 3-204 Parameters for AmbaUSB API **AmbaUSBH\_Uvc\_GetActiveDeviceEx ()**.

#### Return:

| Return Value | Description |
|--------------|-------------|
| 0            | UX_SUCCESS  |

Table 3-205 Returns for API **AmbaUSBH\_Uvc\_GetActiveDeviceEx ()**.

#### Example:

```
UINT32 DeviceNum = 3;  
UINT8 IdArray[3] = {0};  
UINT32 cnt = 0, id = 0;  
  
AmbaUSBH_Uvc_GetActiveDeviceEx(IdArray, &DeviceNum);  
for (cnt=0; cnt<DeviceNum; cnt++) {  
    AmbaPrint("active device [%d]", IdArray[cnt]);  
}
```

#### See Also:

None



### 3.4.68 AmbaUSBH\_Uvc\_GetItControlInfoEx

#### API Syntax:

```
UINT32 AmbaUSBH_Uvc_GetItControlInfoEx(UINT32 id,  
                                         UVCH_CONTROL_ITEM *item,  
                                         UVC_CAMERA_TERMINAL_CONTROL_SELECTOR selector)
```

#### Function Description:

The extension function of **AmbaUSBH\_Uvc\_GetItControlInfo()**. The application calls this API to get the specified Input Terminal control information by the *id* parameter.

#### Parameters:

| Type                                    | Parameters | Description   |
|---|------------|---|
| UINT32                                  | id         | [Input] The ID of the UVC device, which is assigned by the device insert/remove callback functions. See Section 3.4.1.14 for details. |
| UVCH_CONTROL_ITEM *                     | item       | [Output] UVC control information. Please refer to Section 3.4.1.5 for more details.   |
| UVC_CAMERA_TERMINAL_CONTROL_SELECTOR OR | selector   | [Input] Index for the control. Please refer to Section 3.4.1.7 for more details.  |

Table 3-206 Parameters for AmbaUSB API **AmbaUSBH\_Uvc\_GetItControlInfoEx ()**.

#### Return:

| Return Value | Description                        |
|--------------|------------------------------------|
| 0            | UX_SUCCESS                         |
| 0x12         | UX_MEMORY_INSUFFICIENT             |
| 0x1C         | UX_PARAMETER_ERROR                 |
| 0x5F         | UX_NO_DEVICE_CONNECTED             |
| 0x84         | UX_HOST_CLASS_VIDEO_WRONG_SELECTOR |

Table 3-207 Returns for API **AmbaUSBH\_Uvc\_GetItControlInfoEx ()**.



**Example:**

```
UVCH_CONTROL_ITEM item = {0};  
AmbaUSBH_Uvc_GetItControlInfoEx(1, &item, UVC_CT_AE_MODE_CONTROL);
```

**See Also:**

***AmbaUSBH\_Uvc\_GetItControlInfo*** ( )

For PROTRULY Confidential



### 3.4.69 AmbaUSBH\_Uvc\_GetPuControlInfoEx

#### API Syntax:

```
UINT32 AmbaUSBH_Uvc_GetPuControlInfoEx(UINT32 id,  
                                         UVCH_CONTROL_ITEM *item,  
                                         UVC_PROCESSING_UNIT_CONTROL_SELECTOR selector)
```

#### Function Description:

The extension function of **AmbaUSBH\_Uvc\_GetPuControlInfo()**. The application calls this API to get the specified Processing Unit control information by the **id** parameter.

#### Parameters:

| Type                                    | Parameters | Description   |
|---|------------|---|
| UINT32                                  | id         | [Input] The ID of UVC device, which is assigned by the device insert/remove callback functions. See Section 3.4.1.14 for details. |
| UVCH_CONTROL_ITEM *                     | item       | [Output] UVC control information. Please refer to Section 3.4.1.5 for more details.   |
| UVC_PROCESSING_UNIT_CONTROL_SELECTOR OR | selector   | [Input] Index for the control. Please refer to Section 3.4.1.8 for more details.  |

Table 3-208 Parameters for AmbaUSB API **AmbaUSBH\_Uvc\_GetPuControlInfoEx ()**.

#### Return:

| Return Value | Description                        |
|--------------|------------------------------------|
| 0            | UX_SUCCESS                         |
| 0x12         | UX_MEMORY_INSUFFICIENT             |
| 0x1C         | UX_PARAMETER_ERROR                 |
| 0x5F         | UX_NO_DEVICE_CONNECTED             |
| 0x84         | UX_HOST_CLASS_VIDEO_WRONG_SELECTOR |

Table 3-209 Returns for API **AmbaUSBH\_Uvc\_GetPuControlInfoEx ()**.



**Example:**

```
UVCH_CONTROL_ITEM item = {0};  
AmbaUSBH_Uvc_GetPuControllInfoEx(1, &item, UVC_PU_BRIGHTNESS_CONTROL);
```

**See Also:**

[AmbarUSBH\\_Uvc\\_GetPuControllInfo\(\)](#)



### 3.4.70 AmbaUSBH\_Uvc\_PrintDeviceValuesEx

#### API Syntax:

```
UINT32 AmbaUSBH_Uvc_PrintDeviceValuesEx(UINT32 id)
```

#### Function Description:

The extension function of **AmbaUSBH\_Uvc\_PrintDeviceValues()**. The application calls this API to print the specified values by the **id** parameter.

#### Parameters:

| Type   | Parameters | Description   |
|--------|------------|---|
| UINT32 | id         | [Input] The ID of the UVC device, which is assigned by the device insert/remove callback functions. See Section 3.4.1.14 for details. |

Table 3-210 Parameters for AmbaUSB API **AmbaUSBH\_Uvc\_PrintDeviceValuesEx ()**.

#### Return:

| Return Value | Description            |
|--------------|------------------------|
| 0            | UX_SUCCESS             |
| 0x5F         | UX_NO_DEVICE_CONNECTED |

Table 3-211 Returns for API **AmbaUSBH\_Uvc\_PrintDeviceValuesEx ()**.

#### Example:

None

#### See Also :

**AmbaUSBH\_Uvc\_PrintDeviceValues ()**



### 3.4.71 AmbaUSBH\_Uvc\_ProbeAndCommitEx

#### API Syntax:

```
UINT32 AmbaUSBH_Uvc_ProbeAndCommitEx(UINT32 id,  
                                UINT32 formatIndex,  
                                UINT32 frameIndex,  
                                UINT32 fps)
```

#### Function Description:

The extension function of **AmbaUSBH\_Uvc\_ProbeAndCommit()**. The application calls this API to probe and commit the specified stream format by the **id** parameter.

#### Parameters:

| Type   | Parameters  | Description   |
|--------|-------------|---|
| UINT32 | id          | [Input] The ID of the UVC device, which is assigned by device insert/remove callback functions. See Section 3.4.1.14 for details. |
| UINT32 | formatIndex | [Input] Format index  |
| UINT32 | frameIndex  | [Input] Frame index   |
| UINT32 | fps         | [Input] Preferred frame per second  |

Table 3-212 Parameters for AmbaUSB API **AmbaUSBH\_Uvc\_ProbeAndCommitEx ()**.

#### Return:

| Return Value | Description                    |
|--------------|--------------------------------|
| 0            | UX_SUCCESS                     |
| 0x12         | UX_MEMORY_INSUFFICIENT         |
| 0x15         | UX_SEMAPHORE_ERROR             |
| 0x23         | UX_TRANSFER_ERROR              |
| 0x25         | UX_TRANSFER_NOT_READY          |
| 0x5B         | UX_HOST_CLASS_INSTANCE_UNKNOWN |
| 0x5F         | UX_NO_DEVICE_CONNECTED         |



| Return Value | Description                           |
|--------------|---------------------------------------|
| 0x86         | UX_HOST_CLASS_VIDEO_UN SUPPORT_FORMAT |

Table 3-213 Returns for API *AmbaUSBH\_Uvc\_ProbeAndCommitEx* ().

**Example:**

None

**See Also :**

*AmbaUSBH\_Uvc\_ProbeAndCommit* ()



### 3.4.72 AmbaUSBH\_Uvc\_GetCurrentProbeSettingEx

#### API Syntax:

```
UINT32 AmbaUSBH_Uvc_GetCurrentProbeSettingEx(UINT32 id,  
                                UINT32 *formatIndex,  
                                UINT32 *frameIndex,  
                                UINT32 *fps)
```

#### Function Description:

The extension function of **AmbaUSBH\_Uvc\_GetCurrentProbeSetting()**. The application calls this API to get the specified probe values by the **id** parameter.

#### Parameters:

| Type    | Parameters  | Description   |
|---------|-------------|---|
| UINT32  | id          | [Input] The ID of the UVC device, which is assigned by device insert/remove callback functions. See Section 3.4.1.14 for details. |
| UINT32* | formatIndex | [Output] Format index   |
| UINT32* | frameIndex  | [Output] Frame index  |
| UINT32* | fps         | [Output] Preferred frame per second   |

Table 3-214 Parameters for AmbaUSB API **AmbaUSBH\_Uvc\_GetCurrentProbeSetting ()**.

#### Return:

| Return Value | Description            |
|--------------|------------------------|
| 0            | UX_SUCCESS             |
| 0x5F         | UX_NO_DEVICE_CONNECTED |

Table 3-215 Returns for API **AmbaUSBH\_Uvc\_GetCurrentProbeSetting ()**.

#### Example:

None

#### See Also :



*AmbaUSBH\_Uvc\_GetCurrentProbeSetting()*

For PROTRULY Confidential Only



### 3.4.73 AmbaUSBH\_Uvc\_SetCurValuesEx

#### API Syntax:

```
UINT32 AmbaUSBH_Uvc_SetCurValuesEx(UINT32 id,  
                                UINT32 unit,  
                                UINT32 selector,  
                                UINT32 value)
```

#### Function Description:

The extension function of **AmbaUSBH\_Uvc\_SetCurValues()**. The application calls this API to set the specified control values for a unit/terminal by the *id* parameter.

#### Parameters:

| Type   | Parameters | Description   |
|--------|------------|---|
| UINT32 | id         | [Input] The ID of the UVC device, which is assigned by the device insert/remove callback functions. See Section 3.4.1.14 for details. |
| UINT32 | unit       | [Input] Unit/terminal ID  |
| UINT32 | selector   | [Input] Selector ID for the unit/terminal   |
| UINT32 | value      | [Input] Value for the selector  |

Table 3-216 Parameters for AmbaUSB API **AmbaUSBH\_Uvc\_SetCurValuesEx ()**.

#### Return:

| Return Value | Description                    |
|--------------|--------------------------------|
| 0            | UX_SUCCESS                     |
| 0x12         | UX_MEMORY_INSUFFICIENT         |
| 0x15         | UX_SEMAPHORE_ERROR             |
| 0x23         | UX_TRANSFER_ERROR              |
| 0x25         | UX_TRANSFER_NOT_READY          |
| 0x5B         | UX_HOST_CLASS_INSTANCE_UNKNOWN |
| 0x5F         | UX_NO_DEVICE_CONNECTED         |



| Return Value | Description                        |
|--------------|------------------------------------|
| 0x84         | UX_HOST_CLASS_VIDEO_WRONG_SELECTOR |

Table 3-217 Returns for API *AmbaUSBH\_Uvc\_SetCurValuesEx()*.

**Example:**

None

**See Also :**

*AmbaUSBH\_Uvc\_SetCurValues()*



### 3.4.74 AmbaUSBH\_Uvc\_StreamingStartEx

#### API Syntax:

```
UINT32 AmbaUSBH_Uvc_StreamingStartEx(UINT32 id, UINT32 AltSetting)
```

#### Function Description:

The extension function of **AmbaUSBH\_Uvc\_StreamingStart()**. The application calls this API to start the specified video streaming by the **id** parameter. The **AltSetting** parameter must be specified value obtained via **AmbaUSBH\_Uvc\_GetAlternateSettingEx()**.

#### Parameters:

| Type   | Parameters | Description   |
|--------|------------|---|
| UINT32 | id         | [Input] The ID of the UVC device, which is assigned by device insert/remove callback functions. See Section 3.4.1.14 for details. |
| UINT32 | AltSetting | [Input] The alternate setting number.   |

Table 3-218 Parameters for AmbaUSB API **AmbaUSBH\_Uvc\_StreamingStartEx ()**.

#### Return:

| Return Value | Description                 |
|--------------|-----------------------------|
| 0            | UX_SUCCESS                  |
| 0x52         | UX_INTERFACE_HANDLE_UNKNOWN |
| 0x5F         | UX_NO_DEVICE_CONNECTED      |

Table 3-219 Returns for API **AmbaUSBH\_Uvc\_StreamingStartEx ()**.

#### Example:

None

#### See Also :

**AmbaUSBH\_Uvc\_StreamingStart ()**



### 3.4.75 AmbaUSBH\_Uvc\_StreamingStopEx

#### API Syntax:

```
UINT32 AmbaUSBH_Uvc_StreamingStopEx(UINT32 id)
```

#### Function Description:

The extension function of **AmbaUSBH\_Uvc\_StreamingStop()**. The application calls this API to stop the specified video streaming by the *id* parameter.

#### Parameters:

| Type   | Parameters | Description   |
|--------|------------|---|
| UINT32 | <i>id</i>  | [Input] The ID of the UVC device, which is assigned by device insert/remove callback functions. See Section 3.4.1.14 for details. |

Table 3-220 Parameters for AmbaUSB API **AmbaUSBH\_Uvc\_StreamingStopEx ()**.

#### Return:

| Return Value | Description                 |
|--------------|-----------------------------|
| 0            | UX_SUCCESS                  |
| 0x52         | UX_INTERFACE_HANDLE_UNKNOWN |
| 0x5F         | UX_NO_DEVICE_CONNECTED      |

Table 3-221 Returns for API **AmbaUSBH\_Uvc\_StreamingStopEx ()**.

#### Example:

None

#### See Also :

[\*\*AmbaUSBH\\_Uvc\\_StreamingStop\(\)\*\*](#)



### 3.4.76 AmbaUSBH\_Uvc\_GetAlternateSettingEx

#### API Syntax:

```
UINT32 AmbaUSBH_Uvc_GetAlternateSettingEx(UINT32 id,  
                                         UVCH_ALTERNATE_SETTING_s *setting,  
                                         UINT32 *length)
```

#### Function Description:

The application calls this API to get the specified supported alternate setting by the ***id*** parameter.

#### Parameters:

| Type                      | Parameters | Description   |
|---------------------------|------------|---|
| UINT32                    | id         | [Input] The ID of the UVC device, which is assigned by device insert/remove callback functions. See Section 3.4.1.14 for details. |
| UVCH_ALTERNATE_SETTING_s* | setting    | [Output] The alternate setting data structure. See Section 3.4.1.15 for details.  |
| UINT32*                   | length     | [Input/Output] The size of the supported alternate setting.   |

Table 3-222 Parameters for AmbaUSB API **AmbaUSBH\_Uvc\_GetAlternateSettingEx ()**.

#### Return:

| Return Value | Description             |
|--------------|-------------------------|
| 0            | UX_SUCCESS              |
| 0x42         | UX_DESCRIPTOR_CORRUPTED |
| 0x5E         | UX_NO_ALTERNATE_SETTING |
| 0x5F         | UX_NO_DEVICE_CONNECTED  |

Table 3-223 Returns for API **AmbaUSBH\_Uvc\_GetAlternateSettingEx ()**.

#### Example:

```
#define ALTERNATE_SETTING_NUM 12
```



```
UVCH_ALTERNATE_SETTING_s    AltArray[ALTERNATE_SETTING_NUM];  
UINT32                      length = ALTERNATE_SETTING_NUM;  
UINT32                      status = 0;  
UINT32                      i = 0;  
status = AmbaUSBH_Uvc_GetAlternateSettingEx(id, AltArray, &length);  
if (status == OK) {  
    for (i=0; i<length; i++) {  
        AmbaPrint("cur_altsetting %d, psize = %d, BW = %d us",  
AltArray[i].AlternateSetting, AltArray[i].MaxPacketSize,  
AltArray[i].BandWidth);  
    }  
}  
}
```

**See Also:**

None



### 3.4.77 AmbaUSBH\_Uvc\_ReadBlockEx

#### API Syntax:

```
UINT32 AmbaUSBH_Uvc_ReadBlockEx(UINT32 id,  
                           UX_EHCI_ISO_REQUEST *request,  
                           UINT32 PacketNum,  
                           void (*complete_func)(UINT32 id, UX_EHCI_ISO_REQUEST *request))
```

#### Function Description:

The extension function of **AmbaUSBH\_Uvc\_ReadBlock()**. The application calls this API to receive the specified video stream by the *id* parameter.

#### Parameters:

| Type                 | Parameters    | Description   |
|----------------------|---------------|---|
| UINT32               | id            | [Input] The ID of the UVC device, which is assigned by device insert/remove callback functions. See Section 3.4.1.14 for details. |
| UX_EHCI_ISO_REQUEST* | request       | [Input] Pointer of request array. See Section 3.4.1.16 for details.   |
| UINT32               | PacketNum     | [Input] Packet number of request array  |
| Callback function    | complete_func | [Input] See Section 3.4.2.1   |

Table 3-224 Parameters for AmbaUSB API **AmbaUSBH\_Uvc\_ReadBlockEx ()**.

#### Return:

| Return Value | Description               |
|--------------|---------------------------|
| 0            | UX_SUCCESS                |
| 0x12         | UX_MEMORY_INSUFFICIENT    |
| 0x13         | UX_NO_TD_AVAILABLE        |
| 0x15         | UX_SEMAPHORE_ERROR        |
| 0x1C         | UX_PARAMETER_ERROR        |
| 0x25         | UX_TRANSFER_NOT_READY     |
| 0x41         | UX_NO_BANDWIDTH_AVAILABLE |



| Return Value | Description                    |
|--------------|--------------------------------|
| 0x53         | UX_ENDPOINT_HANDLE_UNKNOWN     |
| 0x54         | UX_FUNCTION_NOT_SUPPORTED      |
| 0x5B         | UX_HOST_CLASS_INSTANCE_UNKNOWN |
| 0x5E         | UX_NO_ALTERNATE_SETTING        |
| 0xFF         | UX_ERROR                       |

Table 3-225 Returns for API *AmbaUSBH\_Uvc\_ReadBlockEx ()*.

**Example:**

None

**See Also :**

*AmbaUSBH\_Uvc\_ReadBlock ()*



### 3.4.78 AmbaUSBH\_Uvc\_ReadAppendEx

#### API Syntax:

```
UINT AmbaUSBH_Uvc_ReadAppendEx (UINT32 id,  
                                  UX_EHCI_ISO_REQUEST *request,  
                                  UINT32 PacketNum,  
                                  void (*complete_func)(UX_EHCI_ISO_REQUEST *request))
```

#### Function Description:

The extension function of *AmbaUSBH\_Uvc\_ReadAppend()*. The application calls this API to receive the specified video streaming successively by the *id* parameter and USB stack would make sure the received data between every transfer request are continuous. USB stack also makes the necessary delay which may block the task.

#### Parameters:

See *AmbaUSBH\_Uvc\_ReadBlockEx()*.

#### Return:

See *AmbaUSBH\_Uvc\_ReadBlockEx()*.

#### Example:

See *AmbaUSBH\_Uvc\_ReadBlockEx()*.

#### See Also :

*AmbaUSBH\_Uvc\_ReadAppend( )*  
*AmbaUSBH\_Uvc\_ReadBlockEx( )*



### 3.4.79 AmbaUSBH\_Uvc\_GetMaxPacketSizeEx

#### API Syntax:

```
UINT32 AmbaUSBH_Uvc_GetMaxPacketSizeEx(UINT32 id , UINT32 *size)
```

#### Function Description:

The extension function of **AmbaUSBH\_Uvc\_GetMaxPacketSize()**. The application calls this API to get the specified maximum packet size by the *id* parameter.

#### Parameters:

| Type    | Parameters | Description   |
|---------|------------|---|
| UINT32  | id         | [Input] The ID of the UVC device, which is assigned by the device insert/remove callback functions. See Section 3.4.1.14 for details. |
| UINT32* | size       | [Output] The maximum packet size of current alternate setting.  |

Table 3-226 Parameters for AmbaUSB API **AmbaUSBH\_Uvc\_GetMaxPacketSizeEx ()**.

#### Return:

| Return Value | Description                |
|--------------|----------------------------|
| 0            | UX_SUCCESS                 |
| 0x5F         | UX_NO_DEVICE_CONNECTED     |
| 0x53         | UX_ENDPOINT_HANDLE_UNKNOWN |

Table 3-227 Returns for API **AmbaUSBH\_Uvc\_GetMaxPacketSizeEx ()**.

#### Example:

```
UINT32 temp = 0;  
if (AmbaUSBH_Uvc_GetMaxPacketSizeEx(1, &temp) == OK) {  
    AmbaPrint("max packet size = %d", temp);  
}
```

#### See Also :

**AmbaUSBH\_Uvc\_GetMaxPacketSize ()**



### 3.4.80 AmbaUSBH\_Uvc\_RegisterCallbackEx

#### API Syntax:

```
UINT32 AmbaUSBH_Uvc_RegisterCallbackEx(USB_HOST_UVC_MULTI_CB_s *cb)
```

#### Function Description:

The extension function of **AmbaUSBH\_Uvc\_RegisterCallback()**. The application calls this API to register the specified callback for the camera insertion/removal events by the *id* parameter.

#### Parameters:

| Type                     | Parameters | Description   |
|--------------------------|------------|---|
| UINT32                   | id         | [Input] The ID of the UVC device, which is assigned by device insert/remove callback functions. See Section 3.4.1.14 for details. |
| USB_HOST_UVC_MULTI_CB_s* | cb         | [input] Pointer to callback functions. See Section 3.4.1.14 for details.  |

Table 3-228 Parameters for AmbaUSB API **AmbaUSBH\_Uvc\_RegisterCallbackEx ()**.

#### Return:

| Return Value | Description        |
|--------------|--------------------|
| 0            | UX_SUCCESS         |
| 0x1C         | UX_PARAMETER_ERROR |

Table 3-229 Returns for API **AmbaUSBH\_Uvc\_RegisterCallbackEx ()**.

#### Example:

```
static USB_HOST_UVC_MULTI_CB_s AmbaUsbHostUvcCb;  
static void Uvch_MediaInsert(UINT32 id)  
{  
    AmbaPrint("Web Cam [%d] Insert", id);  
}
```



```
static void Uvch_MediaRemove(UINT32 id)
{
    AmbaPrint("Web Cam [%d] Remove", id);
}

int AppUvch_Init(void)
{
    AmbaUsbHostUvcCb.MediaInsert = Uvch_MediaInsert;
    AmbaUsbHostUvcCb.MediaRemove = Uvch_MediaRemove;
    AmbaUSBH_Uvc_RegisterCallbackEx(&AmbaUsbHostUvcCb);

    return 0;
}
```

See Also:

*AmbaUSBH\_Uvc\_RegisterCallback()*



### 3.4.81 AmbaUSBH\_Uvc\_GetBacklightCtrlInfoEx

#### API Syntax:

```
UINT32 AmbaUSBH_Uvc_GetBacklightCtrlInfoEx(UINT32 id,  
                                UVCH_CTRL_INFO_s *info)
```

#### Function Description:

The extension function of **AmbaUSBH\_Uvc\_GetBacklightCtrlInfo()**. The application calls this API to get the specified backlight control value by the *id* parameter.

#### Parameters:

| Type              | Parameters | Description   |
|-------------------|------------|---|
| UINT32            | id         | [Input] The ID of the UVC device, which is assigned by device insert/remove callback functions. See Section 3.4.1.14 for details. |
| UVCH_CTRL_INFO_s* | info       | [Output] Pointer to control information. See Section 3.4.1.6 for details.   |

Table 3-230 Parameters for AmbaUSB API **AmbaUSBH\_Uvc\_GetBacklightCtrlInfoEx ()**.

#### Return:

| Return Value | Description                    |
|--------------|--------------------------------|
| 0            | UX_SUCCESS                     |
| 0x1C         | UX_PARAMETER_ERROR             |
| 0x5F         | UX_NO_DEVICE_CONNECTED         |
| 0x84         | UX_HOST_WRONG_CONTROL_SELECTOR |

Table 3-231 Returns for API **AmbaUSBH\_Uvc\_GetBacklightCtrlInfoEx ()**.

#### Example:

```
UVCH_CTRL_INFO_s info = {0};  
  
if (AmbaUSBH_Uvc_GetBacklightCtrlInfoEx(id, &info) == OK) {  
    AmbaPrint("Backlight %d ~ %d, cur %d", info.min, info.max, info.cur);  
}
```



See Also:

*AmbaUSBH\_Uvc\_GetBacklightCtrlInfo ()*

For PROTRULY Confidential Only



### 3.4.82 AmbaUSBH\_Uvc\_SetBacklightCtrlEx

#### API Syntax:

```
UINT32 AmbaUSBH_Uvc_SetBacklightCtrlEx(UINT32 id, UINT32 ctrl)
```

#### Function Description:

The extension function of **AmbaUSBH\_Uvc\_SetBacklightCtrl()**. The application calls this API to set the specified backlight control value, which is in the range defined by **AmbaUSBH\_Uvc\_GetBacklightCtrlInfoEx()**, by the *id* parameter.

#### Parameters:

| Type   | Parameters | Description   |
|--------|------------|---|
| UINT32 | id         | [Input] The ID of the UVC device, which is assigned by device insert/remove callback functions. See Section 3.4.1.14 for details. |
| UINT32 | ctrl       | [Input] Backlight control value.  |

Table 3-232 Parameters for AmbaUSB API **AmbaUSBH\_Uvc\_SetBacklightCtrlEx ()**.

#### Return:

| Return Value | Description            |
|--------------|------------------------|
| 0            | UX_SUCCESS             |
| 0x5F         | UX_NO_DEVICE_CONNECTED |

Table 3-233 Returns for API **AmbaUSBH\_Uvc\_SetBacklightCtrlEx ()**.

#### Example:

None

#### See Also :

**AmbaUSBH\_Uvc\_SetBacklightCtrl()**



### 3.4.83 AmbaUSBH\_Uvc\_GetBrightnessCtrlInfoEx

#### API Syntax:

```
UINT32 AmbaUSBH_Uvc_GetBrightnessCtrlInfoEx(UINT32 id,  
UVCH_CTRL_INFO_s *info)
```

#### Function Description:

The extension function of **AmbaUSBH\_Uvc\_GetBrightnessCtrlInfo()**. The application calls this API to get the specified brightness control value by the *id* parameter.

#### Parameters:

| Type              | Parameters | Description   |
|-------------------|------------|---|
| UINT32            | id         | [Input] The ID of the UVC device, which is assigned by device insert/remove callback functions. See Section 3.4.1.14 for details. |
| UVCH_CTRL_INFO_s* | info       | [Output] Pointer to control information. See Section 3.4.1.6 for details.   |

Table 3-234 Parameters for AmbaUSB API **AmbaUSBH\_Uvc\_GetBrightnessCtrlInfoEx ()**.

#### Return:

| Return Value | Description                    |
|--------------|--------------------------------|
| 0            | UX_SUCCESS                     |
| 0x1C         | UX_PARAMETER_ERROR             |
| 0x5F         | UX_NO_DEVICE_CONNECTED         |
| 0x84         | UX_HOST_WRONG_CONTROL_SELECTOR |

Table 3-235 Returns for API **AmbaUSBH\_Uvc\_GetBrightnessCtrlInfoEx ()**.

#### Example:

```
UVCH_CTRL_INFO_s info = {0};  
  
if (AmbaUSBH_Uvc_GetBrightnessCtrlInfoEx(id, &info) == OK) {  
    AmbaPrint("Brightness %d ~ %d, cur %d", info.min, info.max, info.cur);  
}
```



See Also :

*AmbaUSBH\_Uvc\_GetBrightnessCtrlInfo( )*

For PROTRULY Confidential Only



### 3.4.84 AmbaUSBH\_Uvc\_SetBrightnessCtrlEx

#### API Syntax:

```
UINT32 AmbaUSBH_Uvc_SetBrightnessCtrlEx(UINT32 id, UINT32 ctrl)
```

#### Function Description:

The extension function of **AmbaUSBH\_Uvc\_SetBrightnessCtrl()**. The application calls this API to set the specified backlight control value, which is in the range defined by **AmbaUSBH\_Uvc\_GetBrightnessCtrlInfoEx()**, by the *id* parameter.

#### Parameters:

| Type   | Parameters | Description   |
|--------|------------|---|
| UINT32 | id         | [Input] The ID of the UVC device, which is assigned by device insert/remove callback functions. See Section 3.4.1.14 for details. |
| UINT32 | ctrl       | [Input] Brightness control value.   |

Table 3-236 Parameters for AmbaUSB API **AmbaUSBH\_Uvc\_SetBrightnessCtrlEx ()**.

#### Return:

| Return Value | Description            |
|--------------|------------------------|
| 0            | UX_SUCCESS             |
| 0x5F         | UX_NO_DEVICE_CONNECTED |

Table 3-237 Returns for API **AmbaUSBH\_Uvc\_SetBrightnessCtrlEx ()**.

#### Example:

None

#### See Also :

**AmbaUSBH\_Uvc\_SetBrightnessCtrl ()**



### 3.4.85 AmbaUSBH\_Uvc\_GetContrastCtrlInfoEx

#### API Syntax:

```
UINT32 AmbaUSBH_Uvc_GetContrastCtrlInfoEx(UINT32 id,  
UVCH_CTRL_INFO_s *info)
```

#### Function Description:

The extension function of **AmbaUSBH\_Uvc\_GetContrastCtrlInfo()**. The application calls this API to get the specified contrast control value by the **id** parameter.

#### Parameters:

| Type              | Parameters | Description   |
|-------------------|------------|---|
| UINT32            | id         | [Input] The ID of UVC device, which is assigned by device insert/remove callback functions. See Section 3.4.1.14 for details. |
| UVCH_CTRL_INFO_s* | info       | [Output] Pointer to control information. See Section 3.4.1.6 for details.   |

Table 3-238 Parameters for AmbaUSB API **AmbaUSBH\_Uvc\_GetContrastCtrlInfoEx ()**.

#### Return:

| Return Value | Description                    |
|--------------|--------------------------------|
| 0            | UX_SUCCESS                     |
| 0x1C         | UX_PARAMETER_ERROR             |
| 0x5F         | UX_NO_DEVICE_CONNECTED         |
| 0x84         | UX_HOST_WRONG_CONTROL_SELECTOR |

Table 3-239 Returns for API **AmbaUSBH\_Uvc\_GetContrastCtrlInfoEx ()**.

#### Example:

```
UVCH_CTRL_INFO_s info = {0};  
  
if (AmbaUSBH_Uvc_GetContrastCtrlInfoEx (id, &info) == OK) {  
    AmbaPrint("Contrast %d ~ %d, cur %d", info.min, info.max, info.cur);  
}
```



See Also :

*AmbaUSBH\_Uvc\_GetContrastCtrlInfo( )*

For PROTRULY Confidential Only



### 3.4.86 AmbaUSBH\_Uvc\_SetContrastCtrlEx

#### API Syntax:

```
UINT32 AmbaUSBH_Uvc_SetContrastCtrlEx(UINT32 id, UINT32 ctrl)
```

#### Function Description:

The extension function of **AmbaUSBH\_Uvc\_SetContrastCtrl()**. The application calls this API to set the specified contrast control value, which is in the range defined by **AmbaUSBH\_Uvc\_GetContrastCtrlInfoEx()**, by the *id* parameter.

#### Parameters:

| Type   | Parameters | Description   |
|--------|------------|---|
| UINT32 | id         | [Input] The ID of UVC device, which is assigned by the device insert/remove callback functions. See Section 3.4.1.14 for details. |
| UINT32 | ctrl       | [Input] Contrast control value.   |

Table 3-240 Parameters for AmbaUSB API **AmbaUSBH\_Uvc\_SetContrastCtrlEx ()**.

#### Return:

| Return Value | Description            |
|--------------|------------------------|
| 0            | UX_SUCCESS             |
| 0x5F         | UX_NO_DEVICE_CONNECTED |

Table 3-241 Returns for API **AmbaUSBH\_Uvc\_SetContrastCtrlEx ()**.

#### Example:

None

#### See Also :

**AmbaUSBH\_Uvc\_SetContrastCtrl( )**



### 3.4.87 AmbaUSBH\_Uvc\_GetHueCtrlInfoEx

#### API Syntax:

```
UINT32 AmbaUSBH_Uvc_GetHueCtrlInfoEx(UINT32 id, UVCH_CTRL_INFO_s *info)
```

#### Function Description:

The extension function of **AmbaUSBH\_Uvc\_GetHueCtrlInfo()**. The application calls this API to get the specified hue control value by the *id* parameter.

#### Parameters:

| Type              | Parameters | Description   |
|-------------------|------------|---|
| UINT32            | id         | [Input] The ID of the UVC device, which is assigned by the device insert/remove callback functions. See Section 3.4.1.14 for details. |
| UVCH_CTRL_INFO_s* | info       | [Output] Pointer to control information. See Section 3.4.1.6 for details.   |

Table 3-242 Parameters for AmbaUSB API **AmbaUSBH\_Uvc\_GetHueCtrlInfoEx ()**.

#### Return:

| Return Value | Description                    |
|--------------|--------------------------------|
| 0            | UX_SUCCESS                     |
| 0x1C         | UX_PARAMETER_ERROR             |
| 0x5F         | UX_NO_DEVICE_CONNECTED         |
| 0x84         | UX_HOST_WRONG_CONTROL_SELECTOR |

Table 3-243 Returns for API **AmbaUSBH\_Uvc\_GetHueCtrlInfoEx ()**.

#### Example:

```
UVCH_CTRL_INFO_s info = {0};  
  
if (AmbaUSBH_Uvc_GetHueCtrlInfoEx (id, &info) == OK) {  
    AmbaPrint("Hue %d ~ %d, cur %d", info.min, info.max, info.cur);  
}
```



See Also :

*AmbaUSBH\_Uvc\_GetHueCtrlInfo ()*

For PROTRULY Confidential Only



### 3.4.88 AmbaUSBH\_Uvc\_SetHueCtrlEx

#### API Syntax:

```
UINT32 AmbaUSBH_Uvc_SetHueCtrlEx(UINT32 id, UINT32 ctrl)
```

#### Function Description:

The extension function of *AmbaUSBH\_Uvc\_SetHueCtrl()*. The application calls this API to set the specified hue control value, which is in the range defined by *AmbaUSBH\_Uvc\_GetHueCtrlInfoEx()*, by the *id* parameter.

#### Parameters:

| Type   | Parameters | Description   |
|--------|------------|---|
| UINT32 | id         | [Input] The ID of the UVC device, which is assigned by device insert/remove callback functions. See Section 3.4.1.14 for details. |
| UINT32 | ctrl       | [Input] Hue control value.  |

Table 3-244 Parameters for AmbaUSB API *AmbaUSBH\_Uvc\_SetHueCtrlEx()*.

#### Return:

| Return Value | Description            |
|--------------|------------------------|
| 0            | UX_SUCCESS             |
| 0x5F         | UX_NO_DEVICE_CONNECTED |

Table 3-245 Returns for API *AmbaUSBH\_Uvc\_SetHueCtrlEx()*.

#### Example:

None

#### See Also :

*AmbaUSBH\_Uvc\_SetHueCtrl()*



### 3.4.89 AmbaUSBH\_Uvc\_GetSaturationCtrlInfoEx

#### API Syntax:

```
UINT32 AmbaUSBH_Uvc_GetSaturationCtrlInfoEx(UINT32 id,  
                                UVCH_CTRL_INFO_s *info)
```

#### Function Description:

The extension function of **AmbaUSBH\_Uvc\_GetSaturationCtrlInfo()**. The application calls this API to get the specified saturation control value by the *id* parameter.

#### Parameters:

| Type              | Parameters | Description   |
|-------------------|------------|---|
| UINT32            | id         | [Input] The ID of the UVC device, which is assigned by device insert/remove callback functions. See Section 3.4.1.14 for details. |
| UVCH_CTRL_INFO_s* | Info       | [Output] Pointer to control information. See Section 3.4.1.6 for details.   |

Table 3-246 Parameters for AmbaUSB API **AmbaUSBH\_Uvc\_GetSaturationCtrlInfoEx ()**.

#### Return:

| Return Value | Description                    |
|--------------|--------------------------------|
| 0            | UX_SUCCESS                     |
| 0x1C         | UX_PARAMETER_ERROR             |
| 0x5F         | UX_NO_DEVICE_CONNECTED         |
| 0x84         | UX_HOST_WRONG_CONTROL_SELECTOR |

Table 3-247 Returns for API **AmbaUSBH\_Uvc\_GetSaturationCtrlInfoEx ()**.

#### Example:

```
UVCH_CTRL_INFO_s info = {0};  
  
if (AmbaUSBH_Uvc_GetSaturationCtrlInfoEx (id, &info) == OK) {  
    AmbaPrint("Saturation %d ~ %d, cur %d", info.min, info.max, info.cur);  
}
```



See Also :

*AmbaUSBH\_Uvc\_GetSaturationCtrlInfo( )*

For PROTRULY Confidential Only



### 3.4.90 AmbaUSBH\_Uvc\_SetSaturationCtrlEx

#### API Syntax:

```
UINT32 AmbaUSBH_Uvc_SetSaturationCtrlEx(UINT32 id, UINT32 ctrl)
```

#### Function Description:

The extension function of **AmbaUSBH\_Uvc\_SetSaturationCtrl()**. The application calls this API to set the specified saturation control value, which is in the range defined by **AmbaUSBH\_Uvc\_GetSaturationCtrlInfoEx()**, by the *id* parameter.

#### Parameters:

| Type   | Parameters | Description   |
|--------|------------|---|
| UINT32 | id         | [Input] The ID of the UVC device, which is assigned by device insert/remove callback functions. See Section 3.4.1.14 for details. |
| UINT32 | ctrl       | [Input] Saturation control value.   |

Table 3-248 Parameters for AmbaUSB API **AmbaUSBH\_Uvc\_SetSaturationCtrlEx()**.

#### Return:

| Return Value | Description            |
|--------------|------------------------|
| 0            | UX_SUCCESS             |
| 0x5F         | UX_NO_DEVICE_CONNECTED |

Table 3-249 Returns for API **AmbaUSBH\_Uvc\_SetSaturationCtrlEx()**.

#### Example:

None

#### See Also :

**AmbaUSBH\_Uvc\_SetSaturationCtrl()**



### 3.4.91 AmbaUSBH\_Uvc\_GetSharpnessCtrlInfoEx

#### API Syntax:

```
UINT32 AmbaUSBH_Uvc_GetSharpnessCtrlInfoEx (UINT32 id,  
                                UVCH_CTRL_INFO_s *info)
```

#### Function Description:

The extension function of **AmbaUSBH\_Uvc\_GetSharpnessCtrlInfo()**. The application calls this API to get the specified sharpness control value by the *id* parameter.

#### Parameters:

| Type              | Parameters | Description   |
|-------------------|------------|---|
| UINT32            | id         | [Input] The ID of the UVC device, which is assigned by device insert/remove callback functions. See Section 3.4.1.14 for details. |
| UVCH_CTRL_INFO_s* | Info       | [Output] Pointer to control information. See Section 3.4.1.6 for details.   |

Table 3-250 Parameters for AmbaUSB API **AmbaUSBH\_Uvc\_GetSharpnessCtrlInfoEx ()**.

#### Return:

| Return Value | Description                    |
|--------------|--------------------------------|
| 0            | UX_SUCCESS                     |
| 0x1C         | UX_PARAMETER_ERROR             |
| 0x5F         | UX_NO_DEVICE_CONNECTED         |
| 0x84         | UX_HOST_WRONG_CONTROL_SELECTOR |

Table 3-251 Returns for API **AmbaUSBH\_Uvc\_GetSharpnessCtrlInfoEx ()**.

#### Example:

```
UVCH_CTRL_INFO_s info = {0};  
  
if (AmbaUSBH_Uvc_GetSharpnessCtrlInfoEx (id, &info) == OK) {  
    AmbaPrint("Sharpness %d ~ %d, cur %d", info.min, info.max, info.cur);  
}
```



See Also :

*AmbaUSBH\_Uvc\_GetSharpnessCtrlInfo ()*

For PROTRULY Confidential Only



### 3.4.92 AmbaUSBH\_Uvc\_SetSharpnessCtrlEx

#### API Syntax:

```
UINT32 AmbaUSBH_Uvc_SetSharpnessCtrlEx(UINT32 id, UINT32 ctrl)
```

#### Function Description:

The extension function of **AmbaUSBH\_Uvc\_SetSharpnessCtrl()**. The application calls this API to set the specified sharpness control value, which is in the range defined by **AmbaUSBH\_Uvc\_GetSharpnessCtrlInfoEx()**, by the *id* parameter.

#### Parameters:

| Type   | Parameters | Description   |
|--------|------------|---|
| UINT32 | id         | [Input] The ID of the UVC device, which is assigned by device insert/remove callback functions. See Section 3.4.1.14 for details. |
| UINT32 | ctrl       | [Input] Sharpness control value.  |

Table 3-252 Parameters for AmbaUSB API **AmbaUSBH\_Uvc\_SetSharpnessCtrlEx()**.

#### Return:

| Return Value | Description            |
|--------------|------------------------|
| 0            | UX_SUCCESS             |
| 0x5F         | UX_NO_DEVICE_CONNECTED |

Table 3-253 Returns for API **AmbaUSBH\_Uvc\_SetSharpnessCtrlEx()**.

#### Example:

None

#### See Also :

**AmbaUSBH\_Uvc\_SetSharpnessCtrl()**



### 3.4.93 AmbaUSBH\_Uvc\_GetGammaCtrlInfoEx

#### API Syntax:

```
UINT32 AmbaUSBH_Uvc_GetGammaCtrlInfoEx (UINT32 id, UVCH_CTRL_INFO_s *info)
```

#### Function Description:

The extension function of **AmbaUSBH\_Uvc\_GetGammaCtrlInfo()**. The application calls this API to get the specified GAMMA control value by the *id* parameter.

#### Parameters:

| Type              | Parameters | Description   |
|-------------------|------------|---|
| UINT32            | id         | [Input] The ID of the UVC device, which is assigned by the device insert/remove callback functions. See Section 3.4.1.14 for details. |
| UVCH_CTRL_INFO_s* | Info       | [Output] Pointer to control information. See Section 3.4.1.6 for details.   |

Table 3-254 Parameters for AmbaUSB API **AmbaUSBH\_Uvc\_GetGammaCtrlInfoEx ()**.

#### Return:

| Return Value | Description                    |
|--------------|--------------------------------|
| 0            | UX_SUCCESS                     |
| 0x1C         | UX_PARAMETER_ERROR             |
| 0x5F         | UX_NO_DEVICE_CONNECTED         |
| 0x84         | UX_HOST_WRONG_CONTROL_SELECTOR |

Table 3-255 Returns for API **AmbaUSBH\_Uvc\_GetGammaCtrlInfoEx ()**.

#### Example:

```
UVCH_CTRL_INFO_s info = {0};  
if (AmbaUSBH_Uvc_GetGammaCtrlInfoEx (id, &info) == OK) {  
    AmbaPrint("Gamma %d ~ %d, cur %d", info.min, info.max, info.cur);  
}
```



See Also :

*AmbaUSBH\_Uvc\_GetGammaCtrlInfo( )*

For PROTRULY Confidential Only



### 3.4.94 AmbaUSBH\_Uvc\_SetGammaCtrlEx

#### API Syntax:

```
UINT32 AmbaUSBH_Uvc_SetGammaCtrlEx(UINT32 id, UINT32 ctrl)
```

#### Function Description:

The extension function of **AmbaUSBH\_Uvc\_SetGammaCtrl()**. The application calls this API to set the specified GAMMA control value, which is in the range defined by **AmbaUSBH\_Uvc\_GetGammaCtrlInfoEx()**, by the *id* parameter.

#### Parameters:

| Type   | Parameters | Description   |
|--------|------------|---|
| UINT32 | id         | [Input] The ID of the UVC device, which is assigned by device insert/remove callback functions. See Section 3.4.1.14 for details. |
| UINT32 | ctrl       | [Input] GAMMA control value.  |

Table 3-256 Parameters for AmbaUSB API **AmbaUSBH\_Uvc\_SetGammaCtrlEx()**.

#### Return:

| Return Value | Description            |
|--------------|------------------------|
| 0            | UX_SUCCESS             |
| 0x5F         | UX_NO_DEVICE_CONNECTED |

Table 3-257 Returns for API **AmbaUSBH\_Uvc\_SetGammaCtrlEx()**.

#### Example:

None

#### See Also :

**AmbaUSBH\_Uvc\_SetGammaCtrl()**



### 3.4.95 AmbaUSBH\_Uvc\_GetWBTemperatureCtrlInfoEx

#### API Syntax:

```
UINT32 AmbaUSBH_Uvc_GetWBTemperatureCtrlInfoEx (UINT32 id,  
                                              UVCH_CTRL_INFO_s *info)
```

#### Function Description:

The extension function of **AmbaUSBH\_Uvc\_GetWBTemperatureCtrlInfo()**. The application calls this API to get the specified white balance temperature control value by the **id** parameter.

#### Parameters:

| Type              | Parameters | Description   |
|-------------------|------------|---|
| UINT32            | id         | [Input] The ID of the UVC device, which is assigned by device insert/remove callback functions. See Section 3.4.1.14 for details. |
| UVCH_CTRL_INFO_s* | Info       | [Output] Pointer to control information. See Section 3.4.1.6 for details.   |

Table 3-258 Parameters for AmbaUSB API **AmbaUSBH\_Uvc\_GetWBTemperatureCtrlInfoEx ()**.

#### Return:

| Return Value | Description                    |
|--------------|--------------------------------|
| 0            | UX_SUCCESS                     |
| 0x1C         | UX_PARAMETER_ERROR             |
| 0x5F         | UX_NO_DEVICE_CONNECTED         |
| 0x84         | UX_HOST_WRONG_CONTROL_SELECTOR |

Table 3-259 Returns for API **AmbaUSBH\_Uvc\_GetWBTemperatureCtrlInfoEx ()**.



**Example:**

```
UVCH_CTRL_INFO_s info = {0};  
if (AmbaUSBH_Uvc_GetWBTemperatureCtrlInfoEx (id, &info) == OK) {  
    AmbaPrint("WBT %d ~ %d, cur %d", info.min, info.max, info.cur);  
}
```

**See Also:**

*AmbaUSBH\_Uvc\_GetWBComponentCtrlInfo()*



### 3.4.96 AmbaUSBH\_Uvc\_SetWBTemperatureCtrlEx

#### API Syntax:

```
UINT32 AmbaUSBH_Uvc_SetWBTemperatureCtrlEx(UINT32 id, UINT32 ctrl)
```

#### Function Description:

The extension function of *AmbaUSBH\_Uvc\_SetWBTemperatureCtrl()*. The application calls this API to set the specified white balance temperature control value, which is in the range defined by *AmbaUSBH\_Uvc\_GetWBTemperatureCtrlInfoEx ()*, by the *id* parameter.

#### Parameters:

| Type   | Parameters | Description   |
|--------|------------|---|
| UINT32 | id         | [Input] The ID of the UVC device, which is assigned by device insert/remove callback functions. See Section 3.4.1.14 for details. |
| UINT32 | ctrl       | [Input] White balance temperature control value.  |

Table 3-260 Parameters for AmbaUSB API *AmbaUSBH\_Uvc\_SetGammaCtrlEx ()*.

#### Return:

| Return Value | Description            |
|--------------|------------------------|
| 0            | UX_SUCCESS             |
| 0x5F         | UX_NO_DEVICE_CONNECTED |

Table 3-261 Returns for API *AmbaUSBH\_Uvc\_SetWBTemperatureCtrlEx ()*.

#### Example:

None

#### See Also :

*AmbaUSBH\_Uvc\_SetWBTemperatureCtrl()*



### 3.4.97 AmbaUSBH\_Uvc\_GetWBComponentCtrlInfoEx

#### API Syntax:

```
UINT32 AmbaUSBH_Uvc_GetWBComponentCtrlInfoEx(UINT32 id,  
                                              UVCH_CTRL_INFO_s *InfoBlue,  
                                              UVCH_CTRL_INFO_s *InfoRed)
```

#### Function Description:

The extension function of ***AmbaUSBH\_Uvc\_GetWBComponentCtrlInfo()***. The application calls this API to get the specified white balance component control value by the *id* parameter.

#### Parameters:

| Type              | Parameters | Description   |
|-------------------|------------|---|
| UINT32            | id         | [Input] The ID of UVC device, which is assigned by device insert/remove callback functions. See Section 3.4.1.14 for details. |
| UVCH_CTRL_INFO_s* | InfoBlue   | [Output] Pointer to control information. See Section 3.4.1.6 for details.   |
| UVCH_CTRL_INFO_s* | InfoRed    | [Output] Pointer to control information. See Section 3.4.1.6 for details.   |

Table 3-262 Parameters for AmbaUSB API ***AmbaUSBH\_Uvc\_GetWBComponentCtrlInfoEx ()***.

#### Return:

| Return Value | Description                    |
|--------------|--------------------------------|
| 0            | UX_SUCCESS                     |
| 0x1C         | UX_PARAMETER_ERROR             |
| 0x5F         | UX_NO_DEVICE_CONNECTED         |
| 0x84         | UX_HOST_WRONG_CONTROL_SELECTOR |

Table 3-263 Returns for API ***AmbaUSBH\_Uvc\_GetWBComponentCtrlInfoEx ()***.



**Example:**

```
UVCH_CTRL_INFO_s InfoBlue = {0};  
UVCH_CTRL_INFO_s InfoRed = {0};  
  
if (AmbaUSBH_Uvc_GetWBComponentCtrlInfoEx (id, &InfoBlue, &InfoRed) == OK)  
{  
    AmbaPrint("WBC Blue %d ~ %d, cur %d", InfoBlue.min, InfoBlue.max,  
    InfoBlue.cur);  
    AmbaPrint("WBC Red %d ~ %d, cur %d", InfoRed.min, InfoRed.max,  
    InfoRed.cur);  
}
```

**See Also :**

*AmbaUSBH\_Uvc\_GetWBComponentCtrlInfo()*



### 3.4.98 AmbaUSBH\_Uvc\_SetWBComponentCtrlEx

#### API Syntax:

```
UINT32 AmbaUSBH_Uvc_SetWBComponentCtrlEx(UINT32 id, UINT16 blue, UINT16  
red)
```

#### Function Description:

The extension function of *AmbaUSBH\_Uvc\_SetWBComponentCtrl()*. The application calls this API to set the specified white balance component control value, which is in the range defined by *AmbaUSBH\_Uvc\_GetWBComponentCtrlInfoEx()*, by the *id* parameter.

#### Parameters:

| Type   | Parameters | Description   |
|--------|------------|---|
| UINT32 | id         | [Input] The ID of the UVC device, which is assigned by device insert/remove callback functions. See Section 3.4.1.14 for details. |
| UINT32 | blue       | [Input] White balance component control value.  |
| UINT32 | red        | [Input] White balance component control value.  |

Table 3-264 Parameters for AmbaUSB API *AmbaUSBH\_Uvc\_SetWBComponentCtrlEx()*.

#### Return:

| Return Value | Description            |
|--------------|------------------------|
| 0            | UX_SUCCESS             |
| 0x5F         | UX_NO_DEVICE_CONNECTED |

Table 3-265 Returns for API *AmbaUSBH\_Uvc\_SetWBComponentCtrlEx()*.

#### Example:

None

#### See Also :

*AmbaUSBH\_Uvc\_SetWBComponentCtrl()*



### 3.4.99 AmbaUSBH\_Uvc\_GetGainCtrlInfoEx

#### API Syntax:

```
UINT32 AmbaUSBH_Uvc_GetGainCtrlInfoEx (UINT32 id, UVCH_CTRL_INFO_s *info)
```

#### Function Description:

The extension function of **AmbaUSBH\_Uvc\_GetGainCtrlInfo()**. The application calls this API to get the specified gain control value by the *id* parameter.

#### Parameters:

| Type              | Parameters | Description   |
|-------------------|------------|---|
| UINT32            | id         | [Input] The ID of the UVC device, which is assigned by the device insert/remove callback functions. See Section 3.4.1.14 for details. |
| UVCH_CTRL_INFO_s* | Info       | [Output] Pointer to control information. See Section 3.4.1.6 for details.   |

Table 3-266 Parameters for AmbaUSB API **AmbaUSBH\_Uvc\_GetGainCtrlInfoEx ()**.

#### Return:

| Return Value | Description                    |
|--------------|--------------------------------|
| 0            | UX_SUCCESS                     |
| 0x1C         | UX_PARAMETER_ERROR             |
| 0x5F         | UX_NO_DEVICE_CONNECTED         |
| 0x84         | UX_HOST_WRONG_CONTROL_SELECTOR |

Table 3-267 Returns for API **AmbaUSBH\_Uvc\_GetGainCtrlInfoEx ()**.

#### Example:

```
UVCH_CTRL_INFO_s info = {0};  
  
if (AmbaUSBH_Uvc_GetGainCtrlInfoEx (id, &info) == OK) {  
    AmbaPrint("Gain %d ~ %d, cur %d", info.min, info.max, info.cur);  
}
```



See Also :

*AmbaUSBH\_Uvc\_GetGainCtrlInfo( )*

For PROTRULY Confidential Only



### 3.4.100 AmbaUSBH\_Uvc\_SetGainCtrlEx

#### API Syntax:

```
UINT32 AmbaUSBH_Uvc_SetGainCtrlEx(UINT32 id, UINT32 ctrl)
```

#### Function Description:

The extension function of *AmbaUSBH\_Uvc\_SetGainCtrl()*. The application calls this API to set the specified gain control value, which is in the range defined by *AmbaUSBH\_Uvc\_GetGainCtrlInfoEx()*, by the *id* parameter.

#### Parameters:

| Type   | Parameters | Description   |
|--------|------------|---|
| UINT32 | id         | [Input] The ID of the UVC device, which is assigned by the device insert/remove callback functions. See Section 3.4.1.14 for details. |
| UINT32 | ctrl       | [Input] Gain control value.   |

Table 3-268 Parameters for AmbaUSB API *AmbaUSBH\_Uvc\_SetGainCtrlEx()*.

#### Return:

| Return Value | Description            |
|--------------|------------------------|
| 0            | UX_SUCCESS             |
| 0x5F         | UX_NO_DEVICE_CONNECTED |

Table 3-269 Returns for API *AmbaUSBH\_Uvc\_SetGainCtrlEx()*.

#### Example:

None

#### See Also :

*AmbaUSBH\_Uvc\_SetGainCtrl()*



### 3.4.101 AmbaUSBH\_Uvc\_GetCurLineFrequencyCtrlEx

#### API Syntax:

```
UINT32 AmbaUSBH_Uvc_GetCurLineFrequencyCtrlEx(UINT32 id, UINT32 *ctrl)
```

#### Function Description:

The extension function of **AmbaUSBH\_Uvc\_GetCurLineFrequencyCtrl()**. The application calls this API to get the specified line frequency control value by the *id* parameter.

#### Parameters:

| Type    | Parameters | Description   |
|---------|------------|---|
| UINT32  | id         | [Input] The ID of the UVC device, which is assigned by the device insert/remove callback functions. See Section 3.4.1.14 for details. |
| UINT32* | ctrl       | [Output] Line frequency value. See Section 3.4.1.9 for details.   |

Table 3-270 Parameters for AmbaUSB API **AmbaUSBH\_Uvc\_GetGainCtrlInfoEx ()**.

#### Return:

| Return Value | Description                    |
|--------------|--------------------------------|
| 0            | UX_SUCCESS                     |
| 0x1C         | UX_PARAMETER_ERROR             |
| 0x5F         | UX_NO_DEVICE_CONNECTED         |
| 0x84         | UX_HOST_WRONG_CONTROL_SELECTOR |

Table 3-271 Returns for API **AmbaUSBH\_Uvc\_GetGainCtrlInfoEx()**.

#### Example:

```
UINT32 info = 0;  
  
if (AmbaUSBH_Uvc_GetCurLineFrequencyCtrlEx (id, &info) == OK) {  
    AmbaPrint("line frequency %d ~ %d, cur %d", info.min, info.max,  
    info.cur);  
}
```



See Also :

*AmbaUSBH\_Uvc\_GetCurLineFrequencyCtrl( )*

For PROTRULY Confidential Only



### 3.4.102 AmbaUSBH\_Uvc\_SetLineFrequencyCtrlEx

#### API Syntax:

```
UINT32 AmbaUSBH_Uvc_SetLineFrequencyCtrlEx(UINT32 id,  
                                         UVC_POWER_LINE_FREQUENCY_e ctrl)
```

#### Function Description:

The extension function of **AmbaUSBH\_Uvc\_SetLineFrequencyCtrl()**. The application calls this API to set the specified line frequency control value by the *id* parameter.

#### Parameters:

| Type                       | Parameters | Description   |
|----------------------------|------------|---|
| UINT32                     | id         | [Input] The ID of the UVC device, which is assigned by device insert/remove callback functions. See Section 3.4.1.14 for details. |
| UVC_POWER_LINE_FREQUENCY_e | ctrl       | Power line frequency control value. See Section 3.4.1.9 for details.  |

Table 3-272 Parameters for AmbaUSB API **AmbaUSBH\_Uvc\_SetLineFrequencyCtrlEx ()**.

#### Return:

| Return Value | Description            |
|--------------|------------------------|
| 0            | UX_SUCCESS             |
| 0x5F         | UX_NO_DEVICE_CONNECTED |

Table 3-273 Returns for API **AmbaUSBH\_Uvc\_SetLineFrequencyCtrlEx ()**.

#### Example:

None

#### See Also :

**AmbaUSBH\_Uvc\_SetLineFrequencyCtrl( )**



### 3.4.103 AmbaUSBH\_Uvc\_GetHueAutoCtrlInfoEx

#### API Syntax:

```
UINT32 AmbaUSBH_Uvc_GetHueAutoCtrlInfoEx (UINT32 id,  
                                UVCH_CTRL_INFO_s *info)
```

#### Function Description:

The extension function of **AmbaUSBH\_Uvc\_GetHueAutoCtrlInfo()**. The application calls this API to get the specified hue auto value by the *id* parameter.

#### Parameters:

| Type              | Parameters | Description   |
|-------------------|------------|---|
| UINT32            | id         | [Input] The ID of the UVC device, which is assigned by device insert/remove callback functions. See Section 3.4.1.14 for details. |
| UVCH_CTRL_INFO_s* | Info       | [Output] Pointer to control information. See Section 3.4.1.6 for details.   |

Table 3-274 Parameters for AmbaUSB API **AmbaUSBH\_Uvc\_GetHueAutoCtrlInfoEx ()**.

#### Return:

| Return Value | Description                    |
|--------------|--------------------------------|
| 0            | UX_SUCCESS                     |
| 0x1C         | UX_PARAMETER_ERROR             |
| 0x5F         | UX_NO_DEVICE_CONNECTED         |
| 0x84         | UX_HOST_WRONG_CONTROL_SELECTOR |

Table 3-275 Returns for API **AmbaUSBH\_Uvc\_GetHueAutoCtrlInfoEx ()**.

#### Example:

```
UVCH_CTRL_INFO_s info = {0};  
  
if (AmbaUSBH_Uvc_GetHueAutoCtrlInfoEx (id, &info) == OK) {  
    AmbaPrint("Hue Auto %d ~ %d, cur %d", info.min, info.max, info.cur);  
}
```



See Also :

[\*AmbaUSBH\\_Uvc\\_GetHueAutoCtrlInfo\( \)\*](#)

For PROTRULY Confidential Only



### 3.4.104 AmbaUSBH\_Uvc\_SetHueAutoCtrlEx

#### API Syntax:

```
UINT32 AmbaUSBH_Uvc_SetHueAutoCtrlEx(UINT32 id, UINT32 ctrl)
```

#### Function Description:

The extension function of **AmbaUSBH\_Uvc\_SetHueAutoCtrl()**. The application calls this API to set the specified hue auto control value, which is in the range defined by **AmbaUSBH\_Uvc\_GetHueAutoCtrlInfoEx ()**, by the *id* parameter.

#### Parameters:

| Type   | Parameters | Description   |
|--------|------------|---|
| UINT32 | id         | [Input] The ID of the UVC device, which is assigned by the device insert/remove callback functions. See Section 3.4.1.14 for details. |
| UINT32 | ctrl       | [Input] Hue Auto control value.   |

Table 3-276 Parameters for AmbaUSB API **AmbaUSBH\_Uvc\_SetHueAutoCtrlEx ()**.

#### Return:

| Return Value | Description            |
|--------------|------------------------|
| 0            | UX_SUCCESS             |
| 0x5F         | UX_NO_DEVICE_CONNECTED |

Table 3-277 Returns for API **AmbaUSBH\_Uvc\_SetHueAutoCtrlEx ()**.

#### Example:

None

#### See Also :

**AmbaUSBH\_Uvc\_SetHueAutoCtrl( )**



### 3.4.105 AmbaUSBH\_Uvc\_GetWBTempAutoCtrlInfoEx

#### API Syntax:

```
UINT32 AmbaUSBH_Uvc_GetWBTempAutoCtrlInfoEx (UINT32 id,  
                                UVCH_CTRL_INFO_s *info)
```

#### Function Description:

The extension function of **AmbaUSBH\_Uvc\_GetWBTempAutoCtrlInfo()**. The application calls this API to get the specified white balance temperature auto value by the *id* parameter.

#### Parameters:

| Type              | Parameters | Description   |
|-------------------|------------|---|
| UINT32            | id         | [Input] The ID of the UVC device, which is assigned by the device insert/remove callback functions. See Section 3.4.1.14 for details. |
| UVCH_CTRL_INFO_s* | Info       | [Output] Pointer to control information. See Section 3.4.1.6 for details.   |

Table 3-278 Parameters for AmbaUSB API **AmbaUSBH\_Uvc\_GetWBTempAutoCtrlInfoEx ()**.

#### Return:

| Return Value | Description                    |
|--------------|--------------------------------|
| 0            | UX_SUCCESS                     |
| 0x1C         | UX_PARAMETER_ERROR             |
| 0x5F         | UX_NO_DEVICE_CONNECTED         |
| 0x84         | UX_HOST_WRONG_CONTROL_SELECTOR |

Table 3-279 Returns for API **AmbaUSBH\_Uvc\_GetWBTempAutoCtrlInfoEx ()**.

#### Example:

```
UVCH_CTRL_INFO_s info = {0};  
  
if (AmbaUSBH_Uvc_GetWBTempAutoCtrlInfoEx (id, &info) == OK) {  
    AmbaPrint("WBT Auto %d ~ %d, cur %d", info.min, info.max, info.cur);  
}
```



See Also :

*AmbaUSBH\_Uvc\_GetWBTempAutoCtrlInfo( )*

For PROTRULY Confidential Only



### 3.4.106 AmbaUSBH\_Uvc\_SetWBTempAutoCtrlEx

#### API Syntax:

```
UINT32 AmbaUSBH_Uvc_SetWBTempAutoCtrlEx(UINT32 id, UINT32 ctrl)
```

#### Function Description:

The extension function of *AmbaUSBH\_Uvc\_SetWBTempAutoCtrl()*. The application calls this API to set the specified white balance temperature auto control value, which is in the range defined by *AmbaUSBH\_Uvc\_GetWBTempAutoCtrlInfoEx ()*, by the *id* parameter.

#### Parameters:

| Type   | Parameters | Description   |
|--------|------------|---|
| UINT32 | id         | [Input] The ID of the UVC device, which is assigned by device insert/remove callback functions. See Section 3.4.1.14 for details. |
| UINT32 | ctrl       | [Input] White balance temperature Auto control value.   |

Table 3-280 Parameters for AmbaUSB API *AmbaUSBH\_Uvc\_SetWBTempAutoCtrlEx ()*.

#### Return:

| Return Value | Description            |
|--------------|------------------------|
| 0            | UX_SUCCESS             |
| 0x5F         | UX_NO_DEVICE_CONNECTED |

Table 3-281 Returns for API *AmbaUSBH\_Uvc\_SetWBTempAutoCtrlEx ()*.

#### Example:

None

#### See Also :

*AmbaUSBH\_Uvc\_SetWBTempAutoCtrl()*



### 3.4.107 AmbaUSBH\_Uvc\_GetWBCompAutoCtrlInfoEx

#### API Syntax:

```
UINT32 AmбаUSBH_Uvc_GetWBCompAutoCtrlInfoEx (UINT32 id,  
                                              UVCH_CTRL_INFO_s *info)
```

#### Function Description:

The extension function of **AmбаUSBH\_Uvc\_GetWBCompAutoCtrlInfo()**. The application calls this API to get the specified white balance component auto value by the **id** parameter.

#### Parameters:

| Type              | Parameters | Description   |
|-------------------|------------|---|
| UINT32            | id         | [Input] The ID of the UVC device, which is assigned by the device insert/remove callback functions. See Section 3.4.1.14 for details. |
| UVCH_CTRL_INFO_s* | Info       | [Output] Pointer to control information. See Section 3.4.1.6 for details.   |

Table 3-282 Parameters for AmbaUSB API **AmбаUSBH\_Uvc\_GetWBCompAutoCtrlInfoEx ()**.

#### Return:

| Return Value | Description                    |
|--------------|--------------------------------|
| 0            | UX_SUCCESS                     |
| 0x1C         | UX_PARAMETER_ERROR             |
| 0x5F         | UX_NO_DEVICE_CONNECTED         |
| 0x84         | UX_HOST_WRONG_CONTROL_SELECTOR |

Table 3-283 Returns for API **AmбаUSBH\_Uvc\_GetWBCompAutoCtrlInfoEx ()**.

#### Example:

```
UVCH_CTRL_INFO_s info = {0};  
  
if (AmбаUSBH_Uvc_GetWBCompAutoCtrlInfoEx (id, &info) == OK) {  
    AmbaPrint("WBC Auto %d ~ %d, cur %d", info.min, info.max, info.cur);  
}
```



See Also :

*AmbaUSBH\_Uvc\_GetWBCompAutoCtrlInfo( )*

For PROTRULY Confidential Only



### 3.4.108 AmbaUSBH\_Uvc\_SetWBCompAutoCtrlEx

#### API Syntax:

```
UINT32 AmbaUSBH_Uvc_SetWBCompAutoCtrlEx(UINT32 id, UINT32 ctrl)
```

#### Function Description:

The extension function of *AmbaUSBH\_Uvc\_SetWBCompAutoCtrl()*. The application calls this API to set the specified white balance component auto control value, which is in the range defined by *AmbaUSBH\_Uvc\_GetWBCompAutoCtrlInfoEx()*, by the *id* parameter.

#### Parameters:

| Type   | Parameters | Description   |
|--------|------------|---|
| UINT32 | id         | [Input] The ID of the UVC device, which is assigned by device insert/remove callback functions. See Section 3.4.1.14 for details. |
| UINT32 | ctrl       | [Input] White balance component Auto control value.   |

Table 3-284 Parameters for AmbaUSB API *AmbaUSBH\_Uvc\_SetWBCompAutoCtrlEx ()*.

#### Return:

| Return Value | Description            |
|--------------|------------------------|
| 0            | UX_SUCCESS             |
| 0x5F         | UX_NO_DEVICE_CONNECTED |

Table 3-285 Returns for API *AmbaUSBH\_Uvc\_SetWBCompAutoCtrlEx ()*.

#### Example:

None

#### See Also :

*AmbaUSBH\_Uvc\_SetWBCompAutoCtrl()*



### 3.4.109 AmbaUSBH\_Uvc\_GetDigitalMultiplierCtrlInfoEx

#### API Syntax:

```
UINT32 AmbaUSBH_Uvc_GetDigitalMultiplierCtrlInfoEx(UINT32 id,  
UVCH_CTRL_INFO_s *info)
```

#### Function Description:

The extension function of **AmbaUSBH\_Uvc\_GetDigitalMultiplierCtrlInfo()**. The application calls this API to get the specified digital multiplier control value by the *id* parameter.

#### Parameters:

| Type              | Parameters | Description   |
|-------------------|------------|---|
| UINT32            | id         | [Input] The ID of the UVC device, which is assigned by device insert/remove callback functions. See Section 3.4.1.14 for details. |
| UVCH_CTRL_INFO_s* | Info       | [Output] Pointer to control information. See Section 3.4.1.6 for details.   |

Table 3-286 Parameters for AmbaUSB API **AmbaUSBH\_Uvc\_GetDigitalMultiplierCtrlInfoEx ()**.

#### Return:

| Return Value | Description                    |
|--------------|--------------------------------|
| 0            | UX_SUCCESS                     |
| 0x1C         | UX_PARAMETER_ERROR             |
| 0x5F         | UX_NO_DEVICE_CONNECTED         |
| 0x84         | UX_HOST_WRONG_CONTROL_SELECTOR |

Table 3-287 Returns for API **AmbaUSBH\_Uvc\_GetDigitalMultiplierCtrlInfoEx ()**.



**Example:**

```
UVCH_CTRL_INFO_s info = {0};  
if (AmbaUSBH_Uvc_GetDigitalMultiplierCtrlInfoEx (id, &info) == OK) {  
    AmbaPrint("digital multiplier %d ~ %d, cur %d", info.min, info.max,  
    info.cur);  
}
```

**See Also :**

***AmbaUSBH\_Uvc\_GetDigitalMultiplierCtrlInfo()***



### 3.4.110 AmbaUSBH\_Uvc\_SetDigitalMultiplierCtrlEx

#### API Syntax:

```
UINT32 AmbaUSBH_Uvc_SetDigitalMultiplierCtrlEx(UINT32 id, UINT32 ctrl)
```

#### Function Description:

The extension function of *AmbaUSBH\_Uvc\_SetDigitalMultiplierCtrl()*. The application calls this API to set the specified digital multiplier control value, which is in the range defined by *AmbaUSBH\_Uvc\_GetDigitalMultiplierCtrlInfoEx ()*, by the *id* parameter.

#### Parameters:

| Type   | Parameters | Description   |
|--------|------------|---|
| UINT32 | id         | [Input] The ID of the UVC device, which is assigned by the device insert/remove callback functions. See Section 3.4.1.14 for details. |
| UINT32 | ctrl       | [Input] digital multiplier control value.   |

Table 3-288 Parameters for AmbaUSB API *AmbaUSBH\_Uvc\_SetDigitalMultiplierCtrlEx ()*.

#### Return:

| Return Value | Description            |
|--------------|------------------------|
| 0            | UX_SUCCESS             |
| 0x5F         | UX_NO_DEVICE_CONNECTED |

Table 3-289 Returns for API *AmbaUSBH\_Uvc\_SetDigitalMultiplierCtrlEx ()*.

#### Example:

None

#### See Also :

*AmbaUSBH\_Uvc\_SetDigitalMultiplierCtrl()*



### 3.4.111 AmbaUSBH\_Uvc\_GetDigitalMultiplierLimitInfoEx

#### API Syntax:

```
UINT32 AmbaUSBH_Uvc_GetDigitalMultiplierLimitInfoEx (UINT32 id,  
                                UVCH_CTRL_INFO_s *info)
```

#### Function Description:

The extension function of **AmbaUSBH\_Uvc\_GetDigitalMultiplierLimitInfo()**. The application calls this API to get the specified digital multiplier limit control value by the **id** parameter.

#### Parameters:

| Type              | Parameters | Description   |
|-------------------|------------|---|
| UINT32            | id         | [Input] The ID of the UVC device, which is assigned by device insert/remove callback functions. See Section 3.4.1.14 for details. |
| UVCH_CTRL_INFO_s* | Info       | [Output] Pointer to control information. See Section 3.4.1.6 for details.   |

Table 3-290 Parameters for AmbaUSB API **AmbaUSBH\_Uvc\_GetDigitalMultiplierLimitInfoEx ()**.

#### Return:

| Return Value | Description                    |
|--------------|--------------------------------|
| 0            | UX_SUCCESS                     |
| 0x1C         | UX_PARAMETER_ERROR             |
| 0x5F         | UX_NO_DEVICE_CONNECTED         |
| 0x84         | UX_HOST_WRONG_CONTROL_SELECTOR |

Table 3-291 Returns for API **AmbaUSBH\_Uvc\_GetDigitalMultiplierLimitInfoEx ()**.



**Example:**

```
UVCH_CTRL_INFO_s info = {0};  
if (AmbaUSBH_Uvc_GetDigitalMultiplierLimitInfoEx (id, &info) == OK) {  
    AmbaPrint("digital multiplier limit %d ~ %d, cur %d", info.min, info.max,  
    info.cur);  
}
```

**See Also :**

***AmbaUSBH\_Uvc\_GetDigitalMultiplierLimitInfo( )***



### 3.4.112 AmbaUSBH\_Uvc\_SetDigitalMultiplierLimitEx

#### API Syntax:

```
UINT32 AmbaUSBH_Uvc_SetDigitalMultiplierLimitEx(UINT32 id, UINT32 ctrl)
```

#### Function Description:

The extension function of [\*\*AmbaUSBH\\_Uvc\\_SetDigitalMultiplierLimit\(\)\*\*](#). The application calls this API to set the specified digital multiplier limit control value, which is in the range defined by [\*\*AmbaUSBH\\_Uvc\\_GetDigitalMultiplierLimitInfoEx \(\)\*\*](#), by the **id** parameter.

#### Parameters:

| Type   | Parameters | Description   |
|--------|------------|---|
| UINT32 | id         | [Input] The ID of the UVC device, which is assigned by the device insert/remove callback functions. See Section 3.4.1.14 for details. |
| UINT32 | ctrl       | [Input] Digital multiplier limit control value.   |

Table 3-292 Parameters for AmbaUSB API **AmbaUSBH\_Uvc\_SetDigitalMultiplierLimitEx ()**.

#### Return:

| Return Value | Description            |
|--------------|------------------------|
| 0            | UX_SUCCESS             |
| 0x5F         | UX_NO_DEVICE_CONNECTED |

Table 3-293 Returns for API **AmbaUSBH\_Uvc\_SetDigitalMultiplierLimitEx ()**.

#### Example:

None

#### See Also :

[\*\*AmbaUSBH\\_Uvc\\_SetDigitalMultiplierLimit\( \)\*\*](#)



### 3.4.113 AmbaUSBH\_Uvc\_GetCurAnalogVideoStandardEx

#### API Syntax:

```
UINT32 AmbaUSBH_Uvc_GetCurAnalogVideoStandardEx(UINT32 id, UINT32 *ctrl)
```

#### Function Description:

The extension function of **AmbaUSBH\_Uvc\_GetCurAnalogVideoStandard()**. The application calls this API to get the specified analog video standard control value by the *id* parameter.

#### Parameters:

| Type    | Parameters | Description   |
|---------|------------|---|
| UINT32  | id         | [Input] The ID of the UVC device, which is assigned by the device insert/remove callback functions. See Section 3.4.1.14 for details. |
| UINT32* | ctrl       | [Output] Analog video standard control value. See Section 3.4.1.10 for details.   |

Table 3-294 Parameters for AmbaUSB API

**AmbaUSBH\_Uvc\_GetCurAnalogVideoStandardEx()**.

#### Return:

| Return Value | Description                    |
|--------------|--------------------------------|
| 0            | UX_SUCCESS                     |
| 0x1C         | UX_PARAMETER_ERROR             |
| 0x5F         | UX_NO_DEVICE_CONNECTED         |
| 0x84         | UX_HOST_WRONG_CONTROL_SELECTOR |

Table 3-295 Returns for API **AmbaUSBH\_Uvc\_GetCurAnalogVideoStandardEx ()**.

#### Example:

```
UINT32 info = 0;  
  
if (Ambarella_Uvc_GetCurAnalogVideoStandardEx (id, &info) == OK) {  
    AmbarellaPrint("video %d ~ %d, cur %d", info.min, info.max, info.cur);  
}
```



See Also :

*AmbaUSBH\_Uvc\_GetCurAnalogVideoStandard( )*

For PROTRULY Confidential Only



### 3.4.114 AmbaUSBH\_Uvc\_SetAnalogVideoStandardEx

#### API Syntax:

```
UINT32 AmbaUSBH_Uvc_SetAnalogVideoStandardEx(UINT32 id,  
                                              UVC_ANALOG_VIDEO_STANDARD_e ctrl)
```

#### Function Description:

The extension function of **AmbaUSBH\_Uvc\_SetAnalogVideoStandard()**. The application calls this API to set the specified analog video standard value by the *id* parameter.

#### Parameters:

| Type                        | Parameters | Description   |
|-----------------------------|------------|---|
| UINT32                      | id         | [Input] The ID of the UVC device, which is assigned by the device insert/remove callback functions. See Section 3.4.1.14 for details. |
| UVC_ANALOG_VIDEO_STANDARD_e | ctrl       | [Input] Analog video standard control value. See Section 3.4.1.10 for details.  |

Table 3-296 Parameters for AmbaUSB API **AmbaUSBH\_Uvc\_SetAnalogVideoStandardEx ()**.

#### Return:

| Return Value | Description            |
|--------------|------------------------|
| 0            | UX_SUCCESS             |
| 0x5F         | UX_NO_DEVICE_CONNECTED |

Table 3-297 Returns for API **AmbaUSBH\_Uvc\_SetAnalogVideoStandardEx ()**.

#### Example:

None

#### See Also :

**AmbaUSBH\_Uvc\_SetAnalogVideoStandard()**



### 3.4.115 AmbaUSBH\_Uvc\_GetCurAnalogLockStatusEx

#### API Syntax:

```
UINT32 AmbaUSBH_Uvc_GetCurAnalogLockStatusEx(UINT32 id, UINT32 *ctrl)
```

#### Function Description:

The extension function of **AmbaUSBH\_Uvc\_GetCurAnalogLockStatus()**. The application calls this API to get the specified current analog video lock status by the *id* parameter.

#### Parameters:

| Type    | Parameters | Description   |
|---------|------------|---|
| UINT32  | id         | [Input] The ID of the UVC device, which is assigned by the device insert/remove callback functions. See Section 3.4.1.14 for details. |
| UINT32* | ctrl       | [Output] Analog lock status. See Section 3.4.1.11 for details.  |

Table 3-298 Parameters for AmbaUSB API **AmbaUSBH\_Uvc\_GetCurAnalogLockStatusEx ()**.

#### Return:

| Return Value | Description                    |
|--------------|--------------------------------|
| 0            | UX_SUCCESS                     |
| 0x1C         | UX_PARAMETER_ERROR             |
| 0x5F         | UX_NO_DEVICE_CONNECTED         |
| 0x84         | UX_HOST_WRONG_CONTROL_SELECTOR |

Table 3-299 Returns for API **AmbaUSBH\_Uvc\_GetCurAnalogLockStatusEx ()**.



**Example:**

```
UINT32 info = 0;  
if (AmbaUSBH_Uvc_GetCurAnalogLockStatusEx (id, &info) == OK) {  
    AmbaPrint("lock status %d ~ %d, cur %d", info.min, info.max, info.cur);  
}
```

**See Also:**

*AmbaUSBH\_Uvc\_GetCurAnalogLockStatus()*



### 3.4.116 AmbaUSBH\_Uvc\_SetAnalogLockStatusEx

#### API Syntax:

```
UINT32 AmbaUSBH_Uvc_SetAnalogLockStatusEx(UINT32 id,  
                                         UVC_ANALOG_VIDEO_LOCK_e status)
```

#### Function Description:

The extension function of **AmbaUSBH\_Uvc\_SetAnalogLockStatus()**. The application calls this API to set the specified analog lock status by the *id* parameter.

#### Parameters:

| Type                        | Parameters | Description   |
|-----------------------------|------------|---|
| UINT32                      | id         | [Input] The ID of the UVC device, which is assigned by device insert/remove callback functions. See Section 3.4.1.14 for details. |
| UVC_ANALOG_VIDEO_STANDARD_e | ctrl       | [Input] Analog lock status. See Section 3.4.1.11 for details.   |

Table 3-300 Parameters for AmbaUSB API **AmbaUSBH\_Uvc\_SetAnalogLockStatusEx ()**.

#### Return:

| Return Value | Description            |
|--------------|------------------------|
| 0            | UX_SUCCESS             |
| 0x5F         | UX_NO_DEVICE_CONNECTED |

Table 3-301 Returns for API **AmbaUSBH\_Uvc\_SetAnalogLockStatusEx()**.

#### Example:

None

#### See Also :

**AmbaUSBH\_Uvc\_SetAnalogLockStatus( )**



### 3.4.117 AmbaUSBH\_Uvc\_H264EnableEx

#### API Syntax:

```
UINT32 AmbaUSBH_Uvc_H264EnableEx(UINT32 id, UINT32 enable)
```

#### Function Description:

The extension function of **AmbaUSBH\_Uvc\_H264EnableEx()**. The application calls this API to enable the specified H.264 stream by the *id* parameter.

#### Parameters:

| Type   | Parameters | Description   |
|--------|------------|---|
| UINT32 | id         | [Input] The id of UVC device, which is assigned by device insert/remove callback functions. See section 3.4.1.14 for details. |
| UINT32 | Enable     | 0: Disable mux h.264 data stream.<br>1: Enable mux h.264 data stream  |

Table 3-302 Parameters for AmbaUSB API **AmbaUSBH\_Uvc\_H264EnableEx ()**.

#### Return:

| Return Value | Description |
|--------------|-------------|
| 1            | TRUE        |
| 0            | FALSE       |

Table 3-303 Returns for API **AmbaUSBH\_Uvc\_H264EnableEx ()**.

#### Example:

None

#### See Also :

[AmbaUSBH\\_Uvc\\_H264Enable\( \)](#)



### 3.4.118 AmbaUSBH\_Uvc\_IsH264SupportedEx

#### API Syntax:

```
UINT32 AmbaUSBH_Uvc_IsH264SupportedEx(UINT32 id)
```

#### Function Description:

The extension function of *AmbaUSBH\_Uvc\_IsH264Supported()*. The application calls this API to check if the specified UVC device supports H.264 stream by the *id* parameter.

#### Parameters:

| Type   | Parameters | Description   |
|--------|------------|---|
| UINT32 | <i>id</i>  | [Input] The ID of the UVC device, which is assigned by the device insert/remove callback functions. See Section 3.4.1.14 for details. |

Table 3-304 Parameters for AmbaUSB API *AmbaUSBH\_Uvc\_IsH264SupportedEx ()*.

#### Return:

| Return Value | Description |
|--------------|-------------|
| 1            | TRUE        |
| 0            | FALSE       |

Table 3-305 Returns for API *AmbaUSBH\_Uvc\_IsH264SupportedEx ()*.

#### Example:

None

#### See Also :

*AmbaUSBH\_Uvc\_IsH264Supported( )*



### 3.4.119 AmbaUSBH\_Uvc\_H264GetVideoConfigInfoEx

#### API Syntax:

```
UINT32 AmbaUSBH_Uvc_H264GetVideoConfigInfoEx(UINT32 id,  
                                              UVCX_VIDEO_CONFIG_S *ConfigMin,  
                                              UVCX_VIDEO_CONFIG_S *ConfigMax,  
                                              UVCX_VIDEO_CONFIG_S *ConfigCur)
```

#### Function Description:

The extension function of *AmbaUSBH\_Uvc\_H264GetVideoConfigInfo()*. The application calls this API to get the specified video configuration of the H.264 extension unit by the *id* parameter.

#### Parameters:

| Type                 | Parameters | Description   |
|----------------------|------------|---|
| UINT32               | id         | [Input] The ID of the UVC device, which is assigned by the device insert/remove callback functions. See Section 3.4.1.14 for details. |
| UVCX_VIDEO_CONFIG_S* | ConfigMin  | [Output] Minimum video configuration of H264 format. See Section 3.4.1.12 for details.  |
| UVCX_VIDEO_CONFIG_S* | ConfigMax  | [Output] Maximum video configuration of H264 format. See Section 3.4.1.12 for details.  |
| UVCX_VIDEO_CONFIG_S* | ConfigCur  | [Output] Current video configuration of H264 format. See Section 3.4.1.12 for details.  |

Table 3-306 Parameters for AmbaUSB API *AmbaUSBH\_Uvc\_H264GetVideoConfigInfoEx ()*.



**Return:**

| Return Value | Description                    |
|--------------|--------------------------------|
| 0            | UX_SUCCESS                     |
| 0x1C         | UX_PARAMETER_ERROR             |
| 0x5F         | UX_NO_DEVICE_CONNECTED         |
| 0x84         | UX_HOST_WRONG_CONTROL_SELECTOR |

Table 3-307 Returns for API **AmbaUSBH\_Uvc\_H264GetVideoConfigInfoEx()**.

**Example:**

```
UVCX_VIDEO_CONFIG_S VideoConfig = {0};  
UVCX_VIDEO_CONFIG_S VideoConfigMin = {0};  
UVCX_VIDEO_CONFIG_S VideoConfigMax = {0};  
AmbaUSBH_Uvc_H264GetVideoConfigInfoEx(1, &VideoConfigMin, &VideoConfigMax,  
&VideoConfig);
```

**See Also :**

**AmbaUSBH\_Uvc\_H264GetVideoConfigInfo()**



### 3.4.120 AmbaUSBH\_Uvc\_H264SetVideoConfigEx

#### API Syntax:

```
UINT32 AmbaUSBH_Uvc_H264SetVideoConfigEx(UINT32 id,  
                                         UVCX_VIDEO_CONFIG_S *config)
```

#### Function Description:

The extension function of **AmbaUSBH\_Uvc\_H264SetVideoConfig()**. The application calls this API to set the specified video configuration of H.264 extension unit by the **id** parameter. The configuration values should be in the range defined by

**AmbaUSBH\_Uvc\_H264GetVideoConfigInfoEx**. The configuration values do not take effect immediately until the application issues **AmbaUSBH\_Uvc\_ProbeAndCommitEx()**. Note that “wWidth”, “wHeight”, “dwFrameInterval” and “bStreamMuxOption” are set by the USB stack and the application does not need to configure them.

#### Parameters:

| Type                 | Parameters | Description   |
|----------------------|------------|---|
| UINT32               | id         | [Input] The ID of the UVC device, which is assigned by the device insert/remove callback functions. See Section 3.4.1.14 for details. |
| UVCX_VIDEO_CONFIG_S* | Config     | [Input] Video configuration of H264 format. See Section 3.4.1.12 for details.   |

Table 3-308 Parameters for AmbaUSB API **AmbaUSBH\_Uvc\_H264SetVideoConfigEx ()**.

#### Return:

| Return Value | Description            |
|--------------|------------------------|
| 0            | UX_SUCCESS             |
| 0x1C         | UX_PARAMETER_ERROR     |
| 0x5F         | UX_NO_DEVICE_CONNECTED |

Table 3-309 Returns for API **AmbaUSBH\_Uvc\_H264SetVideoConfigEx ()**.



**Example:**

None

**See Also :**

*AmbaUSBH\_Uvc\_H264SetVideoConfig( )*

For PROTRULY Confidential Only



### 3.4.121 AmbaUSBH\_Uvc\_H264RequestIdrEx

#### API Syntax:

```
UINT32 AmbaUSBH_Uvc_H264RequestIdrEx(UINT32 id)
```

#### Function Description:

The extension function of *AmbaUSBH\_Uvc\_H264RequestIdr()*. The application calls this API for requesting the next frame of the specified UVC device as an IDR frame with new SPS and PPS by the *id* parameter.

#### Parameters:

| Type   | Parameters | Description   |
|--------|------------|---|
| UINT32 | <i>id</i>  | [Input] The ID of the UVC device, which is assigned by the device insert/remove callback functions. See Section 3.4.1.14 for details. |

Table 3-310 Parameters for AmbaUSB API *AmbaUSBH\_Uvc\_H264RequestIdrEx ()*.

#### Return:

| Return Value | Description                        |
|--------------|------------------------------------|
| 0            | UX_SUCCESS                         |
| 0x12         | UX_MEMORY_INSUFFICIENT             |
| 0x15         | UX_SEMAPHORE_ERROR                 |
| 0x1C         | UX_PARAMETER_ERROR                 |
| 0x5B         | UX_HOST_CLASS_INSTANCE_UNKNOWN     |
| 0x5F         | UX_NO_DEVICE_CONNECTED             |
| 0x84         | UX_HOST_CLASS_VIDEO_WRONG_SELECTOR |

Table 3-311 Returns for API *AmbaUSBH\_Uvc\_H264RequestIdrEx ()*.

#### Example:

None



See Also :

*AmbaUSBH\_Uvc\_H264Requestldr( )*

For PROTRULY Confidential Only



### 3.4.122 AmbaUSBH\_Uvc\_H264GetRateCtrlModeInfoEx

#### API Syntax:

```
UINT32 AmbaUSBH_Uvc_H264GetRateCtrlModeInfoEx (UINT32 id,  
                                              UVCH_CTRL_INFO_s *info)
```

#### Function Description:

The extension function of **AmbaUSBH\_Uvc\_H264GetRateCtrlModeInfo()**. The application calls this API to get the specified rate control mode information by the *id* parameter.

#### Parameters:

| Type              | Parameters | Description   |
|-------------------|------------|---|
| UINT32            | id         | [Input] The ID of the UVC device, which is assigned by the device insert/remove callback functions. See Section 3.4.1.14 for details.   |
| UVCH_CTRL_INFO_s* | Info       | [Output] Pointer to control information. See Section 3.4.1.6 for details.<br><br>Bits 0-3 Modes<br>0x00: Reserved<br>0x01: CBR<br>0x02: VBR<br>0x03: Constant QP<br><br>Bits 4-7 Flags<br>0x10: Fixed_frame_rate_flag |

Table 3-312 Parameters for AmbaUSB API **AmbaUSBH\_Uvc\_H264GetRateCtrlModeInfoEx ()**.



**Return:**

| Return Value | Description                        |
|--------------|------------------------------------|
| 0            | UX_SUCCESS                         |
| 0x12         | UX_MEMORY_INSUFFICIENT             |
| 0x15         | UX_SEMAPHORE_ERROR                 |
| 0x1C         | UX_PARAMETER_ERROR                 |
| 0x5B         | UX_HOST_CLASS_INSTANCE_UNKNOWN     |
| 0x5F         | UX_NO_DEVICE_CONNECTED             |
| 0x84         | UX_HOST_CLASS_VIDEO_WRONG_SELECTOR |

Table 3-313 Returns for API **AmbaUSBH\_Uvc\_H264GetRateCtrlModeInfoEx()**.

**Example:**

```
UVCH_CTRL_INFO_s info = {0};  
if (AmbaUSBH_Uvc_H264GetRateCtrlModeInfoEx (id, &info) == OK) {  
    AmbaPrint ("Rate Ctrl Mode %d ~ %d, cur %d", info.min, info.max, info.cur);  
}
```

**See Also:**

[\*\*AmbaUSBH\\_Uvc\\_H264GetRateCtrlModeInfo\(\)\*\*](#)



### 3.4.123 AmbaUSBH\_Uvc\_H264SetRateCtrlModeEx

#### API Syntax:

```
UINT32 AmbaUSBH_Uvc_H264SetRateCtrlModeEx (UINT32 id, UINT32 mode)
```

#### Function Description:

The extension function of **AmbaUSBH\_Uvc\_H264SetRateCtrlMode()**. The application calls this API to set the specified rate control mode, which is in the range defined by **AmbaUSBH\_Uvc\_H264GetRateCtrlModeInfoEx()**, by the *id* parameter.

#### Parameters:

| Type   | Parameters | Description  |
|--------|------------|--|
| UINT32 | id         | [Input] The ID of the UVC device, which is assigned by the device insert/remove callback functions. See Section 3.4.1.14 for details.                            |
| UINT32 | mode       | [Input] Rate control mode.<br>Bits 0-3 Modes:<br>0x00: Reserved<br>0x01: CBR<br>0x02: VBR<br>0x03: Constant QP<br>Bits 4-7 Flags:<br>0x10: fixed_frame_rate_flag |

Table 3-314 Parameters for AmbaUSB API **AmbaUSBH\_Uvc\_H264SetRateCtrlModeEx()**.

#### Return:

| Return Value | Description                    |
|--------------|--------------------------------|
| 0            | UX_SUCCESS                     |
| 0x12         | UX_MEMORY_INSUFFICIENT         |
| 0x15         | UX_SEMAPHORE_ERROR             |
| 0x1C         | UX_PARAMETER_ERROR             |
| 0x5B         | UX_HOST_CLASS_INSTANCE_UNKNOWN |



| Return Value | Description                        |
|--------------|------------------------------------|
| 0x5F         | UX_NO_DEVICE_CONNECTED             |
| 0x84         | UX_HOST_CLASS_VIDEO_WRONG_SELECTOR |
| 0x85         | UX_HOST_CLASS_VIDEO_SET_VALUE_FAIL |

Table 3-315 Returns for API *AmbaUSBH\_Uvc\_H264SetRateCtrlModeEx()*.

**Example:**

None

**See Also :**

*AmbaUSBH\_Uvc\_H264SetRateCtrlMode( )*



### 3.4.124 AmbaUSBH\_Uvc\_H264GetTemporalScaleModelInfoEx

#### API Syntax:

```
UINT32 AmbaUSBH_Uvc_H264GetTemporalScaleModelInfoEx(UINT32 id,  
                                                    UVCH_CTRL_INFO_s *info)
```

#### Function Description:

The extension function of **AmbaUSBH\_Uvc\_H264GetTemporalScaleModelInfo()**. The application calls this API to get the specified temporal scale control mode information by the *id* parameter.

#### Parameters:

| Type              | Parameters | Description   |
|-------------------|------------|---|
| UINT32            | id         | [Input] The ID of the UVC device, which is assigned by the device insert/remove callback functions. See Section 3.4.1.14 for details.                                 |
| UVCH_CTRL_INFO_s* | Info       | [Output] Pointer to control information. See Section 3.4.1.6 for details.<br>0x00: No Temporal Enhancement Layer<br>0x01- 0x07: Number of Temporal Enhancement Layers |

Table 3-316 Parameters for API **AmbaUSBH\_Uvc\_H264GetTemporalScaleModelInfoEx ()**.



**Return:**

| Return Value | Description                        |
|--------------|------------------------------------|
| 0            | UX_SUCCESS                         |
| 0x12         | UX_MEMORY_INSUFFICIENT             |
| 0x15         | UX_SEMAPHORE_ERROR                 |
| 0x1C         | UX_PARAMETER_ERROR                 |
| 0x5B         | UX_HOST_CLASS_INSTANCE_UNKNOWN     |
| 0x5F         | UX_NO_DEVICE_CONNECTED             |
| 0x84         | UX_HOST_CLASS_VIDEO_WRONG_SELECTOR |

Table 3-317 Returns for API **AmbaUSBH\_Uvc\_H264GetTemporalScaleModeInfoEx()**.

**Example:**

```
UVCH_CTRL_INFO_s info = {0};  
if (AmbaUSBH_Uvc_H264GetTemporalScaleModeInfoEx (id, &info) == OK) {  
    AmbaPrint("Temporal scale mode %d ~ %d, cur %d", info.min, info.max,  
    info.cur);  
}
```

**See Also:**

[AmbaUSBH\\_Uvc\\_H264GetTemporalScaleModeInfo\(\)](#)



### 3.4.125 AmbaUSBH\_Uvc\_H264SetTemporalScaleModeEx

#### API Syntax:

```
UINT32 AmbaUSBH_Uvc_H264SetTemporalScaleModeEx(UINT32 id, UINT32 mode)
```

#### Function Description:

The extension function of *AmbaUSBH\_Uvc\_H264SetTemporalScaleMode()*. The application calls this API to set the specified temporal scale mode, which is in the range defined by *AmbaUSBH\_Uvc\_H264GetTemporalScaleModeInfoEx ()*, by the *id* parameter.

#### Parameters:

| Type   | Parameters | Description   |
|--------|------------|---|
| UINT32 | id         | [Input] The ID of the UVC device, which is assigned by device insert/remove callback functions. See Section 3.4.1.14 for details. |
| UINT32 | mode       | [Input] Temporal scale mode.<br>0x00: No Temporal Enhancement Layer<br>0x01- 0x07: Number of Temporal Enhancement Layers          |

Table 3-318 Parameters for API *AmbaUSBH\_Uvc\_H264SetTemporalScaleModeEx ()*.

#### Return:

| Return Value | Description                        |
|--------------|------------------------------------|
| 0            | UX_SUCCESS                         |
| 0x12         | UX_MEMORY_INSUFFICIENT             |
| 0x15         | UX_SEMAPHORE_ERROR                 |
| 0x1C         | UX_PARAMETER_ERROR                 |
| 0x5B         | UX_HOST_CLASS_INSTANCE_UNKNOWN     |
| 0x5F         | UX_NO_DEVICE_CONNECTED             |
| 0x84         | UX_HOST_CLASS_VIDEO_WRONG_SELECTOR |
| 0x85         | UX_HOST_CLASS_VIDEO_SET_VALUE_FAIL |

Table 3-319 Returns for API *AmbaUSBH\_Uvc\_H264SetTemporalScaleModeEx ()*.



**Example:**

None

**See Also :**

*AmbaUSBH\_Uvc\_H264SetRateCtrlMode( )*

For PROTRULY Confidential Only



### 3.4.126 AmbaUSBH\_Uvc\_H264GetSpatialScaleModeInfoEx

#### API Syntax:

```
UINT32 AmbaUSBH_Uvc_H264GetSpatialScaleModeInfoEx(UINT32 id,  
                                                 UVCH_CTRL_INFO_s *info)
```

#### Function Description:

The extension function of *AmbaUSBH\_Uvc\_H264GetSpatialScaleModeInfo()*. The application calls this API to get the specified spatial scale control mode information by the *id* parameter.

#### Parameters:

| Type              | Parameters | Description  |
|-------------------|------------|--|
| UINT32            | id         | [Input] The ID of the UVC device, which is assigned by the device insert/remove callback functions. See Section 3.4.1.14 for details.                              |
| UVCH_CTRL_INFO_s* | Info       | [Output] Pointer to control information. See Section 3.4.1.6 for details.<br>0x00: No Spatial Enhancement Layer<br>0x01-0x08: Number of Spatial Enhancement Layers |

Table 3-320 Parameters for API *AmbaUSBH\_Uvc\_H264GetSpatialScaleModeInfoEx ()*.



**Return:**

| Return Value | Description                        |
|--------------|------------------------------------|
| 0            | UX_SUCCESS                         |
| 0x12         | UX_MEMORY_INSUFFICIENT             |
| 0x15         | UX_SEMAPHORE_ERROR                 |
| 0x1C         | UX_PARAMETER_ERROR                 |
| 0x5B         | UX_HOST_CLASS_INSTANCE_UNKNOWN     |
| 0x5F         | UX_NO_DEVICE_CONNECTED             |
| 0x84         | UX_HOST_CLASS_VIDEO_WRONG_SELECTOR |

Table 3-321 Returns for API **AmbaUSBH\_Uvc\_H264GetSpatialScaleModeInfoEx()**.

**Example:**

```
UVCH_CTRL_INFO_s info = {0};  
if (AmbaUSBH_Uvc_H264GetSpatialScaleModeInfoEx (id, &info) == OK) {  
    AmbaPrint("Spatial scale mode %d ~ %d, cur %d", info.min, info.max,  
    info.cur);  
}
```

**See Also:**

***AmbaUSBH\_Uvc\_H264GetSpatialScaleModeInfo()***



### 3.4.127 AmbaUSBH\_Uvc\_H264SetSpatialScaleModeEx

#### API Syntax:

```
UINT32 AmbaUSBH_Uvc_H264SetSpatialScaleModeEx (UINT32 id, UINT32 mode)
```

#### Function Description:

The extension function of ***AmbaUSBH\_Uvc\_H264SetSpatialScaleMode()***. The application calls this API to set the specified spatial scale mode, which is in the range defined by ***AmbaUSBH\_Uvc\_H264GetSpatialScaleModeInfoEx ()***, by the *id* parameter.

#### Parameters:

| Type   | Parameters | Description   |
|--------|------------|---|
| UINT32 | id         | [Input] The ID of the UVC device, which is assigned by the device insert/remove callback functions. See Section 3.4.1.14 for details. |
| UINT32 | mode       | [Input] Spatial scale mode.<br>0x00: No Spatial Enhancement Layer<br>0x01-0x08: Number of Spatial Enhancement Layers                  |

Table 3-322 Parameters for API ***AmbaUSBH\_Uvc\_H264SetSpatialScaleModeEx ()***.

#### Return:

| Return Value | Description                        |
|--------------|------------------------------------|
| 0            | UX_SUCCESS                         |
| 0x12         | UX_MEMORY_INSUFFICIENT             |
| 0x15         | UX_SEMAPHORE_ERROR                 |
| 0x1C         | UX_PARAMETER_ERROR                 |
| 0x5B         | UX_HOST_CLASS_INSTANCE_UNKNOWN     |
| 0x5F         | UX_NO_DEVICE_CONNECTED             |
| 0x84         | UX_HOST_CLASS_VIDEO_WRONG_SELECTOR |
| 0x85         | UX_HOST_CLASS_VIDEO_SET_VALUE_FAIL |

Table 3-323 Returns for API ***AmbaUSBH\_Uvc\_H264SetSpatialScaleModeEx ()***.



**Example:**

None

**See Also :**

*AmbaUSBH\_Uvc\_H264SetSpatialScaleMode( )*

For PROTRULY Confidential Only



### 3.4.128 AmbaUSBH\_Uvc\_H264GetFrameRateConfigInfoEx

#### API Syntax:

```
UINT32 AmbaUSBH_Uvc_H264GetFrameRateConfigInfoEx (UINT32 id,  
                                                UVCH_CTRL_INFO_s *info)
```

#### Function Description:

The extension function of **AmbaUSBH\_Uvc\_H264GetFrameRateConfigInfo()**. The application calls this API to get the specified frame rate configuration information by the *id* parameter.

#### Parameters:

| Type              | Parameters | Description   |
|-------------------|------------|---|
| UINT32            | id         | [Input] The ID of the UVC device, which is assigned by the device insert/remove callback functions. See Section 3.4.1.14 for details. |
| UVCH_CTRL_INFO_s* | Info       | [Output] Pointer to control information. See Section 3.4.1.6 for details.<br>In 100 ns frame intervals.                               |

Table 3-324 Parameters for API **AmbaUSBH\_Uvc\_H264GetFrameRateConfigInfoEx ()**.

#### Return:

| Return Value | Description                        |
|--------------|------------------------------------|
| 0            | UX_SUCCESS                         |
| 0x12         | UX_MEMORY_INSUFFICIENT             |
| 0x15         | UX_SEMAPHORE_ERROR                 |
| 0x1C         | UX_PARAMETER_ERROR                 |
| 0x5B         | UX_HOST_CLASS_INSTANCE_UNKNOWN     |
| 0x5F         | UX_NO_DEVICE_CONNECTED             |
| 0x84         | UX_HOST_CLASS_VIDEO_WRONG_SELECTOR |

Table 3-325 Returns for API **AmbaUSBH\_Uvc\_H264GetFrameRateConfigInfoEx ()**.



**Example:**

```
UVCH_CTRL_INFO_s info = {0};  
if (AmbaUSBH_Uvc_H264GetFrameRateConfigInfoEx (id, &info) == OK) {  
    AmbaPrint("frame rate config %d ~ %d, cur %d", info.min, info.max,  
    info.cur);  
}
```

**See Also:**

***AmbaUSBH\_Uvc\_H264GetFrameRateConfigInfo( )***

For PROTRULY Confidential Only



### 3.4.129 AmbaUSBH\_Uvc\_H264SetFrameRateConfigEx

#### API Syntax:

```
UINT32 AmbaUSBH_Uvc_H264SetFrameRateConfigEx (UINT32 id, UINT32 FrameRate)
```

#### Function Description:

The extension function of **AmbaUSBH\_Uvc\_H264SetFrameRateConfig()**. The application calls this API to set the specified frame rate configuration, which is in the range defined by **AmbaUSBH\_Uvc\_H264GetFrameRateConfigInfoEx ()**, by the *id* parameter.

#### Parameters:

| Type   | Parameters | Description   |
|--------|------------|---|
| UINT32 | id         | [Input] The ID of the UVC device, which is assigned by device insert/remove callback functions. See Section 3.4.1.14 for details. |
| UINT32 | FrameRate  | [Input] Frame rate configuration.<br>In 100 ns frame intervals  |

Table 3-326 Parameters for API **AmbaUSBH\_Uvc\_H264SetFrameRateConfigEx ()**.

#### Return:

| Return Value | Description                        |
|--------------|------------------------------------|
| 0            | UX_SUCCESS                         |
| 0x12         | UX_MEMORY_INSUFFICIENT             |
| 0x15         | UX_SEMAPHORE_ERROR                 |
| 0x1C         | UX_PARAMETER_ERROR                 |
| 0x5B         | UX_HOST_CLASS_INSTANCE_UNKNOWN     |
| 0x5F         | UX_NO_DEVICE_CONNECTED             |
| 0x84         | UX_HOST_CLASS_VIDEO_WRONG_SELECTOR |
| 0x85         | UX_HOST_CLASS_VIDEO_SET_VALUE_FAIL |

Table 3-327 Returns for API **AmbaUSBH\_Uvc\_H264SetFrameRateConfigEx ()**.

#### Example:



None

**See Also :**

*AmbaUSBH\_Uvc\_H264SetFrameRateConfig( )*

For PROTRULY Confidential Only



### 3.5 Simple Class APIs

This section describes the APIs for Simple Class.

For PROTRULY Confidential Only



### 3.5.1      Callback functions

#### 3.5.1.1    **MediaInsert**

##### **API Syntax:**

```
void (*MediaInsert) (void);
```

##### **Function Description:**

This function is called when a media insert event occurs.

##### **Parameters:**

None

##### **Return:**

None

##### **Example:**

None

##### **See Also:**

None



### 3.5.1.2 MediaRemove

#### API Syntax:

```
void (*MediaRemove) (void);
```

#### Function Description:

This function is called when a media remove event occurs.

#### Parameters:

None

#### Return:

None

#### Example:

None

#### See Also:

None



### 3.5.2 AmbaUSBH\_Simple\_RegisterCallback

#### API Syntax:

```
UINT32 AmbaUSBH_Simple_RegisterCallback (USB_HOST_SIMPLE_CB_s *cb)
```

#### Function Description:

The application calls this API to register callback functions of device insert/remove events.

#### Parameters:

| Type                  | Parameters | Description                       |
|-----------------------|------------|-----------------------------------|
| USB_HOST_SIMPLE_CB_s* | cb         | See Sections 3.5.1.1 and 3.5.1.2. |

Table 3-328 Parameters for AmbaUSB API **AmbaUSBH\_Simple\_RegisterCallback()**.

#### Return:

| Return Value | Description        |
|--------------|--------------------|
| 0            | UX_SUCCESS         |
| 0x1C         | UX_PARAMETER_ERROR |

Table 3-329 Returns for API **AmbaUSBH\_Simple\_RegisterCallback()**.

#### Example:

None

#### See Also:



### 3.5.3 AmbaUSBH\_Simple\_SetPidVid

#### API Syntax:

```
void AmbaUSBH_Simple_SetPidVid (UINT32 pid, UINT32 vid)
```

#### Function Description:

The application calls this API to set the product ID and vendor ID of specific Simple Class devices.

#### Parameters:

| Type   | Parameters | Description |
|--------|------------|-------------|
| UINT32 | pid        | Product ID  |
| UINT32 | vid        | Vendor ID   |

Table 3-330 Parameters for AmbaUSB API **AmbaUSBH\_Simple\_SetPidVid()**.

#### Return:

None

#### Example:

None

#### See Also :



### 3.5.4 AmbaUSBH\_Simple\_GetBulkInEndpointList

#### API Syntax:

```
UINT32 AmbaUSBH_Simple_GetBulkInEndpointList (UINT8 *buf, UINT32 *len)
```

#### Function Description:

The application calls **AmbaUSBH\_Simple\_GetBulkInEndpointList** or **AmbaUSBH\_Simple\_GetBulkInEndpoint** to get the bulk-in endpoint list of the Simple Class device.

#### Parameters:

| Type    | Parameters | Description                               |
|---------|------------|---|
| UINT8*  | buf        | The pointer of buffer store endpoint list |
| UINT32* | len        | The length of endpoint list               |

Table 3-331 Parameters for AmbaUSB API **AmbaUSBH\_Simple\_GetBulkInEndpointList()**.

#### Return:

| Return Value | Description                    |
|--------------|--------------------------------|
| 0            | UX_SUCCESS                     |
| 0x5B         | UX_HOST_CLASS_INSTANCE_UNKNOWN |
| 0x53         | UX_ENDPOINT_HANDLE_UNKNOWN     |
| 0x52         | UX_INTERFACE_HANDLE_UNKNOWN    |
| 0xFF         | No bulk-in endpoint found      |

Table 3-332 Returns for API **AmbaUSBH\_Simple\_GetBulkInEndpointList()**.

#### Example:

```
UINT8 UsbBuf[8] = {0};  
UINT32 ActualLength = 0;  
UINT32 unRet = 0;  
int count = 0;  
unRet = AmbaUSBH_Simple_GetBulkInEndpointList(UsbBuf, &ActualLength);  
if (unRet != 0) {
```



```
AmbaPrint("[Error] %s(): can not get bulk-in endpoint, 0x%x",
__func__, unRet);

    return RetVal;
}

for (count = 0; count < ActualLength; count++) {
    AmbaPrint("Bulk In Endpoint [%d] = 0x%x", count, UsbBuf[count]);
}
```

For PROTRULY Confidential Only



### 3.5.5 AmbaUSBH\_Simple\_GetBulkOutEndpointList

#### API Syntax:

```
UINT32 AmbaUSBH_Simple_GetBulkOutEndpointList (UINT8 *buf, UINT32 *len)
```

#### Function Description:

The application calls **AmbaUSBH\_Simple\_GetBulkOutEndpointList** or **AmbaUSBH\_Simple\_GetBulkOutEndpoint** to get the bulk-out endpoint list of the Simple Class device.

#### Parameters:

| Type    | Parameters | Description                               |
|---------|------------|---|
| UINT8*  | buf        | The pointer of buffer store endpoint list |
| UINT32* | len        | The length of endpoint list               |

Table 3-333 Parameters for AmbaUSB API **AmbaUSBH\_Simple\_GetBulkOutEndpointList()**.

#### Return:

| Return Value | Description                    |
|--------------|--------------------------------|
| 0            | UX_SUCCESS                     |
| 0x5B         | UX_HOST_CLASS_INSTANCE_UNKNOWN |
| 0x53         | UX_ENDPOINT_HANDLE_UNKNOWN     |
| 0x52         | UX_INTERFACE_HANDLE_UNKNOWN    |
| 0xFF         | No bulk-in endpoint found      |

Table 3-334 Returns for API **AmbaUSBH\_Simple\_GetBulkOutEndpointList()**.

#### Example:

```
UINT8 UsbBuf[8] = {0};  
UINT32 ActualLength = 0;  
UINT32 unRet = 0;  
int count = 0;  
unRet = AmbaUSBH_Simple_GetBulkOutEndpointList(UsbBuf, &ActualLength);  
if (unRet != 0) {
```



```
    AmbaPrint("[Error] %s(): can not get bulk-out endpoint, 0x%x",  
    __func__, unRet);  
  
    return RetVal;  
}  
  
for (count = 0; count < ActualLength; count++) {  
    AmbaPrint("Bulk Out Endpoint [%d] = 0x%x", count, UsbBuf[count]);  
}
```

For PROTRULY Confidential Only



### 3.5.6 AmbaUSBH\_Simple\_ControlRequest

#### API Syntax:

```
UINT32 AmbaUSBH_Simple_ControlRequest (UINT32 request,  
                                         UINT32 type,  
                                         UINT32 value,  
                                         UINT32 index,  
                                         UINT8 *buf,  
                                         UINT32 length)
```

#### Function Description:

The application calls this API to send vendor request to Simple Class device via control pipe.

#### Parameters:

| Type   | Parameters | Description                                       |
|--------|------------|---|
| UINT32 | request    | bRequest. See USB 2.0 spec for more details.      |
| UINT32 | type       | bmRequestType. See USB 2.0 spec for more details. |
| UINT32 | value      | wValue. See USB 2.0 spec for more details.        |
| UINT32 | index      | wIndex. See USB 2.0 spec for more details.        |
| UINT8* | buf        | Pointer to buffer if length field is not zero.    |
| UINT32 | length     | wLength. See USB 2.0 spec for more details.       |

Table 3-335 Parameters for AmbaUSB API **AmbaUSBH\_Simple\_ControlRequest()**.

#### Return:

| Return Value | Description                     |
|--------------|---------------------------------|
| 0            | UX_SUCCESS                      |
| 0x5B         | UX_HOST_CLASS_INSTANCE_UNKNOWN  |
| 0x51         | UX_CONFIGURATION_HANDLE_UNKNOWN |
| 0x25         | UX_TRANSFER_NOT_READY           |
| 0x0D         | Semaphore timeout               |

Table 3-336 Returns for API **AmbaUSBH\_Simple\_ControlRequest()**.



**Example:**

```
UINT8 UsbBuf[64] = {0};

UINT32 count = 0;
UINT32 unRet = 0;

UINT32 request = 0x59; // request supported by device
UINT32 type = 0xC0; // bit 7 =1 (device to host), bit[6:5] = 2 (vendor)
UINT32 value = 0;
UINT32 index = 0; // interface 0
UINT32 len = 20;

AmбаPrint("request = 0x%x, type = 0x%x, value = 0x%x, index = 0x%x, len
= %d", request,
          type, value, index, len);

unRet = AmбаUSBH_Simple_ControlRequest(request,
                                         type,
                                         value,
                                         index,
                                         UsbBuf,
                                         len);

if (unRet != OK) {
    AmбаPrint("Simple Control Request Result: 0x%X", unRet);
} else {
    AmбаPrint("Simple Control Request Result: 0x%X", unRet);
    for (count = 0; count < len; count++) {
        AmбаPrint(" [%d] 0x%x", count, UsbBuf[count]);
    }
}
```



### 3.5.7 AmbaUSBH\_Simple\_BulkRead

#### API Syntax:

```
UINT32 AmbaUSBH_Simple_BulkRead (UINT8 *DataPtr,  
                                  UINT32 len,  
                                  UINT32 *ActualLen,  
                                  UINT32 EpAddr,  
                                  UINT32 timeout)
```

#### Function Description:

The application calls this API to get data from the Simple Class device via Bulk-In pipe. Note that there is a transfer block of 16K bytes in bulk-in pipe. The parameter “len” field decides how many transfer block USB host will use for receiving data from the USB device. The USB host will wait till each transfer block is filled with data (less than or equal to 16K bytes) or until there is a transfer timeout. For example, if the application issues a BulkRead with “len” being 32K bytes but gets only 8 bytes; the USB host will keep waiting until there is another Bulk-In data transfer or until the current transfer is expired. If there’s another Bulk-In data received with 12 bytes during this period, the returning “ActualLen” would be 20 bytes but the data will be located in DataPtr (8 bytes) and DataPtr+16K (12 bytes) separately. To avoid this confusion, Ambarella recommends that the application should develop its protocol to get the intended data length before issuing BulkRead.

The simple class device must send an additional zero-length packet if the transfer length is a multiple of 512 bytes.

#### Parameters:

| Type    | Parameters | Description                             |
|---------|------------|---|
| UINT8*  | DataPtr    | Pointer to data buffer.                 |
| UINT32  | len        | The request length of USB host.         |
| UINT32* | *ActualLen | The actual data length. Must be <= len. |
| UINT32  | EpAddr     | Endpoint address.                       |



| Type   | Parameters | Description   |
|--------|------------|---|
| UINT32 | timeout    | Timeout value in ms. 0 means no waiting.<br>0xFFFFFFFF means waiting forever. |

Table 3-337 Parameters for AmbaUSB API **AmbaUSBH\_Simple\_BulkRead()**.

**Return:**

| Return Value | Description                     |
|--------------|---------------------------------|
| 0            | UX_SUCCESS                      |
| 0x5B         | UX_HOST_CLASS_INSTANCE_UNKNOWN  |
| 0x51         | UX_CONFIGURATION_HANDLE_UNKNOWN |
| 0x25         | UX_TRANSFER_NOT_READY           |
| 0x5C         | UX_TRANSFER_TIMEOUT             |
| 0x53         | UX_ENDPOINT_HANDLE_UNKNOWN      |

Table 3-338 Returns for API **AmbaUSBH\_Simple\_BulkRead()**.

**Example:**

```
UINT8 UsbBuf[1024*16] = {0};  
UINT32 ActualLength = 0;  
UINT32 unRet = 0;  
int ep_address = 0x82;  
  
unRet = AmbaUSBH_Simple_BulkRead(UsbBuf,  
                                1024*16,  
                                &ActualLength,  
                                ep_address,  
                                1000);
```



### 3.5.8 AmbaUSBH\_Simple\_BulkWrite

#### API Syntax:

```
UINT32 AmbaUSBH_Simple_BulkWrite (UINT8 *DataPtr,  
                                  UINT32 len,  
                                  UINT32 *ActualLen,  
                                  UINT32 EpAddr,  
                                  UINT32 timeout)
```

#### Function Description:

The application calls this API to send data to Simple Class device via Bulk-Out pipe.

The simple class host will send an additional zero-length packet if the transfer length is a multiple of 512 bytes.

#### Parameters:

| Type    | Parameters | Description   |
|---------|------------|---|
| UINT8*  | DataPtr    | Pointer to data buffer.   |
| UINT32  | len        | The request length of the USB host.   |
| UINT32* | *ActualLen | The actual data length. Must be <= len.                                       |
| UINT32  | EpAddr     | Endpoint address  |
| UINT32  | timeout    | Timeout value in ms. 0 means no waiting.<br>0xFFFFFFFF means waiting forever. |

Table 3-339 Parameters for AmbaUSB API **AmbaUSBH\_Simple\_BulkWrite()**.



Return:

| Return Value | Description                     |
|--------------|---------------------------------|
| 0            | UX_SUCCESS                      |
| 0x5B         | UX_HOST_CLASS_INSTANCE_UNKNOWN  |
| 0x51         | UX_CONFIGURATION_HANDLE_UNKNOWN |
| 0x25         | UX_TRANSFER_NOT_READY           |
| 0x5C         | UX_TRANSFER_TIMEOUT             |
| 0x53         | UX_ENDPOINT_HANDLE_UNKNOWN      |

Table 3-340 Returns for API **AmbaUSBH\_Simple\_BulkWrite()**.

**Example:**

```
UINT8 UsbBuf[1024*16] = {0};  
UINT32 ActualLength = 0;  
UINT32 unRet = 0;  
int ep_address = 0x01;  
memset(UsbBuf, 0xaa, 1024*16);  
unRet = AmbaUSBH_Simple_BulkWrite(UsbBuf,  
                                1024*16,  
                                &ActualLength,  
                                ep_address,  
                                1000);
```



## Appendix 1. Additional Resources

Please contact an Ambarella representative for digital copies of documents.

For PROTRULY Confidential Only



## Appendix 2. Important Notice

All Ambarella design specifications, datasheets, drawings, files, and other documents (together and separately, "materials") are provided on an "as is" basis, and Ambarella makes no warranties, expressed, implied, statutory or otherwise with respect to the materials, and expressly disclaims all implied warranties of noninfringement, merchantability and fitness for a particular purpose.

The information contained herein is believed to be accurate and reliable. However, Ambarella assumes no responsibility for the consequences of use of such information.

Ambarella Incorporated reserves the right to correct, modify, enhance, improve, and otherwise change its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete.

All products are sold subject to Ambarella's terms and conditions of sale supplied at the time of order acknowledgment. Ambarella warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with its standard warranty. Testing and other quality control techniques are used to the extent

Ambarella deems necessary to support this warranty. Ambarella assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using Ambarella components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

Ambarella does not warrant or represent that any license, either expressed or implied, is granted under any Ambarella patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which Ambarella products or services are used. Information published by Ambarella regarding third-party products or services does not constitute a license from Ambarella to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from Ambarella under the patents or other intellectual property of Ambarella. Reproduction of information from Ambarella documents is not permissible without prior approval from Ambarella. Ambarella products are not authorized for use in safety-critical applications (such as life support) where a failure of the product would reasonably be expected to cause severe personal injury or death, unless officers of the parties have executed an agreement specifically governing such use. Customers



acknowledge and agree that they are solely responsible for all legal, regulatory and safety-related requirements concerning their products and any use of Ambarella products in such safety-critical applications, notwithstanding any applications-related information or support that may be provided by Ambarella. Further, Customers must fully indemnify Ambarella and its representatives against any damages arising out of the use of Ambarella products in such safety-critical applications. Ambarella products are neither designed nor intended for use in automotive and military/aerospace applications or environments. Customers acknowledge and agree that any such use of Ambarella products is solely at the Customer's risk, and they are solely responsible for compliance with all legal and regulatory requirements in connection with such use.

For PROTRULY Only



## Appendix 3. Revision History

NOTE: Page numbers for previous drafts may differ from page numbers in the current version.

| Version | Date          | Comments   |
|---------|---------------|--|
| 1.0     | 8 August 2013 | Initial draft version  |
| 1.3     | 21 Jan 2015   | Add description for AmbaUSBD_Msc_Mount() and AmbaUSBD_Msc_UnMount()  |
| 1.3     | 30 Jan 2015   | Add A9 block diagram in AmbaUSBH_SystemSetPhy0Owner().   |
| 1.3     | 06 Feb 2015   | Add AmbaUSB_System_SetMemoryPool()   |
| 1.3     | 10 Feb 2015   | Add AmbaUSBD_Mtp_SetSupportedEvents()<br>Add more descriptions in UVC Device Class APIs.<br>Add UVC Host APIs.                                       |
| 1.3     | 26 Feb 2015   | Rename AmbaUSB_System_SetMemoryPool() to AmbaUSBD_System_SetMemoryPool()<br>Add AmbaUSBH_System_SetMemoryPool()<br>Add AmbaUSBH_System_ClassUnHook() |
| 1.3     | 09 Mar 2015   | Add AmbaUSBD_System_SetBulkInTimeout()<br>Add AmbaUSB_System_SetDeviceConfigDetectTimeout()<br>Add Section 2.9, Stream Class APIs.                   |
| 1.4     | 2 Apr 2015    | Add Section 3.4 MTP Class APIs   |
| 1.5     | 11 May 2015   | Add Section 3.5 UVC Class APIs   |
| 1.6     | 15 July 2015  | Remove original 3.3 since it's the same as 3.5.<br>Modify Section 2.7.2, 2.7.3.<br>Add Section 3.1.8, 3.4.1.8 – 3.4.1.12, 3.4.16 – 3.4.64, 3.5       |
| 1.7     | 21 Aug 2015   | Updated Section 3.4: Added UVCH extension APIs   |

Table A3-1. Revision History.