AMX Module Interface Specification

for the

Vision HDbitT
HDMI Extender System

TABLE OF CONTENTS

Revision History

Date	Initials	Comments
17/09/2018	KDA	v1.0.0 Initial release

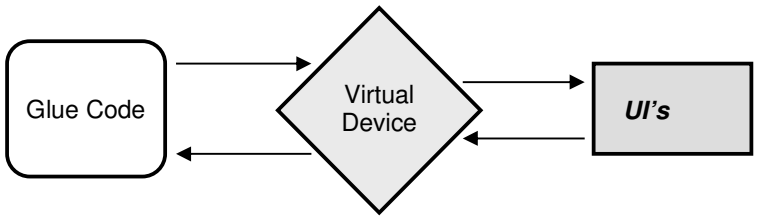
Introduction

This is a reference manual to describe the implementation process for the Vision_HDbitT_ar_v1_0_0.axs module. It also provides a list of Send_Command's available for mainline coding of the core components.

Overview

The Vision_HDbitT_ar_v1_0_0.axs has 4 parameters passed into it.
1st is the "Real" device for the module,
2nd is the "Virtual" device for the module,
3rd is the "Real" device to be the "Server" for the module,
4th is an array of UI's (User Interfaces) for the Module.
The "Virtual" device is used as the communications portal of the module. It will be used to control the module/device via send_commands and it will provide feedback and information about the module/device via send_strings.

The following diagram gives a graphical view of the interface between the interface code and the NetLinx module.



Implementation

To interface to the Vision_HDbitT_ar_v1_0_0.axs module, the programmer must perform the following steps:

1. Define the "Real" device ID in the Define_Device section of the NetLinx code.
2. Define the "Virtual" device ID in the Define_Device section of the NetLinx code.
3. Define the "Server" device ID in the Define_Device section of the NetLinx code.
4. Define the "UI Array" devices ID for the Module in the Define_Device section of the Netlinx code.

An example of how to do this is shown below.

```
DEFINE_DEVICE
    dvDevice        =     0:02:0    // The real device used for sending packets to the actual device
    dvServer        =     0:03:0    // The real device used for receive packets to the actual device
    vdvHDbitT       = 32768:01:0    // The virtual device used for communications to the module
    dvUI_1          = 10001:03:0    // UI number 1 for this module
    dvUI_2          = 10002:03:0    // UI number 2 for this module
    dvUI_3          = 10003:03:0    // UI number 3 for this module

DEFINE_VARIABLE
    dvUI_Array      = { dvUI_1, dvUI_2, dvUI_3 }    // An array of UI's for the module

DEFINE_START        // Place define_module calls to the very end of the define_start section.
// Comm module
DEFINE_MODULE 'Vision_HDbitT_ar_v1_0_0' mdlHDbitT (dvDevice, vdvHDbitT, dvServer, dvUI_Array)
```

Note on UI Array: AT Controls provides AMX modules that will have inbuilt UI status and technical control directly from the module. An example of the buttons and data fields of this have been provided in the IDE workspace "User Interface" section. If these are not required, simple at a "dummy" address (one not used anywhere in your code) for the UI array. The UI Array can be from 1 to any number of UIs.

Note on Start: Upon processor bootup, the module will set an internal random delay start time between 15 and 120 seconds after which it will attempt to communicate with the Vision device. The PROPERTY-IP_ADDRESS,###.###.###.### MUST be passed in before the communication is attempted.

Command Interface

Utilising the module for control of the module/device is done via the "Virtual" device and via command events (NetLinx command *send_command*) sent to the "Virtual" device. The common commands that should be supported by the module are listed below.
In the following table, <> represent a required value, but does not form part of the string/data.
In the following table, [] represent an optional value, but does not form part of the string/data.
Notes: There are 3 basic types of send_commands for the DSXM v2 Module. They are:
1) PROPERTY These are commands that will work at any time and are usually configuration type.
2) ? These are Query commands that will work anytime to provide status of the module.
3) Others These will only work when the module is "running" and controlling a device.
Module Understanding: The module utilises a generic communications method for communicating to the Vision IP devices. The command sets in this manual describe the Device commands first and then the generic communications commands. Some commands are applicable to both mechanisms.
The Vision device requires a server in the AMX module for return packets to be received. By default, this module sets up a server utilising its own IP Address and the IP port used to communicate with the Vision device itself.

Device Commands

Command – Property	Description
PROPERTY-BROADCAST,txt	Sets the state of the module to Broadcast Status changes txt = TRUE = Any change of state will be broadcast to glue code txt = FALSE = No change of status will be broadcast to glue code Note: Refer also to QUERY commands
PROPERTY-IP_ADDRESS,###.###.###.###	Sets the IP Address of the Vision device to communicate with The IP Address can be entered as URL if desired PROPERTY-IP_ADDRESS,10.1.1.112 PROPERTY-IP_ADDRESS,vision_tx@atcontrols Note: This command MUST be passed in before the module will attempt to communicate with the Vision device
PROPERTY-IP_PORT,####	Sets the IP Port for the communications with the Vision device PROPERTY-IP_PORT,9001 Note: By default this is set to 9001 (factory default of the Vision device)
Command – Query	Description
BROADCAST?	Query the "Broadcast" state of the module Response example: BROADCAST-TRUE When TRUE, any state change of the components of the module will be broadcast to the glue code By default, this is FALSE
INFORMATION?	Query all the information about the system This reposts the following data: Device Name Device Group ID Device MAC Address Server State Broadcast State Module Name and Version Comms component Version
IP_ADDRESS?	Query the "IP Address" the module is using to communicate with the Vision device Response example: IP_ADDRESS-###.###.###.###
IP_PORT?	Query the "IP Port" the module is using to communicate with the Vision device Response example: IP_PORT-####
VERSION?	Query the "Version" number of this module This will respond with the module version number: MODULE_VERSION-v1.0.0
Command – Normal	Description
GROUP_ID-## or CHANNEL-##	Set the device to the specified GROUP_ID (stream channel) ## = The GROUP_ID (0-99) GROUP_ID-1 // Sets the device to GROUP_ID 1 CHANNEL-53 // Sets the device to GROUP_ID 53 Note: When this is sent to a TX device it changes what stream it is pushing out on. When sent to an RX device it changes what stream it connects to. Be careful when changing a TX device that the GROUP_ID is not currently being transmitted by any other TX device

PASSBACK-state	Instruct the module to passback any responses
	state = TRUE // Any responses received from the device are passed to glue
	FALSE // Any responses received from the device are not passed

POLL-txt	Instruct the module to POLL for the specified information
	txt = MAC_ADDRESS // Polls for the device MAC Address
	= DEVICE_NAME // Polls for the Name of the device
	= GROUP_ID // Polls for the current GROUP_ID of the Device
	= CHANNEL // Polls for the current CHANNEL (group_id) of the device

SET_DEVICE_NAME-txt	Set the device name of the device txt = The NAME to set on the device (up to 32 characters) e.g. SET_DEVICE_NAME-RX_Display 1

Communications Commands

Command – Property (Setup)	Description
PROPERTY-COMMS_TIMING,<x>	Sets the "Communications Base Timing" value for the module. <0> or <RESET> : Sets timing back to default number of 100ms <1> .. <?> : Sets timing to the specific ms value Note: The module by default uses 100ms (1/10th second) comms basic timing. Note: Should not be set above 1000 (1sec) as this will effect second counters
PROPERTY-COMMUNICATIONS_REPORTING,<x>	Sets the "Communications Reporting" state for the module Refer to STRING_FEEDBACK section for details on these reports. <FALSE> or <OFF> : Turns Reporting OFF (default) <TRUE> or <ON> : Turns Reporting ON
PROPERTY-DEBUG,<x>[y][z]	Sets the "Debug" state for the module <0> or <OFF> : Turns Debug OFF <1> or <ON> : Turns Debug ON (to basic level reporting) <2> : Sets debug to level 2 which includes FUNCTIONS <3> : Sets debug to level 3 which includes ERRORS [D] : y is optional to have debug include Date stamping [T] : z is optional to have debug include Time stamping
PROPERTY-DEBUG_DEVICE,<x>,<y>,<z>	Sets the "Debug Device" for sending of debug messages <x> : Device <y> : Port <z> : System Note: xyz can also be expressed as d:p:s Examples: PROPERTY-DEBUG_DEVICE,5001:2:0 PROPERTY-DEBUG_DEVICE,5001,2,0 Note: By default the debug device is 0:1:0
PROPERTY-FEEDBACK_TIMING,<x>	Sets the "Feedback Timing" value for the module (1/10sec). Note: Will only accept values between 1 and 10.
PROPERTY-IP_ADDRESS,<x>	Sets the "IP Address" of the controlling device <x> : The IP Address or URL of the device to be controlled Note: Only applicable for IP type communications
PROPERTY-IP_PORT,<x>	Sets the "IP Port" of the controlling device <x> : The IP Port of the device to be controlled Note: Only applicable for IP type communications Note: By default this module will set Telnet port 23
PROPERTY-MC_SERVER_DEVICE,<x>,<y>,<z>	Sets the "MC (Multicast) Server Device" for receiving Multicast packets <x> : Device <y> : Port <z> : System Note: xyz can also be expressed as d:p:s Examples: PROPERTY-MC_SERVER_DEVICE,0,4,0 PROPERTY-REAL_DEVICE,0,4,0 *Note: This is really only applicable when the IP Protocol method is UDP in the Multicast IP Address range.* In some cases you may need to receive Multicast Packets on the network rather than point-to-point TCP methods. If a valid MC_SERVER_DEVICE command is received by the module it will open up the "listening" socket to receive packets. A Multicast Server will use the supplied IP Address and IP Port when opening this server. As described above this is really ONLY valid when using Multicast IP Addresses.
PROPERTY-REINIT,<x>	Triggers the "Initialisation" sequence of the module <x> : No of seconds before executing This command affectively re-initialises the module to run through the bootup sequence again. If x = 0, the module will be stopped.

PROPERTY-RESPONSE_TIMEOUT,<x>	Sets the "Response Timeout" value for communicating with the control device. <x> : The number of 1/10ths of seconds to wait for a response from the device. Examples: PROPERTY-RESPONSE_TIMEOUT,40 This sets that the module will wait up to 4 seconds for a response from a device when it sends a string. Note 1: 40 is the default value set by the module. Note 2: If like with UDP communications there will be no response from the device, set this value to 0.
PROPERTY-START_DELAY,<x>	Sets the "Start Delay" of the Module <x> : No of seconds to delay the start Explanation: By default these modules have a delayed start from bootup which is a randomly generated number between x and y seconds. This ensures that modules start up in the processor at different times rather than them all starting at once and placing demands on the processor. (ATC have found this improves the reliability of Virtual devices coming online, rebuild_events and virtual_channel/level_count settings) Using this command, ALL modules can be started exactly when the programmer wants from bootup.
PROPERTY-START_MODULE	Sets the "Running" state of the module This command starts the module processing. (see below in "RUNNING?")
PROPERTY-STOP_MODULE	Sets the "Running" state of the module This command stops the module from processing. (see below in "RUNNING?")
PROPERTY-UI_ONLINE_REFRESH_DELAY,<x>	Sets the "Touch Panel Online Refresh Delay" used for delaying the refreshing of the touch panel text buttons when the panels come online. If not set this setting is default to a random value between 20 and 120 deci seconds. <x> : Delay time measured in Deci seconds (1/10 sec)
Command - Query	Description
COMMS_METHOD?	Query the "Communications" settings of the module This will respond with the communications method for controlling the device: COMMS_METHOD-TYPE, INFO, SUB_INFO TYPE = IP, RS232 or VIRTUAL INFO = when IP = IP Address when RS232 = Baud Rate settings when VIRTUAL = The virtual device number Note: Virtual would mean this module is communicating through another module for data (strings) to/from the real device. SUB_INFO = when IP = IP Port
COMMS_VERSION?	Query the "Version" of the communications component of the module This will respond with the communications component version: COMMS_VERSION-v2.21.11
DEBUG-<x>[y][z]	Sets the "Debug" state of the module <0> or <OFF> : Turns Debug OFF <1> or <ON> : Turns Debug ON (to basic level reporting) <2> : Sets debug to level 2 which includes FUNCTIONS <3> : Sets debug to level 3 which includes ERRORS [D] : y is optional to have debug include Date stamping [T] : z is optional to have debug include Time stamping Note: DEBUG is the only nonstandard command that can be sent that will be actioned even when the module is not "running".
DEBUG?	Query the "Debug" state of the Module Module responds with the debug state: DEBUG-X<D><T> X = Debug Level (1 – 3) D = If date stamp is turned ON (not included if not) T = if time stamp is turned ON (not included if not)
DATE?	Query the "Date" this version was written This will respond with the module date and time of writing: MODULE_DATE-dd/mm/yy hh:mm:ss
INFO?	Query the "Info" of this module This will respond with all the information about the module: MODULE_INFO-Module_File_Name.axs, v2_x_y, dd/mm/yy hh:mm:ss, ### X = The minor version number Y = The bug fixes version number ### = The total lines of the module
IP_ADDRESS?	Query the "IP Address" settings of the module This will respond with the IP Address of the controlling device: IP_ADDRESS-xxx.xxx.xxx.xxx (or URL) Note: Only applicable in IP Type communications.

IP_PORT?	Query the "IP Port" settings of the module This will respond with the IP Port used for IP communications to the device: IP_PORT-#### Note: Only applicable in IP Type communications.
NAME?	Query the "Name" of this module This will respond with all the Module file name: MODULE_NAME-Module_File_Name_v2_x_y.axs e.g. Vision_HDbitT_ar_v1_0_0
PASSWORD?	Query the "Password" settings of the module This will respond with the password stored for communications to the device: PASSWORD-string
RESPONSE_TIMEOUT?	Query the "Response Timeout" time of the module. This will respond with what this setting currently is. RESPONSE_TIMEOUT-40 1/10THS SECONDS Note: When a string (packet) is sent to a device, the module by default will use a timeout of 4 seconds to wait for a response. But, if the device will not respond (often the case with UDP communications), this setting needs to set to 0 so communications happens faster.
RESPONSE_APPEND?	Query the "Response Append" flag of the module. This will respond with what this setting currently is. RESPONSE_APPEND-TRUE (or FALSE)
RESPONSE_PREPEND?	Query the "Response Prepend" flag of the module. This will respond with what this setting currently is. RESPONSE_PREPEND-TRUE (or FALSE)
RUNNING?	Query the "Running" state of the module This will respond with all the running (processing) state of the module: MODULE-RUNNING or STOPPED Note: When a module is stopped it will not be doing any processing for control. No data to and from the device No processing of UI inputs No timelines running in the module
AUTHENTICATION?	Query the state of the Authentication AUTHENTICATION-OFF AUTHENTICATION-USERNAME (It is waiting at username entry) AUTHENTICATION-PASSWORD (it is waiting at password entry) AUTHENTICATION-CONFIRMATION (it is waiting at a response of OK) AUTHENTICATION-OK (module assumes all Authentication is OK)
Command – Normal	Description
CLEAR_STACK	This command will clear any strings stacked to be sent to the device. Note: No parameters are required with this command.
FORCE_DELAY-<x>	This command will force a delay in any communications to a device. <x> : The number of 1/10ths seconds to force a delay. Example: "'FORCE_DELAY-20'" This would cause the module to not communicate with the device for 2 seconds. Once this timer has expired any stacked strings will be processed as normal. Note: Any string command can be sent to the module during the "FORCE_DELAY" period and will be processed, in order, immediately the timer has expired.
STRING-<x>	Sends a string to the device being communicated to be this module. <x> : The actual string (including terminators) that needs to be sent to the device. Example: "'STRING-Hello',$0D,$0A" This would send the string "'Hello',$0D,$0A" to the device. Any response to that string will be passed back by the module via a send_string to glue code. Note: The module handles all communications flow control. So multiple strings can be sent via this command without needing to wait for the response. The module will deliver them to the device using flow control and return any responses if/when they are received.

Module Control/Feedback Channels- Standard

#	Description	Channel FB	Address FB
10	Group_ID Set Keypad 0	Momentary	Digital Readout – Group_ID
11	Group_ID Set Keypad 1	Momentary	
12	Group_ID Set Keypad 2	Momentary	
13	Group_ID Set Keypad 3	Momentary	
14	Group_ID Set Keypad 4	Momentary	
15	Group_ID Set Keypad 5	Momentary	
16	Group_ID Set Keypad 6	Momentary	
17	Group_ID Set Keypad 7	Momentary	
18	Group_ID Set Keypad 8	Momentary	
19	Group_ID Set Keypad 9	Momentary	
Note:	Pulse vdvVirtual of channels 10-19 also acts the same		
67	Module Info display 1		Module file name
68	Module Info display 2		Module version
69	Module Info display 3		Module build date time
70	Module Info display 4		Module virtual device
71	Module Info display 5		Module real device
72	Module Info display 6		Module comms method
73	Module Info display 7		Module comms connect data
74	Module Info display 8		Module debug device
75	Module Info display 9		Spare display field
76	Module Info display 10		Module running state
77 – 250	SNAPI Compliant channels (added in when programmed)	SNAPI Standards	SNAPI Standards
251	Communications OK / Device Name	Channel	Device Name Text field
252	Data Initialised (Status of when Data has been gathered)	Channel	
253	Lamp Warming (On during lamp warm up – displays)	Channel	
254	Lamp Cooling (On during lamp cooling – displays)	Channel	
255	Power status	Channel	
256	MC (Multicast) Server Online status	Channel	
257	Data Transmit	Pulse	
258	Data Receive	Pulse	
259	Communicating to device (on between data sent and received)	Channel	
260	Connection (status of communications connection with device)	Channel	
261	Text Status 1		Text feedback – module specific
262	Text Status 2		Text feedback – module specific
263	Text Status 3		Text feedback – module specific
264	Text Status 4		Text feedback – module specific
265	Text Status 5		Text feedback – module specific
266	Text Status 6		Text feedback – module specific
267	Gather Data (on while data is being gathered from device)	Channel	
268	Refresh UI (updates all status to the UI/s)	Pulse	
269	Debug (status of Debug state)	Channel	Text of Debug State
270	Module Running (status of the processing state of the module)	Channel	
271	Diagnostics Status – Time & Type 1 + Diagnostics On/Off	Channel (On/Off)	Display of Time & Type 1
272	Diagnostics Status – Time & Type 2 + Diagnostics ASCII/HEX	Channel (Mode)	Display of Time & Type 2
273	Diagnostics Status – Time & Type 3 + Diagnostics HEX only	Channel (Mode)	Display of Time & Type 3
274	Diagnostics Status – Time & Type 4 + Diagnostics BINARY only	Channel (Mode)	Display of Time & Type 4
275	Diagnostics Status – Time & Type 5 + Diagnostics Clear	Pulse (Clear)	Display of Time & Type 5
276	Diagnostics Status – Time & Type 6		Display of Time & Type 6
277	Diagnostics Status – Time & Type 7		Display of Time & Type 7
278	Diagnostics Status – Time & Type 8		Display of Time & Type 8
279	Diagnostics Status – Time & Type 9		Display of Time & Type 9
280	Diagnostics Status – Time & Type 10		Display of Time & Type 10
281	Diagnostics Status – Time & Type 11		Display of Time & Type 11
282	Diagnostics Status – Time & Type 12		Display of Time & Type 12
283	Diagnostics Status – Time & Type 13		Display of Time & Type 13
284	Diagnostics Status – Time & Type 14		Display of Time & Type 14
285	Diagnostics Status – Time & Type 15		Display of Time & Type 15
286	Diagnostics Status – Data 1		Display of Data (String) 1
287	Diagnostics Status – Data 2		Display of Data (String) 2
288	Diagnostics Status – Data 3		Display of Data (String) 3
289	Diagnostics Status – Data 4		Display of Data (String) 4
290	Diagnostics Status – Data 5		Display of Data (String) 5
291	Diagnostics Status – Data 6		Display of Data (String) 6
292	Diagnostics Status – Data 7		Display of Data (String) 7
293	Diagnostics Status – Data 8		Display of Data (String) 8
294	Diagnostics Status – Data 9		Display of Data (String) 9
295	Diagnostics Status – Data 10		Display of Data (String) 10
296	Diagnostics Status – Data 11		Display of Data (String) 11
297	Diagnostics Status – Data 12		Display of Data (String) 12
298	Diagnostics Status – Data 13		Display of Data (String) 13
299	Diagnostics Status – Data 14		Display of Data (String) 14
300	Diagnostics Status – Data 15		Display of Data (String) 15

Module Control/Feedback Channels- Module Specific

#	Description	Channel FB	Address FB
301	Brand		Text field
302	Model		Text field
303	MAC Address		Text field
304			
305	Control Method		Text field

Module String Feedback

The module will issue unsolicited status feedback notices to inform the programmer the state of communications connection and authentication status, plus other information.
These are sent by the module as strings to glue code.

String	Description
BROADCAST-txt	This string lets the glue code the current state of BROADCAST TRUE = The module will broadcast any state changes to glue code FALSE = The module will not broadcast any state changes to glue code
COMMUNICATIONS-ONLINE,<x>	Reporting status of the communications connection state: Where x = TRUE – Connection to the device has been established FALSE – Connection to the device has been dropped (or lost)
COMMUNICATIONS-AUTHENTICATION,<x>	Reporting status of the Authentication state with a device: Where x = OFF – Authentication is not required USERBNAME – The server is waiting for system to submit a Username PASSWORD – The server is waiting for system to submit a Password CONFIRMATION – System has submitted UN/PW and awaiting Confirmation OK – The Authentication process is complete and all is Good (OK) RESET – Authentication process has been Reset, waiting a connection
ERROR:txt	This string lets the glue code know of any errors that have occurred These are explicit in description
FORCE_DELAY-<x>	This string lets the glue code know when the comms has been forced into a delay. The string is sent when the force delay is actually started and then again when it has timed out. x = TRUE or FALSE
GROUP_ID-##	This string lets the glue code the current Group_ID set in the device ## = The Group ID set (0-99) UNKNOWN = When the module is unaware of the group ID of the device
MODULE_INITIALISED-<x>	This string lets the glue code know when the module has initialised or not. x = TRUE or FALSE
SERVER-txt	This string lets the glue code the current Server state (for receiving packets from the device) LISTENING = The server is waiting to receive a connection CONNECTED = The device has connected and is communicating OFFLINE = The server is not initialised UNKOWN = The module is unaware of the server state

NOTE: By default the COMMUNICATIONS- reporting is OFF. A PROPERTY send_command must be used to turn it on. This makes this added feature backwards compatible.
PROPERTY-COMMUNICATIONS_REPORTING,TRUE or FALSE