

# **Agilent 35900e OpenLab Protocol Documentation**

Volker Mauel  
Head of Technology

September 16, 2023

# 1 Overview

## 1.1 The two services


Initial communication from an OpenLab AIC to a 35900E happens in two stages. The first stage utilizes the Telnet Service, the second stage uses the instrument service.

### 1.1.1 Telnet Service


The telnet service (listening on Port **23**) allows users to configure basic instrument parameters for integration into their environment. In here, there IP Address, Subnet Mask and default route can be configured.

After establishing the connection, the device answers with the following header:

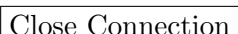
```
\r\n===\xff\xfc\x01\r\n
Agilent_35900_Series_II\r\n
Please_type_"?"_for_HELP,_or_"_"_for_current_settings\r\n
```

During the Action **Get Serial Number and Firmware**, the AIC sends the  Command via Telnet, and the device answers with the following information:

```
_ _ _===JetDirect_Telnet_Configuration===\n\r
Firmware_Rev.:_E.02.04.32\r\n
MAC_Address:_00:aa:bb:cc:dd:ee\r\n
Config_By:_USER_SPECIFIED\r\n\r\n
IP_Address:_1.2.3.4\r\n
Subnet_Mask:_255.255.255.0\r\n
Default_Gateway:_1.2.3.1\r\n
DHCP_Config:_Disabled\r\n
>
```

The  command closes the socket and the connection switches over to the Instrument Service listening on Port **9100**.

## 2 States and transitions

Figure 1 shows an overview of the possible instrument states and the transitions between them. Notably for an instrument once in the **offline**-State during a run, it can only come back into **idle** by using the  Button in the OpenLab Software and reconnecting all devices at once.

## 3 Instrument Service detailed command documentation

During the startup of the device, the clock is initialized. This is a relative timer that is being used throughout the protocol.

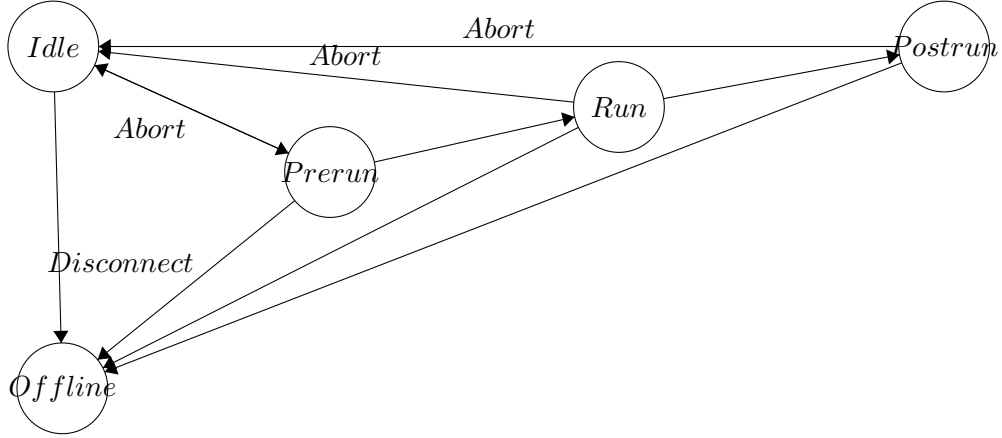


Figure 1: Possible States for the Module and their transitions

```
var_start_time=datetime.datetime.now()
```

Additionally, only the "Run" state is currently implemented in the code. The other states should behave accordingly.

### 3.1 Run initialization

The nomenclature differs between the OpenLab software and the instrument protocol:

Protocol	OpenLab
AXPRE	Prerun
AXINTO	Running
AXPOST	Postrun

During initialization of a run, a timetable is being sent to the instrument. Example with comments added - normally each line is only terminated by `\n` without further whitespaces/newlines in between:

Listing 1: Timetable initialisation

```
#TimeTable - Conditional - Row -- AXINTO - requires - the - command - AR_START - (ARST)
TTCR-AXINTO, -AR_START\n
```

```
#TimeTable - Conditional - Row -- AXPOST - requires - the - command - AR_STOP - (ARSP)
TTCR-AXPOST, -AR_STOP\n
```

```
#TimeTable - Operation -- For - AXINTO - the - previous - State - is - AXPRE
TTOP-AXINTO, -0; -TTSP-AXPRE\n
TTOP-AXINTO, -0; -TTDS-AXPRE\n
```

```
#TimeTable - Operation -- For - AXPOST - the - previous - State - is - AXINTO
```

```
TTOP-AXPOST, - 0; -TTSP-AXINTO\n
TTOP-AXPOST, - 0; -TTDS-AXINTO\n
```

```
#TimeTable-Operation -- Set-AXINTO-planned-runtime-to-180000ms(=3-minutes)
#Behave-like-receiving-the-ARSP(=Stop)-command-afterwards
```

```
TTOP-AXINTO, - 180000 ; -ARSP\n
```

```
#TTOP-SY(system)-NO(-op) -- do-nothing-in-AXPRE, -disables-this
TTOP-AXPRE, - 0; -SYNO\n
```

```
#TimeTable-Entry -- activates-the-commands-from-before
```

```
TTEN-AXPRE\n
TTEN-AXINTO\n
TTEN-AXPOST\n
```

When this message is received, the implementation shall take note of the highlighted value, as this is the runtime in milliseconds.

## 3.2 AREV

The **AREV** command is responsible for returning a Reference Time after an injection. The exact moment of setting this time depends on the instrument receiving the ARST command. Request:

Listing 2: Request (AIC→ Instrument)

```
AREV\n
```

Responses:

Listing 3: Response (Instrument→ AIC) - No injection yet

```
AREV-NONE; -NONE\n
```

Listing 4: Response (Instrument→ AIC) - Injection happend and run ongoing

```
AREV-HOST, - 123456 , - 223; -NONE\n
```

The highlighted value (123456) is calculated as follows:

Listing 5: calculation of injection time

```
# returns the measurement start time in milliseconds
injection_time = (time_of_measurement_start - start_time).
total_seconds() * 1000
```

Listing 6: Response (Instrument→ AIC) - Run finished

```
AREV-HOST, - 123456 , - 223; -HOST, - 789123 , - 255\n
```

After a run, the end is signaled from the instrument to OpenLab / AIC by the AREV command also. In this example, the run started at 123456ms after instrument start and ran until 789123ms after instrument start (=665667ms). So AXINTO was most likely set to be around that number.

### 3.3 ARST

The **ARST** command starts a run and sets the reference time in order to align signals in the OpenLab online view.

```
var time_of_measurement_start = datetime.now().millis
var running = True
```

### 3.4 ARSP

The **ARSP** command ends a run. In regular runs this is most likely not used or only used as an internal placeholder. In regular runs, when the time set from the timetable elapses, the AIC requests timestamps by **AREV**, knows the run has ended, and officially end the run by sending additional **ARGR** commands.

Else, this is the **ABORT** command to immediately stop a run.

### 3.5 AVSS

Requests the System Status for Channel A values. Since this is one of the more complex commands, multiple colors will be used in the listing and explained further down.

Listing 7: Request (AIC→ Instrument) System State Channel A

```
AVSS\n
```

Listing 8: Response 1 (Instrument→ AIC) System State Channel A

```
AVSS-ON,-0,-5,-1,-123456789\n
```

The **status text** can only be ON or OFF. In general this is ON, with OFF only used for a short time after a measurement. The **status code** has 3 different values. When not in an active measurement, this will be 0. In an active run, this value changes to 5, except shortly after a run, where this changes to 14. The **queue size** contains the number of values currently buffered on the instrument side and ready to be received. This is 0 when no values are ready to be received, otherwise greater than 0. The **time** returns the elapsed milliseconds since the instrument has been turned on. This value shall only increase in one session, otherwise there is an error.

### 3.6 AVDF

The **AVDF** command requests the preparation of values from the queue to be requested via **AVRD** shortly.

Listing 9: Request (AIC→ Instrument) System State Channel A

AVDF-HEX, - 3\n

Requests the preparation of 3 values. The previous AVSS command must have returned at least 3. This value needs to be saved until the next AVR D command.

### 3.7 AVR D

Reads the prepared number of values from the instruments measurement queue. This should be seen in connection with the **AVSS** and **AVDF** commands.

Listing 10: Request (AIC→ Instrument) Read Values from Queue

AVRD\n

Listing 11: Response (Instrument→ AIC) Read Values from Queue

AVRD-HEX, - 002 ; 0023F73C 0023F725\n

### 3.8 ARSS

The **ARSS** command requests the channel A RUN SYSTEM STATE.

Listing 12: Request (AIC→ Instrument) Read Values from Queue

ARSS\n

Listing 13: Response (Instrument→ AIC) Read Values from Queue

ARSS- statustext , - statuscode\n

The **statustext** is either READY in regular online mode, RUN during a measurement or NOT\_READY shortly after a measurement until a **ARGR** command is received. The **statuscode** in these cases is 0 for READY, 5 for RUN and 14 for NOT\_READY. These status values are not user-facing and only for internal handling of state.

### 3.9 AVSL

Requests or sets the value sampling rate

Listing 14: Requesting (AIC→ Instrument) Sampling rate currently set

AVSL- ?\n

Listing 15: Response (Instrument→ AIC) Sampling rate currently set

AVSL- 1000\n

The value 1000 in this case means once per second (1Hz).

Listing 16: Setting (AIC→ Instrument) Sampling rate

```
AVSL-1000\n
```

Listing 17: Response to setting Sampling Rate (Instrument→ AIC)

```
AVSL-1000\n
```

For a sampling rate of 10Hz, the value will be 100 etc.

### 3.10 ARGR

The **ARGR** command is being sent shortly before an injection is happening and can be seen as a "GET READY" method. There is no expected answer from the Instrument to the AIC to this command.

### 3.11 ATRD

The **ATRD** command is a basic PING/PONG mechanic.

Listing 18: Request (AIC→ Instrument)

```
ATRD\n
```

Listing 19: Response (Instrument→ AIC)

```
ATRD-255\n
```

### 3.12 ARBM

The **ARB**M command requests configures the button mode for channel A.

Listing 20: Request (AIC→ Instrument)

```
ARB?-?\n
```

Listing 21: Response (Instrument→ AIC)

```
ARB-OFF, -OFF\n
```

Setting the button mode works accordingly

Listing 22: Request (AIC→ Instrument)

```
ARB-OFF, -OFF\n
```

Listing 23: Response (Instrument→ AIC)

```
ARB-OFF, -OFF\n
```

### 3.13 TTSS

The **TTSS** command requests the timetable system status

Listing 24: Request (AIC→ Instrument)

```
TTSS state \n
```

The **state** here is either AXPRES, AXINTO or AXPOST as defined above.

Listing 25: Response (Instrument→ AIC)

```
TTSS state , - current-status , - currenttime , - plannedtime \n
```

The **current-status** is either ENABLED if this state should be running after ARST, DISABLED if this state should be skipped, or RUNNING if this is the currently active state. The **currenttime** is milliseconds since Start of the RUN (not instrument!). The **plannedtime** is what gets set in the init phase (see 3.1).

In our implementation, only the Prerun and Run state are actively used.

If no run is active, the response to a request will always be

Listing 26: Response (Instrument→ AIC)

```
TTSS state , - current-status , - DISABLED , - 1 , - 0 \n
```

Additionally, if this is our internal trigger to set a RUN\STOPTIME and set the **ARSS** state to NOT\READY until **ARGR** is received again. By setting RUN\STOPTIME, the next **AREV** request will return that the run ended.

NOTE: In our implementation, we keep the RUNNING state until ARGR is received.

### 3.14 SYSN

The **SYSN** command returns the system Serial Number

Listing 27: Request (AIC→ Instrument)

```
SYSN \n
```

Listing 28: Response (Instrument→ AIC)

```
SYSN -LIFERADIO1 \n
```

### 3.15 SYID

The **SYID** command returns the system ID and Firmware.

Listing 29: Request (AIC→ Instrument)

```
SYID \n
```

Listing 30: Response (Instrument→ AIC)

```
SYID -HP35900E , -Rev -E.02.04.32 \n
```