ASCII Communication Protocol

Purpose

The following describes details of how to communicate to ModuSystems Controllers using the RS-232 communication channel.

Serial Port Configuration

ModuSystems communicates through the an RS-232 serial port at 9600 baud, no parity, 1 stop bit. The UART supports a 16 bit incoming and outgoing FIFO. There is no hardware or software handshake protocol.

Connecting ModuSystems to a WINTEL PC requires a "null modem" style cable or adaptor. Note that terminal hardware may require loopback or hardware enabling to satisfy host hardware protocol.

Procedure Message Format

In general messages involve 3 letter commands. Some commands are in the form:

<procedure> <parameters> <CR>

There might be no parameters, one, or several depending on the function. Parameters are separated by spaces. The space between the procedure and the first parameter is optional. Adding a space improves readability. The command is terminated with a carriage return (ASCII 13) character. After receiving a command, ModuSystems will transmit the following information:

<CRLF> <error number> <return value> <prompt>

The prompt is the "greater than" symbol. The error number indicates if the command was completed successfully. A "0" indicates a success, other numbers represent errors. If the first number is not 0, the remaining information in the response is not valid. After the error number is *a* space followed by the return value of the command. All commands have return values. Most return values reflect controller state. Some return values confirm information that was sent when no specific return value is associated with the command. The response terminates with the "greater than" symbol as the default prompt.

Hooking up a terminal, or using a terminal program on the PC permits exercising the board and viewing this transaction behavior. Typing just the "CR" key produces a null command which returns 0 for error and 0 for return value:

```
> {type "enter"}
0 0>
```

The command RWD (Reset Watchdog) which has no parameters can be typed with the following response:

```
>RWD {enter}
0 1>
```

The first 0 indicates no command syntax error was encountered. The "1" return value indicates that the command did indeed reset the watchdog. The value "1" corresponds to "true" and "0" corresponds to false.

An example of a command which requires parameters would be the output command, SetOutputBit (SOB). The first parameter is the bit number and the second the desired output value.

```
> SOB 27 1
0 1>
```

Parameters are separated by spaces or commas.

Object Message Format

Another message form reflects the "object oriented" nature of the on-board software system and has the following form:

<receiver> <verb> <parameters>

The receiver can be either a single motor axis or a coordinated group of axes. Individual axes are presented as the "A" array numbered 1 to 16 (A is for "Axis"). An example of an object message to an individual axis would be the MTT command, SetMotorType:

```
>A1 MTT SRV
0 0>
```

Here the "receiver" is A1, axis 1, the first "element" in the 16 xis array. MTT is the "verb" which acts on the "receiver". SRV is a named constant. The parameter could have been a number also.

Groups of axes are represented as the "C" array numbered 1 to 10 ("C" is for Coordinated group). Before a group can be directed, it must be described with the INI

command. The INI command takes as parameters the axis numbers that constitute the coordinated group. For example, a two axis coordinated group could be associated with group 1 as follows:

```
>C1 INI 1 2
0 2>
```

Ten groups are available. Here the first group is being described as running the first and second axis on the controller board. The answer, "2" corresponds to the dimension of the group which should be the same as the number of parameters provided.

Many commands apply to both individual axes as well as to groups. For example, turning on the motors for axis 5 and for group 1 would be done with the same command, just different receivers:

```
>A5 MTR ON
0 1>C1 MTR ON
0 1>
```

Performing motion requires configuring the board for the motors being used, setting motion attributes, turning motors on, and directing their motion.

Command Reference

Command Summary

IO Operations

INB..... InputBit ..... Return level of specified input
CIO ..... ConfigureIOBitAsOutput . Instructs I/O bit to behave as output signal
SOE ..... SetOutputEnable ..... Tell output bits to become active
SOB ..... SetOutputBit..... Change state of output bit on hardware

Safety

RWD..... ResetWatchdog ..... Allow tripped safety system to resume servo activity
WHT ..... WatchdogHasTripped ... Returns status of watchdog system
UHD ..... UserHasDisabled ..... Indicates if any disable input is asserted

Axis and Coordinated Group Commands

Attributes

INI ..... Init ..... Associate axes into coordinated group
DSP..... Dispose ..... Release axis group relationship
RSA..... ResetAllocation ..... Clear all group relationships
MTT..... MotorType ..... Configures motor for servo or stepper operation
ENA..... Enable ..... Allow amplifier to power motor
MTR ..... Motor ..... Turns motor operation on and off
LIV ..... LoopInversion ..... Include an additional sign inversion in control law
CIV ..... CoordinateInversion .... Reverse which way is regarded as the positive direction
ACL..... Accel ..... Set acceleration rate for trapezoidal moves
DCL..... Decel ..... Deceleration rate for trapezoidal moves
SPD..... Speed ..... Set speed of slew phase of trapezoidal moves
GAI ..... Gain ..... Set compensation parameter for servo
ZER..... Zero ..... Compensation parameter for servo
ITG..... Integrator ..... Compensation parameter to eliminate steady state error
ERL..... ErrorLimit ..... Set permissible tracking error before disable occurs
PLT..... PositiveLimit ..... Set boundary for movement in the positive direction
NLT..... NegativeLimit ..... Set boundary for movement in the negative direction
ACP ..... ActualPosition ..... Define current position coordinate
CAP ..... CapturePosition ..... Return position recorded when latch event occurred
COP ..... CommandedPosition ... Set commanded position for non-trapezoidal moves
DEP ..... DestinationPosition ..... Return absolute coordinate of end of move
ERP ..... ErrorPosition ..... Return discrepency between current and ideal position
AIC ..... ArmInputCapture..... Prepares axis to latch position based on input signal
AXC ..... ArmIndexCapture..... Prepares axis to latch position based on index signal
TRQ ..... CommandedTorque..... Set output voltage when not servoing
PFV ..... ProfileVelocity ..... Return current ideal profile velocity

Motion

MVT ..... MoveTo..... Move to absolute coordinate

MVB..... MoveBy ..... Move to relative coordinate
BMT ..... BeginMoveTo ..... Start move to absolute coordinate
BMB ..... BeginMoveBy ..... Start move to relative coordinate
MAC..... MoveAlongCurve ..... Perform coordinated multiaxis motion along curve
BMC..... BeginMoveAlongCurve . Begin coordinated curved motion
AMT ..... AppendMoveTo ..... Add absolute vector segment to curve description
AMB ..... AppendMoveBy ..... Add vector segment to curve relative to last segment
ARC ..... AppendArc ..... Add circular or helical arc to continuous path curve
CLR ..... Clear ..... Erase any established motion curve info
LNK..... LinkToBuffer ..... Assocate curve buffer with axis group
JOG ..... Jog ..... Move indefintely at constant speed
STP ..... Stop ..... Gently stops any motion that may be in progress
BST..... BeginStop ..... Begins to stop but immediately does next instruction
ABT ..... Abort ..... Suddenly aborts any motion that may be in progress
CHT ..... CaptureHasTripped ..... Indicate if latch event has occurred
MIF..... MoveIsFinished ..... Return true if move has finished

User Task Control

BUT ..... BeginUserTask ..... Spawn independent application behavior in controller
AUT ..... AbortUserTask ..... Terminate an independent behavior in controller
SUT..... ScheduleUserTask..... Spawns independent period application behavior
PUT..... SuspendUserTask..... Cause task in controller to become inactive
RUT ..... ResumeUserTask ..... Cause suspended activity to become active again
UTP..... UserTaskPresent ..... Indicates if particular task is currently present in controller

User Variable Control

USB ..... UserBoolean ..... Manipulate user boolean in controller
USL ..... UserLongint ..... Manipulate user longint in controller
USS ..... UserSingle ..... Manipulate user single in controller