# **CORIO2 Universal Scaler**

# **User Control Specification - Protocol 2**

Compatible with: C2-7000 Series units

Release 0.1 Beta\*

Last edit date: 10th October 2005

#### **Brief**

This control specification for protocol 2 outlines how to control a CORIO2 Universal Scaler unit via an RS232 link, using ASCII-based commands, currently only the C2-7000 series of unit supports this protocol (requiring firmware version 30 or above). It details how to send and receive serial data to perform many of the functions that a user has access to on the front panel.

(Please note that this specification applies ONLY to those units bearing the CORIO2 logo.)

Web: <a href="http://www.tvone.co.uk">http://www.tvone.co.uk</a>
Email: tech.europe@tvone.com

All the latest updates are at <a href="http://www.tvone.com/support">http://www.tvone.com/support</a> All trademarks and copyrights are acknowledged. E&OE.

### Disclaimer / limited warranty

No warranty is made either expressed or implied including but not limited to any implied warranties of merchantability or fitness for a particular purpose. In no event shall the manufacturer and/or any supplier of the CORIO2 product be liable for errors found within, or be liable for any direct indirect or consequential damages or loss in connection with the purchase or use of the hardware software or manual. The sole and exclusive liability regardless of the form of action shall not exceed the replacement cost of the CORIO2 materials.

By using this specification you have indicated that you have agreed to the terms listed above.

### Revision history

0.1 Beta release – \*The following functions will become available in the next release

Reading and Writing of the button functions

Reading Writing and Execution of macro functions

#### Communication standard

You should read this specification in conjunction with the User Manual.

Packets of ASCII data containing hexadecimal numbers are exchanged between the CORIO2 unit and controller via an RS232 link.

The RS232 standard is 57600 baud, 8 bits, no parity and 1 stop bit.

No flow control is used - however all control packets start with and ASCII 'F', end with carriage-return (13 decimal) and all such packets sent to the CORIO2 unit will be acknowledged (thereby provided software handshaking).

It may take around 20ms (0.02 seconds) for an RS232 command to be actioned and acknowledged.

ASCII-hex data is used where a number is encoded into its hexadecimal equivalent with leading zeros - eg. Where '00' is decimal value 0, '80' is decimal 128 and 'FF' is decimal 255.

Any gap of more than 1 second between the characters of a control command sent using the RS232 port will cause a time-out - and previous characters sent will be lost.

Write packets (sending command functions to the CORIO2 unit) are always 20 characters long (including a carriage return (EOP) at the end). The CORIO2 unit will respond with a full 20 character message indicating what has changed. This returned payload will reflect the actual value of the parameter changed, if the user requests a value out of bounds then the limit value is used, and the payload will then reflect the limit value used.

Read packets (sent to request information from the CORIO2 unit) are always 14 characters long (including a carriage return at the end), the response from the scaler will be a 20 byte message with the Write flag (since it is 'writing' the value back to the host) and the ACK flag set.

The ACK flag will be returned as 0 if the command is invalid for some reason – for example a bad FUNCTION, WINDOW, OUTPUT or PAYLOAD value. An ACK=0 message will be otherwise identical to the one you sent, so you know exactly which message has the error.

Any changes made to the unit using the front panel controls will also cause the full 20 byte message to be sent indicating the change that has occurred, thus enabling a program to stay 'in-sync' with the unit. In some cases (such as the execution of a macro) multiple 20 bytes messages will be sent indicating all the parameters that have been changed.

Only one message can be sent to the unit, another message can't be sent until a specific response is received from the scaler (the user should look for a message with the same WINDOW, OUTPUT and FUNCTION values as they sent). If no message is received back within 1 second, there is likely to be a hardware communication problems (or wrong baud rate, etc.).

## Packet format

Below is a representation of data packets sent / received to / from the CORIO2 unit:

The table below details the function of each part of the packet:

SOP This is always the ASCII letter if to indicate the packet start.  ASCII-hex byte to indicate the type of command being sent. Each bit in the byte has a different function. Currently only the following bits are defined: Bit 7 = Write (i) or Read (i) request. Messages from the scaler are always Writes. Bit 6 = AGK bit. Should be set to 10 for messages to the scaler. ACK=1 returned means message was okay. ACK=0 returned means an error was present in the message. Bit 5 = 0 Reserved for future use. Bit 4 = 0 Reserved for future use. Bit 3 = 0 Reserved for future use. Bit 1 = 0 Reserved for future use. Bit 0 = 0 Reserved for future use. DHA  This byte has multiple uses.  SOURCE  ON THE FORD AND ASSOCIATE ASSO	Packet part	Function										
ASCII-hex byte to indicate the type of command being sert. Each bit in the byte has a different function. Currently only the following bits are defined: Bit 7 = Write (0) or Read (1) request. Messages from the scaler are always Writes. Bit 6 = ACK bit. Should be set to 0 for messages to the scaler. ACK=1 returned means message was okay. ACK=0 returned means an error was present in the message. Bit 5 = 0 Reserved for future use. Bit 3 = 0 Reserved for future use. Bit 3 = 0 Reserved for future use. Bit 2 = 1 This bit "must" be set for this protocol (Protocol 2). Bit 1 = 0 Reserved for future use. Bit 0 = 0 Reserved for future use. Byte to indicate the source channel to be altered (if appropriate). This byte has multiple uses.  SOURCE When Function number is within the Adjust Sources section (see later): This byte should be set to 0 if the command does not relate to an input-specific parameter.  Ox10 = REBI, 0x11 = REBIQ, 0x12 = REBIQ Ox30 = CV1, 0x31 = CV2, 0x32 = CV3 Ox40 = YC1, 0x41 = YC2, 0x42 = YC3 Ox50 = SDI1, 0x51 = SDI2 OxD0 = OUT1, 0xD1 = OUT2 OxF0 = TC1, 0xF1 = TC2  Or - for Macro related commands: Bit 7.4 = 0 Reserved Bit 3.0 = Reserved (set to 0). Bit 5.0 = Represents the window to be adjusted. Byte of the function code. (Remainder [7.0] are in FUNC LOW) Eg. If the function code is 0x234, and we want to adjust Output 2, then this byte is 0x12 ASCII-hax byte to indicate the lowest 8 bits of the actual function to set or receive (e.g. change Program source). A later table details all the functions available, and each one will normally relate to an option on the CORIO2.  PAYLOAD A Series of ASCII-hex bytes carrying the data to send. A SciI-hax byte to indicate the lowest 8 bits of the actual function to set or receive (e.g. change Program source). A later table details all the function of the payloa												
Each bit in the byte has a different function. Currently only the following bits are defined: Bit 7 = Write (b) or Read (1) request. Messages from the scaler are always Writes. Bit 6 = ACK bit. Should be set to 0 for messages to the scaler. ACK=1 returned means message was okay. ACK=0 returned means an error was present in the message. Bit 5 = 0 Reserved for future use. Bit 3 = 0 Reserved for future use. Bit 2 = 1 This bit "must" be set for this protocol (Protocol 2). Bit 1 = 0 Reserved for future use. Bit 2 = 0 Reserved for future use. Bit 0 = 0 Reserved for future use. Byte to indicate the source channel to be altered (if appropriate).  When Function number is within the Adjust Sources section (see later):  When Function number is within the Adjust Sources section (see later):  When Function number is within the Adjust Sources section (see later):  When Function number is within the Adjust Sources section (see later):  When Function number is within the Adjust Sources section (see later):  When Function number is within the Adjust Sources section (see later):  When Function number is within the Adjust Sources section (see later):  Whon Sources (and the Adjust Sources) and the Adjust Sources (and Sources) and th												
Bit 7 = Write (0) or Read (1) request. Messages from the scaler are always Writes. Bit 6 = ACK bit. Should be set to 0 for messages to the scaler. ACK=1 returned means message was okay. ACK=0 returned means an error was present in the message. Bit 3 = 0 Reserved for future use. Bit 4 = 0 Reserved for future use. Bit 2 = 1 This bit "must" be set for this protocol (Protocol 2). Bit 1 = 0 Reserved for future use. Bit 0 = 0 Reserved for future use. CHA  This byte has multiple uses.  When Function number is within the Adjust Sources section (see later):  Official of the source channel to be altered (if appropriate).  When Function number is within the Adjust Sources section (see later):  Official of the source channel to be altered (if appropriate).  This byte should be set to 0 if the command does not relate to an input-specific parameter.  Official of the source of th	CIVID											
Bit 6 = ACK bit. Should be set to 0 for messages to the scaler. ACK=1 returned means message was okay. ACK=0 returned means an error was present in the message.  Bit 5 = 0 Reserved for future use. Bit 5 = 0 Reserved for future use. Bit 2 = 1 Reserved for future use. Bit 2 = 1 This bit "must" be set for this protocol (Protocol 2). Bit 1 = 0 Reserved for future use. Bit 0 = 0 Reserved for future use. Byte to indicate the source channel to be altered (if appropriate).  SOURCE When Function number is within the Adjust Sources section (see later):  WACRO Byte to indicate the source channel to be altered (if appropriate).  This byte should be set to 0 if the command does not relate to an input-specific parameter.  0x10 = RGB1, 0x11 = RGB2, 0x12 = RGB3 0x30 = Cv1, 0x31 = Cv2, 0x32 = Cv3 0x30 = Cv1, 0x31 = Cv2, 0x32 = Cv3 0x30 = Cv1, 0x31 = Cv2, 0x32 = Cv3 0x50 = SDI1, 0x51 = SDI2 0x50 = SDI1, 0x51 = CV12 0xF0 = TC1, 0xF1 = TC2  Or - for Macro related commands: Bit 7.3 = 0 Reserved Bit 3.0 = Macro number  Or - when not used (for the majority of commands) Bit 7.3 = 0 Reserved (set to 0). Bit 7.3 = 0 Reserved (set to 0). Bit 7.3 = 0 Reserved (set to 0). Bit 7.3 = Number representing the window to adjust 0 = Output 1, 1 = Output 2. Bit 5.2 = Reserved (set to 0). Bit 1.3 = Bit 9.8 8 of the function code. (Remainder [7.0] are in FUNC LOW.) Bit 1.3 = Bit 9.8 8 of the function code (Remainder [7.0] are in FUNC LOW.) Bit 1.3 = Bit 9.8 8 of the function code is 0x34, and we want to adjust Output 2, then this byte is 0x12  FUNC HIGH A SCII-hex byte to indicate the lowest 8 bits of the actual funct												
okay. ACK=0 returned means an error was present in the message. Bit 5 = 0 Reserved for future use. Bit 3 = 0 Reserved for future use. Bit 3 = 0 Reserved for future use. Bit 2 = 1 This bit "must" be set for this protocol (Protocol 2). Bit 1 = 0 Reserved for future use. Bit 0 = 0 Reserved for future use. Bit 0 = 0 Reserved for future use. Bit 1 = 0 Reserved for future use. Bit 1 = 0 Reserved for future use. Bit 0 = 0 Reserved for futu												
Bit 5 = 0 Reserved for future use. Bit 4 = 0 Reserved for future use. Bit 3 = 0 Reserved for future use. Bit 3 = 0 Reserved for future use. Bit 1 = 0 Reserved for future use. Bit 1 = 0 Reserved for future use. Bit 1 = 0 Reserved for future use. Bit 0 = 0 Reserved for future use.  When Function number is within the Adjust Sources section (see later):  When Function number is within the Adjust Sources section (see later):  When Function number is within the Adjust Sources section (see later):  When Function number is within the Adjust Sources section (see later):  When Function number is within the Adjust Sources section (see later):  When Function number is within the Adjust Sources section (see later):  When Function number is within the Adjust Sources section (see later):  When Function number is within the Adjust Sources section (see later):  When Function number is within the Adjust Sources section (see later):  When Function later section (see later):  When Function later section section (see later):  When Function later section sect												
Bit 4 = 0 Reserved for future use. Bit 3 = 0 Reserved for future use. Bit 2 = 1 This bit "must" be set for this protocol (Protocol 2). Bit 1 = 0 Reserved for future use. Bit 0 = 0 Reserved for future use. Bit 1 = 0 Reserved for future use. Bit 1 = 0 Reserved for future use. Bit 1 = 0 Reserved for future use. Bit 0 = 0 Reserved for future use. Bit 0 = 0 Reserved for future use. Bit 1 = 0 Reserved for future use. Bit 0 = 0 Reserved for future use. Bit 3 = 0 Reserved for future use. Bit 3 = 0 Reserved for future use. Bit 3 = 0 Reserved for future use. Bit 7 = Reserved (for the majority of commands) Bit 7 = 0 Reserved for future use. Bit 7 = Reserved (set to 0). Bit 6 = 0 Represents the window to be adjusted. E.g. Window 'A' is sent as '41' since 0x41 is ASCII for 'A'. 0x61 is ASCII for 'a' and is sent as '61'. DUTPUT Bit 3 = 2 Reserved (set to 0). Bit 7 = Reserved (set to 0). Bit 7 = Bit 7 = 8 A Box future use for future use for future use. Bit 3 = Bit 7 = Reserved (set to 0). Bit 1 = Bit 7 = Reserved (set to 0). Bi												
Bit 3 = 0 Reserved for future use. Bit 2 = 1 This bit "must" be set for this protocol (Protocol 2). Bit 1 = 0 Reserved for future use. Bit 0 = 0 Reserved for future use. CHA  This byte has multiple uses.  SOURCE  When Function number is within the Adjust Sources section (see later):  When Function number is within the Adjust Sources section (see later):  When Function number is within the Adjust Sources section (see later):  When Function number is within the Adjust Sources section (see later):  When Function number is within the Adjust Sources section (see later):  When Function number is within the Adjust Sources section (see later):  When Function number is within the Adjust Sources section (see later):  Whin Sources (Not 1 = RGB2, 0x12 = RGB3) 0x30 = CV1, 0x31 = CV2, 0x32 = CV3 0x30 = CV1, 0x31 = CV2, 0x32 = CV3 0x30 = CV1, 0x31 = CV2, 0x32 = CV3 0x30 = CV1, 0x31 = CV2, 0x32 = CV3 0x50 = SDI1, 0x51 = SDI2 0x50 = TC1, 0xF1 = TC2  Or – for Macro related commands: Bit 7.4 = 0 Reserved Bit 3.0 = Macro number  Or – when not used (for the majority of commands) Bit 7.4 = 0 Reserved Bit 7.4 = 0 Reserved Bit 7.4 = 0 Reserved (set to 0).  Bit 6.0 = Represents the window to be adjusted. E.g. Window 'A' is sent as '41' since 0x41 is ASCII for 'A'. 0x61 is ASCII for 'a' and is sent as '61'.  Bit 7.4 = Sumber representing the window to adjust 0 = Output 1, 1 = Output 2. Bit 7.4 = Number representing the window to adjust 0 = Output 1, 1 = Output 2. Bit 7.4 = Number representing the window to adjust 0 = Output 1, 1 = Output 2. Bit 7.4 = Number representing the window to adjust 0 = Output 1, 1 = Output 2. Bit 7.4 = Number representing the window to adjust 0 = Output 1, 1 = Output 2. Bit 7.4 = Number representing the window to adjust 0 = Output 1, 1 = Output 2. Bit 7.4 = Number representing the window to adjust 0 = Output 1, 1 = Output 2. Bit 7.4 = Number representing the window to adjust 0 = Output 1, 1 = Output 2. Bit 7.4 = Number representing the window to adjust 0 = Output 1, 1 = Output 2. Bit 7.4 = Number representing t												
Bit 2 = 1 This bit "must" be set for this protocol (Protocol 2). Bit 1 = 0 Reserved for future use. Bit 0 = 0 Reserved for future use. Bit 0 = 0 Reserved for future use. Bit 0 = 0 Reserved for future use. This byte has multiple uses.  When Function number is within the Adjust Sources section (see later):  When Function number is within the Adjust Sources section (see later):  Byte to indicate the source channel to be altered (if appropriate). This byte should be set to 0 if the command does not relate to an input-specific parameter.  0x10 = R8F1, 0x11 = R6B2, 0x12 = RGB3 0x30 = CV1, 0x31 = CV2, 0x32 = CV3 0x40 = YC1, 0x41 = YC2, 0x42 = YC3 0x50 = SDI1, 0x51 = SDI2 0xD0 = OUT1, 0x51 = SDI2 0xD0 = OUT1, 0x51 = TC2  Or - for Macro related commands: Bit 7.4 = 0 Reserved Bit 30 = Macro number  Or - when not used (for the majority of commands) Bit 7.3 = 0 Reserved Bit 7.4 = 0 Reserved Bit 7.4 = 0 Reserved (set to 0). Bit 6.0 = Represents the window to be adjusted. E.g. Window 'A' is sent as '41' since 0x41 is ASCII for 'A'. 0x61 is ASCII for 'a' and is sent as '61'.  BUTPUT Bit 7.4 = Number representing the window to adjust 0 = Output 1, 1 = Output 2. Bit 3.2 = Reserved (set to 0). Bit 3.2 = Reserved (set to 0). Bit 3.2 = Reserved (set to 0). Bit 3.3 = Reserved (set to 0). ASCII-hex byte to indicate the lowest 8 bits of the actual function to set or receive (e.g. change Program source). A later table details all the functions available, and each one will normally relate to an option on the CORIO2.  PAYLOAD A Series of ASCII-hex bytes carrying the data to send.  Read requests have no payload - the payload is in the data sent back. Write packets require a payload, and this is always in 'triple-bytes' - ie, 3 bytes are required, MSB first. eg. '000001' is 1 in decimal, '010000' is 65536 in decimal, and 'FFFFFO' is -16 in decimal.  CS ASCII-hex byte that is the (check) sum of all previous bytes (excluding the SOP 'F' character). Eg. The command fo400410082000001'?? Instead. This is purely for test and debugging - you s												
Bit 1 = 0 Reserved for future use. Bit 0 = 0 Reserved for future use. This byte has multiple uses.  When Function number is within the Adjust Sources section (see later): What Sources When Function number is within the Adjust Sources section (see later): What Sources When Function number is within the Adjust Sources section (see later): What Sources When Function number is within the Adjust Sources section (see later): What Sources When Function number is within the Adjust Sources section (see later): What Sources When Function number is within the Adjust Sources section (see later): What Sources When Function number is within the Adjust Sources section (see later): What Sources When Function number is within the Adjust Sources section (see later): What Sources When Function number is within the Adjust Sources section (see later): What Sources What Sources When Function number is within the Adjust Sources section (see later): What Sources What Sources When Function number is within the Adjust Sources section (see later): What Sources What Sources When Function section of if the command does not relate to an input-specific parameter.  What Sources What Sources What Sources What Sources What Sources When Function section of the Adjust Sources What												
Bit 0 = 0 Reserved for future use.  This byte has multiple uses.  When Function number is within the Adjust Sources section (see later):  When Function number is within the Adjust Sources section (see later):  When Function number is within the Adjust Sources section (see later):  When Function number is within the Adjust Sources section (see later):  When Function number is within the Adjust Sources section (see later):  When Function number is within the Adjust Sources section (see later):  When Function number is within the Adjust Sources section (see later):  When Function number and source channel to be altered (if appropriate).  This byte should be set to 0 if the command does not relate to an input-specific parameter.  0x10 = RGB1, 0x11 = RGB2, 0x12 = RGB3 0x30 = CV1, 0x31 = CV2, 0x32 = CV3 0x40 = VC1, 0x41 = CV2, 0x42 = VC3 0x40 = VC1, 0x41 = CV2, 0x42 = VC3 0x50 = SDI1, 0x51 = SDI2 0xD0 = OUT1, 0x51 = SDI2 0xD0 = OUT1, 0x51 = SDI2 0xF0 = TC1, 0xF1 = TC2  Or - for Macro related commands: Bit 7.4 = 0 Reserved Bit 30 = Macro number  Or - when not used (for the majority of commands) Bit 7.4 = 0 Reserved (set to 0). Bit 6.0 = Represents the window to be adjusted. E.g. Window 'A' is sent as '41' since 0x41 is ASCII for 'A'. 0x61 is ASCII for 'a' and is sent as '61'.  Bit 7.4 = Number representing the window to adjust 0 = Output 1, 1 = Output 2.  Bit 3.2 = Reserved (set to 0).  Bit 1.0 = Bits 9 & 8 of the function code. (Remainder [7.0] are in FUNC LOW.) Eg. If the function code is 0x234, and we want to adjust Output 2, then this byte is 0x12  FUNC LOW  ASCII-hex byte to indicate the lowest 8 bits of the actual function to set or receive (e.g. change Program source).  A later table details all the functions available, and each one will normally relate to an option on the CORIO2.  A Series of ASCII-hex bytes carrying the data to send.  Read requests have no payload - the payload is in the data sent back.  Write packets require a payload, and this is always in 'triple-bytes' - ie. 3 bytes are required, MSB fi		· · · · · ·										
This byte has multiple uses.  When Function number is within the Adjust Sources section (see later):  When Function number is within the Adjust Sources section (see later):  When Function number is within the Adjust Sources section (see later):  When Function number is within the Adjust Sources section (see later):  When Function number is within the Adjust Sources section (see later):  When Function number is within the Adjust Sources section (see later):  When Function number is within the Adjust Sources section (see later):  When Function number is within the Adjust Sources section (see later):  When Function number is within the Adjust Sources section (see later):  When Function number is within the Adjust Sources section (see later):  When Function number is within the Adjust Sources section (see later):  When Function number is within the Adjust Sources section (see later):  When Function number is within the Adjust Sources section (see later):  When Function number is within the Adjust Sources section (see later):  When Function number is within the Adjust Sources section (see later):  When Function number is within the Adjust Sources section (see later):  When Function number is within the Adjust Sources section input se												
When Function number is within the Adjust Sources section (see later):  When Function number is within the Adjust Sources section (see later):  When Function number is within the Adjust Sources section (see later):  Byte to indicate the source channel to be altered (if appropriate).  This byte should be set to 0 if the command does not relate to an input-specific parameter.  0x10 = RGB1, 0x11 = RGB2, 0x12 = RGB3 0x30 = CV1, 0x31 = CV2, 0x32 = CV3 0x40 = VC1, 0x41 = CV2, 0x32 = CV3 0x50 = SDI1, 0x51 = SDI2 0xD0 = OUT1, 0xD1 = OUT2 0xF0 = TC1, 0xF1 = TC2  Or – for Macro related commands:  Bit 7.4 = 0 Reserved Bit 3.0 = Macro number  Or – when not used (for the majority of commands) Bit 7.0 = 0  WINDOW/ Bit 6.0 = Represents the window to be adjusted. E.g. Window 'A is sent as '41' since 0x41 is ASCII for 'A'. 0x61 is ASCII for 'a' and is sent as '61'.  DITPUT  Bit 7.4 = Number representing the window to adjust 0 = Output 1, 1 = Output 2.  Bit 3.2 = Reserved (set to 0).  Bit 1.0 = Bits 9 & 8 of the function code. (Remainder [7.0] are in FUNC LOW.) Eg. If the function code is 0x234, and we want to adjust Output 2, then this byte is 0x12  FUNC LOW  ASCII-hex byte to indicate the lowest 8 bits of the actual function to set or receive (e.g. change Program source).  A later table details all the functions available, and each one will normally relate to an option on the CORIO2.  PAYLOAD  A series of ASCII-hex bytes carrying the data to send.  Read requests have no payload, and this is always in 'triple-bytes' - ie. 3 bytes are required, MSB first. eg. 000001' is 1 in decimal, '010000' is 65536 in decimal, and 'FFFFFO' is -16 in decimal.  ASCII-hex byte that is the (check) sum of all previous bytes (excluding the SOP 'F' character). Eg. The command F0400410082000001 has the checksum of 04+00+41+00+82+00+00+01=C8, so the complete command to send is F0400410082000001 has the checksum of 04+00+41+00+82+00+00+01=C8, so the complete command to send is F0400410082000001 has the splaced. This is purely for test and debugging - y	СНА											
Byte to indicate the source channel to be altered (if appropriate).  This byte should be set to 0 if the command does not relate to an input-specific parameter.  0x10 = RGB1, 0x11 = RGB2, 0x12 = RGB3 0x30 = CV1, 0x31 = CV2, 0x32 = CV3 0x40 = YC1, 0x41 = YC2, 0x42 = YC3 0x50 = SDI1, 0x51 = SDI2 0xD0 = OUT1, 0xD1 = OUT2 0xF0 = TC1, 0xF1 = TC2  Or – for Macro related commands: Bit 74 = 0 Reserved Bit 30 = Macro number  Or – when not used (for the majority of commands) Bit 70 = 0  WINDOW/ DGO / Bit 60 = Represents the window to be adjusted. E.g. Window'A' is sent as '41' since 0x41 is ASCII for 'A'. 0x61 is ASCII for 'a' and is sent as '61'.  DUTPUT Bit 74 = Number representing the window to adjust 0 = Output 1, 1 = Output 2. Bit 32 = Reserved (set to 0). Bit 10 = Bits 9 & 8 of the function code. (Remainder [70] are in FUNC LOW.) Eg. If the function code is 0x234, and we want to adjust Output 2, then this byte is 0x12  FUNC LOW ASCII-hex byte to indicate the lowest 8 bits of the actual function to set or receive (e.g. change Program source).  A series of ASCII-hex bytes carrying the data to send. Read requests have no payload - the payload is in the data sent back. Write packets require a payload, and this is always in 'triple-bytes' - ie. 3 bytes are required, MSB first. eg. '000001' is 1 in decimal, '1010000' is 65536 in decimal, and 'FFFFFO' is -16 in decimal.  CS ASCII-hex byte that is the (check) sum of all previous bytes (excluding the SOP 'F' character). Eg. The command F0400410082000001 has the checksum of 04+00+41+00+82+00+00+01=C8, so the complete command to send is F0400410082000001?? Instead. This is purely for test and debugging - you should normally use a checksum to ensure data validity.	CHA	This byte has multiple uses.										
Byte to indicate the source channel to be altered (if appropriate).  This byte should be set to 0 if the command does not relate to an input-specific parameter.  0x10 = RGB1, 0x11 = RGB2, 0x12 = RGB3 0x30 = CV1, 0x31 = CV2, 0x32 = CV3 0x40 = YC1, 0x41 = YC2, 0x42 = YC3 0x50 = SDI1, 0x51 = SDI2 0xD0 = OUT1, 0xD1 = OUT2 0xF0 = TC1, 0xF1 = TC2  Or – for Macro related commands: Bit 74 = 0 Reserved Bit 30 = Macro number  Or – when not used (for the majority of commands) Bit 70 = 0  WINDOW/ DGO / Bit 60 = Represents the window to be adjusted. E.g. Window'A' is sent as '41' since 0x41 is ASCII for 'A'. 0x61 is ASCII for 'a' and is sent as '61'.  DUTPUT Bit 74 = Number representing the window to adjust 0 = Output 1, 1 = Output 2. Bit 32 = Reserved (set to 0). Bit 10 = Bits 9 & 8 of the function code. (Remainder [70] are in FUNC LOW.) Eg. If the function code is 0x234, and we want to adjust Output 2, then this byte is 0x12  FUNC LOW ASCII-hex byte to indicate the lowest 8 bits of the actual function to set or receive (e.g. change Program source).  A series of ASCII-hex bytes carrying the data to send. Read requests have no payload - the payload is in the data sent back. Write packets require a payload, and this is always in 'triple-bytes' - ie. 3 bytes are required, MSB first. eg. '000001' is 1 in decimal, '1010000' is 65536 in decimal, and 'FFFFFO' is -16 in decimal.  CS ASCII-hex byte that is the (check) sum of all previous bytes (excluding the SOP 'F' character). Eg. The command F0400410082000001 has the checksum of 04+00+41+00+82+00+00+01=C8, so the complete command to send is F0400410082000001?? Instead. This is purely for test and debugging - you should normally use a checksum to ensure data validity.	SOURCE	When Function number is within the Adjust Sources section (see later):										
WACRO NUMBER  Byte to indicate the source channel to be altered (if appropriate).  This byte should be set to 0 if the command does not relate to an input-specific parameter.  0x10 = RGB1, 0x11 = RGB2, 0x12 = RGB3  0x30 = CV1, 0x31 = CV2, 0x32 = CV3  0x40 = YC1, 0x41 = YC2, 0x42 = YC3  0x50 = SDI1, 0x51 = SDI2  0xD0 = OUT1, 0xD1 = OUT2  0xF0 = TC1, 0xF1 = TC2  Or - for Macro related commands:  Bit 74 = 0 Reserved  Bit 30 = Macro number  Or - when not used (for the majority of commands)  Bit 7 = Reserved (set to 0).  0GO / Bit 60 = Represents the window to be adjusted.  E.g. Window 'A' is sent as '41' since 0x41 is ASCII for 'A'. 0x61 is ASCII for 'a' and is sent as '61'.  DUTPUT  Bit 74 = Number representing the window to adjust 0 = Output 1, 1 = Output 2.  Bit 32 = Reserved (set to 0).  **CUNC HIGH  Bit 10 = Bits 9 & 8 of the function code. (Remainder [70] are in FUNC LOW.)  E.g. If the function code is 0x234, and we want to adjust Output 2, then this byte is 0x12  FUNC LOW  ASCII-hex byte to indicate the lowest 8 bits of the actual function to set or receive (e.g. change Program source).  A later table details all the functions available, and each one will normally relate to an option on the CORIO2.  PAYLOAD  A series of ASCII-hex bytes carrying the data to send.  Read requests have no payload - the payload is in the data sent back.  Write packets require a payload, and this is always in 'triple-bytes' - ie. 3 bytes are required, MSB first.  e.g. '000001' is 1 in decimal, '010000' is 85536 in decimal, and 'FFFFFO' is -16 in decimal.  CS  ASCII-hex byte that is the (check) sum of all previous bytes (excluding the SOP 'F' character).  E.g. The command F0400410082000001 has the checksum of 04+00+41+00+82+00+00+01=C8, so the complete command to send is F0400410082000001?? Instead. This is purely for test and debugging - you should normally use a checksum to ensure data validity.	or	Which i and to in that the Adjust Courses Scotlon (See later).										
NUMBER  This byte should be set to 0 if the command does not relate to an input-specific parameter.  0x10 = RGB1, 0x11 = RGB2, 0x12 = RGB3 0x30 = CV1, 0x31 = CV2, 0x32 = CV3 0x40 = YC1, 0x41 = YC2, 0x42 = YC3 0x50 = SDI1, 0x51 = SDI2 0xD0 = OUT1, 0xD1 = OUT2 0xF0 = TC1, 0xF1 = TC2  Or - for Macro related commands: Bit 7.4 = 0 Reserved Bit 3.0 = Macro number  Or - when not used (for the majority of commands) Bit 7.0 = 0  WINDOW / 0GO / 0GO / Bit 6.0 = Represents the window to be adjusted. E.g. Window 'A' is sent as '41' since 0x41 is ASCII for 'A'. 0x61 is ASCII for 'a' and is sent as '61'.  DUTPUT Bit 7.4 = Number representing the window to adjust 0 = Output 1, 1 = Output 2. Bit 3.2 = Reserved (set to 0). Bit 1.0 = Bits 9 & 8 of the function code. (Remainder [7.0] are in FUNC LOW.) Eg. If the function code is 0x234, and we want to adjust Output 2, then this byte is 0x12  EUNC LOW  A Scil-hex byte to indicate the lowest 8 bits of the actual function to set or receive (e.g. change Program source). A later table details all the functions available, and each one will normally relate to an option on the CORIO2.  PAYLOAD A series of ASCII-hex bytes carrying the data to send. Read requests have no payload - the payload is in the data sent back. Write packets require a payload, and this is always in 'triple-bytes' - ie. 3 bytes are required, MSB first. eg. '000001' is 1 in decimal, '010000' is 65536 in decimal, and 'FFFFO' is -16 in decimal.  ASCII-hex byte that is the (check) sum of all previous bytes (excluding the SOP 'F' character). Eg. The command F0400410082000001 has the checksum to be replaced by 2 question marks, so in the previous example you could send F0400410082000001? Instead. This is purely for test and debugging - you should normally use a checksum to ensure data validity.	-	Byte to indicate the source channel to be altered (if appropriate)										
Ox10 = RGB1, 0x11 = RGB2, 0x12 = RGB3 0x30 = CV1, 0x31 = CV2, 0x32 = CV3 0x40 = YC1, 0x41 = YC2, 0x42 = YC3 0x50 = SDI1, 0x51 = SDI2 0xD0 = OUT1, 0xD1 = OUT2 0xF0 = TC1, 0xF1 = TC2  Or - for Macro related commands: Bit 74 = 0 Reserved Bit 30 = Macro number  Or - when not used (for the majority of commands) Bit 70 = 0  MINDOW / OGO / Bit 60 = Represents the window to be adjusted. E.g. Window 'A' is sent as '41' since 0x41 is ASCII for 'A'. 0x61 is ASCII for 'a' and is sent as '61'.  DUTPUT Bit 74 = Number representing the window to adjust 0 = Output 1, 1 = Output 2. Bit 32 = Reserved (set to 0). Bit 10 = Bits 9 & 8 of the function code. (Remainder [70] are in FUNC LOW.) Eg. If the function code is 0x234, and we want to adjust Output 2, then this byte is 0x12  FUNC LOW ASCII-hex byte to indicate the lowest 8 bits of the actual function to set or receive (e.g. change Program source). A later table details all the functions available, and each one will normally relate to an option on the CORIO2.  PAYLOAD A series of ASCII-hex bytes carrying the data to send. Read requests have no payload - the payload is in the data sent back. Write packets require a payload, and this is all ways in 'triple-bytes' - ie. 3 bytes are required, MSB first. eg. '000001' is 1 in decimal, '010000' is 65536 in decimal, and 'FFFFF0' is -16 in decimal.  CS ASCII-hex byte that is the (check) sum of all previous bytes (excluding the SOP 'F' character). Eg. The command F0400410082000001 has the checksum of 04+00+41+00+82+00+00+01=C8, so the complete command to send is F0400410082000001?? Instead. This is purely for test and debugging - you should normally use a checksum to ensure data validity.												
0x30 = CV1, 0x31 = CV2, 0x32 = CV3 0x40 = YC1, 0x41 = YC2, 0x42 = YC3 0x50 = SDI1, 0x51 = SDI2 0xD0 = OUT1, 0xD1 = OUT2 0xF0 = TC1, 0xF1 = TC2  Or – for Macro related commands: Bit 7.4 = 0 Reserved Bit 30 = Macro number  Or – when not used (for the majority of commands) Bit 7.0 = 0  MINDOW / Dit 7.0 = 0  MINDOW / Bit 6.0 = Represents the window to be adjusted. E.g. Window 'A' is sent as '41' since 0x41 is ASCII for 'A'. 0x61 is ASCII for 'a' and is sent as '61'.  DUTPUT Bit 7.4 = Number representing the window to adjust 0 = Output 1, 1 = Output 2. Bit 3.2 = Reserved (set to 0). Bit 1.0 = Bits 9 & 8 of the function code. (Remainder [7.0] are in FUNC LOW.) Eg. If the function code is 0x234, and we want to adjust Output 2, then this byte is 0x12  FUNC LOW ASCII-hex byte to indicate the lowest 8 bits of the actual function to set or receive (e.g. change Program source). A later table details all the functions available, and each one will normally relate to an option on the CORIO2.  PAYLOAD A series of ASCII-hex bytes carrying the data to send. Read requests have no payload - the payload is in the data sent back. Write packets require a payload, and this is always in 'triple-bytes' - ie. 3 bytes are required, MSB first. eg. '000001' is 1 in decimal, '010000' is 65536 in decimal, and 'FFFFF0' is -16 in decimal.  CS ASCII-hex byte that is the (check) sum of all previous bytes (excluding the SOP 'F' character). Eg. The command F0400410082000001 has the checksum of 04+00+410+0+82+00+00+01=C8, so the complete command to send is F04004100820000017? Instead. This is purely for test and debugging - you should normally use a checksum to ensure data validity.	NOMBER											
Ox40 = YC1, 0x41 = YC2, 0x42 = YC3 0x50 = SDI1, 0x51 = SDI2 0xD0 = OUT1, 0xD1 = OUT2 0xF0 = TC1, 0xF1 = TC2  Or – for Macro related commands: Bit 74 = 0 Reserved Bit 30 = Macro number  Or – when not used (for the majority of commands) Bit 70 = 0  MINDOW / Bit 7 = Reserved (set to 0). Bit 60 = Represents the window to be adjusted. E.g. Window 'A' is sent as '41' since 0x41 is ASCII for 'A'. 0x61 is ASCII for 'a' and is sent as '61'. DUTPUT Bit 74 = Number representing the window to adjust 0 = Output 1, 1 = Output 2. Bit 32 = Reserved (set to 0). Eg. If the function code is 0x234, and we want to adjust Output 2, then this byte is 0x12  FUNC HOM ASCII-hex byte to indicate the lowest 8 bits of the actual function to set or receive (e.g. change Program source). A later table details all the functions available, and each one will normally relate to an option on the CORIO2.  PAYLOAD A series of ASCII-hex bytes carrying the data to send. Read requests have no payload - the payload is in the data sent back. Write packets require a payload, and this is always in 'triple-bytes' - ie. 3 bytes are required, MSB first. eg. '000001' is 1 in decimal, '010000' is 65536 in decimal, and 'FFFFF0' is -16 in decimal.  CS ASCII-hex byte that is the (check) sum of all previous bytes (excluding the SOP 'F' character). Eg. The command F0400410082000001 has the checksum of 04+00+41+00+82+00+00+01=C8, so the complete command to send is F0400410082000001 (S. A short-cut for debugging allows the checksum to be replaced by 2 question marks, so in the previous example you could send F0400410082000001? Instead. This is purely for test and debugging - you should normally use a checksum to ensure data validity.												
Ox50 = SDI1, 0x51 = SDI2 0xD0 = OUT1, 0xD1 = OUT2 0xF0 = TC1, 0xF1 = TC2  Or - for Macro related commands: Bit 74 = 0 Reserved Bit 30 = Macro number  Or - when not used (for the majority of commands) Bit 70 = 0  WINDOW / Bit 60 = Represents the window to be adjusted. E.g. Window 'A' is sent as '41' since 0x41 is ASCII for 'A'. 0x61 is ASCII for 'a' and is sent as '61'.  DUTPUT Bit 74 = Number representing the window to adjust 0 = Output 1, 1 = Output 2. Bit 32 = Reserved (set to 0). Bit 10 = Bits 9 & 8 of the function code. (Remainder [70] are in FUNC LOW.) Eg. If the function code is 0x234, and we want to adjust Output 2, then this byte is 0x12  FUNC LOW ASCII-hex byte to indicate the lowest 8 bits of the actual function to set or receive (e.g. change Program source). A later table details all the functions available, and each one will normally relate to an option on the CORIO2.  PAYLOAD A series of ASCII-hex bytes carrying the data to send. Read requests have no payload - the payload is in the data sent back. Write packets require a payload, and this is always in 'triple-bytes' - ie. 3 bytes are required, MSB first. eg. '000001' is 1 in decimal, '010000' is 65536 in decimal, and 'FFFFFO' is -16 in decimal.  CS ASCII-hex byte that is the (check) sum of all previous bytes (excluding the SOP 'F' character). Eg. The command F0400410082000001 has the checksum of 04+00+41+00+82+00+00+01=C8, so the complete command to send is F04004100820000012'. Instead. This is purely for test and debugging - you should normally use a checksum to ensure data validity.												
OxD0 = OUT1, 0xD1 = OUT2 0xF0 = TC1, 0xF1 = TC2  Or – for Macro related commands: Bit 74 = 0 Reserved Bit 30 = Macro number  Or – when not used (for the majority of commands) Bit 70 = 0  WINDOW / .OGO / Bit 60 = Represents the window to be adjusted. Bit 74 = Number representing the window to adjust 0 = Output 1, 1 = Output 2. Bit 74 = Number representing the window to adjust 0 = Output 1, 1 = Output 2. Bit 32 = Reserved (set to 0). Bit 10 = Bits 9 & 8 of the function code. (Remainder [70] are in FUNC LOW.) Eg. If the function code is 0x234, and we want to adjust Output 2, then this byte is 0x12  FUNC LOW  A SCII-hex byte to indicate the lowest 8 bits of the actual function to set or receive (e.g. change Program source). A later table details all the functions available, and each one will normally relate to an option on the CORIO2.  A series of ASCII-hex bytes carrying the data to send. A series of ASCII-hex bytes carrying the data to send. Read requests have no payload - the payload is in the data sent back. Write packets require a payload, and this is always in 'triple-bytes' - ie. 3 bytes are required, MSB first. eg. '000001' is 1 in decimal, '010000' is 65536 in decimal, and 'FFFFFO' is -16 in decimal.  CS  ASCII-hex byte that is the (check) sum of all previous bytes (excluding the SOP 'F' character). Eg. The command F040041008200000172 Instead. This is purely for test and debugging - you should normally use a checksum to be replaced by 2 question marks, so in the previous example you could send F04004100820000017? Instead. This is purely for test and debugging - you should normally use a checksum to ensure data validity.												
OxF0 = TC1, 0xF1 = TC2  Or – for Macro related commands: Bit 74 = 0 Reserved Bit 30 = Macro number  Or – when not used (for the majority of commands) Bit 70 = 0  WINDOW / DGO / Bit 60 = Represents the window to be adjusted. E.g. Window 'A' is sent as '41' since 0x41 is ASCII for 'A'. 0x61 is ASCII for 'a' and is sent as '61'.  DUTPUT Bit 74 = Number representing the window to adjust 0 = Output 1, 1 = Output 2. Bit 32 = Reserved (set to 0). Bit 10 = Bits 9 & 8 of the function code. (Remainder [70] are in FUNC LOW.) Eg. If the function code is 0x234, and we want to adjust Output 2, then this byte is 0x12  FUNC LOW  ASCII-hex byte to indicate the lowest 8 bits of the actual function to set or receive (e.g. change Program source). A later table details all the functions available, and each one will normally relate to an option on the CORIO2.  PAYLOAD A series of ASCII-hex bytes carrying the data to send. A 3 bytes Read requests have no payload - the payload is in the data sent back. Write packets require a payload, and this is always in 'triple-bytes' - ie. 3 bytes are required, MSB first. eg. '000001' is 1 in decimal, '010000' is 65536 in decimal, and 'FFFFF0' is -16 in decimal.  CS  ASCII-hex byte that is the (check) sum of all previous bytes (excluding the SOP 'F' character). Eg. The command F0400410082000001C8. A short-cut for debugging allows the checksum to be replaced by 2 question marks, so in the previous example you could send F0400410082000001? Instead. This is purely for test and debugging - you should normally use a checksum to ensure data validity.												
Or – for Macro related commands: Bit 7.4 = 0 Reserved Bit 3.0 = Macro number  Or – when not used (for the majority of commands) Bit 7.0 = 0  WINDOW / DGO / Bit 6.0 = Represents the window to be adjusted. E.g. Window 'A' is sent as '41' since 0x41 is ASCII for 'A'. 0x61 is ASCII for 'a' and is sent as '61'.  DUTPUT Bit 7.4 = Number representing the window to adjust 0 = Output 1, 1 = Output 2. Bit 3.2 = Reserved (set to 0).  FUNC HIGH Bit 1.0 = Bits 9 & 8 of the function code. (Remainder [7.0] are in FUNC LOW.) E.g. If the function code is 0x234, and we want to adjust Output 2, then this byte is 0x12  FUNC LOW ASCII-hex byte to indicate the lowest 8 bits of the actual function to set or receive (e.g. change Program source). A later table details all the functions available, and each one will normally relate to an option on the CORIO2.  A series of ASCII-hex bytes carrying the data to send. A series of ASCII-hex bytes carrying the data to send. A series of ASCII-hex bytes carrying the data to send. A series of ASCII-hex bytes carrying the data to send. A scil-hex byte that is the (check) sum of all previous bytes (excluding the SOP 'F' character). E.g. The command F0400410082000001 has the checksum of 04+00+41+00+82+00+00+01=C8, so the complete command to send is F0400410082000001C8. A short-cut for debugging allows the checksum to be replaced by 2 question marks, so in the previous example you could send F0400410082000001? Instead. This is purely for test and debugging - you should normally use a checksum to ensure data validity.												
Bit 74 = 0 Reserved Bit 30 = Macro number  Or – when not used (for the majority of commands) Bit 70 = 0  WINDOW / Bit 60 = Represents the window to be adjusted. BORDER E.g. Window 'A' is sent as '41' since 0x41 is ASCII for 'A'. 0x61 is ASCII for 'a' and is sent as '61'.  DUTPUT Bit 74 = Number representing the window to adjust 0 = Output 1, 1 = Output 2. Bit 32 = Reserved (set to 0). BIt 10 = Bits 9 & 8 of the function code. (Remainder [70] are in FUNC LOW.) E.g. If the function code is 0x234, and we want to adjust Output 2, then this byte is 0x12  FUNC LOW ASCII-hex byte to indicate the lowest 8 bits of the actual function to set or receive (e.g. change Program source). A later table details all the functions available, and each one will normally relate to an option on the CORIO2.  PAYLOAD A series of ASCII-hex bytes carrying the data to send. Read requests have no payload - the payload is in the data sent back. Write packets require a payload, and this is always in 'triple-bytes' - ie. 3 bytes are required, MSB first. e.g. '000001' is 1 in decimal, '010000' is 65536 in decimal, and 'FFFFFO' is -16 in decimal.  CS ASCII-hex byte that is the (check) sum of all previous bytes (excluding the SOP 'F' character). E.g. The command F0400410082000001 has the checksum of 04+00+41+00+82+00+00+01=C8, so the complete command to send is F04004100820000017? Instead. This is purely for test and debugging - you should normally use a checksum to ensure data validity.												
Bit 74 = 0 Reserved Bit 30 = Macro number  Or – when not used (for the majority of commands) Bit 70 = 0  WINDOW / Bit 60 = Represents the window to be adjusted. BORDER E.g. Window 'A' is sent as '41' since 0x41 is ASCII for 'A'. 0x61 is ASCII for 'a' and is sent as '61'.  DUTPUT Bit 74 = Number representing the window to adjust 0 = Output 1, 1 = Output 2. Bit 32 = Reserved (set to 0). BIt 10 = Bits 9 & 8 of the function code. (Remainder [70] are in FUNC LOW.) E.g. If the function code is 0x234, and we want to adjust Output 2, then this byte is 0x12  FUNC LOW ASCII-hex byte to indicate the lowest 8 bits of the actual function to set or receive (e.g. change Program source). A later table details all the functions available, and each one will normally relate to an option on the CORIO2.  PAYLOAD A series of ASCII-hex bytes carrying the data to send. Read requests have no payload - the payload is in the data sent back. Write packets require a payload, and this is always in 'triple-bytes' - ie. 3 bytes are required, MSB first. e.g. '000001' is 1 in decimal, '010000' is 65536 in decimal, and 'FFFFFO' is -16 in decimal.  CS ASCII-hex byte that is the (check) sum of all previous bytes (excluding the SOP 'F' character). E.g. The command F0400410082000001 has the checksum of 04+00+41+00+82+00+00+01=C8, so the complete command to send is F04004100820000017? Instead. This is purely for test and debugging - you should normally use a checksum to ensure data validity.		Or – for Macro related commands:										
Or – when not used (for the majority of commands) Bit 7.0 = 0  MINDOW / Bit 7 = Reserved (set to 0).  OGO / Bit 6.0 = Represents the window to be adjusted. E.g. Window 'A' is sent as '41' since 0x41 is ASCII for 'A'. 0x61 is ASCII for 'a' and is sent as '61'.  DUTPUT Bit 7.4 = Number representing the window to adjust 0 = Output 1, 1 = Output 2. Bit 3.2 = Reserved (set to 0). Bit 1.0 = Bits 9 & 8 of the function code. (Remainder [70] are in FUNC LOW.) Eg. If the function code is 0x234, and we want to adjust Output 2, then this byte is 0x12  FUNC LOW ASCII-hex byte to indicate the lowest 8 bits of the actual function to set or receive (e.g. change Program source). A later table details all the functions available, and each one will normally relate to an option on the CORIO2.  PAYLOAD A series of ASCII-hex bytes carrying the data to send. Read requests have no payload - the payload is in the data sent back. Write packets require a payload, and this is always in 'triple-bytes' - ie. 3 bytes are required, MSB first. eg. '000001' is 1 in decimal, '010000' is 65536 in decimal, and 'FFFFF0' is -16 in decimal.  CS ASCII-hex byte that is the (check) sum of all previous bytes (excluding the SOP 'F' character). Eg. The command F0400410082000001 has the checksum of 04+00+41+00+82+00+00+01=C8, so the complete command to send is F0400410082000001(2s. A short-cut for debugging allows the checksum to be replaced by 2 question marks, so in the previous example you could send F0400410082000001?? Instead. This is purely for test and debugging - you should normally use a checksum to ensure data validity.												
Bit 70 = 0  WINDOW / LOGO / Bit 60 = Represents the window to be adjusted. E.g. Window 'A' is sent as '41' since 0x41 is ASCII for 'A'. 0x61 is ASCII for 'a' and is sent as '61'.  DUTPUT Bit 74 = Number representing the window to adjust 0 = Output 1, 1 = Output 2. Bit 32 = Reserved (set to 0).  BIT 10 = Bits 9 & 8 of the function code. (Remainder [70] are in FUNC LOW.) Eg. If the function code is 0x234, and we want to adjust Output 2, then this byte is 0x12  FUNC LOW  ASCII-hex byte to indicate the lowest 8 bits of the actual function to set or receive (e.g. change Program source). A later table details all the functions available, and each one will normally relate to an option on the CORIO2.  PAYLOAD A series of ASCII-hex bytes carrying the data to send. Read requests have no payload - the payload is in the data sent back. Write packets require a payload, and this is always in 'triple-bytes' - ie. 3 bytes are required, MSB first. eg. '000001' is 1 in decimal, '010000' is 65536 in decimal, and 'FFFFF0' is -16 in decimal.  CS ASCII-hex byte that is the (check) sum of all previous bytes (excluding the SOP 'F' character). Eg. The command F04004100820000001 has the checksum of 04+00+41+00+82+00+00+01=C8, so the complete command to send is F0400410082000001C8. A short-cut for debugging allows the checksum to be replaced by 2 question marks, so in the previous example you could send F0400410082000001? Instead. This is purely for test and debugging - you should normally use a checksum to ensure data validity.		Bit 30 = Macro number										
Bit 70 = 0  WINDOW / LOGO / Bit 60 = Represents the window to be adjusted. E.g. Window 'A' is sent as '41' since 0x41 is ASCII for 'A'. 0x61 is ASCII for 'a' and is sent as '61'.  DUTPUT Bit 74 = Number representing the window to adjust 0 = Output 1, 1 = Output 2. Bit 32 = Reserved (set to 0).  BIT 10 = Bits 9 & 8 of the function code. (Remainder [70] are in FUNC LOW.) Eg. If the function code is 0x234, and we want to adjust Output 2, then this byte is 0x12  FUNC LOW  ASCII-hex byte to indicate the lowest 8 bits of the actual function to set or receive (e.g. change Program source). A later table details all the functions available, and each one will normally relate to an option on the CORIO2.  PAYLOAD A series of ASCII-hex bytes carrying the data to send. Read requests have no payload - the payload is in the data sent back. Write packets require a payload, and this is always in 'triple-bytes' - ie. 3 bytes are required, MSB first. eg. '000001' is 1 in decimal, '010000' is 65536 in decimal, and 'FFFFF0' is -16 in decimal.  CS ASCII-hex byte that is the (check) sum of all previous bytes (excluding the SOP 'F' character). Eg. The command F04004100820000001 has the checksum of 04+00+41+00+82+00+00+01=C8, so the complete command to send is F0400410082000001C8. A short-cut for debugging allows the checksum to be replaced by 2 question marks, so in the previous example you could send F0400410082000001? Instead. This is purely for test and debugging - you should normally use a checksum to ensure data validity.												
WINDOW / LOGO / Bit 60 = Represents the window to be adjusted. E.g. Window 'A' is sent as '41' since 0x41 is ASCII for 'A'. 0x61 is ASCII for 'a' and is sent as '61'.  DUTPUT Bit 74 = Number representing the window to adjust 0 = Output 1, 1 = Output 2. Bit 32 = Reserved (set to 0). Bit 10 = Bits 9 & 8 of the function code. (Remainder [70] are in FUNC LOW.) Eg. If the function code is 0x234, and we want to adjust Output 2, then this byte is 0x12  FUNC LOW ASCII-hex byte to indicate the lowest 8 bits of the actual function to set or receive (e.g. change Program source). A later table details all the functions available, and each one will normally relate to an option on the CORIO2.  PAYLOAD A series of ASCII-hex bytes carrying the data to send. Read requests have no payload - the payload is in the data sent back. Write packets require a payload, and this is always in 'triple-bytes' - ie. 3 bytes are required, MSB first. eg. '000001' is 1 in decimal, '010000' is 65536 in decimal, and 'FFFFFO' is -16 in decimal.  CS ASCII-hex byte that is the (check) sum of all previous bytes (excluding the SOP 'F' character). Eg. The command F0400410082000001 has the checksum of 04+00+41+00+82+00+00+01=C8, so the complete command to send is F0400410082000001C8. A short-cut for debugging allows the checksum to be replaced by 2 question marks, so in the previous example you could send F0400410082000001? Instead. This is purely for test and debugging - you should normally use a checksum to ensure data validity.		Or – when not used (for the majority of commands)										
Bit 60 = Represents the window to be adjusted. E.g. Window 'A' is sent as '41' since 0x41 is ASCII for 'A'. 0x61 is ASCII for 'a' and is sent as '61'.  Bit 74 = Number representing the window to adjust 0 = Output 1, 1 = Output 2. Bit 32 = Reserved (set to 0). Bit 10 = Bits 9 & 8 of the function code. (Remainder [70] are in FUNC LOW.) Eg. If the function code is 0x234, and we want to adjust Output 2, then this byte is 0x12  FUNC LOW ASCII-hex byte to indicate the lowest 8 bits of the actual function to set or receive (e.g. change Program source). A later table details all the functions available, and each one will normally relate to an option on the CORIO2.  PAYLOAD A series of ASCII-hex bytes carrying the data to send. Read requests have no payload - the payload is in the data sent back. Write packets require a payload, and this is always in 'triple-bytes' - ie. 3 bytes are required, MSB first. eg. '00001' is 1 in decimal, '010000' is 65536 in decimal, and 'FFFFF0' is -16 in decimal.  ASCII-hex byte that is the (check) sum of all previous bytes (excluding the SOP 'F' character). Eg. The command F0400410082000001 has the checksum of 04+00+41+00+82+00+00+01=C8, so the complete command to send is F0400410082000001C8. A short-cut for debugging allows the checksum to be replaced by 2 question marks, so in the previous example you could send F0400410082000001? Instead. This is purely for test and debugging - you should normally use a checksum to ensure data validity.		Bit 70 = 0										
E.g. Window 'A' is sent as '41' since 0x41 is ASCII for 'A'. 0x61 is ASCII for 'a' and is sent as '61'.  Bit 74 = Number representing the window to adjust 0 = Output 1, 1 = Output 2.  Bit 32 = Reserved (set to 0).  Bit 10 = Bits 9 & 8 of the function code. (Remainder [70] are in FUNC LOW.)  Eg. If the function code is 0x234, and we want to adjust Output 2, then this byte is 0x12  ASCII-hex byte to indicate the lowest 8 bits of the actual function to set or receive (e.g. change Program source).  A later table details all the functions available, and each one will normally relate to an option on the CORIO2.  PAYLOAD  A series of ASCII-hex bytes carrying the data to send.  Read requests have no payload - the payload is in the data sent back.  Write packets require a payload, and this is always in 'triple-bytes' - ie. 3 bytes are required, MSB first. eg. '000001' is 1 in decimal, '010000' is 65536 in decimal, and 'FFFFO' is -16 in decimal.  CS  ASCII-hex byte that is the (check) sum of all previous bytes (excluding the SOP 'F' character).  Eg. The command F0400410082000001 has the checksum of 04+00+41+00+82+00+00+01=C8, so the complete command to send is F0400410082000001C8.  A short-cut for debugging allows the checksum to be replaced by 2 question marks, so in the previous example you could send F0400410082000001? Instead. This is purely for test and debugging - you should normally use a checksum to ensure data validity.	WINDOW /											
Bit 74 = Number representing the window to adjust 0 = Output 1, 1 = Output 2.  Bit 32 = Reserved (set to 0).  Bit 10 = Bits 9 & 8 of the function code. (Remainder [70] are in FUNC LOW.)  Eg. If the function code is 0x234, and we want to adjust Output 2, then this byte is 0x12  ASCII-hex byte to indicate the lowest 8 bits of the actual function to set or receive (e.g. change Program source).  A later table details all the functions available, and each one will normally relate to an option on the CORIO2.  A series of ASCII-hex bytes carrying the data to send.  Read requests have no payload - the payload is in the data sent back.  Write packets require a payload, and this is always in 'triple-bytes' - ie. 3 bytes are required, MSB first. eg. '000001' is 1 in decimal, '010000' is 65536 in decimal, and 'FFFFF0' is -16 in decimal.  CS  ASCII-hex byte that is the (check) sum of all previous bytes (excluding the SOP 'F' character).  Eg. The command F0400410082000001 has the checksum of 04+00+41+00+82+00+00+01=C8, so the complete command to send is F0400410082000001C8.  A short-cut for debugging allows the checksum to be replaced by 2 question marks, so in the previous example you could send F0400410082000001? Instead. This is purely for test and debugging - you should normally use a checksum to ensure data validity.	LOGO /											
Bit 32 = Reserved (set to 0).  Bit 10 = Bits 9 & 8 of the function code. (Remainder [70] are in FUNC LOW.)  Eg. If the function code is 0x234, and we want to adjust Output 2, then this byte is 0x12  ASCII-hex byte to indicate the lowest 8 bits of the actual function to set or receive (e.g. change Program source).  A later table details all the functions available, and each one will normally relate to an option on the CORIO2.  PAYLOAD  A series of ASCII-hex bytes carrying the data to send.  Read requests have no payload - the payload is in the data sent back.  Write packets require a payload, and this is always in 'triple-bytes' - ie. 3 bytes are required, MSB first. eg. '000001' is 1 in decimal, '010000' is 65536 in decimal, and 'FFFFF0' is -16 in decimal.  CS  ASCII-hex byte that is the (check) sum of all previous bytes (excluding the SOP 'F' character).  Eg. The command F0400410082000001 has the checksum of 04+00+41+00+82+00+00+01=C8, so the complete command to send is F0400410082000001C8.  A short-cut for debugging allows the checksum to be replaced by 2 question marks, so in the previous example you could send F0400410082000001? Instead. This is purely for test and debugging - you should normally use a checksum to ensure data validity.	BORDER											
Bit 10 = Bits 9 & 8 of the function code. (Remainder [70] are in FUNC LOW.)  Eg. If the function code is 0x234, and we want to adjust Output 2, then this byte is 0x12  ASCII-hex byte to indicate the lowest 8 bits of the actual function to set or receive (e.g. change Program source).  A later table details all the functions available, and each one will normally relate to an option on the CORIO2.  PAYLOAD  A series of ASCII-hex bytes carrying the data to send.  Read requests have no payload - the payload is in the data sent back.  Write packets require a payload, and this is always in 'triple-bytes' - ie. 3 bytes are required, MSB first. eg. '000001' is 1 in decimal, '010000' is 65536 in decimal, and 'FFFFF0' is -16 in decimal.  CS  ASCII-hex byte that is the (check) sum of all previous bytes (excluding the SOP 'F' character).  Eg. The command F0400410082000001 has the checksum of 04+00+41+00+82+00+00+01=C8, so the complete command to send is F0400410082000001C8.  A short-cut for debugging allows the checksum to be replaced by 2 question marks, so in the previous example you could send F0400410082000001?? Instead. This is purely for test and debugging - you should normally use a checksum to ensure data validity.	OUTPUT	Bit 74 = Number representing the window to adjust 0 = Output 1, 1 = Output 2.										
Eg. If the function code is 0x234, and we want to adjust Output 2, then this byte is 0x12  ASCII-hex byte to indicate the lowest 8 bits of the actual function to set or receive (e.g. change Program source).  A later table details all the functions available, and each one will normally relate to an option on the CORIO2.  PAYLOAD  A series of ASCII-hex bytes carrying the data to send.  Read requests have no payload - the payload is in the data sent back.  Write packets require a payload, and this is always in 'triple-bytes' - ie. 3 bytes are required, MSB first. eg. '000001' is 1 in decimal, '010000' is 65536 in decimal, and 'FFFFF0' is -16 in decimal.  CS  ASCII-hex byte that is the (check) sum of all previous bytes (excluding the SOP 'F' character).  Eg. The command F0400410082000001 has the checksum of 04+00+41+00+82+00+00+01=C8, so the complete command to send is F0400410082000001C8.  A short-cut for debugging allows the checksum to be replaced by 2 question marks, so in the previous example you could send F0400410082000001?? Instead. This is purely for test and debugging - you should normally use a checksum to ensure data validity.	&											
ASCII-hex byte to indicate the lowest 8 bits of the actual function to set or receive (e.g. change Program source).  A later table details all the functions available, and each one will normally relate to an option on the CORIO2.  PAYLOAD  A series of ASCII-hex bytes carrying the data to send.  Read requests have no payload - the payload is in the data sent back.  Write packets require a payload, and this is always in 'triple-bytes' - ie. 3 bytes are required, MSB first. eg. '000001' is 1 in decimal, '010000' is 65536 in decimal, and 'FFFFF0' is -16 in decimal.  CS  ASCII-hex byte that is the (check) sum of all previous bytes (excluding the SOP 'F' character).  Eg. The command F0400410082000001 has the checksum of 04+00+41+00+82+00+00+01=C8, so the complete command to send is F0400410082000001C8.  A short-cut for debugging allows the checksum to be replaced by 2 question marks, so in the previous example you could send F0400410082000001?? Instead. This is purely for test and debugging - you should normally use a checksum to ensure data validity.	FUNC HIGH											
Source).  A later table details all the functions available, and each one will normally relate to an option on the CORIO2.  PAYLOAD  A series of ASCII-hex bytes carrying the data to send.  Read requests have no payload - the payload is in the data sent back.  Write packets require a payload, and this is always in 'triple-bytes' - ie. 3 bytes are required, MSB first. eg. '000001' is 1 in decimal, '010000' is 65536 in decimal, and 'FFFFF0' is -16 in decimal.  CS  ASCII-hex byte that is the (check) sum of all previous bytes (excluding the SOP 'F' character).  Eg. The command F0400410082000001 has the checksum of 04+00+41+00+82+00+00+01=C8, so the complete command to send is F0400410082000001C8.  A short-cut for debugging allows the checksum to be replaced by 2 question marks, so in the previous example you could send F0400410082000001?? Instead. This is purely for test and debugging - you should normally use a checksum to ensure data validity.												
A later table details all the functions available, and each one will normally relate to an option on the CORIO2.  A series of ASCII-hex bytes carrying the data to send.  Read requests have no payload - the payload is in the data sent back.  Write packets require a payload, and this is always in 'triple-bytes' - ie. 3 bytes are required, MSB first. eg. '000001' is 1 in decimal, '010000' is 65536 in decimal, and 'FFFFF0' is -16 in decimal.  ASCII-hex byte that is the (check) sum of all previous bytes (excluding the SOP 'F' character).  Eg. The command F0400410082000001 has the checksum of 04+00+41+00+82+00+00+01=C8, so the complete command to send is F0400410082000001C8.  A short-cut for debugging allows the checksum to be replaced by 2 question marks, so in the previous example you could send F0400410082000001?? Instead. This is purely for test and debugging - you should normally use a checksum to ensure data validity.	FUNC LOW											
CORIO2.  A series of ASCII-hex bytes carrying the data to send. Read requests have no payload - the payload is in the data sent back. Write packets require a payload, and this is always in 'triple-bytes' - ie. 3 bytes are required, MSB first. eg. '000001' is 1 in decimal, '010000' is 65536 in decimal, and 'FFFFF0' is -16 in decimal.  ASCII-hex byte that is the (check) sum of all previous bytes (excluding the SOP 'F' character). Eg. The command F0400410082000001 has the checksum of 04+00+41+00+82+00+00+01=C8, so the complete command to send is F0400410082000001C8. A short-cut for debugging allows the checksum to be replaced by 2 question marks, so in the previous example you could send F0400410082000001?? Instead. This is purely for test and debugging - you should normally use a checksum to ensure data validity.												
A series of ASCII-hex bytes carrying the data to send. Read requests have no payload - the payload is in the data sent back. Write packets require a payload, and this is always in 'triple-bytes' - ie. 3 bytes are required, MSB first. eg. '000001' is 1 in decimal, '010000' is 65536 in decimal, and 'FFFFF0' is -16 in decimal.  ASCII-hex byte that is the (check) sum of all previous bytes (excluding the SOP 'F' character). Eg. The command F0400410082000001 has the checksum of 04+00+41+00+82+00+00+01=C8, so the complete command to send is F0400410082000001C8. A short-cut for debugging allows the checksum to be replaced by 2 question marks, so in the previous example you could send F0400410082000001?? Instead. This is purely for test and debugging - you should normally use a checksum to ensure data validity.												
Read requests have no payload - the payload is in the data sent back.  Write packets require a payload, and this is always in 'triple-bytes' - ie. 3 bytes are required, MSB first. eg. '000001' is 1 in decimal, '010000' is 65536 in decimal, and 'FFFFF0' is -16 in decimal.  ASCII-hex byte that is the (check) sum of all previous bytes (excluding the SOP 'F' character).  Eg. The command F0400410082000001 has the checksum of 04+00+41+00+82+00+00+01=C8, so the complete command to send is F0400410082000001C8.  A short-cut for debugging allows the checksum to be replaced by 2 question marks, so in the previous example you could send F0400410082000001?? Instead. This is purely for test and debugging - you should normally use a checksum to ensure data validity.												
Write packets require a payload, and this is always in 'triple-bytes' - ie. 3 bytes are required, MSB first. eg. '000001' is 1 in decimal, '010000' is 65536 in decimal, and 'FFFFF0' is -16 in decimal.  ASCII-hex byte that is the (check) sum of all previous bytes (excluding the SOP 'F' character).  Eg. The command F0400410082000001 has the checksum of 04+00+41+00+82+00+00+01=C8, so the complete command to send is F0400410082000001C8.  A short-cut for debugging allows the checksum to be replaced by 2 question marks, so in the previous example you could send F0400410082000001?? Instead. This is purely for test and debugging - you should normally use a checksum to ensure data validity.	PAYLOAD											
eg. '000001' is 1 in decimal, '010000' is 65536 in decimal, and 'FFFFF0' is -16 in decimal.  ASCII-hex byte that is the (check) sum of all previous bytes (excluding the SOP 'F' character).  Eg. The command F0400410082000001 has the checksum of 04+00+41+00+82+00+00+01=C8, so the complete command to send is F0400410082000001C8.  A short-cut for debugging allows the checksum to be replaced by 2 question marks, so in the previous example you could send F0400410082000001?? Instead. This is purely for test and debugging - you should normally use a checksum to ensure data validity.	x 3 bytes											
ASCII-hex byte that is the (check) sum of all previous bytes (excluding the SOP 'F' character).  Eg. The command F0400410082000001 has the checksum of 04+00+41+00+82+00+00+01=C8, so the complete command to send is F0400410082000001C8.  A short-cut for debugging allows the checksum to be replaced by 2 question marks, so in the previous example you could send F0400410082000001?? Instead. This is purely for test and debugging - you should normally use a checksum to ensure data validity.												
Eg. The command F0400410082000001 has the checksum of 04+00+41+00+82+00+00+01=C8, so the complete command to send is F0400410082000001C8.  A short-cut for debugging allows the checksum to be replaced by 2 question marks, so in the previous example you could send F0400410082000001?? Instead. This is purely for test and debugging - you should normally use a checksum to ensure data validity.												
complete command to send is F0400410082000001C8. A short-cut for debugging allows the checksum to be replaced by 2 question marks, so in the previous example you could send F0400410082000001?? Instead. This is purely for test and debugging - you should normally use a checksum to ensure data validity.	CS											
A short-cut for debugging allows the checksum to be replaced by 2 question marks, so in the previous example you could send F0400410082000001?? Instead. This is purely for test and debugging - you should normally use a checksum to ensure data validity.		Eg. The command F0400410082000001 has the checksum of 04+00+41+00+82+00+00+01=C8, so the										
example you could send F0400410082000001?? Instead. This is purely for test and debugging - you should normally use a checksum to ensure data validity.												
should normally use a checksum to ensure data validity.												
EOP This is a carriage return (no line-feed) - ASCII code 13 (decimal).												
	EOP	This is a carriage return (no line-feed) - ASCII code 13 (decimal).										

## **Function list**

These are listed in menu-order.

A CORIO2 unit and manual should be used to determine the actual function of each code, as only the menu text is listed here

Function codes are given in hexadecimal and adjustment range is in decimal (but always sent as hexadecimal!).

The mode of operation also restricts what Window and Output can be used the follow table show the allowed combinations:

Mode	Allowed Window and Output combinations
Switcher	Output 1 (0x00) and Window A (0x41) / Window Z (0x5A) / Logo a (0x61)
Independent	Output 1 (0x00) and Window A (0x41) / Window Z (0x5A) / Logo a (0x61)  OR
	Output 2 (0x01) and Window B (0x42) / Window Z (0x5A) / Logo b (0x62)
Dual PIP	Any combination of Output and Window

The following table is a list of all menu functions and there related function and valid range of adjustment.

Menu text Cha	nnels Function (Hex)	Range of adjustment (decimal)					
Top level							
Mode	109	0 = Switcher 1 = Independent 2 = Dual PIP					
Adjust outputs	•						
Lock source (connector)	149	0x10 = RGB1, 0x11 = RGB2, 0x12 = RGB3 0x30 = CV1, 0x31 = CV2, 0x32 = CV3 0x40 = YC1, 0x41 = YC2, 0x42 = YC3 0x50 = SDI1, 0x51 = SDI2 0xD0 = OUT1, 0xD1 = OUT2 0xF0 = TC1, 0xF1 = TC2					
Lock method	10A	02 = Off, Genlock, Lock & Mix					
Output resolution	83	11000					
Output image type analogue	E2	0 = RGBHV 2 = RGsB 3 = YUV 4 = tlYUV 7 = tlRGB					
Output image type digital	16C						
Background Y	13B	16235					
Background U	13C	16240					
Background V	13D	16240					
CCIR Output Standard	101	0 = NTSC/PAL, 1 = PAL-M/PAL-N					
Output CV/YC IRE	133	-7.512.5					
Output CV/YC Hue (degrees)	139	-2222					
Output SC/H Phase	85	-180180					
Output Luma Bandwidth	134	0,1,2 = Low, Medium, High					
Output Chroma Bandwidth	135	0,1,2 = Low, Medium, High					
Output Chroma delay	137	-43					
PAL WSS	130	0 = Off 1 = 4:3 Full format 2 = 14:9 Letterbox centre 3 = 14:9 Letterbox top 4 = 16:9 Letterbox centre 5 = 16:9 Letterbox top					

	1	6 - > 16:0 Lotterboy contro					
		6 = >16:9 Letterbox centre					
		7 = 14:9 Full format					
Tale	445	8 = 16:9 Full format					
Take	11E	0->1 = Perform a Preview -> Program transition					
Adjust Windows		0.40 DODA 0.44 DODO 0.40 DODO					
Program source / Window source	82	0x10 = RGB1, 0x11 = RGB2, 0x12 = RGB3					
(connector)		0x30 = CV1, 0x31 = CV2, 0x32 = CV3					
		0x40 = YC1, 0x41 = YC2, 0x42 = YC3					
		0x50 = SDI1, 0x51 = SDI2					
		0xD0 = OUT1, 0xD1 = OUT2					
Mindow Frankla	400	0xF0 = TC1, 0xF1 = TC2					
Window Enable	12B	01 = Off, On					
Zoom level %	86	1001000					
Zoom level H %	103	1001000 (only used in Advanced A/R mode)					
Zoom level V %	105	1001000 (only used in Advanced A/R mode)					
Aspect ratio in	107	0.1:19.99:1					
H/V zoom pan % (H)	9F	0100					
H/V zoom pan % (V)	A0	0100					
Freeze	9C	01 = Off, On					
H/V out shift (H)	AD	-100100					
H/V out shift (V)	AE	-100100					
Lock pixel offset	14A	-20472047					
Lock line offset	14B	-20472047					
Shrink level %	87	10100					
Shrink level H %	104	10100 (only used in Advanced A/R mode)					
Shrink level V %	106	10100 (only used in Advanced A/R mode)					
H/V shr. pos.% (H)	DA	0100					
H/V shr. pos.% (V)	DB	0100					
Aspect Adjust	102	01 = Simple, Advanced					
Flicker reduction	92	03 = Off, Low, Med, High					
Image smoothing	A1	02 = Off, Med, High					
Image shooting	95	03 = Off, Horiz., Vertical, H & V					
De-glitch	A3	01 = Off, On					
Max fade level	10F	0100 = Fade level %					
Layer priority	144	05 = Layer priority					
Headphone volume	FD	-1615 (-16=Mute)					
·	FD	-1615 (-16=Mute)					
Adjust keyers							
Keyer enable	127	0 1 = Off, On					
Y key min/max (min)	AF	0255					
Y key min/max (max)	B2	0255					
Y key Softness	121	0255					
Y key Invert	122	01 = Off, On					
U key min/max (min)	B0	0255					
U key min/max (max)	B3	0255					
U key Softness	123	0255					
U key Invert	124	01 = Off, On					
V key min/max (min)	B1	0255					
V key min/max (max)	B4	0255					
V key Softness	125	0255					
V key Invert	156	01 = Off, On					
Logos	1 100	15					
Logo enable	12B	01 = Off, On					
Logo number	143	09 Logo selection					
H/V out shift (H)	AD	0100 %					
H/V out shift (V)	AE	0100 %					
Max fade level	10F	0100%					
	144	05					
Layer priority	144	UU					
Borders  Border applie	450	0.1.0# 00					
Border enable	150	01 = Off, On					

Border H size		152	099
Border V size		151	099
Border H offset		153	099
Border V offset		154	099
Border Opacity		158	0 (fully transparent)100 (solid)
Border Y		155	16235
Border U		156	16240
Border V		157	16240
Adjust sources	<b>L</b>		1
Source to adjust		0x116	0x10 = RGB1, 0x11 = RGB2, 0x12 = RGB3
			0x30 = CV1, 0x31 = CV2, 0x32 = CV3
			0x40 = YC1, 0x41 = YC2, 0x42 = YC3
			0x50 = SDI1, 0x51 = SDI2
			0xD0 = OUT1, $0xD1 = OUT2$
			0xF0 = TC1, 0xF1 = TC2
Testcard	010	DC	010
TL pos. adj. (left)	09	B6	-100100
TL pos. adj. (top)	09	B7	-100100
BR size adj. (right)	09	DE	-100100
BR size adj. (bottom)	09	DF	-100100
Audio input	010	D0	09 = Channels 1 10 on A2-2000
Audio vol	010	CF	-1615 (-16=Mute)
Bal	010	D1	-1515
Input pixel phase	02	91	031
RGB input type	02	C1	0 = Auto
l	02		1 = D-RGB
			2 = D-YUV
			3 = A-RGB
			4 = A-YUV
RGB contr. (red)	02	C5	75150
RGB contr. (green)	02	C6	75150
RGB contr. (blue)	02	C7	75150
De-int.	09	B8	05 = Normal, Auto, Film 3:2, M.comp.low,
			M.comp.med., M.comp.high
(Film mode detected)	09	E3	01 = Not detected, Detected
Bright	38	BB	0180
Contrast	38	BC	0180
Saturation	38	B9	0180
Hue	38	BA	-180180
Sharpness	38	80	-7+7
Luma delay	38	BD	-43
Transitions	10.10		1110
Transition type		112	02 = Cut, fade, wipe
Switching fade time		F5	0 (off) to 50 (5.0 seconds)
Wipe type		145	0 = Left -> Right
vvipe type		145	1 = Right -> Left
			2 = Up -> Down
			3 = Down -> Up
			4 = Diagonal
			5 = Diamond
Wipe Size		146	102000
Adjust resolutions		1 170	102000
	ie to adiust' value	to the corre	ct value first, and only then change the other values -
			should not adjust the 'Image to adjust' entry using the front
panel whilst also accessing it v		y. THE USERS	should het dajust the image to dajust entry using the nont
Image to adjust	14 110202	81	11000
innago to adjust		CA	0.1 = Off. On

6

0..1 = Off, On

10000..200000

10000..200000

64..2047

CA

ΒE

BF

96

Interlaced

H.freq.crse

H.freq.fine

H/V active (H)

## CORIO2 Universal Scaler User Control Specification Protocol 2

H/V active (V) H/V start (H) H/V start (V) Clks/I Lines/f H/V sync (H)	97 8B 8C 8D 8E 8F	642047 01023 01023 644095 642047
H/V start (V) Clks/l Lines/f	8C 8D 8E	01023 644095
Clks/l Lines/f	8D 8E	644095
Lines/f	8E	
IH/V sync (H)	O.	81023
H/V sync (V)	90	11023
Sync polarity	94	03 = ++, +-, -+,
System	<del></del>	05 = 11, 1 , 1,
SW (Software version)	D2	Read only
PT (Product type)	C4	Read only
BT (Board type)	C2	Read only
Advanced menus	11D	01, Off, On
Store	C8	Set to 1 to store
Buzzer	CB	01 = Off, On
Power cycles	D6	Read only
Firmware updates	DD	Read only
Hours in use	D7	Read only
Resolutions	D8	Read only
Number of testcards	D9	Read only
Number of logos	14F	Read only
Board temp. (deg.C)	CD	Read only
Air temp. (deg.C)	148	Read only
Regulators temp.(deg.C)	147	Read only
PLD temp. (deg.C)	111	Read only
Fan speed (rpm)	CE	Read only
Led brightness	12C	0100
RS232 Baud rate	AB	05 = 9600, 19200, 28800, 33600, 38400, 57600
TAC number 0	15D	Read only
TAC number 1	15E	Read only
TAC number 2	15F	Read only
TAC number 3	160	Read only
TAC number 4	161	Read only
TAC number 5	162	Read only
Hidden		
Front panel lock	FC	0 = unlocked, 1 = locked

#### CORIO2 Universal Scaler User Control Specification Protocol 2

## **Examples**

Each example show the packet send to the unit and its response. When a byte is not required to be sent it is indicated by a '- 'in the table below. Each character shown below is sent as a ASCII character so F0400 is sent as 'F' '0' '4' '0' '0'.

Packet sent								Packet returned									
S O P	CMD	СНА	WIN	OUT	FUN	PAY	CS	E O P	S O P	CMD	СНА	WIN	OUT	FUN	PAY	CS	E O P
Set	Set 1B Source to RGB2																
F	04	00	42	00	82	000011	D9	C R	F	44	00	42	00	82	000011	19	C R
Set	Set 1A Enable advanced aspect control																
F	04	00	41	01	02	000001	??	C R	F	44	00	42	01	02	000001	8A	C R
Set	t 1A Sh	nrink to	110 –	- invali	d max	for shrink is	s 100										
F	04	00	41	00	87	00006E	??	C R	F	44	00	41	00	87	000064	70	C R
	Read 1C Zoom level – invalid as window C does not exist																
F	84	00	43	00	86	-	??	C R	F	04	00	43	00	86	000000	CD	C R
Re	Read 1B Zoom level																
F	84	00	42	00	86	-	??	C R	F	44	00	42	00	86	000064	70	C R

All the latest updates are at <a href="http://www.tvone.com/support">http://www.tvone.com/support</a>