Mini Gateway 100

USER MANUAL

ART Logics Ⓒ Shanghai

1 Content

2 Introduction

The **Mini Gateway 100** is an embedded test sequencer able to run fully configurable test scenarios controlling different devices and instruments. Equipped with **CAN**, **Ethernet** and **UART** data interfaces, it allows the user to integrate a wide range of instruments into its test environment.
In a test environment, The Mini Gateway 100 establishes a network connecting the attached instruments with different data interfaces, in which it acts as a **master** running the test sequences by executing actions, sending commands and retrieving results from the instruments in a **Real-Time** and **synchronized** manner.
About the configuration of the test sequences, a defined **ASCII protocol** of request/response is used to define the test scenarios and then upload them to the Gateway. The user can create the configuration of the test sequences manually with the defined protocol or use the user-friendly **Test Manager Software** provided.
In addition, the Gateway can be used as a **standalone** device via control buttons and switches to run the test sequences saved in the built-in **SD Card**.

3 System Architecture

The Test System architecture is composed of a **Control Computer**, The **Mini Gateway 100** and one or more **instruments**, in the form of a network of test devices executing real-time test plans. The Mini Gateway 100 is the only master node and can access to any device connected in the network through the three data interfaces (Ethernet, UART and CAN).
The communication between the Control Computer and the Gateway is through the **Ethernet** interface with configurable IP address and Port number. The user can use a multiple of Mini Gateway 100 devices considering the Control PC Software can handle more than one Gateway. Each Gateway has a **unique address**, and depending on the message's header, the specific Gateway will accept and respond to the message. As different Gateways cannot share the same address, only one Gateway can accept a Control Computer's message if this one is addressed to the related Gateway. Also, an **acknowledgement** is always returned.
The communication between the Gateway and the instruments connected to its network is done via three data interfaces: **CAN**, **Ethernet** and **UART** with configurable parameters. These parameters will be explained in details in a coming section.

3.1 Communication Control to Gateway

To send a **command** from the Control Computer to the Gateway, the header of the message has to start by '@', followed by the address of the Gateway and then followed by two 'XX' or '11' depending on the command type.

The Gateway parses the **first characters** of the message. If the first one is a request character ('@') and the two followings is its address, it will accept the message. Otherwise, the message will be **rejected**. Then, the Gateway will execute this command and send an acknowledge back to the Control Computer.

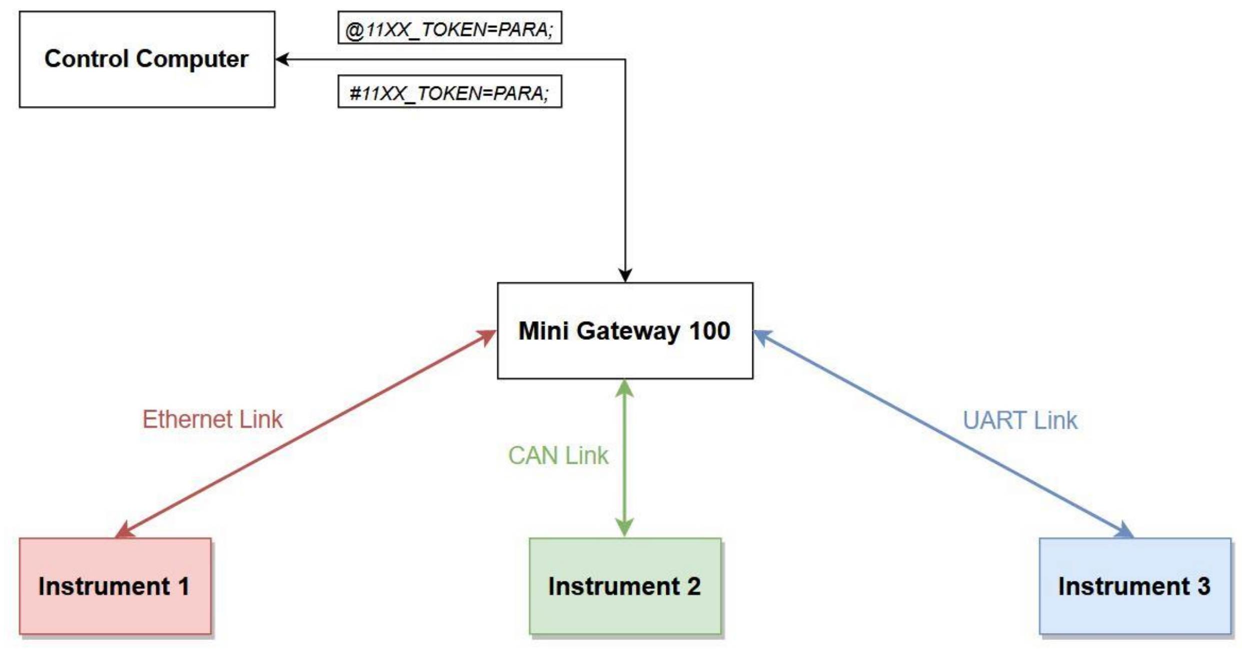


Figure 1 System Structure

This section describes the protocol commands used to remotely control the Mini Gateway 100.

- **Communication Interface**

The Gateway uses the **TCP protocol** for communication with the PC Control Software. When an instrument is connected to the Gateway, all the **instrument resources** can be accessed through the Gateway from any computer or equipment.

Parameters of the Ethernet TCP port are the following:

- **IP address**: 192.168.0.42
- **Port number**: 6025

- **Communication Timing**

To unsure a good use of the **Gateway Test System**, as the ART-Logics Control Software does it, the user should follow several rules:

- When a command is sent, the next one can be sent only after receiving the reply to the first one.
- If no reply is received after 1.5 second, the command is cancelled.

- **Communication Syntax**

The command is an **ASCII chain** sent from the Control Computer to the Gateway, following this scheme:

@NNXX_token=parameters;

Example: @12XX_GETDIG=1;

Index	Length	Content	Mandatory	Description
1	1	**@**	Y	Address header for each command
2	2	**NN**	Y	The Gateway address ***NN*** from 00 to FF
3	2	**nn**	Y	'XX' or '11' depending on the command
4	1	**_**	Y	Separator
5	x	**token**	Y	The token length is not fixed. Tokens are case sensitive, and must use caps
6	x	**parameters**	N	Depending on token. Starts with "=". Parameters separated by comma
7	1	**;**	Y	End flag of the command. When it is received, the command is processed

Figure 2 Command ASCII characters

The acknowledge is an **ASCII chain** sent back from the Gateway to the Control Computer, following this scheme:

#NNnn_token=parameters;

Example: #12XX_SYSID=MINI_GATEWAY_100_01_01_45;

Index	Length	Content	Mandatory	Description
1	1	**#**	Y	Address header for each request
2	2	**NN**	Y	The Gateway address ***NN*** from 00 to FF
3	2	**nn**	Y	'XX' or '11' depending on the command
4	1	**_**	Y	Separator
5	x	**token**	Y	The token length is not fixed. Tokens are case sensitive, and must use caps
6	x	**parameters**	N	Depending on token. Starts with "=". Parameters are separated by comma
7	1	**;**	Y	End flag of the command. When it is received, the command is processed

Figure 3 Response ASCII characters

- ## Communication Commands List

Name	Syntax
`SETDIG`	`@NN11_SETDIG=<Channel>;`
`CLRDIG`	`@NN11_CLRDIG=<Channel>;`
`GETDIG`	`@NN11_GETDIG=<Channel>;`
`MSGTX`	`@NN11_MSGTX=<Port>,<MsgContent>;`
`MSGRX`	`@NN11_MSGRX=<Port>,<MessageName>,<ByteNumber>;`
`CONFIG`	`@NN11_CONFIG=<Port>,<ParameterList>;`
`TSTRT`	`@NN11_TSTRT;`
`TSTOP`	`@NN11_TSTOP;`

PROCESS	@NN11_PROCESS=<ID>,DEFINE [,<Granularity>,<TotalStep>];
	@NN11_PROCESS=<ID>,<END|START|STOP|DELETE|RESULT|RESX>;
	@NN11_PROCESS=QUERY;
	@NN11_PROCESS=<ID>,<StepNum>,<Command>,<Parameter>;

SYSID	@NNXX_SYSID;

SEQUENCE	@NN11_SEQUENCE=<ID>,DEFINE,<Pid>,<cycles>,<delay>[,<Pid>,<cycle>,<delay>];
	@NN11_SEQUENCE=START,<ID>[,<ID>];
	@NN11_SEQUENCE=STOP[,<ID>];

SYSTIME	@NNXX_SYTIME ;

3.1.1 SETDIG command

Purpose	Set a digital level output channel to a high level "1".
Syntax	**@n_SETDIG=**<Channel>;
Example	@1211_SETDIG=3;
Description	The <*Channel*> parameter defines the digital output channel in the Gateway to access for writing. Valid value for it is 1 to 5.
Acknowledge	When the Gateway receives the **@n_SETDIG** command and execute it successfully, **#n_SETDIG** returns the status of the Digital Output.

3.1.2 CLRDIG command

Purpose	Set the digital level of and output channel to a low level "0".
Syntax	**@n_CLRDIG=**<Channel>;
Example	@1111_CLRDIG=2;
Description	The <*Channel*> parameter defines the digital output channel in the Gateway to access for writing. Valid value for it is to 5.
Acknowledge	When the Gateway receives the **@n_CLRDIG** command and execute it successfully, **#n_CLRDIG** returns the status of the Digital Output.

3.1.3 GETDIG command

Purpose	Read the level of a digital input channel.
Syntax	**@n_GETDIG=**<Channel>;
Example	@2211_GETDIG=1;

Description	The <***Channel***> parameter defines the digital input channel in the Gateway to access for reading. Valid value for it is 1 to 5.
Acknowledge	#n_GETDIG=<Data>; When the Gateway receives the **@n_GETDIG** command and execute it successfully, **#n_GETDIG** with digital channel's status will be sent back to PC. <Data> defines all digital channel's status, if digital channel's status is high, the corresponding bit will be set to 1. Otherwise, it will be set to 0

3.1.4 TSTRT command

Purpose	Notify the Gateway that the configuration state is over and test will be launched.
Syntax	@n_TSTRT;
Example	@2311_TSTRT;
Description	There is no any parameter in the request. The command will initialize CAN, Ethernet and UART buses. No new CONFIG commands can be sent after this command until a TSTOP command is sent.
Acknowledge	#n_TSTRT**;** When the Gateway receives the **@n_TSTRT** command, **#n_TSTRT** will be sent back to PC.

3.1.5 TSTOP command

Purpose	Ask the Gateway to clear all previous configurations.
Syntax	@n_TSTOP;
Example	@1111_TSTOP;
Description	There is no any parameter in the request. The Gateway will clear all previous configurations, stop and delete all processes. It will stop any CAN, Ethernet and UART pending communication. A new test with new configuration can be launched following this command.
Acknowledge	#n_TSTOP**;**

When the Gateway receives the **@n_TSTOP** command, **#n_TSTOP** will be sent back to PC.

3.1.6 SYSID command

Purpose	Read the software version of the Mini Gateway 100.
Syntax	@n_SYSID;
Example	@11XX_SYSID;
Description	This command returns the software version of the Mini Gateway 100.
Acknowledge	#11XX_SYSID=MINI_GATEWAY_100_01_01_45;

3.1.7 SYSTIME command

Purpose	Read the time information.
Syntax	@n_SYSTIME;
Example	@55XX_SYSTIME;
Description	This command returns the time information in the Gateway in ms. The value returned represents the duration since the Gateway has been turned on.
Acknowledge	#13XX_SYSTIME=3450;

3.1.8 PROCESS command

Purpose	Define a process in the Gateway. A process is a timely defined test plan that runs in a loop.
Syntax	**@n_PROCESS=**<ID>,**DEFINE** [,<Granularity>,<TotalStep>]; **@n_PROCESS=**<ID>,**END,START|QUERY|STOP|DELETE|RESULT**; **@n_PROCESS=**<ID>,<StepNum>,<Command>,<Parameter>;

Example	@0511 _PROCESS=1,DEFINE,400,100;
	@0511_PROCESS=1,START;
	@0511_PROCESS=1,10,GETDIG,1;

Description	The <ID> parameter defines the process ID, which is the only identifier of a process. It is a decimal number. The valid range is from 1 to 255. 32 processes can be defined in one Gateway at a time. The DEFINE is used to define a new process. It is the first command to define a process. The <ID> in it should not be previously used to refer to another process. If no other parameter following it, the information (<Granularity>,<TotalStep>) of the process will be returned in its answer. The END is used to end the definition of a process. The START is used to run a process defined before. The STOP is used to stop a running process. The DELETE is used to delete one predefined process or all process. After it is executed successfully, the process will not be available any more. NOTE: If the process is running, the command will fail. The RESULT is used to get the result of the measurement commands of the process. Its answer will be the combination of their value, which are separated by comma ",". The QUERY is used to get the draft information of how many processes have been defined in the Gateway. Its answer includes all the number of the defined processes. The <Granularity> defines the minimum time (in ms) resolution of the process. It's an integer number. Valid value for is 10, 20, 30,…,65530. The <TotalStep> defines how many steps that the process will endure. Valid value for it is 1 to 4294967295. It must be at least one step more than the last step number. The total time that the process will last is < TotalStep > * <Granularity>. The <StepNum> defines when the <Command> will be executed. The <Command> defines the operation that the Gateway will operate at the <StepNum>. It can be the following commands: SETDIG, CLRDIG, GETDIG, MSGTX, MSGRX,. NOTE: 1. @n_PROCESS=<ID>,<StepNum>,<Command>,<Parameter>; is used to define an operation in the process. 2. The <TotalStep> must be bigger than last <StepNum>. 3. The command to add a command in the step table will fail in the follow situations: • The process has not been defined. • The <Command> or <Parameter> is wrong or not support in process. • The process is running. • The <StepNum> is less than the previous <StepNum> saved in the process.
Acknowledge	#n_PROCESS;

When the Gateway receives the @n_PROCESS command, #n_PROCESS with the same parameter will be sent back to PC.
For RESULT:
Since the length of result answer can exceed the limit for one answer string, then it's possible that more than one answer for the result query are available. The first answer of this command is identified with "BEGIN", and the last one is ended with "END"
#0511_PROCESS=5,RESULT,LOOP=1,BEGIN,1,0x11FFEE44,0,1;
#0511_PROCESS=5,RESULT,LOOP=1,0,1,1,1,0,1,0;
#0511_PROCESS=5,RESULT,LOOP=1,0,1,0,0xEACF2586,END;
For QUERY:
If there is any process defined in the Gateway, its answer will look like the follow (1,5,100 and 255 are the process ID defined in it.):
#1105_PROCESS=QUERY,4 DEFINED,1,5,100,255;
Else, it will be
#1105_PROCESS=QUERY,0 DEFINED;
For DEFINE:
If the process has been defined in the controller module, its answer will be as follow:
#1105_PROCESS=1,DEFINE,200,1000,1,9,LOOP=11;
The parameters in sequence are explained below one by one:
1 : the process ID
DEFINE: Key word to identify the answer
200: Granularity
1000: Total steps
1: Running state. 1 means it's running, and 0 means it's not running.
9: The numbers of the commands have been added to the process.
LOOP=11: The number of how many times the process has ran since the last "START" command. It will be set to 0 when the process is stopped.
For @n_PROCESS=<ID>,<StepNum>,<Command>,<Parameter>;
If executed successfully, the answer with the same parameters and SRAM space will be sent back to PC.
#n_PROCESS=<ID>,<StepNum>,<Command>,<Parameter>,<SRAM>;
<SRAM> is used to indicated how much SRAM is left, it's a float data and decreases from 1 to 0.

3.1.9 SEQUENCE command

Purpose	Launch processes in sequential mode
Syntax	**@n_SEQUENCE**=<Sid>,DEFINE,<Pid>,<cycles>,<delay> [,<Pid>,<cycles>,<delay>]; **@n_SEQUENCE=**START,<Sid>[,<Sid>]**;**

@n_SEQUENCE=STOP**[**,<Sid>]**;**

Example

@0511_SEQUENCE=1,DEFINE,20,1,0,6,2,20,500,40,500;
@0511_SEQUENCE=START,1,2;
@0511_SEQUENCE=STOP,2;
@0511_SEQUENCE=STOP;

Description

The **DEFINE** is used to define a new sequence. It is the first command to define a sequence. Parameters are as follow:

- **<Sid>:** Sequence ID, is a decimal number. Valid value is from 1 to 5. Only 5 sequences can be defined in one Gateway at a time.
- **<Pid>:** Process ID, is a decimal number. Valid value is from 1 to 255. Only 32 different processes can be defined in one Gateway.
- **<cycles>:** number of loops a process will be executed. A process can be called again later in the sequence. It is an integer value. Valid value is from 1 to 4294967295.
- **<delay>:** in ms. Number of time to wait before the next process will be started. It is an integer number and multiple of 10 or 0. Valid values are 0,10,20,...,4294967290.

The <ID> in it should not be previously used to refer to another sequence. The sequence should not be running before being defined. The definition should only be composed of already defined processes. A correct sequence definition must have at least one complete Item, a complete Item is made of a Process ID <Pid>, a cycle number <cyces> and a delay <delay> which can be equal to 0.
A sequence can be defined again during test execution as long as it is not previously running and that all mission profiles defined inside are currently stopped.
To start the sequence with a delay, the first three parameters can be set as follow: <Pid> = 0, <cycles> = 1 (this parameter will not be taken into account in this case), <delay> = x with x the requested delay time in ms.
The **START** is used to run a sequence defined before. The sequence to start must be previously defined or stopped. At least one Sid must be used in the command.
Parameters are as follow:

- **<Sid>:** Sequence ID, is a decimal number. Valid value is from 1 to 5. Only 5 sequences can be defined in one Gateway at a time.

The **STOP** is used to stop a running sequence. The sequence to stop must be already started.
Parameters are as follow:

- **<Sid>:** Sequence ID, is a decimal number. This parameter is **optional**, if not specified all sequences will be stopped. Valid value is from 1 to 5. Only 5 sequences can be defined in one Gateway at a time.

Acknowledge	Command returns same parameters:
	@0511_SEQUENCE=1,DEFINE,20,1,0,6,2,20,500,40;
	#0511_SEQUENCE=1,DEFINE,20,1,0,6,2,20,500,40;
	@0511_SEQUENCE=START,1,2;
	#0511_SEQUENCE=START,1,2;
	@0511_SEQUENCE=STOP,2;
	#0511_SEQUENCE=STOP,2;

3.1.10 CONFIG command

Purpose	Configure the CAN, Ethernet and UART message related information.
Syntax	**@n_CONFIG=**<Port>,<ParameterList>**;** **For CAN communication** **@n_CONFIG=**CAN1,BAUDRATE,<Baudrate Value>; **@n_CONFIG=**CAN1,TX|RX,<MessageName>,<CAN mode>,<CAN Id>; **For Ethernet communication** **@n_CONFIG**=ETH1,MACADDR,<MAC Address>; **@n_CONFIG**=ETH1,IP4ADDR,<Source IPv4 Address>,<Mask IPv4 Address>,<Gateway IPv4 Address>; **@n_CONFIG**=ETH1,<Protocol>,<Channel Name>,BIND,<Source IPv4 Address>,<Source Port Number>,CONNECT,<Destination IPv4 Address>,<Destination IPv4 Port Number>;

Example	@0511_CONFIG=CAN1,BAUDRATE,500K;
	@0511_CONFIG=CAN1,TX,REQDIG,EXT,0X16302190;
	@0511_CONFIG=CAN1,TX,CAN1TX,STD,0X112;
	@0511_CONFIG=CAN1,RX,CAN1RX,STD,0X5AA;
	@0511_CONFIG=ETH1,MACADDR,1B:63:0A:8E:00:CE;
	@0511_CONFIG=ETH1,IP4ADDR,192.168.0.11,255.255.255.0,192.168.0.1;
	@0511_CONFIG=ETH1,TCP,AGILENT,BIND,192.168.0.15,41569,CONNECT,192.168.0.19,5025;

Description	The parameter <Port> defines the port to be configured. Valid values are CAN1, ETH1 and UART1 The <*ParameterList*> parameter defines configuration to be downloaded to the Gateway. For different configuration, the parameter will be different. The option defined in mission profile will be used here as parameter. However, only the option related with CAN/ETH/UART in mission profile can be used as this command's parameter. The Configuration is available for the current test, once the test started, configuration is fixed and can not be changed. To reconfigure a <Port>, test must be stoped by sending TSTOP command.

For <Port> is **CAN1**, the valid <ParameterList> are listed below:

- **BAUDRATE,<Baudrate Value>:** Defines the baudrate of the CAN interface 1. Valid values are 10K, 20K, 33.3K, 40K, 83.3K, 100K, 125K, 250K, 500K and 1000K.
- **TX|RX**,**<MessageName>,<CAN mode>,<CAN Id>:** Start definition of a CAN message for transmission (TX) or reception (RX) on CAN1 or CAN2:
  - The parameter **<TX|RX>** defines the operation mode of a CAN message. TX is for transmission and RX is for reception.
  - The parameter **<MessageName>** defines the alias of a CAN message. Any name can be use as an alias to identify this message. However, the alias name cannot include the space character and it must be unified in the mission profile scope. Its length should be **less than 12** bytes.
  - The parameter **<CAN mode>** defines the standard or extended mode of this CAN message. Valid values are **STD** or **EXT**.
  - The parameter **<CAN Id>** defines the ID of this CAN message. Valid values are any hexadecimal number preceded by "0X" from 0X00 to 0X7FF in **STD** and from 0X00 to 0X1FFFFFFF for **EXT** mode.

For <Port> is **ETH1**, the valid <ParameterList> are listed below:

- **MACADDR,<MAC Address>:** Define the MAC address of the Gateway
  - <MAC Address>: Valid MAC Address is 6 bytes in hexadecimal format separated by ':' or '-' characters.
- **IP4ADDR,<Source IPv4 Address>,<Mask IPv4 Address>,<Gateway IPv4 Address>**:Define the IP address, Mask address and the Network Gateway address of the Gateway.
  - <Source IPv4 Address>: Sets the IP v4 source address of the Ethernet interface.
  - <Mask IPv4 Address> : Sets the Subnet mask IP v4 address of the Ethernet interface.
  - <Gateway IPv4 Address>: Sets the default gateway IP v4 address of the local network.
- **<Protocol>,<Channel Name>,BIND,<Source IPv4 Address>,<Source Port Number>,CONNECT,<Destination IPv4 Address>,<Destination IPv4 Port Number>**: Define the parameters of an Ethernet channel.
  - <Protocol>: The Ethernet channel protocol. Valid values are TCP or UDP.
  - <Channel Name>: defines the alias of a ETH channel. Any name can be use as an alias to identify this message. However, the alias name cannot include the space character and it must be

unified in the mission profile scope. Its length should be **less than 12** bytes.
- <Source IPv4 Address>: Sets the IP v4 source address of the Ethernet channel.
- <Source Port Number> : Sets the source port number of the Ethernet channel.
- <Destination IPv4 Address>: Sets the IP v4 destination address of the Ethernet channel.
- <Destination Port Number>: Sets the destination port number of the Ethernet channel.

Acknowledge

When the Gateway receives the @n_CONFIG command and execute it successfully, #n_CONFIG with the same parameters will be sent back to PC.
#n_CONFIG=<port>,<ParameterList>**;**

3.1.11 MSGTX command

Purpose

Write a message in the specified communication port.

Syntax

@n_MSGTX=<Port>,<MsgContent>**;**
@n_MSGTX=CAN1,<MessageName>,<Data>;
@n_MSGTX=ETH1,<ChannelName>,<Data>;

Example

@0511_MSGTX=CAN1,RdReq,0X01FF;
@0511_MSGTX=ETH1,RIGOL,IDN*?;

Description

The <*Port*> parameter defines the port of Gateway, where the message is sent. Valid value for it can be: CAN1, ETH1 and UART1.
The <*MsgContent*> parameter defines the data to be sent.
- For **CAN1**, it is <*MessageName*> and <*Data*>*,* the detail of <*MsgContent*> is explained below.
- The <*MessageName*> must be defined by the CONFIG command before the MSGTX command is used. Its length should be less than 12 bytes.
- The <Data> is a string representing hexadecimal bytes to be sent. The maximum length is 8 bytes.
- For **ETH1**, it is <*MessageName*> and <*Data*>*,* the detail of <*MsgContent*> is explained below.
- The <*MessageName*> must be defined by the CONFIG command before the MSGTX command is used. Its length should be less than 255 bytes.
- The <Data> is a string representing hexadecimal bytes to be sent, i.e. starting with 0X and only including the characters 0, 1, 2, 3, 4, 5, 6, 7,

8, 9, A, B, C, D, E, F (such as "0X0A120E06") or the ASCII string to send.

Acknowledge

#n_MSGTX=<Port>[,<MessageName>],<MsgContent>;
When the Gateway receives the **@n_MSGTX** command and execute it successfully, **#n_MSGTX** with the same parameters will be sent back to PC.
If the message is sent in multiframe format, the number of bytes is returned instead of the full message.
Ex: #11_MSGTX=ETH1,ETH1TX,22B_DATASENT;

3.1.12 CONFIG command

Purpose

Read a message from the specified communication port.

Syntax

@n_MSGRX=<Port>,<MessageName>,<ByteNumber>;
@n_MSGRX=<Port>,CLEARMSG;

Example

@0511_MSGRX=CAN1,RdReq,26;
@0511_MSGRX=ETH1,ETH1RX,255;
@05_MSGRX=CAN1,CLEARMSG;

Description

The <*Port*> parameter defines the message type to be sent in the port of the Gateway. Valid value for it can be: CAN1, ETH1 and UART1.
- <*MessageName*>*:* defines the message name to be read in the Gateway. It should be defined by CONFIG command before this command is used. It should only be provide when <*Port*> is CAN1, ETH1 and UART1.
- <*ByteNumber*>: defines how many bytes the Gateway needs to read.
- CLEARMSG: this command will clear all stored messages in the Gateway for the specific port.

Acknowledge

When the Gateway receives the @n_MSGRX command and execute it successfully, #n_MSGRX will be sent back to PC. Two possible options:

- Message requested:

#n_MSGRX=<Port>[,<MessageName>],<Data>;
If a valid message is read, the content of this message will be provided in the format of byte flow in <Data>, it is started with "0X".
If there is no message read in a certain period, <Data> will not be provided.
If the message is a multiframe, the token ",END" is added at the end of the data.
If the message is longer than 250 bytes, the acknowledgment is divided in two strings, with the tokens "BEGIN" and "END" indicating the start and end of the data:

```
#1111_MSGRX=CAN1,CanAlias,BEGIN,0X11223344[…]2233;
#1111_MSGRX=CAN1,CanAlias,445566[…]445566,END;
```

- ClearMessage requested:

```
#n_MSGRX=<Port>,CLEARMSG[,<ClearMode>];
```

If a CLEARMSG command is sent and executed successfully, a #n_MSGRX will be sent back to PC with the same parameters.

4 Operational Modes

Performing a **test plan** using the Mini Gateway 100 has to go through three main stages:

Configuration stage: The user prepares the setup by connecting the instruments and devices used in the test, and also write the test sequence plans using the **Art Logics Test Software** or **manually** following the Gateway's commands protocol.

Test run stage: After the hardware and software configurations are done. The user can run the test sequences as planned for a defined period of time or until the user stops the test manually.

Result analysis stage: The user can check the real time results while the test is running or after the test finished. The results are saved as a csv file.

In the Test run stage, the user has the choice to use the Mini Gateway 100 in two operational modes, the **PC Control Mode** or **the Standalone Mode**.

The PC Control/Standalone Mode Switch on the front side of the Gateway is used to choose the current working mode.

4.1 Standalone Mode

In this mode, the test scenarios are saved in the **SD Card** and the user can run these tests **independently** from the PC Control. Therefore, the system architecture is reduced to the Mini Gateway 100 and the test instruments in the Standalone Mode.

In the configuration stage, after the configuration of the test scenarios is done, the user can upload the tests to the SD Card by sending the process and sequence commands to the Gateway. A test scenario is defined by a **unique ID** and once the tests are saved, the user can manually control the progress of the test using control **buttons** and **switches**.

The front side of the Mini Gateway 100 has 4 buttons and 1 switch for the user to be used to control the progress of the tests manually in the Standalone mode.

Step/Non-Step Switch: The user can choose between two modes of controlling the processes. In the **Non-Step mode**, the buttons are used to jump and change between the processes, but in the **Step mode**, the Gateway stays on one process and the buttons are used to jump between the steps inside this process.

Start/Stop Button: In the **Non-Step mode**, start or stop the current process. Equivalent to sending the commands @n_PROCESS=<ID>,START; and @n_PROCESS=<ID>,STOP;

In the **Step mode**, repeat the execution of the current step of the current process.

Next Button: In the **Non-Step mode**, stop the current process running and start the next process saved in the SD Card. If only one process is defined, clicking this button has no effect.

In the **Step mode**, execute the next step of the current process after finishing the execution of the current step. If it is the last step, the next step will be the first step of the process.

Previous Button: In the **Non-Step mode**, stop the current process running and start the previous process saved in the SD Card. If only one process is defined, clicking this button has no effect.

In the **Step mode**, execute the previous of the current process after finishing the execution of the current step. If it is the first step, the previous step is the last step of the process.

Reset Button: Reset the Mini Gateway 100. This will delete any test configurations saved in the internal memory of the Gateway.

The user is free to jump between the operational modes and also the Step/Non-Step modes during his test to be able to achieve and produce the results wanted.

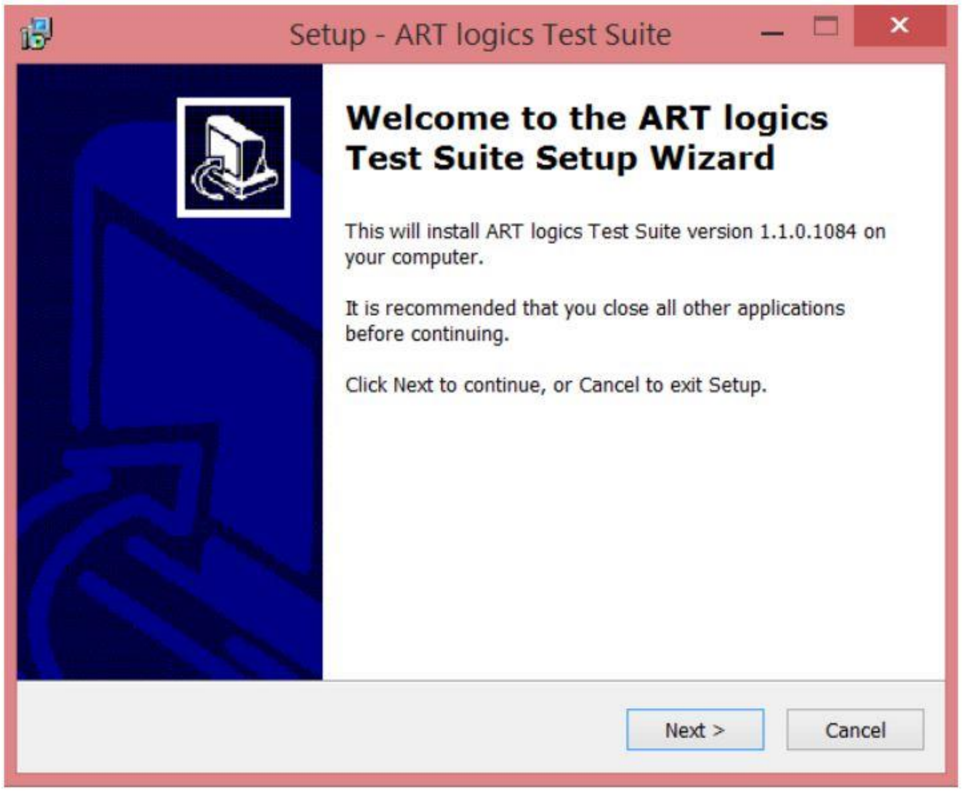
4.2 PC Control Mode

In this mode, the user uses the **Art Logics Test Software** or any **third-party Software tool** to upload the test plans and run it on the Gateway. Also, a computer needs to be connected to the Gateway via **Ethernet** to upload the tests, run it and retrieve the results.

The ART Logics Test Suite (abbr. ALTS) Software, an adjustable and graphical tool, is able to configure the test plans and scenarios, control the test execution and gather extracted equipment data.

4.2.1 Installation

Art Logics will provide the ALTS installation as a exe file. By double clicking the exe file we can start the installation of the software on our computer.


Setup - ART logics Test Suite

Welcome to the ART logics Test Suite Setup Wizard

This will install ART logics Test Suite version 1.1.0.1084 on your computer.

It is recommended that you close all other applications before continuing.

Click Next to continue, or Cancel to exit Setup.

Figure 4 ALTS Setup window

4.2.2 Create a new project

After the installation is done, open the ALTS and start by creating a new project. The ALTS is based on a Project system. A project is containing all the configuration of the test. Therefore, it is essential to begin by creating a new project: click on **File -> Projects -> New Project**.

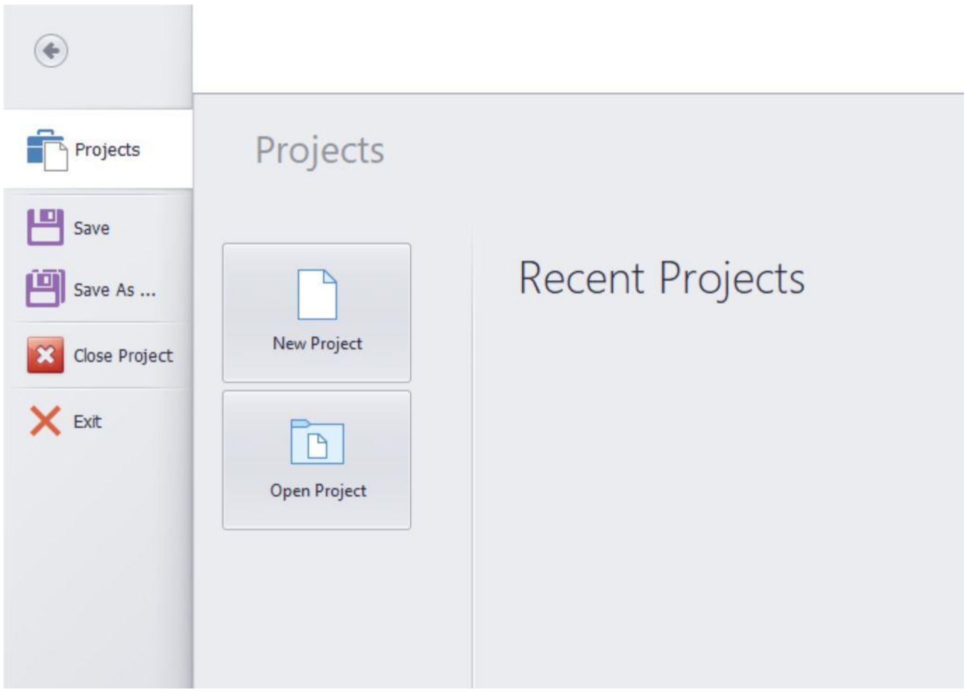


Figure 5 Create new project menu

4.2.3 Create a Mission Profile

After creating a new project, it is time to start the configuration of our test in the following view.

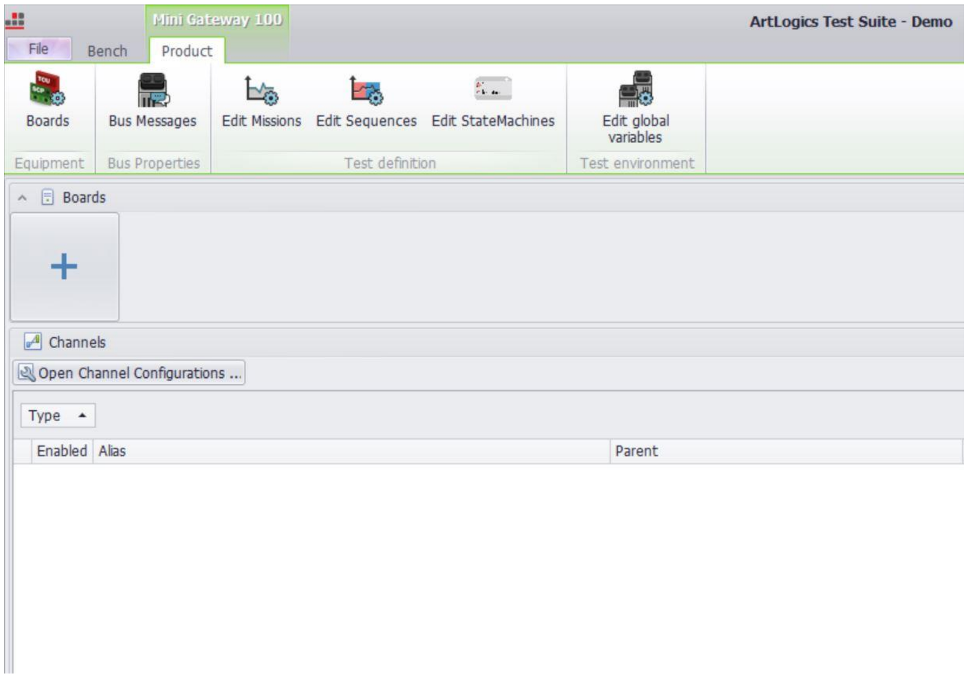
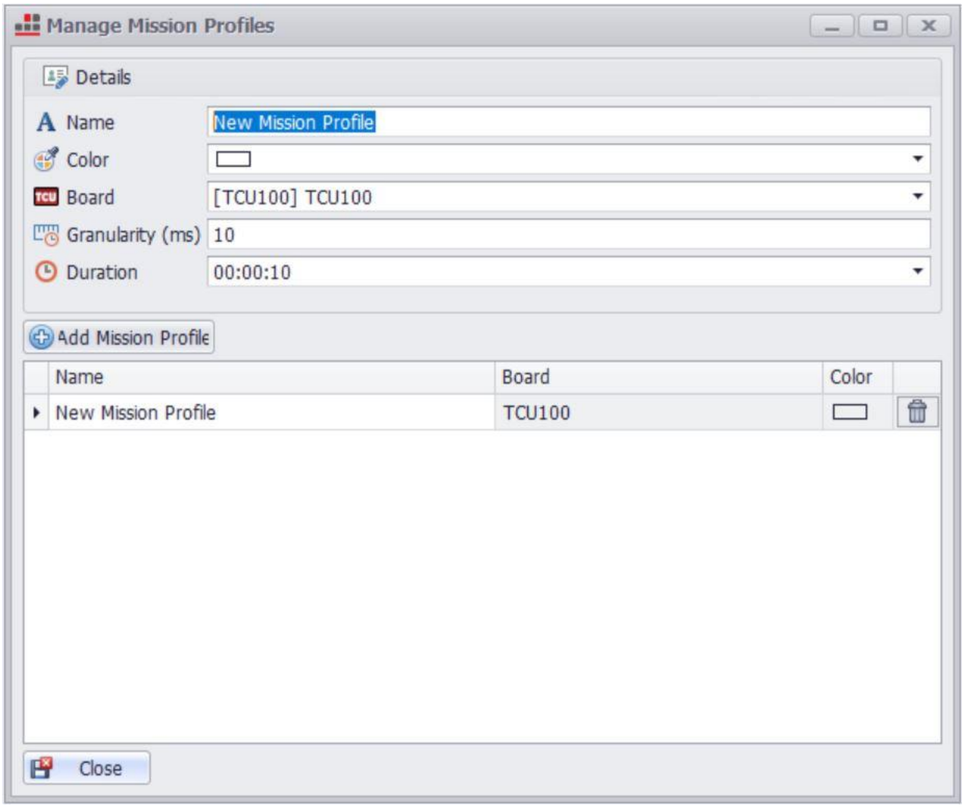


Figure 6 Project view

In order to create the first Mission Profile, click on **Edit Missions -> Manage Mission Profiles -> Add Mission Profile**. For information, a Mission Profile is a container of all the Action / Event that will be performed during the test, like a measurement, sending and receiving CAN or Ethernet messages.

We have to choose the granularity of the Mission Profile which is the duration of each step, and the duration of the whole loop.



Name	New Mission Profile
Color	
Board	[TCU100] TCU100
Granularity (ms)	10
Duration	00:00:10

Name	Board	Color
New Mission Profile	TCU100	▭

Figure 7 Create a Mission Profile view

The following display should appear to start adding actions in the Mission Profile. It is needed then to add the channels on which the actions will be performed, by clicking on Select Channels.

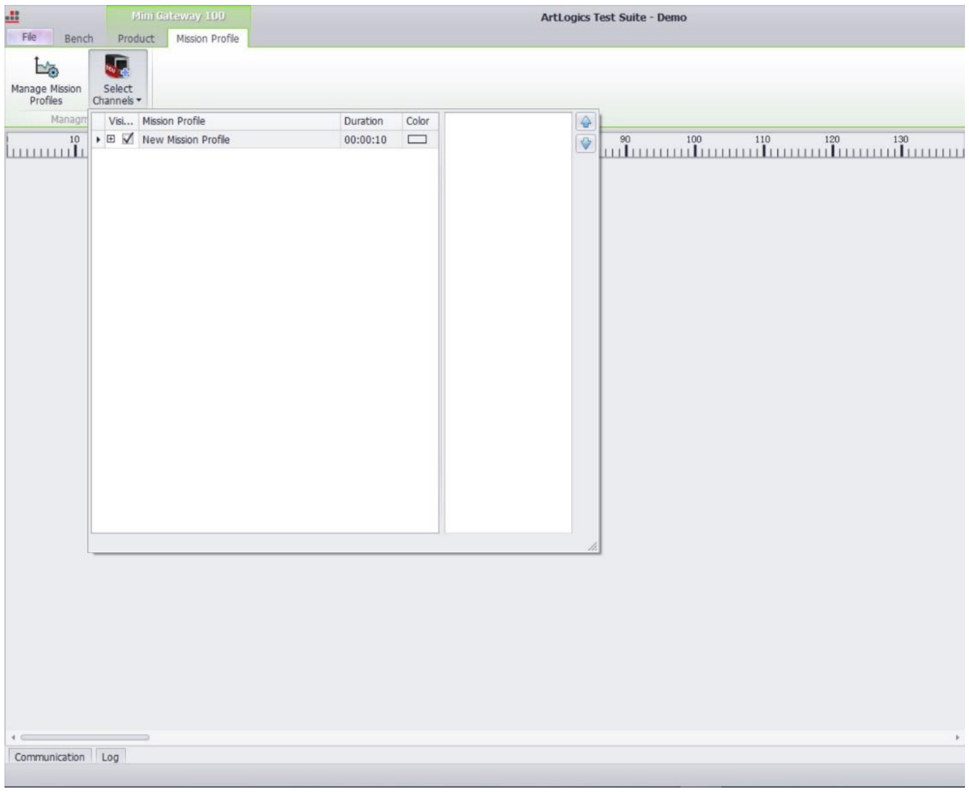
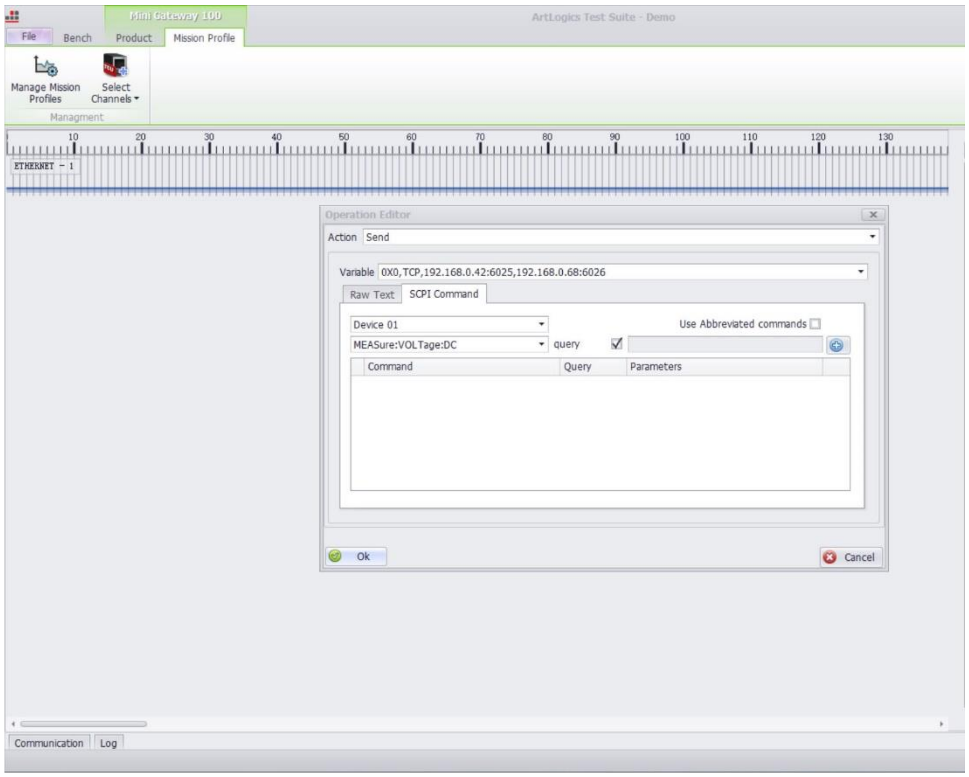


Figure 8 Select Channels view

For example, we will add the **Ethernet Channel**.

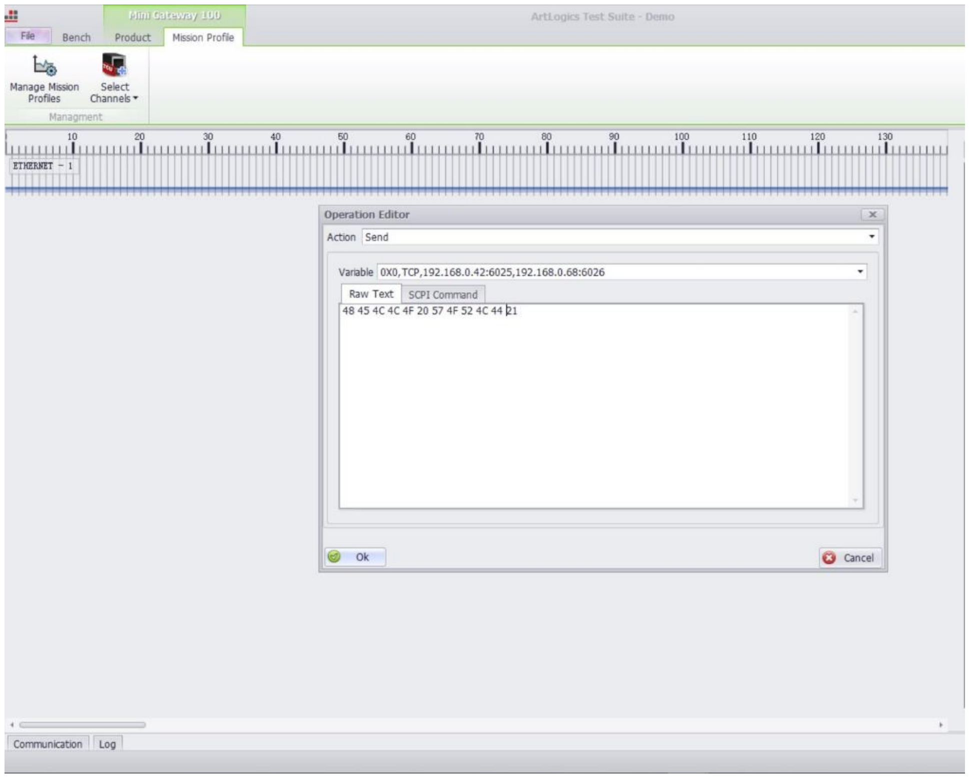
Now, we can start adding Events (also called Actions) inside the Channel added. To add a new Event, double click on the channel on the channel and the timing to do the action.

For our Ethernet channel action, we can add two actions: **Send** and **Receive**. If we want to send an Ethernet frame, we can send a raw message or a SCPI message.



Action: Send
Variable: 0X0,TCP,192.168.0.42:6025,192.168.0.68:6026
Device 01
MEASure:VOLTage:DC query ✓

Figure 9 Ethernet SCPI message



Action: Send
Variable: 0X0,TCP,192.168.0.42:6025,192.168.0.68:6026

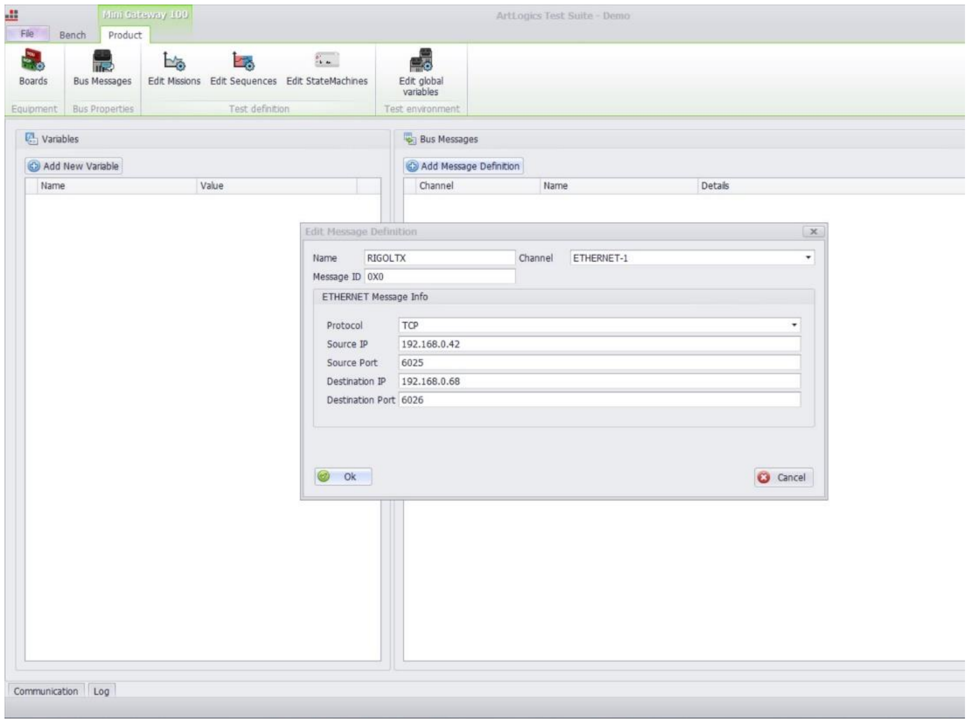
```
48 45 4C 4C 4F 20 57 4F 52 4C 44 21
```

Figure 10 Ethernet raw message

Notes:

- Depending on the channel, a form will open in order to specify what action to perform and to configure the event.
- Some channels may require to first set up some settings as for the CAN, which will require configuring it with its baudrate.

To add special parameters for the special Channels like CAN and Ethernet, click on **Bus Messages on the main view -> Add Message Definition**.

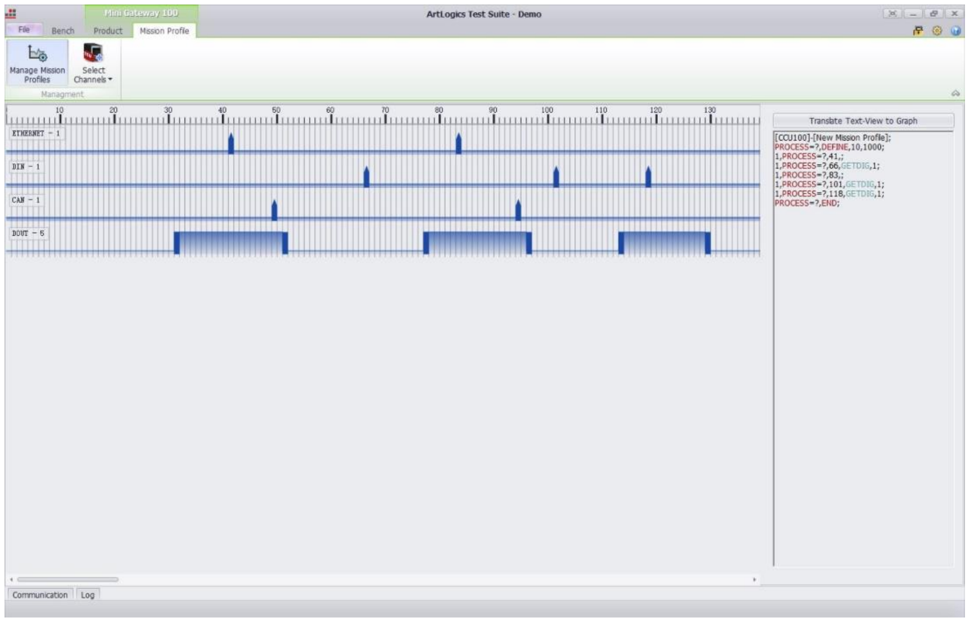


Name	RIGOLTX	Channel	ETHERNET-1
Message ID	0X0		

Protocol	TCP
Source IP	192.168.0.42
Source Port	6025
Destination IP	192.168.0.68
Destination Port	6026

Figure 11 Ethernet Channel Config view

A Mission Profile with Ethernet, CAN and Digital Outputs will have the following view.



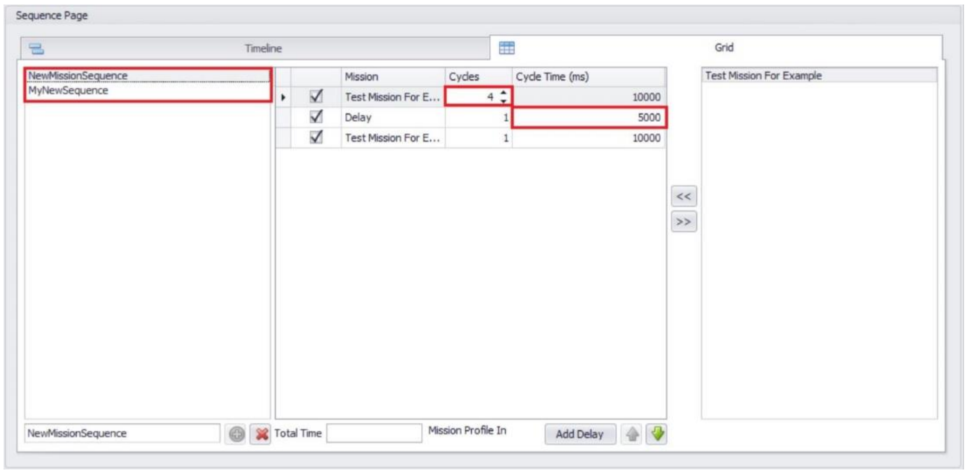
```
[CCU100]-[New Mission Profile];
PROCESS=?,DEFINE,10,1000;
1,PROCESS=?,41,;
1,PROCESS=?,66,GETDIG,1;
1,PROCESS=?,83,;
1,PROCESS=?,101,GETDIG,1;
1,PROCESS=?,118,GETDIG,1;
PROCESS=?,END;
```

Figure 12 Mission Profile view

4.2.4 Create a Sequence

A **sequence** of test is a list of Mission profile performed sequentially, one after the other. To create a new sequence of test, click **on Edit Sequences -> Add New Sequence**.

Click on Add Mission Profile to insert a Mission Profile in the sequence and click on Add Delay to insert a delay.



	Mission	Cycles	Cycle Time (ms)
✓	Test Mission For E...	4	10000
✓	Delay	1	5000
✓	Test Mission For E...	1	10000

Figure 13 Sequence view

4.2.5 Create a Flowchart

After defining a workspace with the Mission Profiles and Sequences, you should be able to create the content of your test from the flowchart. To do it, the user should click on Edit **StateMachines ->Manage Flowcharts -> Add**.

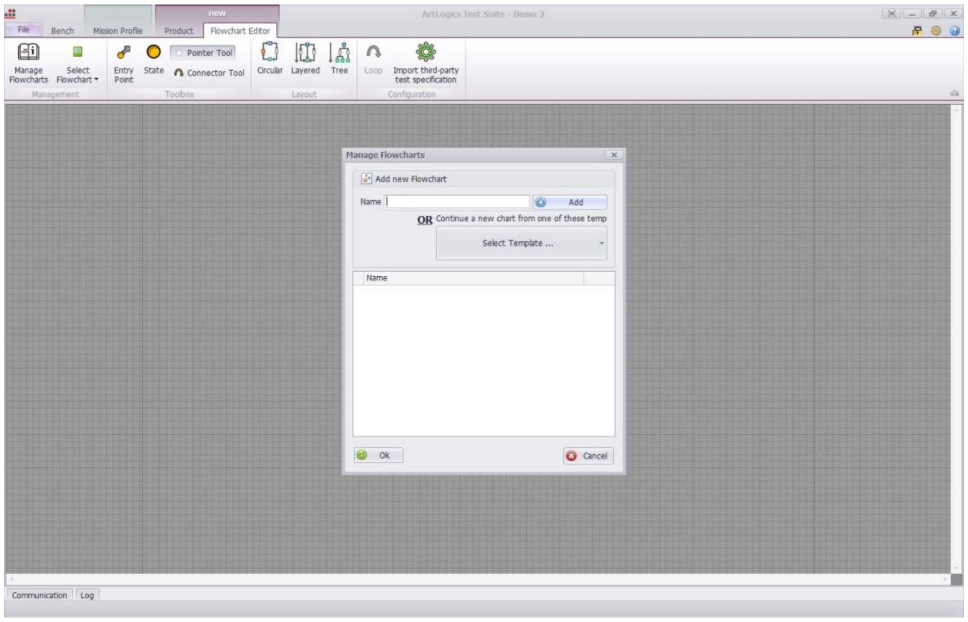
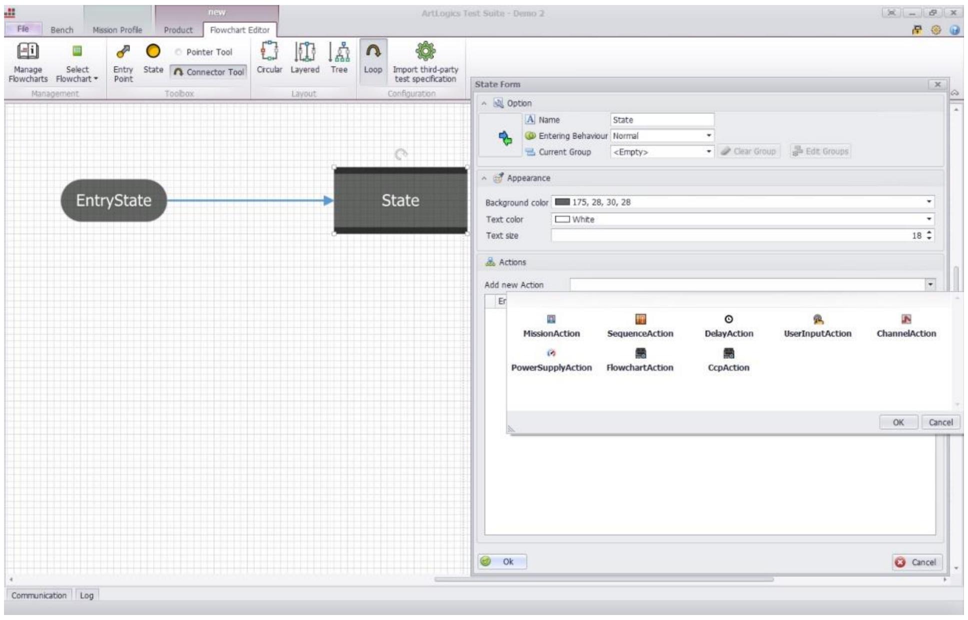


Figure 14 Create a Flowchart

The **Flowchart** contains **states** which are containers where you can insert a list of actions; this list of actions will be executed sequentially. An action may execute Mission Profiles, Sequences, Flowchart, wait a user action, send emails and so on…

To create a state, first click on the “state” menu item, then to click on the canvas to give a position to the state.

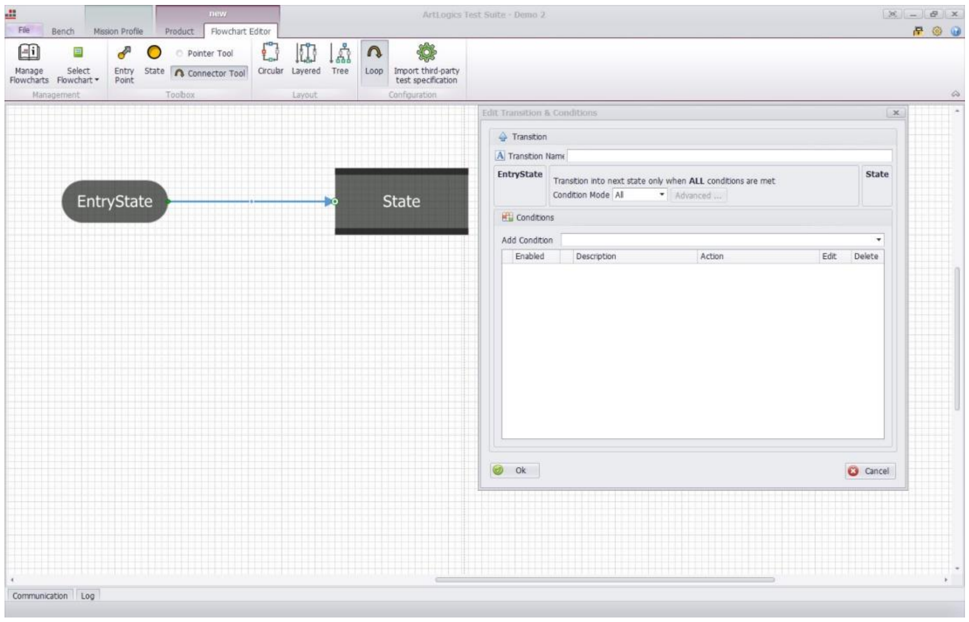
To set the properties of the state, double click on the state or to do a right click on the state then to do a left click on property.



Name: State
Entering Behaviour: Normal
Current Group: <Empty>
Background color: 175, 28, 30, 28
Text color: White
Text size: 18

Figure 15 Create a State in a Flowchart

When we have more than one state, the user can link between them using transition connections by clicking on the Connection Tool. Also, a transition can be defined as a condition to go from a state to the next one. To define the condition double click on the transition connection on the Flowchart.



EntryState — Transition into next state only when **ALL** conditions are met — State

Condition Mode: All

Figure 16 State Condition definition

5 Properties

5.1 General operation

→ Real-time commands test sequencer and controller.
→ Built in Sd Card to store test scenarios.
→ User friendly user interface software tools to create test sequence plans and view results.
→ Full support for the SCPI products.
→ 5-Volt supply at coaxial power connector.
→ Voltage supply via USB.
→ Operating temperature range is -30 to 60 °C.

5.2 CAN operation

→ 2 CAN ports compliant with the CAN specifications 2.0A/B.
→ Transmission and reception of CAN messages using 2D sub connection.
→ 120 Ω CAN termination can be activated via a jumper.
→ CAN bit rates from 10 kbit/s up to 1 Mbit/s.

5.3 Ethernet operation

→ 1 10/100 Mbps Ethernet port.
→ Transmission and reception of TCP and UDP messages.
→ Configurable IP, Mask and Gateway addresses.
→ Configurable Ports.

5.4 UART operation

→ 1 RS-232 port and 1 RS-485 port.
→ Transmission and reception of UART messages via 2D sub connections.
→ UART bit rates from 9600 bit/s to 1 Mbit/s.

6 Copyright

7 Contacts

Tel： +86(21) 6107 5469

Cell: +86 139 1860 5295

Address: Room 403, No.1, Lane 600,Tianshan Road, Changning District, Shanghai, China, Zip:200051

Website: www.art-logics.com

Support: nadir.ait@art-logics.com