

XSYS Controller White Paper

(August, 1 2001)

- The XSYS Controller operating system can be used on 8, 16, 32, 64, or 128 bit architectures.
- The operating system (henceforth referred to as 'xsys os') is a full featured os that incorporates most data acquisition features, and can easily be expanded.
- The programming languages used for the xsys os are a combination of C/C++ and assembly.
- The custom drivers developed for the xsys os were not derived from any existing libraries or source code.
- Engineering behind the xsys os is based on years of solid formal research and development, and has been well documented from the inception date of January, 6 2001.
- Xsyscontroller.com is a trademark of RCS Programming.
- Rcsprogramming.com is a trademark of RCS Programming

The Architecture

The Xsys Controller is (at its basic definition) a data acquisition and distribution system, but should be considered primarily a network device. The Xsys Controller is also what can be referred to as a 'smart' interface, where the 'smart' refers to the embedded xsys os, and the interfacing is typically between a client and host type architecture. The protocol for communication is based on a FIFO method (first in first out) with string 'Terminator', and multi-threading takes place on the host machine that will process the user interface and logic. There are capabilities for multi-threading in future revisions, but it has not been enabled at this time.

The RCS Programming Run Time Editor is a good example of the type of client side host that should be used to provide the user interface and logic that is typically necessary for an integrated system, but any 'socket' capable software will work. This means that the XSYS Controller can be treated like a 'black box' interface for almost any system, and the versatility of the hardware lends this 'box' to be used as the expansion as well. While specific boxes may be developed in the future (IR, RS232, etc.) the cost of the XSYS-1 allows expansion by simply buying another one! The XSYS can also be used in conjunction with other systems because of the open protocol architecture based on a simple string terminated by a logical identifier.

The string format for the protocol was established as a string terminated command to make the communication more versatile, and much easier for debugging. The strings are assembled either on the fly, or hard coded externally from any device capable of transmitting ASCII strings over a network socket using the standard TCP/IP protocols, or HTTP protocols (unencrypted). The complete protocol with examples can be found in this paper, along with the XSYS Controller specifications.

XSYS hardware specifications

(revision 1A)

- 6 outputs that can be used as:
 - o IR Ports (up to 2 emitters per port).
 - o Digital outputs for closure to ground (0 volts) or sourcing up to 200 milliamps of power to power small devices (like relays).
 - o One-way RS232 ports up to 13200 baud (variable).
- 23 Inputs (dry contact or voltage to 12V DC).
- 11 Analog Voltage Inputs (up to 10V DC).
- 4 Analog Outputs (up to 10 volts DC, 40 milliamps).
- 3 RS232 ports up to 115200 baud (three wire only).
- 1 XNET (RS485) port for local expansion network.
- 10BaseT network connection.
- TCP/IP, UDP, HTTP, DHCP protocols.
- The IR ports are capable of sending data up to 60 khz.
- The IR Ports require a buffering resistor. If the IR ports are used for digital output (including one-way RS232) no buffering resistor is required.
- All digital outputs are rated for sourcing up to 200 milliamps of power (i.e. you can steer the equivalent voltage of a small appliance AC adapter).
- All digital inputs have basic protection circuitry, but are limited to 12 volts (or the power supply voltage).
- Network communications is 10BaseT, asynchronous.
- Voltage outputs are limited to 10 volts DC, 40 milliamps.
- Voltage inputs are limited to 10 volts DC, 40 milliamps.
- Inputs 1 to 23 can be used as closure (to ground) or voltage sensing.
- Input 0 is used as the RESET input (short this to ground, and reset power for 'Factory Default' configuration).
- The input switching threshold is 2.4 volts.
- The inputs are 'pulled up' to the sourcing voltage terminal (i.e. 12 volts).

Protocol

- * Note (all 'Bytes' are assumed to be ASCII or UTF-8 characters).
- * All characters transmitted to the Xsys are assumed to be ASCII characters.

Internal Specifications

- Input buffers are limited 512k bytes.
- Output buffers are limited to 1024 bytes (including terminator).
- Baud rates:
 - o 1200
 - o 2400
 - o 4800
 - o 9600
 - o 14400
 - o 19200
 - o 28800
 - o 31250 (Midi)
 - o 38400
 - o 57600
 - o 115200
- Sequences are limited to 32 bytes.

- **NVRAM** is limited to 1024 bytes (including terminator).
- **DOMAIN** names are limited to 32 bytes (not required at this time).
- **HOST** names are limited to 32 bytes.
- IP Addresses (i.e. **NETMASK**, **GATEWAY**, **NAMESERVER**, etc.) are expected to be in the standard notation of 10.0.1.128 (ASCII chars).
- Port number range is limited to 65,535.
- Socket **TIMEOUT** is limited to 65,535 seconds.
- **IRLEVEL** max value = 9.99.
- **DAC** max value = 9.99.
- **VOL** max value = 10.

* **NAMESERVER**, **DOMAIN**, and **GATEWAY** are not required at this time.

Terminators

The terminators are relatively intuitive, and based on a port steering protocol which is similar to a 'file extension' in that the terminator indicates what the system should do with the received data string. An example would be:

"Power ON COM1"

The 'COM1' indicates that the string should be sent to RS232 communications port 1, and everything before that is the data string to be sent. There is no other parsing, and all data that is not the terminator (the space before the terminator is required, and is considered part of the terminator) will be communicated to the RS232 communications port 1 at the assigned baud rate. If a mixture of ASCII, HEX, and/or binary is required, the string needs to be formatted at the host.

/*--- OUTPUT ---*/

(RS232 ports)
COM0 (Output buffer is 1024 bytes)
COM1
COM2

(RS485 Port)
XNET

(IR Ports)
XIR1
XIR2
XIR3
XIR4
XIR5
XIR6

Example: "0,98,0,10,314,187,22,88,22,187,22,88,22,3861,0 **XIR3**"

Byte 1 = Carrier (0 = Disable OFF, 1 = Disable ON)
 Byte 2 = PWM (Calculated value)
 Byte 3 = FM
 Byte 4 = Packet length (IR packet only)

(One way RS232 Ports - variable baud rate from 4100 to 13200)

XSER1
(to)
XSER16

Example: "20,CI102T **XSER8**" (20 = baud rate value or 'timing' value)

(Digital Ouputs)

OUTPUT (Port, Logic) Parameters go first (ex. 1,3 **OUTPUT**)

- The Ouputs can be controlled based on the 'Logic' value:
- 0 = OFF
- 1 = ON
- 2 = TOGGLE
- 3 = PULSE

Example: "1,3 **OUTPUT**" (Pulses output 1)

VOL
DAC
GET

/*--- UTILITY ---*/

DIGIN

DIGOUT

IRTIMER

IRLEVEL

LOOP232

NETID

WRITE

READ

STORE

SEQ1_# (Where # = any stored sequence number from 1 to 4)

SEQ2_#

SEQ3_#

SEQ4_#

RESET

STATUS

CONFIG

/*--- NETWORK ---*/

IPADDRESS

MASTERADDRESS

MASTERPORT

SUBNET

HOSTNAME

GATEWAY

NAMESERVER

DOMAIN

DHCP

PORT

TIMEOUT

/*--- RESERVED ---*/

COMA

COMB

COMC

COMD

(examples - parameters go first)

DELAY 1 COMA (max 9, for milliseconds delay between incoming bytes)

1200 COMA (1200 to 115200 for baud rate)

8BIT COMA

7BIT COMA

EVEN COMA
ODD COMA