XCONSOLE COMMUNICATION PROTOCOL (April, 2008)

Serial Settings for the master terminal:

- **115200 baud**
- **8 data bits**
- **2 stop bits**
- **No Parity**
- **No Flow Control**

Serial Settings for the Xilica device:

- **115200 baud**
- **Device# set as the same as the protocol Device# (Device 1 in the system menu of the device refers to Device 0 (<20>) in the protocol)**

Format used is readable ASCII (96 values for each character) unless specified otherwise:

Value	ASCII	Denote
0	0x20	<20>
:	:	:
95	0x7F	<7F>

Note: All Device#, Channel#, Aux# starts from 0 (<20> in the protocol) for Device 1, Channel 1, Aux 1 respectively. This is called zero-based in computer programming.

PROTOCOL STRING

<01><READ/WRITE><SENDER>*<HEADER><VALUE>...<PROCESS>*<CHECKSUM><02>

1 1 1 1 Varies 1 1 1 <= size in bytes

<READ/WRITE> - Read = 'R' or <52>, Write = 'W' or <57>

<SENDER> - Device number of the sender: <7F> is normally used.

The bytes in RED can be inserted many times inside a single Protocol String. There is no limit in the number of <HEADER><VALUE> combo as long as the whole block (from <01> to <02>) is less than 256 bytes. When a <HEADER><VALUE> is received, the corresponding new parameters are stored in the memory. However, it is processed only when a <PROCESS> (<1F>) is received. The <PROCESS> can be inserted more than once if multiple actions are desired, eg. Mute multiple channels at the same time.

<HEADER>

The <HEADER> is used to identify the types of value following the header.

Command Header	<03 to 07>; <03> = 4 bytes <VALUE>… <07> = 8 bytes <VALUE> Command ID in ASCII character
Device Header	<08> Device # where this command is sent to
I/O Header	<09> Input / Output section of the device
Channel Header	<0A> Channel #
Aux Header	<0B> For extra information such as EQ Num, FBX Num ,etc
Column Header	<0C> Used in Special Commands like Level Increment
Data Header	<10 to 18>; <10> = 1 byte <VALUE> … <18> = 9 bytes <VALUE> The parameter data value to be changed

<VALUE>

<VALUE> must always follow the <HEADER>.

Command Value	For the commands, related table in this document
Device Value	<20> for Device 0... <2F> for Device 15
I/O Value	<20> for Input, <21> for Output
Channel Value	<20> for Channel 0... <27> for Channel 7
Aux Value	<20> for Aux # 0... <1F> for Aux #31
Column Value	Used in Special Commands, see related table in this document
Data Value	For the list of Data values, see related table in this document

Note: For Data values > <60> (96 in decimal), convert the data into a Base 96 number. Add <20> to each digit to get the Hex bytes. This restricts all Data value hex bytes between <20> and <7F>.

<CHECKSUM>

The Checksum is calculated from <01> to the character before the checksum, which should be <1F>.

The formula used is:

```
byte i, nChksum;
byte nTx[256];

for (i = 0, nChecksum = 0; i++) nChecksum += nTx[i]; // Add all hex number before the checksum
                                                     // nTx is the buffer holding all transmit data
nChecksum %= 0x60;    // Modulo of <60> (Remainder when divided by <60>)
nChecksum += 0x20;    // Add <20> to become readable ascii
```

So <CHECKSUM> is the value of nChecksum.

<PROCESS>

The process byte is always <1F>. It tells the device to process the information (Command, Device, I/O, Channel, Aux commands and values) before it.

EXAMPLES

Example #1 – Mute:
<01><57><7F><03><4D><55><54><30><08><20><09><21><0A><22><10><21><1F><2E><02>

To calculate the checksum, use the formula above.
The sum of all hex values in byte = <CE>
<CE> Modulo <60> is <0E>
Add <20> becomes <2E>, therefore CHECKSUM = <2E>.

This string will update the MUT0 command for Device 0, Output 2 (all zero-based). The mute will be turn on as specified by <10><21>.

Example #2 – EQ Frequency:
<01><57><7F><03><45><51><46><30><08><20><09><20><0A><23><0B><25><10><21><1F><44><02>

This string will update the EQF0 command for Device 0, Input 3, Aux 5 (Or EQ#5) (all zero-based).

To calculate the checksum, use the formula above.
The sum of all hex values in byte= <E4>
<E4> Modulo <60> is <24>
Add <20> becomes <44>, therefore CHECKSUM = <44>.

Example #3 – Program Recall:
<01><57><7F><03><25><50><52><30><08><20><10><20><1F><68><02>

This example will recall Program 0 from Device 0 (all zero-based).

To calculate the checksum, use the formula above.
The sum of all hex values in byte = <48>
<48> Modulo <60> is <48>
Add <20> becomes <68>, therefore CHECKSUM = <68>.

COMMAND AND CORRESPONDING VALUE TABLES

AUDIO COMMAND	ID	Data Min	Data Max	Value Min	Value Max	Value Step
Meter	MTR0					
Mute	MUT0	0	1	OFF	ON	
Mix Gain	MIC0	0	15	0dB	45dB	3dB
Signal Level	LVL0	0	220	-40dB	+15dB	0.25dB
Signal Polarity	POL0	0	1	+	-	
Signal Delay	DLY3	0	62400	0ms	650ms	1/96ms
EQ Type	EQT0	0	4	PEQ / LO-SH / HI-SH / AP-1 / AP-2		
EQ Frequency	EQF0	0	29980	20Hz	30000	1Hz
EQ Bandwidth	EQB0	0	359	0.02 oct / 0 deg	3.61oct/179.5 deg	0.01oct/ 0.5 deg
EQ Level	EQL0	0	180	-30dB	+15dB	0.25dB
EQ Bypass	EQb0	0	1	OFF	ON	
GEQ Level	EQL1	0	180	-30dB	+15db	0.25dB
GEQ Bypass	EQb1	0	1	OFF	ON	
Crossover Type	XRT0	0	3	OFF / BUTWRTH / LINK-RI / BESSEL		
Crossover Frequency	XRF0	0	29980	20Hz	30000	1Hz
Crossover Slope	XRS0	0	7	6dB	48dB	6dB
FIR Type	XFE0	0	1	OFF / FIR		
FIR Frequency	XFF0	0	29980	20Hz	30000	1Hz
Compressor Threshold	CMT0	0	80	-20dBu	+20dBu	0.5dBu
Compressor Attack	CMA0	0	106	0.3ms	100ms	0.1ms / 1ms
Compressor Release	CMR0	0	4	2x / 4x / 8x / 16x / 32x		
Compressor Ratio	CMX0	0	39	1	40	1
Limiter Threshold	LMT0	0	80	-20dBu	+20dBu	0.5dBu
Limiter Attack	LMA0	0	106	0.3ms	100ms	0.1ms / 1ms
Limiter Release	LMR0	0	4	2x / 4x / 8x / 16x / 32x		
Mixer	MIX0	0	161	OFF	0.0dB	0.25dB
Channel Name	CHN0	0	84	'_'	';'	

CHARACTER CODE MAP

_ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789abcdefghijklmnopqrstuvwxyz-<>?,./~!@#$%^&*()=+';

SPECIAL COMMAND	ID	COLUMN	Data
Level Increment	#LVL	1 (<21>)	#Steps of 0.25dB
Level Decrement	#LVL	0 (<20>)	#Steps of 0.25dB

SYSTEM COMMAND	ID
Down Sync	%SD0
Up Sync	%SU0
Lock Password	%LP0
Lock Key	%LK0
Channel In Number	%nI0
Channel Out Number	%nO0
Company	%NC0
Sampling Frequency	%nS0
Product Name	%NP0
Version	%NV0
Device Name	%ND0
Device Number	%nD0
Program Number	%Pn0
Program Name	%PN0
Program Recall	%PR0
Program Store	%PS0
Program Download	%PD0
Program Upload	%PU0
Reset	%RS0
Ethernet Info	%EN0
XPanel Info	%XP0

ADVANCED SPECIFIC PROCEDURE

Program Download

1. PC sends Download ON command (%PD0 with data = <21>)
2. PC starts sending each parameter (just like downsync) for program #30
3. PC sends Program Store command (%PS0 with data = 29 (<3D>))
4. PC receives Program Store command (%PS0 with data = 29 (<3D>)) as acknowledgement
5. Repeats step 2 to 4 for the rest of the program in descending order
6. PC sends Download OFF command (%PD0 with data = <20>)

Program Upload

1. PC sends Upload ON command (%PU0 with data = <21>)
2. PC sends Program Recall command (%PR0 with data = 29 (<3D>))
3. PC receives Program Recall command (%PR0 with data = 29 (<3D>)) as acknowledgement
4. PC sends Upsync command (%SU0) to receive data for program #30
   ** Upsync will always send system parameter such as Channel #, Password, etc. Simply ignore them and take the main data part **
5. Repeats step 2 to 4 for the rest of the program in descending order
6. PC sends Upload OFF command (%PU0 with data = <20>), the firmware will automatically reload the data in memory (program #31) upon completion

Downsync (Reset All)

1. PC sends Downsync ON command (%SD0 with data = <21>)
2. PC starts sending a parameter to the firmware
3. PC receives Meter (%MTR0), repeat step 2 for another parameter. When all parameters are sent, go to next step.
4. PC sends Downsync OFF command (%SD0 with data = <20>)
5. PC receives Downsync OFF command as acknowledgement (%SD0 with data = <20>)

Password, Program Name, Device Name

Each character of the string is sent along with the Aux header/value. The value of the character is based on the character code map. For example, a password for “ABCD” would be packed like this:
<01><57><7F><03><25><4C><50><30><08><20><0B><20><10><21><1F><0B><21><10><22><0B>
<22><10><23><0B><23><10><24><4E><02>