**Model 4100**

**Isolated High Power Stimulator**

#### Matlab Interface Source Code

#### Revision 4.0

Table of Contents

[Introduction 1](#_Toc88139673)

[AMS\_4100 communication protocol 2](#_Toc88139674)

[Example Communication Files 6](#_Toc88139675)

[GUI Communication Files 7](#_Toc88139676)

[Example GUI 10](#_Toc88139677)

# Introduction

A-M Systems Models 4100 stimulator functions are set with commands sent from the front panel or from a computer over Ethernet or USB. Computer communication over USB relies on drivers from FTDI that will automatically install if connected to the internet. Ethernet communication is done using the telnet protocol.

This document briefly describes the Matlab files written by A-M Systems, and is divided into four sections. The first section describes the AMS\_4100 communication protocol and data structure. This is the basic communication structure used in the basic communication Example files and the GUI example. The second section describes the two example files and show how to set up basic communication with the instrument. The third section describes the two main communication files that the GUI uses to communicate with the instrument. The last section describes the example GUI (ams4100\_GUI.m) and its many supporting m files.

# AMS\_4100 communication protocol

## Introduction

Both transmit and receive protocols operate with commands which are a series of string “words” followed by a carriage-return 0x0d. Words are delimited by “whitespace” which is either 1 or more comma or blanks. If a word is one of the “reserved words defined in the table below, the stream just has to contain enough characters to uniquely specify it. For example, the word “get” can be specified by “g”, “ge” or “get”. Commands executed without error return an asterix (\*) after the response if one is required. Commands with errors return a question mark (?)

All SET commands must be prefaced with the individual instruments pin number. This allows the instrument to have secure communication if necessary. This number solely defined on the instruments front panel. The pin can be set by navigating the Configuration page and selecting the network setup. The instrument will come with a default pin number of 1001.

**Timing starts with the command “set active run”. Timing stops with the command “set active stop”. Any changed value will not take effect until timing is restarted (or if already stopped then started).**

If amplitude values entered are above 50V then a “dangerous voltage” message will be sent to the computer with a set active start command and the program must respond with an OK.

## Command Syntax

| **1st Word** | **2nd Word** | **3rd Word** | **4th Word** | **5th Word** | **Comments** |
| --- | --- | --- | --- | --- | --- |
| get | revision |  |  |  | Returns firmware revison |
|  | active |  |  |  | Returns the current status from this list: “Unknown”,  “Ready monitor”,  “Ready low output”,  “Ready HIGH OUTPUT”,  “Waiting for gate or Trigger”,  “Monitoring pulses”,  “Generating pulses”,  “MON. UNSAFE PULSES", or  “DANGEROUS PULSES” |
|  | network |  |  |  | Returns the IP address, IP Mask, and IP gateway |
|  | menu | menu # | item # |  | Returns the instrument setting for the specified menu # and item #.  Values are returned in signed 64 bit integers, representing uV, uA, or us.  For example -21V = -21,000,000  See Menu Tables Below. |
|  | condition |  |  |  | Returns two characters representing several instrument internal switches,  First character bits:  Bit 7 always =0, Bit 6 always =1, Bit 5 always=0,  Bit 4=1 if >200V,  Bit 3=1 if >100uA,  Bit 2=1 if FPGA is generating,  Bit 1=1 if the FPGA is loaded, waiting/generating  Bit 0=1 if the enable button is depressed.  Second Character bits:  Bit 7 always =0, Bit 6 always =1, Bit 5 always=0,  Bit 4=always 0  Bit 3=always 0  Bit 2=1 if the relay is open  Bit 1=1 if the Front panel is on free run  Bit 0=1 if the front panel has been changed |
|  |  |  |  |  |  |
| *PIN* Set | active | run/stop |  |  | Either starts or stops the generation of pulses. |
|  | network | IP address | mask | gateway | String in "format" separated by "whitespace" |
|  | display | menu # |  |  | Sets the LCD display on the front panel to the value of menu |
|  | menu | menu # | item # | value | Sets the value of the item in menu to the value. Value is a 64 bit signed integer value in uV,uA, or us. Microseconds have only positive values. |
|  | trigger | none/ one/ free-run |  |  | Generates an output trigger:  \*free-run starts output without trigger,  \*none cancels free-run and system waits for a  hardware trigger  \*one generates a single trigger |
|  | Relay | Open/ Close |  |  | Opens or closes the relay on the output of the instrument. |

Examples

Note: <CR> represents a carriage return (ascii hex 0x0d or decimal 13):

<LF> represents a line feed (ascii hex 0x0a or decimal 10):

Get the firmware revision

“get rev<CR>” or “g r<CR>”

10The instrument will return

“get rev<CR>M1\_F1<CR><CR><LF>M1\_F1<CR><LF>\*<CR><LF>”

Get Event Type for Library 2

“g m 11 2<CR>”

The instrument will return:

“g m 11 2<CR><CR><LF>1<CR><LF>\*<CR><LF>”

Set Event Type to Ramp for Library 1 (assuming the *PIN* is ‘1001’)

“1001 s m 10 2 3<CR>”

The instrument will return:

“1001 s m 10 2 3<CR><CR><LF>\*<CR><LF>”

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Menu Name** | **Menu #** | **Item #** | **Item Name** | **Values** | | | | | | | | |
| General | 0 | 0 | Mode | 0 = Int Volt | 1 = Int Current | 2 = Ext 20V/V | 3 = Ext 10 ma/V | 4 = Ext 1 ma/V | 5 = Ext 100 uA/V |  |  |  |
|  |  | 1 | Monitor | 0 = 0.1V/V | 1 = 1V/V | 2 = 10V/V | 3 = 20V/V | 4 = 10uA/V | 5 = 100uA/V | 6 = 1mA/V | 7=10mA/V |  |
|  |  | 2 | Trig | 0 = Rising | 1 = Falling |  |  |  |  |  |  |  |
|  |  | 3 | Auto | 0 = None | 1 = Count | 2 = Fill |  |  |  |  |  |  |
|  |  | 4 | Save | Save the settings on the instrument | | |  |  |  |  |  |  |
|  |  | 5 | Output | 0 = On | 1 = Off | (leaves the output enabled but at 0V or 0A) | | |  |  |  |  |
| Configuration | 1 | 0 | Rates | 0 = Period | 1 = Frequency |  |  |  |  |  |  |  |
|  |  | 1 | Sync1 | 0 = TrainDel | 1 = TrainDur | 2 = EvDel | 3 = EvntDur1 | 4 = EvntDur2 | 5 = EvntDur3 | 6 = EvntTotalDur | 7 = Clock-us | 8 = Clock\_ms |
|  |  | 2 | Sync2 | 0 = TrainDel | 1 = TrainDur | 2 = EvDel | 3 = EvntDur1 | 4 = EvntDur2 | 5 = EvntDur3 | 6 = EvntTotalDur | 7 = Clock-us | 8 = Clock\_ms |
| UniformEvent | 4 | 0 | Library # | Integer value: | 1 to 20 |  |  |  |  |  |  |  |
| Train | 7 | 0 | Type | 0 = Uniform | 1 = Mixed |  |  |  |  |  |  |  |
|  |  | 1 | Delay | Time value 0 to 90,000,000,000us steps of 1us | | | |  |  |  |  |  |
|  |  | 2 | Durat | Time value 2 to 90,000,000,000us steps of 1us | | | |  |  |  |  |  |
|  |  | 3 | Period | Time value 2 to 90,000,000,000us steps of 1us | | | |  |  |  |  |  |
|  |  | 4 | Number | Quantity 0 to 99999 steps of 1 | | |  |  |  |  |  |  |
|  |  | 5 | H/O | 0 = Hold | 1 = Offset |  |  |  |  |  |  |  |
|  |  | 6 | Level | Amplitude value -200,000,000 to 200,000,000 steps of 1. 1uV or 1uA | | | | |  |  |  |  |
| Event List | 8 | 5 | Event 1 | Library number for corresponding Event 1 to 20 steps of 1 | | |  |  |  |  |  |  |
|  |  | 6 | Event 2 |  |  |  |  |  |  |
|  |  | **…** | **…** |  |  |  |  |  |  |
|  |  | 14 | Event 10 |  |  |  |  |  |  |
|  |  | 23 | Event 11 |  |  |  |  |  |  |
|  |  | 24 | Event 12 |  |  |  |  |  |  |
|  |  | **…** | **…** |  |  |  |  |  |  |
|  |  | 32 | Event 20 |  |  |  |  |  |  |
| Library # 1-20 | 10 to 30 | 2 | Type | 0 = Mono | 1 = Biphase | 2 = Asym | 3 = Ramp |  |  |  |  |  |
|  |  | 3 | Delay | Time value 0 to 90,000,000,000us steps of 1us | | | |  |  |  |  |  |
|  |  | 4 | Number | Quantity 0 to 99999 steps of 1 | | | |  |  |  |  |  |
|  |  | 5 | Period | Time value 2 to 90,000,000,000us steps of 1us | | | |  |  |  |  |  |
|  |  | 6 | Duration 1 | Time value 1 to 90,000,000,000us steps of 1us | | | |  |  |  |  |  |
|  |  | 7 | Amplitude1 | Amplitude value -200,000,000 to 200,000,000 steps of 1. 1uV or 1uA | | | | |  |  |  |  |
|  |  | 8 | Interphase | Time value 0 to 90,000,000,000us steps of 1us | | | |  |  |  |  |  |
|  |  | 9 | Duration 2 | Time value 0 to 90,000,000,000us steps of 1us | | | |  |  |  |  |  |
|  |  | 10 | Amplitude2 | Amplitude value -200,000,000 to 200,000,000 steps of 1. 1uV or 1uA | | | | |  |  |  |  |

# Example Communication Files

## EXAMPLE\_1\_USB.m

This file goes through step by step instructions to send data to the instrument using a USB/Serial port connection from a computer running Matlab. The instrument must be turned and and connected via a USB cable to your computer running Matlab. The USB connection uses an FTDI driver that should be automatically installed when connected to your computer. Your can also download the USB Serial Converter driver files from ftdichip.com ([VCP Drivers - FTDI (ftdichip.com)](https://ftdichip.com/drivers/vcp-drivers/) )

## EXAMPLE\_1\_Ethernet.m

This file goes through step by step instructions to send data to the instrument using an ethernet port connection from a computer running Matlab. Matlab needs to know how you set your IP address on the instrument, and the computer running Matlab need to be able to find that IP address, typically this is solved by putting the computer and instrument on the same network. The instrument does not use DHCP.

# GUI Communication Files

## ComConstants.m

This file contains a structure of constants that define the instrument variable locations, and values that represent variable settings. For example to set the train type to mixed the string sent would be “set menu 7 0 1”. This is represented by:

“*PIN* set menu constants.menu.train constants.train.type value.train.type.mixed”

## ams4100\_hClass.m

This file is a handle class that has properties and get set functions for each instrument variable. There are also methods for starting/stopping the instrument and sending trigger values. The class needs to be instantiated with three input values.

1. a port to communicate with the instrument.

The port is a string that defines the serial port ( ‘COM’ )

or an IP address ( ’10.0.0.80’).

1. The second value is the instruments PIN number set on the front panel.

The pin number defaults to 1001. If you choose a different pin number you need to include this variable

1. The third value is used for debugging, it defaults to TRUE, if it is set to FALSE then there will be no communication with the instrument, and no the error messages will be suppressed.

Example: inst=ams4100\_hClass('COM9',1001)

ams4100\_hClass with properties:

DoComms: 1

Active: 'Ready low output'

Output: 'intVolt'

Trigger: 'rising'

Auto: 'none'

Monitor: 'scale\_1VperV'

Sync1: 'eventDuration1'

Sync2: 'eventDelay'

PeriodOrFreq: 'period'

TrainType: 'mixed'

TrainDelay: 2

TrainDur: 600000

TrainPeriod: 1000000

TrainQuantity: 1

TrainLevel: 0

TrainFrequency: 0

OffsetOrHold: 'offset'

EventID: 1

LibID: 1

EventType: 'ramp'

EventDelay: 0

EventDur1: 200000

EventDur2: 200000

EventDur3: 200000

EventPeriod: 1000000

EventQuantity: 1

EventFrequency: 0

EventAmp1: 5000000

EventAmp2: -5000000

EventList: [3 2 1 4 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1 -1]

UniformNumber: 3

HighVflag: [0 1]

HighIflag: [0 0]

Generating: [0 0]

Running: [0 0]

EnableButtonIn: [0 0]

PIN 1001

PortSuccess: 1

Revision: 'M1,F1'

Network: '10.0.0.80'

SerialNumber: '0'

values: [1x1 struct]

loading: 0

Port: 'COM9'

PortInfo: [1x1 serial]

ActiveComms:

Setting or getting a variable is done by using the instantiated class:

EX: >> inst.EventDur1

out=: g m 10 6 reply = g m 10 6~200000~\*

ans =200000

EX: >> inst.EventDur3=10000;

out=: *PIN* s a stop reply = *PIN* s a stop~\*

out=: *PIN* s m 10 9 10000 reply = *PIN* s m 10 9 10000~\*

out=: *PIN* s a run reply = *PIN* s a run~\*

As can be seen the command window will display the communication back and forth from the instrument.

To set values for a library position follow these steps:

1. Set the library number to the one to be changed :

inst.LibID=3; this will set the library to be changed to position 3

1. Set all the event parameters to the desired values:

inst.EventType= 3; 0 = Mono, 1 = Biphase, 2 = Asym, 3 = Ramp

inst.EventDelay: 0; in microsecond

inst EventDur1: 500; in microsecond

inst .EventDur2: 0; in microsecond

inst .EventDur3: 500; in microsecond

inst .EventPeriod: 2000; in microsecond

inst .EventQuantity: 1;

inst .EventAmp1: 12000000; in microvolts

inst .EventAmp2: -5000000; in microvolts

To view the values of any library positions follow these steps:

1. Set the library number to the one you want to view :

inst.LibID=1;

1. Then look at any event value:

inst .EventAmp1

“out=: g m 12 7 reply = g m 12 7~9000000~\*

ans =

9000000”

inst.LibID=3;

inst .EventAmp1

“out=: g m 10 7 reply = g m 10 7~12000000~\*

ans =

12000000”

To have outputs of more than one event type Train Type must be in ‘mixed’ mode and the EventList must have more than one defined library number.

1. Set TrainType

inst. TrainType=1;

1. Set the Event List to some pattern

Inst.EventList=[1 2 1 1 2];

If LibID 1 and 2 have been defined the output will have event 1 followed by 2 followed by 1 then 1 then 2.

To disable the output execute the command:

Inst..Stop;

“out=: 1001 s a stop reply = 1001 s a stop~\*”

To enable the output execute the command:

Inst.Run;

“out=: 1001 s a run reply = 1001 s a run~\* ~\*”

To send the current configured Train output once execute the command:

Inst. .GoOnce;

“out=: 1001 s t one reply = 1001 s t one~\* “

To send the current configured Train output continuously execute the command:

Inst. GoFreeRun

“out=: 1001 s t free reply = 1001 s t free~\* “

To stop the continuously output execute the command:

Inst. StopFreeRun

“out=: 1001 s t none reply = 1001 s t none~\*”

To control the front panel enable Relay (timing continues) issue the command

Inst. IsoOutput=0 ‘ the output relay is eneabled

“out=: 1001 s m 0 5 0 reply = 1001 s m 0 5 0~\*”

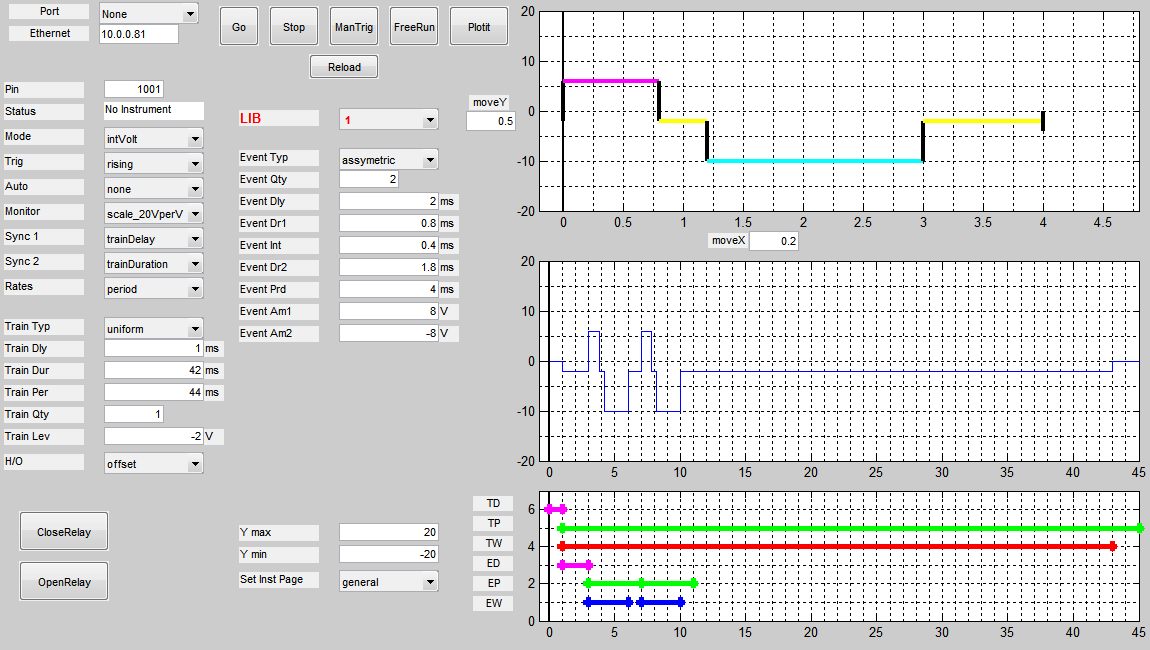
Inst. .IsoOutput=1 ‘ the output relay is disabled

“out=: 1001 s m 0 5 1 reply = 1001 s m 0 5 1~\*”

# Example GUI

The main file for the GUI is ams400\_GUI.m. It MATLAB file that creates the Model 4100 user interface with no inputs or outputs to the function. This is just an example and should be considered a beta program.

Example:



Required functions that must be in the same folder or in your MATLAB path.

CLASSES:

ams4100\_hClass.m - The class that communicates with the instrument

ComConstants.m - The sturcture with that enumerates the constats.

FUNCTIONS:

checkTimes.m - verifies AMS4100 time values are valid. Returns string error.

controlsGUI.m - Cell of GUI cotrols, each row has Name, Label, and type

defaultData.m - Fills the AMS4100 GUI with default data.

DrawBiphasic.m - Draws the interactive Biphasic Event Graph

DrawMonophasic.m - Draws the interactive Monophasic Event Graph

DrawRamp.m - Draws the interactive Ramp Event Graph

getOffsetAmps.m - Returns the pre-train, train, amp1 and amp2 values

based on the train event level and offset type.

getTimeValues.m - Returns the event time values for the desired event.

getValueNames.m - Generates a valid variable name based on the

comConstants stucture.

IDnum.m - Processes changes to the EventID, LibID and EventList

uicontrols

LoadWindow.m - Loads the window uicontrols with data from the instrument,

or local Data if there is no instrument connection

Plotit.m - Plots the time response for the events.

processUserInput.m - takes a change to the uicontrol, and updates the

localData and sends info to the instrument.

RetAmp1.m and RetAmp3.m - Given the graphical Y value of a line it

returns the Event aplitude

SetCom.m - Sets the serial port communication for the instrument

SetTimeAMS.m - Takes the time values from the instrument and converts

it to a string for the uicontrols

timeNum.m - Takes a uicontrol time value and makes sure it is in the

appropriate range

trainNum.m - Takes a uicontrol train number and makes sure it is in the

appropriate range

UpdateEvents.m - Updates the uicontrol event data when there is a

change to the libID, or EventID.