Argus ETMobile

Real-Time Communication with External Devices

MANUAL VERSION 1.1

July, 2016

Argus Science LLC

techsupport@argusscience.com
Web site: www.argusscience.com

Table of Contents

1 Network communication with ETMobile

Commands can be sent to the ETMobile (ETM) application from an external device via TCP/IP network connection. Streaming Data and/or video can be received from ETM by the external device via either TCP or UDP. Commands are used to send XDAT values to ETM; to open, close and name data files on ETM; to start and stop recording to data files on ETM; and to initiate and stop streaming real-time data and/or video from ETM to the external device.

An application called ETRemote is provided with ETM systems. ETRemote can communicate with ETM from an external Win7 PC and uses the communication protocol described in this manual. The same communication protocol can be used to create custom applications, with C#, VC++, ETNetLib, MFC, as well as with scripting application like Paradigm, e-Prime, and MatLab.

The ETM IPv4 address (or addresses if more than network is available) and Server Port number are displayed on the ETM "Network Configuration" dialog (*Network button under Scene heading*). Either a LAN cable or WiFi may be used, but cannot be the same LAN or WiFi network being used to communicate between the ETM PC and DTU.
The network to be used for external communication must be set, on the ETM PC, to "obtain IP address dynamically".

If WiFi connection will be used between the ETM PC and DTU, then the built-in LAN port can be used for external comm. In this case use the Network Config tool, provided on the ETM PC, to set "LAN to Internet" and "WLAN to DTU".

If the built in LAN port, on the ETM PC is being used to connect to the DTU, then the built in WiFi on the ETM PC can be used for external comm. In this case use the Network Config tool, to set "LAN to DTU" and "WLAN to Internet". Alternately a USB to Network adapter can be used to create a second LAN on the ETM PC for external comm. This second network must be set, on the ETM PC, to "obtain IP address dynamically". In this case it will need to be done with Win7 "Network and Internet" dialogs (the ETM Network Config tool will only configure the built in LAN port).

A TCP "command socket" must be opened by the external device, and connected to ETM to enable command communication. When a command channel is opened, command messages can be sent to ETM by the external device, and in certain cases ETM will send messages to the external device on this channel. Streaming data or video requires that a second TCP or UDP "data socket" be opened by the external device and connected to the ETM application.

Supported commands and received data and video message formats are described in section 2, "ETM Message Format".

To send commands from an external device

On ETM:

- Open the Network Configuration dialog. If more than one network is available, select the one to be used from the drop down "IPv4" list.
- Click the "Listen" button on "Network Configuration" dialog.

On the External device:

- Create a TCP Socket which will be referred to as the "command socket".
- connect the command socket with the ETM IP address and port number.
- send one of the supported commands to the command socket.

To receive streaming UDP data or video from ETM

On ETM:

- Select the proper address and port, and click "Listen as previously described.
- "Send Data Result" and/or "Send Video" must be checked on the *Network Configuration* dialog.

On the External device:

- If a command channel is not already opened, create a TCP "command" socket and connect it to the ETM as previously described.
- Create a UDP socket for receiving data ("UDP data socket"), and a memory buffer in which to copy data that comes to the UDP socket receive buffer.
- Send the CMD_START_SDATA_UDP (or CMD_START_SVIDEO) message to the command socket.
- Data (or video) will begin streaming to the UDP data socket. Copy data from the UDP receive buffer to a memory buffer for processing.
- Use the CMD_STOP_SDATA_UDP (or CMD_STOP_SVIDEO_UDP) command message to halt the data or video streaming.

To receive streaming TCP data or video from an ETM data channel

On ETM:

- Select the proper address and port, and click "Listen as previously described.
- "Send Data Result" and/or "Send Video" must be checked on the *Network Configuration* dialog.

On the External device:

- If a command channel is not already opened, create a TCP "command" socket and connect it to the ETM as previously described.
- Create a TCP socket for receiving data ("TCP data socket") and a memory buffer in which to copy data that comes to the TCP socket receive buffer.
- Send the CMD_SET_CONNECT_TYPE to the command socket with the proper argument for "send data" or "send video".
- Connect the TCP data socket with ETM IP address and port number. Data (or video) will begin streaming as soon as the connection is made. Copy data from the TCP receive buffer to a memory buffer for processing.
- Close the TCP data socket connection to halt data streaming.

To receive a single data item value from ETM command channel
On the External device:

- If a command channel is not already opened, create a TCP "command socket" and connect it to the ETM as previously described.
- Send CMD_GET_DATA_ITEM with data ID set to desired item (list of IDs is provided at the end of this section and again in section 2)
- Read the response on the command socket channel. Examine the response to first confirm that it is a non-error response to the CMD_GET_DATA_ITEM command, and then to get the data item value. The response format is specified in section 2 along with all variable IDs, and the corresponding variable types.
- If the data ID is set to one of the variables marked "NA" (not available from ETM), 0 will be returned as the data item value.

Remember to close all sockets when through.

All commands start with a 4 byte Argus Signature, and include a "byte order" checksum. For the Signature and all other messages elements specified, the least significant byte is sent first; so the first byte sent will be hex53 (least significant byte of the Argus Signature) followed by hex47, etc.

To compute the checksum sum all of the bytes in the message (other than the checksum itself), ignore all but the least significant byte of the sum, and take the "twos compliment" negative of that byte. If all bytes in the resulting message are summed (including the checksum) the least significant byte of the sum should be zero.

For example, to set XDAT=100 (hex64) send the following 20 bytes in the order listed. The values below are hex values.

Byte 0: 53
Byte 1: 47
Byte 2: 41
Byte 3: 20
Byte 4: 14
Byte 5: 00
Byte 6: 00
Byte 7: 00
Byte 8: 05
Byte 9: 00
Byte 10: 00
Byte 11: 00
Byte 12: 83
Byte 13: 00
Byte 14: 00
Byte 15: 00
Byte 16: 64
Byte 17: 00
Byte 18: 00
Byte 19: 00

Bytes 0-3 are the ArgusSignature; bytes 4-7 are the message size; bytes 8-11 are the Set_XDAT command; bytes 12-15 are the checksum; and bytes 16-19 are the XDAT

value to be set. To calculate the check sum first sum bytes 0 through 11 and bytes 16 through 19 yielding dec381=hex17D. Consider only the least significant byte (hex7D). Take the twos compliment negative of hex7D to get hex83.

To have an external device command the ETM to record data, first use the CMD_SET_DATAFILE_NAME command to specify a file name for the ETM csv file. . If ET3Space is opened, this will set a file name for both the ET3Space ehd file, and the ETM csv file. If ET3 Space is in use, CMD_OPEN_DATAFILE will open an ehd file with the same name as the csv file. Use CMD_START_DATAFILE_RECORDING to start recording both the csv and ehd file (if an ehd file has been opened) and CMD_STOP_DATAFILE_RECORDING can be used to pause recording. If recording is restarted after a pause, without closing the ehd file or naming a new csv file, recording resumes as a new "segment" on the same ehd file. In the case of the csv file, the ETM will automatically start a new csv file with the previous name and an incremented number appended to the end of the name. Close both ehd and csv files with CMD_CLOSE_DATAFILE.

Note that external control can be mixed with local ETM control. For example, a data file can be named (and opened on ET3Space) manually on the ETM application, and an external device can then command recording to start and stop, etc. A start recording command from an external device will have no affect on ET3Space if a file is not opened, a stop recording command will have no affect if recording has not started, and so forth.

The data message format (CMD_DATA_MSG), for real-time streaming data sent from ETM, is described in the "ETM Message Format" section. The message item called "CheckState" is an 8 byte (64 bit) value specifying the contents of the data buffer. Each bit represents one of the possible data items from the list of possible data items shown in the table, at the end of this section. If the bit is set, the corresponding data item is present. Included data items will be in the data buffer in the same order that they appear in the table. The least significant bit of the 8 byte CheckState value (the least significant bit of byte 47 in the data message) is bit 0 and corresponds to the fist item listed in the table. Note that there are only 41 data items, so bits 41-63 of CheckState will always be 0. The list of data items at the end of this section specifies the CheckState bit associated with each item.

The CheckState bit only indicates the presence or absence of a data item. The size of each item, if present, is that listed in the table at the end of this section. Items marked "NA" are not available from ETMobile systems, and will never be included in the streaming data. Some integer values have "scale factors", which are also listed in the table. To create a floating point value with the intended units of measure (inches or centimeters for distance values, degrees for angle values, and pixels for pupil diameter) first convert the value to a float, then multiply by the scale factor listed in the table. An explanation of each data item can be found in section 3 of this document.

All data sent by ETM uses the "little endian" convention (least significant byte of each word is stored in smallest memory address).

Most commands sent to the ETM PC over the "command socket" are unidirectional. In some cases the command starts data streaming from the ETM over the "data socket", but no response is expected on the command socket. Command 25, however, does elicit a response on the command socket. The response begins with the ArgusSignature (bytes 0-3) and message size (bytes 4-7) followed by the "Response cmd" (bytes 8-12). The "Response Cmd, starting at Byte 8 should have the most significant bit set. The next bit (bit14 of byte 8) will be set if an error has been detected. The least significant byte of the "Response Cmd" (byte 12) should contain the number of the command being responded to. The response checksum is computed just as described earlier for commands.

Commands 14 – 17 are sent by the ETM program to an external application if a command socket has been opened, and if "Auto-Record Remote Screen", on the ETM Nework Configuration dialog, is checked. They are intended for communication with ETRemote, but can be received by any external application that has opened a command channel with ETM. Each time ETM starts recording a data file, it will send the CMD_OPEN_SVFILE and the CMD_START_SVFILE_RECORDING command. The external application is free to either make use of them or ignore them. If the external application is ETRemote, when it receives these commands it will open a screen file and start recording. When ETM stops recording data, it will send CMD_STOP_SVFILE_RECORDING. This allows ETRemote (or any other external app) to record a video of the stimulus screen that is exactly synchronized with the start and end of the data file recorded on ETM.

Sample C#, VC++, ETNetLib, and MFC programs, using the ETM network communication protocol, are provided as part of the ETRemote installation. After installing ETRemote, these samples can be found in *C:\Program Files\Argus Science\ETRemote\ET_SDK_Samples*.

CheckState bit	Data item	Type	Size (bytes)	Scale factor
0	start_of_record	Byte	1	1
1	status	Byte	1	1
2	overtime_count	Uint16	2	1
3	mark_value	Byte	1	1
4	XDAT	UInt16	2	1
5	CU_video_field_num	UInt16	2	1
6	pupil_pos_horz	UInt16	2	1
7	pupil_pos_vert	UInt16	2	1
8	pupil_diam	UInt16	2	0.01
9	pupil_height	NA		
10	cr_pos_horz	UInt16	2	1
11	cr_pos_vert	UInt16	2	1
12	cr_diam	NA		
13	horz_gaze_coord	Int16	2	0.1
14	vert_gaze_coord	Int16	2	0.1
15	horz_gaze_offset	NA		
16	vert_gaze_offset	NA		
17	hdtrk_X	Int16	2	0.01
18	hdtrk_Y	Int16	2	0.01
19	hdtrk_Z	Int16	2	0.01
20	hdtrk_az	Int16	2	0.01
21	hdrtk_el	Int16	2	0.01
22	hdtrk_rl	Int16	2	0.01
23	EH_scene_number	Byte	1	1
24	EH_gaze_length	Single	4	1
25	EH_horz_gaze_coord	Single	4	1
26	EH_vert_gaze_coord	Single	4	1
27	eyeplot_x	Single	4	1
28	eyeplot_y	Single	4	1
29	EH_eyelocation_X	Int16	2	0.01
30	EH_eyelocation_Y	Int16	2	0.01
31	EH_eyelocation_Z	Int16	2	0.01
32	EH_gaze_dir_X	Int16	2	0.001
33	EH_gaze_dir_Y	Int16	2	0.001
34	EH_gaze_dir_Z	Int16	2	0.001
35	aux_sensor_X	NA		
36	aux_sensor_Y	NA		
37	aux_sensor_Z	NA		
38	aux_sensor_az	NA		
39	aux_sensor_el	NA		
40	aux_sensor_rl	NA		

2 ETM Message Format

Byte 0: ArgusSignature // Argus Signature (0x20414753): " AGS", 4 bytes
Byte 4: MsgSize // Total message size, 4 bytes
Byte 8: Cmd // Message Command, 1 byte, other bytes reserved to 0
Byte 12: Checksum // Message Checksum, 1 byte, other bytes reserved to 0
Byte 16: Argument // Optional, depends on command type

```
// Command Type
#define CMD_START_DATAFILE_RECORDING   1   //start recording data on ETM  PC
#define CMD_STOP_DATAFILE_RECORDING    2   //stop recording data on ETM PC
#define CMD_OPEN_DATAFILE              3   //open a data file on ETM PC
#define CMD_CLOSE_DATAFILE             4   //close currently open data file on ETM PC
#define CMD_SET_XDAT                   5   //set an XDAT value on ETM PC
#define CMD_SET_DATAFILE_NAME          6   //set the data file name on ETM PC
#define CMD_SET_CONNECT_TYPE           7   //set a TCP command socket for data or video
#define CMD_START_SDATA_UDP            8   //start sending UDP data from ETM
#define CMD_STOP_SDATA_UDP             9   //stop sending UDP data from ETM
#define CMD_START_SVIDEO_UDP           10  //start sending UDP video from ETM
#define CMD_STOP_SVIDEO_UDP            11  //stop sending UDP video from ETM
#define CMD_START_SVFILE_RECORDING     14  //start recording material sent from ETM
#define CMD_STOP_SVFILE_RECORDING      15  //stop recording material sent from ETM
#define CMD_OPEN_SVFILE                16  //open a file to record material sent from ETM
#define CMD_CLOSE_SVFILE               17  //close file

// Special Command Type with Response
#define CMD_GET_DATAITEM               25

// Response bit for network command
#define CMD_RSP_BIT                    0x80000000
#define CMD_RSP_ERR_BIT                0x40000000

// Data / Video Command Type Message
#define CMD_DATA_MSG                   0x81
#define CMD_JPEG_MSG                   0x82
```

// Normal Command Type 1 - 17

1. **CMD_START_DATAFILE_RECORDING (1)**

Byte 0: Argus Signature (0x20414753) //Argus Signature " AGS"
Byte 4: MsgSize (0x00000010) // Total message size: 16
Byte 8: Cmd (0x00000001) // Message Command: Start Recording Data
Byte 12: Checksum (0x000000ef) // Message Checksum: checksum byte

2. **CMD_STOP_DATAFILE_RECORDING (2)**

Byte 0: Argus Signature (0x20414753) //Argus Signature " AGS"
Byte 4: MsgSize (0x00000010) // Total message size: 16
Byte 8: Cmd (0x00000002) // Message Command: Stop Recording Data
Byte 12: Checksum (0x000000ee) // Message Checksum: checksum byte

3. **CMD_OPEN_DATAFILE (3)**

Byte 0: Argus Signature (0x20414753) //Argus Signature " AGS"
Byte 4: MsgSize (0x00000010) // Total message size: 16
Byte 8: Cmd (0x00000003) // Message Command: Open Data File
Byte 12: Checksum (0x000000ed) // Message Checksum: checksum byte

4. **CMD_CLOSE_DATAFILE (4)**

Byte 0: Argus Signature (0x20414753) //Argus Signature " AGS"
Byte 4: MsgSize (0x00000010) // Total message size: 16
Byte 8: Cmd (0x00000004) // Message Command: Close Data File
Byte 12: Checksum (0x000000ec) // Message Checksum: checksum byte

5. **CMD_SET_XDAT (5)**

Byte 0: Argus Signature (0x20414753) //Argus Signature " AGS"
Byte 4: MsgSize (0x00000014) // Total message size: 20
Byte 8: Cmd (0x00000005) // Message Command: Set XDAT
Byte 12: Checksum // Message Checksum: checksum byte
Byte 16: Argument (XDAT) // XData Value

6. **CMD_SET_DATAFILE_NAME (6)**

Byte 0: Argus Signature (0x20414753) //Argus Signature " AGS"
Byte 4: MsgSize // Total message size: 16 + Size of Data File Name
Byte 8: Cmd (0x00000006) // Message Command: Set XDAT
Byte 12: Checksum // Message Checksum: checksum byte
Byte 16: Argument (Char String) // Data File Name Char String

7. **CMD_SET_CONNECT_TYPE (7)**

Byte 0: Argus Signature (0x20414753) //Argus Signature " AGS"
Byte 4: MsgSize (0x00000014) // Total message size: 20
Byte 8: Cmd (0x00000007) // Message Command: Set Connect Type
Byte 12: Checksum // Message Checksum: checksum byte
Byte 16: Argument // Connect Type

- #define SOCKET_TYPE_SDATA_TCP 3 // Send Data to Remote via TCP / IP
- #define SOCKET_TYPE_SVIDEO_TCP 7 // Send Video to Remote via TCP / IP

- #define SOCKET_TYPE_RVIDEO_TCP 9 // Receive Video from Remote via TCP / IP

8. CMD_START_SDATA_UDP (8)

Byte 0: Argus Signature (0x20414753) //Argus Signature " AGS"
Byte 4: MsgSize (0x00000014) // Total message size: 20
Byte 8: Cmd (0x00000008) // Message Command: Start Sending UDP Data
Byte 12: Checksum // Message Checksum: checksum byte
Byte 16: Argument (Port Number) // UDP Port Number

9. CMD_STOP_SDATA_UDP (9)

Byte 0: Argus Signature (0x20414753) //Argus Signature " AGS"
Byte 4: MsgSize (0x00000010) // Total message size: 16
Byte 8: Cmd (0x00000009) // Message Command: Stop Sending UDP Data
Byte 12: Checksum (0x000000e7) // Message Checksum: checksum byte

10. CMD_START_SVIDEO_UDP (10)

Byte 0: Argus Signature (0x20414753) //Argus Signature " AGS"
Byte 4: MsgSize (0x00000014) // Total message size: 20
Byte 8: Cmd (0x0000000a) // Message Command: Start Sending UDP Video
Byte 12: Checksum // Message Checksum: checksum byte
Byte 16: Argument (Port Number) // UDP Port Number

11. CMD_STOP_SVIDEO_UDP (11)

Byte 0: Argus Signature (0x20414753) //Argus Signature " AGS"
Byte 4: MsgSize (0x00000010) // Total message size: 16
Byte 8: Cmd (0x0000000b) // Message Command: Stop Sending UDP Video
Byte 12: Checksum (0x000000e5) // Message Checksum: checksum byte

14. CMD_START_SVFILE_RECORDING (14)

Byte 0: Argus Signature (0x20414753) //Argus Signature " AGS"
Byte 4: MsgSize (0x00000010) // Total message size: 16
Byte 8: Cmd (0x0000000e) // Message Command: Start Recording Screen Video
Byte 12: Checksum (0x000000e2) // Message Checksum: checksum byte

15. CMD_STOP_SVFILE_RECORDING (15)

Byte 0: Argus Signature (0x20414753) //Argus Signature " AGS"
Byte 4: MsgSize (0x00000010) // Total message size: 16
Byte 8: Cmd (0x0000000f) // Message Command: Stop Recording Screen Video
Byte 12: Checksum (0x000000e1) // Message Checksum: checksum byte

16. CMD_OPEN_SVFILE (16)

Byte 0: Argus Signature (0x20414753) //Argus Signature " AGS"
Byte 4: MsgSize // Total message size: 16 + Size of Data File Name
Byte 8: Cmd (0x00000010) // Message Command: Open Screen Video File
Byte 12: Checksum // Message Checksum: checksum byte
Byte 16: Argument (Char String) // Screen Video File Name Char String

17. CMD_CLOSE_SVFILE (17)

Byte 0: Argus Signature (0x20414753) //Argus Signature " AGS"
Byte 4: MsgSize (0x00000010) // Total message size: 16
Byte 8: Cmd (0x00000011) // Message Command: Close Screen Video File
Byte 12: Checksum (0x000000df) // Message Checksum: checksum byte

// Special Command Type for EyeTracRemote (command sent by ETM to an open command socket)

25. CMD_GET_DATA_ITEM (25)

Byte 0: Argus Signature (0x20414753) //Argus Signature " AGS"
Byte 4: MsgSize // Total message size: 20
Byte 8: Cmd (0x00000017) // Message Command: Get Eye Data Item Value
Byte 12: Checksum // Message Checksum: checksum byte
Byte 16: DataID // Data Item ID

// Response
Byte 0: Argus Signature (0x20414753) //Argus Signature " AGS"
Byte 4: MsgSize // Total message size: 48
Byte 8: Response Cmd // Response Command:
// No Error: CMD_RSP_BIT | MD_GET_DATAITEM
// Error: CMD_RSP_BIT | CMD_RSP_ERR_BIT | MD_GET_DATAITEM
Byte 12: Checksum // Message Checksum: checksum byte
Byte 16: FrameNo // Current Frame Number
Byte 20: Reserved // Reserved bytes
Byte 24: TimeStamp // Time Stamp of this frame
Byte 32: UpdateRate // Frame Update Rate
Byte 36: DataID // Data Item ID
Byte 40: DataVal // Data Item Value
Byte 44: Reserved // Reserved bytes

```
// Data File Items defination
#define DATAID_START_OF_RECORD          0     // DataVal: Unsigned Integer Type
#define DATAID_STATUS                   1     // DataVal: Unsigned Integer Type
#define DATAID_OVERTIME_COUNT           2     // DataVal: Unsigned Integer Type
#define DATAID_MARK_VALUE               3     // DataVal: Unsigned Integer Type

#define DATAID_XDAT                     4     // DataVal: Unsigned Integer Type
#define DATAID_CU_VIDEO_FIELD_NUM       5     // DataVal: Unsigned Integer Type

#define DATAID_PUPIL_POS_HORZ           6     // DataVal: Floating Point Type
#define DATAID_PUPIL_POS_VERT           7     // DataVal: Floating Point Type

#define DATAID_PUPIL_DIAM               8     // DataVal: Floating Point Type
#define DATAID_PUPIL_HEIGHT             9     // DataVal: Floating Point Type

#define DATAID_CR_POS_HORZ              10    // DataVal: Floating Point Type
#define DATAID_CR_POS_VERT              11    // DataVal: Floating Point Type

#define DATAID_CR_DIAM                  12    // Not available from ETM
```

```
#define DATAID_HORZ_GAZE_COORD          13      // DataVal: Floating Point Type
#define DATAID_VERT_GAZE_COORD          14      // DataVal: Floating Point Type

#define DATAID_HORZ_GAZE_OFFSET         15      // Not available from ETM
#define DATAID_VERT_GAZE_OFFSET         16      // Not available from ETM

#define DATAID_HDTRK_X                  17      // DataVal: Floating Point Type
#define DATAID_HDTRK_Y                  18      // DataVal: Floating Point Type
#define DATAID_HDTRK_Z                  19      // DataVal: Floating Point Type

#define DATAID_HDTRK_AZ                 20      // DataVal: Floating Point Type
#define DATAID_HDTRK_EL                 21      // DataVal: Floating Point Type
#define DATAID_HDTRK_RL                 22      // DataVal: Floating Point Type

#define DATAID_EH_SCENE_NUMBER          23      // DataVal: Integral Type
#define DATAID_EH_GAZE_LENGTH           24      // DataVal: Floating Point Type

#define DATAID_EH_HORZ_GAZE_COORD       25      // DataVal: Floating Point Type
#define DATAID_EH_VERT_GAZE_COORD       26      // DataVal: Floating Point Type

#define DATAID_EYEPLOT_X                27      // DataVal: Floating Point Type
#define DATAID_EYEPLOT_Y                28      // DataVal: Floating Point Type

#define DATAID_EH_EYE_LOCATION_X        29      // DataVal: Floating Point Type
#define DATAID_EH_EYE_LOCATION_Y        30      // DataVal: Floating Point Type
#define DATAID_EH_EYE_LOCATION_Z        31      // DataVal: Floating Point Type

#define DATAID_EH_GAZE_DIR_X            32      // DataVal: Floating Point Type
#define DATAID_EH_GAZE_DIR_Y            33      // DataVal: Floating Point Type
#define DATAID_EH_GAZE_DIR_Z            34      // DataVal: Floating Point Type

#define DATAID_AUX_SENSOR_X             35      // Not available from ETM
#define DATAID_AUX_SENSOR_Y             36      // Not available from ETM
#define DATAID_AUX_SENSOR_Z             37      // Not available from ETM

#define DATAID_AUX_SENSOR_AZ            38      // Not available from ETM
#define DATAID_AUX_SENSOR_EL            39      // Not available from ETM
#define DATAID_AUX_SENSOR_RL            40      // Not available from ETM
```

// Data / Video Command Type Message

Data Message Format: CMD_DATA_MSG(0x81)

Byte 0: Argus Signature (0x20414753) //Argus Signature " AGS"
Byte 4: MsgSize // Total message size: 56 + DataSize
Byte 8: Cmd (0x00000081) // Message Command: Data Message
Byte 12: Checksum (0x00000000) // Message Checksum: Reserved to 0 for Data Message
Byte 16: DataSize // Selected Data Size
Byte 20: FrameSize(0x00000000) // Video Frame Size: 0 since no video
Byte 24: FrameNo // Current Frame Number
Byte 28: Reserved // reserved bytes
Byte 32: TimeStamp // Time Stamp of this frame
Byte 40: UpdateRate // Frame Update Rate
Byte 44: Reserved // reserved bytes

Byte 48: CheckState // Data Selected State
Byte 56: BufStart // Data Buffer

Video Message Format: CMD_JPEG_MSG(0x82)

Byte 0: Argus Signature (0x20414753) //Argus Signature " AGS"
Byte 4: MsgSize // Total message size: 56 + DataSize + FrameSize
Byte 8: Cmd (0x00000082) // Message Command: JPEG Message
Byte 12: Checksum (0x00000000) // Message Checksum: Reserved to 0 for JPEG Message
Byte 16: DataSize // Selected Data Size
Byte 20: FrameSize // Video Frame Size (Encoded JPEG Image)
Byte 24: FrameNo // Current Frame Number
Byte 28: Reserved // reserved bytes
Byte 32: TimeStamp // Time Stamp of this frame
Byte 40: UpdateRate // Frame Update Rate
Byte 44: Reserved // reserved bytes
Byte 48: CheckState // Data Selected State
Byte 56: BufStart // Data Buffer
Byte 56 + DataSize: Buffer // Video Buffer

3 Data Item Explanation

Start of record byte – fixed value 0xFA

Status byte – contains eye tracer status information.

Bit	Meaning (if 1)
0 (least significant)	Head tracker enabled, monocular system or left eye binocular
1	Head tracker enabled, right eye (binocular system only)
2	Cornea Reflection found, right eye (binocular system only)
3	Pupil Found, right eye (binocular system only)
4	Cornea Reflection found, monocular system or left eye binocular
5	Pupil Found, monocular system or left eye binocular
6	Right eye data was simulated for left/right eye data synchronization (binocular only)
7	Left eye data was simulated for left/right eye data synchronization (binocular only)

overtime count, 2 bytes, unsigned integer. Shows how many records were lost prior to this one. Typically contains the value zero.

Mark value byte – will be last integer "Mark" value entered by user.

XDAT – 16 bit integer set by external device.

CU video field number – Internal field (or record) number kept by system. It is the number of video fields received from the eye camera since the ETM program was activated, and rolls over to 0 after every 65535 fields. Useful mostly for debugging purposes.

pupil_pos – coordinates proportional to horizontal (0 to 640) and vertical ()0 to 480) pupil position with respect to the eye camera field of view.

Pupil_diam – value proportional to diameter of the pupil image on the eye camera.

cr_pos -- coordinates proportional to horizontal (0 to 640) and vertical ()0 to 480) position of the primary corneal reflection with respect to the eye camera field of view.

gaze_coord – horizontal and vertical coordinates of computed point of gaze with respect to the head mounted scene camera 640 x 480 pixel field of view (fov), multiplied by 10. Convert to float and multiply by the scale factor 0.1 to get the pixel coordinate value. Note that fractional pixel positions are represented. Also note that the values are signed. Negative values represent positions to the left, or above the scene camera fov, while values greater than 640 or 480 represent positions to the right of , or below the camera fov.

hdtrk – X, Y, Z position values and azimuth, elevation, and roll orientation values received by the ETM ET3Space system from a head tracker. Values are multiplied by 100, and stored or transmitted as signed integers. Convert to float and multiply by the scale factor 0.1 to get

original values. Position values are in units of inches or centimeters (depending on which unit system was set in ET3Space configuration dialog). Angles are in degrees.

EH_scene_number – number of scene plane first intersected by line of gaze, as computed by the ET3Space feature.

EH_gaze_length – Distance from the eye to the point of gaze on the scene plane designated by EH_scene_number, as computed by the ET3Space feature. The value is recorded or transmitted as a single precision floating point value with units of either inches or centimeters (depending on which unit system was set in ET3Space configuration dialog).

EH_gaze_coord – The point of gaze in scene plane coordinates, on the scene plane designated by EH_scene_number, as computed by the ET3Space feature. Coordinates are with respect to the coordinate frame defined on the scene plane by the ET3Space environment specifications. The "horizontal" value is the Y scene plane coordinate, and the "vertical" value is the Z coordinate. The value is recorded or transmitted as a single precision floating point value with units of either inches or centimeters (depending on which unit system was set in ET3Space configuration dialog).

4 Sample Programs

Sample programs are provided to demonstrate communication with ETM from C++ or C#, applications. The examples include source code where appropriate. One set of C++ samples demonstrate very basic communicate with ETM, using just the Command Prompt window rather than a Graphic User Interface (GUI). C# and C++/MFC samples demonstrate slightly more sophisticated communication and include a GUI, and use a multi-threading library provided by Argus.

A ".Net" Com library, called ETNetLib.dll, is provided and the C# and C++/MFC samples use this library to implement communication. ETNetLib uses multi-threading, and the users application must support multi-threading if the library is used. All samples are normally installed as part of the ETRemote installation, and can be found in Program Files -> ArgusScience-> ETRemote -> ET_SDK_Samples.

4.1 ETNetLib.DLL

ETNetLib is a COM library that provides an interface to ETM. It can be used to send commands to ETM and to receive streaming data or video using a separate ETM data socket channel. It uses multi-threading and an application must support multi-threading to use the library.

For example, use "ConnectET" to open both an ETM command and data socket. Be sure to click "Listen" on the ETM Network configuration dialog before trying to connect. The library will initiate and maintain a thread to receive and buffer streaming data from ETM. Use "GetConnectStatus" to ensure that valid connections are open. Use "GetDataItemValue" to get the latest available sample of any data value being streamed from ETM. The value returned will be properly scaled and returned as the appropriate data type (unsigned integer, integer, or double). Use "SendETCmd" to send commands to ETM via the command channel.

See the sample C# and C++/MFC programs for detailed examples. Both the compiled ETNetLib.dll and the source code, with a complete Microsoft Visual Studio project, are provided.

4.2 C++

The C++ examples are executed by typing commands on the "Command Prompt" window (Start->All Programs->Accessories->Command Prompt). Results also display on the Command Prompt window. Source code and Visual Studio project files are included for all samples. Be sure that the external PC and ETM PC are connected to the same LAN. In all cases, if a command socket connection with ETM is not already open, be sure to click "Listen" on the ETM Network configuration dialog before running the sample program. All sample programs include a "readme" text file with instructions. The following sample programs are included.

- **ETGetDataItemValue**
  Demonstrate receiving single ETM data values on the command channel
  Demonstration of receiving single ETM data values on the command channel. Be sure that "Send Data Result" is checked on the ETM Network Configuration dialog. The command socket connection to ETM is used to return a single item value. The ETM IP address and port number as well as the desired data item are specified as command line arguments.

  Command: *ETGetDataItemValue IPAddress IPPort DataID*
  where IPAddress and IPPort are the address and port number shown on the ETM Network configuration dialog, and DataID is the ID number of the desired data item (see section 2, command 25).

- **ETReceiveData**
  Demonstrate receiving streaming data from ETM on a data channel
  Be sure that "Send Data Result" is checked on the ETM Network Configuration dialog. Both a command and a data socket connection to ETM are opened, and the data socket is used to receive streaming data. The ETM IP address and port number are specified in the command line. A single data record, containing all data items being reported by ETM, is displayed on the Command Prompt window before the ETM connection is closed. (Program can be modified to loop and receive multiple data records before closing.)

  Command: *ETReceiveData IPAddress IPPort*
  where IPAddress and IPPort are the address and port number shown on the ETM Network configuration dialog

- **ETReceiveImage**
  Demonstrate receiving image data from ETM on a data channel
  Be sure that "Send Video" is checked on the ETM Network Configuration dialog. Both a command and a data socket connection to ETM are opened, and the data socket is used to receive image data. The ETM IP address and port number as well as a jpg file name are specified in the command line. A single jpg image from the ETM display is read from ETM and stored in the specified file. The command and data socket connections are then closed.

  Command: *ETReceiveImage IPAddress IPPort ImageFileName*
  where IPAddress and IPPort are the address and port number shown on the ETM Network configuration dialog and ImageFileName is the name under which a jpg image file will be stored.

- **ETRecordData**
  Demonstrate control of ETM data files
  This sample program sends commands to ETM to name and open data files, and to start and stop recording data to the file. The ETM IP address and port number as well as a data file name are specified in the command line. An ETM command socket is opened. A command is sent to ETM to open a file with the specified name, and then

and a command is sent to begin recording. The ETM should show that a file has been opened and is recording. When the <Esc> key is pressed, on the PC running the ETRecordData sample program, a command will be sent to ETM to Stop recording, close the data file, and close the command connection.

Command: *ETRecordData IPAddress IPPort FileName*
where IPAddress and IPPort are the address and port number shown on the ETM Network configuration dialog and IPPortFileName is the name of the data file to be opened on ETM.

- **ETSetXDAT**
  Demonstrate setting XDAT values on ETM
  The ETM IP address and port number as well as the desired XDAT value are specified in the command line. An ETM command socket is opened, and a command is sent to set the specified XDAT value. The XDAT value should appear on the ETM Screen.

  Command: *ETRecordData IPAddress IPPort FileName*
  where IPAddress and IPPort are the address and port number shown on the ETM Network configuration dialog, and XDAT is an integer between 0 and 65535. The command socket connection is then closed.

4.3 C#

The C# sample programs use the ETNetLib COM Object library and demonstrate receiving realtime data, controlling ETM data files, and sending XDAT values to ETM. ETNetLib must be "registered" on the PC running the sample C# programs. If EyeTRACRemote was installed on the PC, this will have been done by the install program. Otherwise use the regsvr32 command to register the dll.

These samples include simple GUIs rather than requiring use of the Command Prompt window. Source code and Visual Studio project files are included for all samples. Be sure that the external PC and ETM PC are connected to the same LAN. In all cases, if a command socket connection with ETM is not already open, be sure to click "Listen" on the ETM Network configuration dialog before running the sample program. All sample programs include a "readme" text file with instructions. The following sample programs are included.

- **ETReceiveData**
  Demonstrate receiving data on a streaming data channel
  Be sure that "Send Data Result" is checked on the ETM Network Configuration dialog. To run the program, double click ETReceiveData.exe in the usual way. On the resulting program window, enter the ETM ip address and port number shown on the ETM Network configuration dialog. Click the "Connect" button to establish a connection to ETM. The program will attempt to open both a command and data socket connection to ETM. If successful, the Connect button will change to a "Disconnect" button. Click the "Receive Data" button to display the most recent data frame being sent over the data channel. Click again to display

another frame. Examine the ETReceiveDataForm.cs file code to see the calls to ETNetLib functions.

- **ETReceiveImage**
  Demonstrate receiving Image data and digital data on a streaming data channel
  Be sure that "Send Data Result" and "Send Video" are checked on the ETM Network Configuration dialog. To run the program, double click ETReceiveImage.exe in the usual way. On the resulting program window, enter the ETM ip address and port number shown on the ETM Network configuration dialog. Click the "Connect" button to establish a connection to ETM. The program will attempt to open both a command and data socket connection to ETM. If successful, the Connect button will change to a "Disconnect" button. Click the "Receive Image" button to display the latest ETM display, and the lastest data frame being sent over the data channel. Look at the ETReceiveImageForm.cs file code to see the calls to ETNetLib functions.

- **ETRecordData**
  Demonstrate control of ETM data files using the ETM command socket channel
  To run the program, double click ETReceiveImage.exe in the usual way. On the resulting program window, enter the ETM IP address and port number shown on the ETM Network Configuration dialog. Click the "Connect" button to establish a command socket connection to ETM. If successful, the Connect button will change to a "Disconnect" button. Type in a file name to be used as the name of the ETM data file, then click "Set File Name". Click the "Open Host File". The file name should appear as the "Track Log" "Video Recording" name on the ETM screen. Use the "Start Recording" and "Stop Recording" buttons to cause recording to start and stop on ETM. This should be properly indicated on the ETM screen. Examine the ETRecordDataForm.cs file code to see the calls to ETNetLib functions.

- **ETSetXDAT**
  Demonstrate setting ETM XDAT values using the ETM command socket channel
  To run the program, double click ETSetXDAT.exe in the usual way. On the resulting program window, enter the ETM IP address and port number shown on the ETM Network configuration dialog. Click the "Connect" button to establish a command socket connection to ETM. If successful, the Connect button will change to a "Disconnect" button. Type in the desired XDAT value and click the "Set" button. The XDAT value should appear on the ETM Screen. Examine the ETSetXDATForm.cs file code to see the calls to ETNetLib functions.

4.4 C++/MFC

A set of 4 sample programs are provided using C++ and using the Micorsoft Foundation Class (MFC) library to generate a GUI. These are identical in program name and functionality to the C# samples, and also use the ETNetLib COM Object library.

5 Communication with Paradigm

Paradigm with the *Paradigm elements for ASL* add-on package, from Perception Research Sytems Inc., is an application designed to facilitate building precisely timed visual display experiments and collecting subject responses. Experiments can be built with a drag and drop designer. A simple script event allows insertion of Python script if needed. The *Paradigm elements for ASL* add-on package provides drag and drop elements which can communicate with Argus ETM, including control of data recording, and setting XDAT values on ETM.

Note that analysis of gaze data from visual display experiments, presented on a stimulus PC monitor, is usually most efficient if a separate head tracker and the optional ET3Space feature is used with ETM; or alternately, if ETAnalysis "Stimulus Tracking" feature is used. In both cases ETRemote (or another user created application) should be used to synchronously record the stimulus screen presentation. Both ET3Space and Stimulus Tracking are described in separate manuals.

To design an experiment that will communicate with ETM, first open a new experiment. Under Devices, select the Device Manager, and highlight "Network Port". Check the "NetworkPort1" check box. Next to "IP Address", enter the IP address shown on the ETM Network Configuration dialog. Click on the Experiment Bar to open the network port. Select the "ASL" tab on the left pane of the Paradigm window, and drag the "Begin Session" element to the "Commands" pane near the bottom right of the Paradigm window. A small dialog window will appear with a drop down menu labeled "XDAT Port Type". Select "EyeTRAC7 port" from the drop down menu, and click "Done".

Follow instructions in the Paradigm documentation to design an experiment using drag and drop events and other Paradigm features. For example, command ETM to start recording at any event by dragging the ASL tab "Start Recording" element to the Command pane. Set an XDAT value during any event by dragging the ASL tab "Send Marker" element to the Command pane and entering the desired XDAT value. See Paradigm documentation for additional features and more detailed instructions.

6 Communication with E-Prime

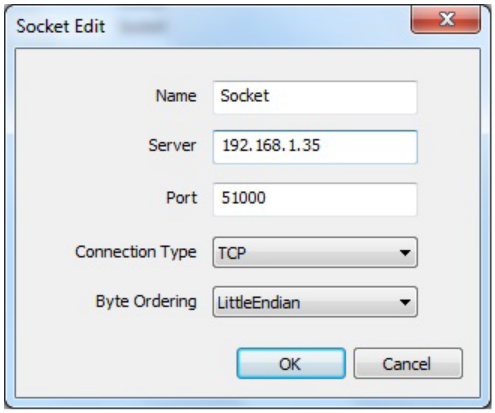
E-Prime, from Psychology Software Tools, Inc., is an application for designing visual display experiments. It includes a programming language called E-Basic as well as graphical drag and drop features. *As noted in the previous section, analysis of gaze data with respect to stimulus screen displays is usually most efficient when optional ET3Space or ETAnalysis Stimulus Tracking features are used.*

Eprime version 2.0 Professional supports network TCIP sockets and can communicate with the ETM. Argus Science provides an E-Basic library script with subroutines that can be called from E-Prime "in-line" script objects. The library routines support control of ETM data recording functions, sending "XDAT" values to ETM, and reading real-time gaze data from ETM. Sample scripts are also provided to demonstrate control of ETM data files, setting XDAT values, and reading real time gaze values.

The library script and samples are normally installed as part of the ETRemote installation, and can be found in Program Files->ArgusScience-> ETRemote-> ET_SDK_Samples->Eprime.

When starting a new experiment design on E-Studio, double click the "Experiment Object" node on the "Structure" diagram to bring up the Properties dialog. Select the Devices tab. If "Socket" is listed, be sure it is checked. If "Socket" is not listed as one of the devices, click the "Add" button, highlight "Socket" and click OK. Double click "Socket" on the Properties, Devices tab, to bring up the Socket Edit dialog.

On the socket edit dialog, set "Name" to "Socket"; set "Server:" to the IP address shown on the ETM Network Configuration dialog; and set "Port", "Connection Type", and "Byte Ordering" as shown in the screen shot below.



Open Argus_ETServer_Eprime_Library.txt in note pad, and use copy/paste to past the text into the E-Studio User Script Window. The library subroutines can now be called from E-Prime "In-line Objects" .

The library subroutines as well as the sample programs that use them are designed to be very short and simple. A user with experience in E-Basic programming can customize these routines to best suit their specific task.

Functionality of each "library" subroutine is described by comment lines included with the code. The Subroutines are:

- ETServer_OpenDataFile
- ETServer_CloseDataFile
- ETServer_StartDataFileRecording
- ETServer_StopDataFileRecording
- SendXDATtoETServer(XDAT As Long)
- ETServer_GetRealTimeFloatData (DataID As Integer, Value As Single, status As Integer)
- ETServer_GetRealTimeIntegerData (DataID As Integer, Value As Integer, status As Integer)

To run one of the sample scripts, select "Open" from the E-Studio File menu, browse to the desired *.es2 file, and click Open. It will not be necessary to copy text from the Argus_ETServer_Eprime_Library, since that will already be part of the sample script. It will, however, be necessary to open the Socket Edit menu, and set "Server" to the ETM IP address, as previously described. Make sure other items on the Socket Edit dialog are also set as previously shown. If a command channel is not already opened on ETM, click the "Listen" button on the ETM Network Configuration dialog before running the script. The following sample scripts are included:

ETServer_Sample_FileOpen_Rec_SendXDAT_FileClose.es2

The sample program will first set XDAT to 0 (call to "SendXDATtoETServer" subroutine), send a command to ETM to open a file (call to "ETServer_OpenDataFile"), and will display a message explaining the demonstration and prompting for <SpaceBar> to start. The opened data file should show as ready to record on the ETM screen.

When <SpaceBar> is pressed E-Prime will send a command to ETM to Start recording on the data file (call to "ETServer_StartDataFileRecording"), and the ETM screen should show that the file is recording.

E-Prime will then sequentially display a set of 6 screens, each with a different background color and a number in the center. Each time a new screen is displayed XDAT will be set to the number shown on the screen. This should be visible on the ETM screen. Each screen will display for several seconds before proceeding to the next display. After all 6 have been displayed, XDAT will be set back to zero ("SendXDATtoETServer"), the file will stop recording ("ETServer_StopDataFileRecording"), and will close ("ETServer_CloseDataFile").

ETServer_Sample_ReadRTData.es2

The sample program will first display an instruction screen. When <SpaceBar> is pressed, E-Prime will loop though ETM commands to send horizontal and vertical gaze data over the command channel ("ETServer_GetRealTimeFloatData" subroutine), and to update the E-Prime display with these values.

7 Communication with MatLab

MatLab, when also equipped with the "Instrument Control Toolbox", can open TCP sockets, and can communicate with Argus ETM systems. Call the tcpip function, with a device IP address and port number as arguments, to create a tcpip object. The ETM IP address and port number can be found on the ETM Network Configuration dialog. For example, create the tcp object *t* , with IP address 192.168.1.35 and port 51000, as follows:

t = tcpip_pc ('192.168.1.35', 51000)

Use fopen to connect. For example:

fopnen(t)

(Note that the "Listen" button on the ETM Network Configuration dialog must be pressed for a connection to be established.) Remember to close the connection and delete the tcpip object at the end of the program. For example:

fclose(t)
delete(t)

A set of function subroutines is provided to send various commands to ETM. Each is in a separate file, with a file name that is the same as the function, and ".m" extension. Each assumes that a tcpip object has already been created and connected with ET7as the ET7 command socket. The tcpip object is the first argument in each function. There are separate functions to send commands for ET7 file open, file close, start and stop recording, set XDAT, and to command ET7 to send the latest value of a data item over the command socket. The latter two functions require a second argument, to specify the desired XDAT value in the first case, and to specify the DataID of the desired data item, in the second case. XDAT must be an integer between 0 and 65535. DataID must be an integer corresponding to one of the data ID values listed in section three, under command 25. The functions are:

- ETServer_OpenDataFile(t)
- ETServer_CloseDataFile(t)
- ETServer_StartDataFileRecording(t)
- ETServer_StopDataFileRecording(t)
- ETServer_SendXDAT(t, XDAT)
- ETServer_GetDataItem(t, DataID)

3 sample Matlab scripts are provided, which call the functions discussed above, so the files containing these functions must be in the current Matlab directory or a directory on the Matlab path. Each sample starts by creating a tcp object and connecting to ETM, as discussed above. The Matlab PC and ETM PC must be connected to the same local area network. When running the sample, the user must first edit the line of code that sets the proper IP address, since this may be different for each system installation.

ETServer_Sample_SendXDAT.m
The sample program creates a TCP object, opens a command socket connection with ETM, pauses for 1 sec, sets XDAT to 500 on ETM by calling the ETServer_SendXDAT function, pauses for another seconds, and then closes the connection and deletes the TCP object. The user must edit the script to include the correct IP address for ETM (as shown on the ET7 Network Configuration dialog), and to change the XDAT value. Be sure that the "Listen" button has been clicked, on the ETM Network configuration dialog, before running the script. The XDAT values set by the sample should appear on the ETM screen.

ETServer_Sample_File_and_XDAT_Commands.m
The sample program creates a TCP object, opens a command socket connection with ETM, pauses for 1 sec, then sends a command to open a data file on ETM, a command to set XDAT=0, and a command to start recording on the opened file. At 5 sec intervals, commands are sent to set XDAT to 100, 150, and 200. After another 5 sec interval, commands are sent to close the connection and delete the TCP object.

ETServer_Sample_Read_RealTime_Data.m
The sample program creates a TCP object, opens a command socket connection with ETM, and pauses for 1 sec. The program then clears the input buffer, and sends a command to ETM to send the latest horizontal gaze position value over the command channel. It then reads the first 40 bytes of the response into an input buffer, and reads a float value starting at the 41st byte into a horizontal gaze position variable. The command and read sequence is repeated for vertical gaze and pupil diameter values. Finally the input buffer is cleared and commands are sent to close the connection and delete the TCP object. Note that the sample program does not check the ET7 response to see if the ASL signature is correct, to see if the data ID on the return value is correct, or to see if the checksum on the return value is correct. This type of error checking can be included if desired. As with the other sample scripts, be sure that the "Listen" button has been clicked, on the ETM Network configuration dialog, before running the script.