SCU10 Product

USER MANUAL

Contents:

Chapter 1

SCU10 Product User Manual

1.1 Introduction

The **SCU10** is a programmable device with 16 **voltage aquisition channels** 16 bits precision and up to 100 kHz rate, 2 **CAN** bus channels, 5 **Digital Inputs** and 5 **Digital Outputs**. It offers three communication interfaces, **Ethernet TCP**, **USB-C** and **RS-232**, to configure the parameters of the voltage input channels and receive the aquisition data.

Also, the **SCU10** is able to run in a standalone mode without a host, by saving the aquisition data in its built-in micro SD Card.

1.2 Properties

- 16 channels differential voltage measurement with 16 bits precision and up to 100 kHz rate.
- 2 High-speed CAN channels (ISO 11898-2), with configurable Bit rate up to 1 Mbit/s.
- 5 Digital Input channels.
- 5 Digital Output channels.
- Selectable voltage input range -/+ 10 V and -/+ 40 V.
- Ethernet TCP, USB-C and RS-232 communication interfaces.
- Built-in SD Card to store the configurations and the data aquisitions.
- Software defined and external trigger signals support.
- External clock reference for synchronization.
- Ready to use C# and LabVIEW APIs.

1.3 Getting Started with the SCU10

To start using the SCU10, we need to provide a power supply source and a communication link.

Note: In the **standalone mode**, we only need to provide the power supply for the SCU10 to work. The operation can start and store the data using the SD Card. Check *Standalone Mode* section for more information.

1.3.1 Power Supply

1. Connect a 24V power supply through the DC power connector.
2. Check the LED on the front panel. If the LED is on, then the board is powered on.

1.3.2 Communication Link

If the **USB-C** cable is used for the communication:

- The device driver provided should be installed.
- Check that the USB COM Port of the device is detected.
- Open the COM Port to start the communication.

If the **RS-232** port is used for the communication:

- Configure the UART to the default parameters (baud 921600, data bits 8, stop bit 1, no parity bit).
- Open the COM Port to start the communication.

If the **Ethernet** interface is used for the communication:

- Connect the SCU10 to your IP local network.
- Optionally, it is useful to ping the IP address to make sure it is connected to the network.
- Establish the TCP connection with the configured IP address and port number (IP address 192.168.0.11, port 6025).

After establishing the connection with the SCU10, send the following command to make sure the communication is working:

```
@11_HELLO;
```

The board should say hello back:

```
[23/03/02,09:07:17.0100,0012]#11XX_HELLO;
```

1.3.3 Check the Software Version

The SYSID command can be used to check the current software version of the device, as well as its resources:

- **Syntax**:

  ```
  @n_SYSID=[<Param>];
  ```

- **Description**:

  <**Param**>: Optional Parameter:

  – *RESOURCES*: When this token is used, the command will return the list of available resources count on the device.

If no parameter, this command returns the software version of the device and board ID number.

- **Response**:

  The command is sent without parameters:

  ```
  #n_SYSID=<Software Version>;
  ```

  The command is sent with RESOURCES parameter:

  ```
  #n_SYSID=RESOURCES,<Resource channels>;
  ```

- **Example**:

  To get the current software version:

  ```
  @11_SYSID;
  #11_SYSID=SCU10_01_01_02_03;
  ```

  To get the available resources of the device:

  ```
  @11_SYSID=RESOURCES;
  #11_SYSID=RESOURCES,VI16,CAN2;
  ```

1.4 Communication Protocol

To send a command to the SCU10, the message always starts with '@' and ends with ';', and it includes the ID of the device, the command and its paramaters in the right order.

Once the device receives the command, it checks the ID and if it matches, the device executes the command and sends back a response starting with '#' and ending with ';' with a header including the timestamp and the size of the response.

Command Format:

```
@<ID>_<COMMAND>=<PARAMETERS>;
```

Response Format:

```
[yy/mm/dd,hh:mm:ss.msec,size]#<ID>_<COMMAND>=<RESULT>;
```

The following is an example of a command and response exchange between a host and the SCU10, to get the device's software version.

Command:

```
@11_SYSID;
```

Response:

```
[23/08/02,18:27:55.0684,0021]#11_SYSID=SCU10_01_01_03_04;
```

1.4.1 Commands List

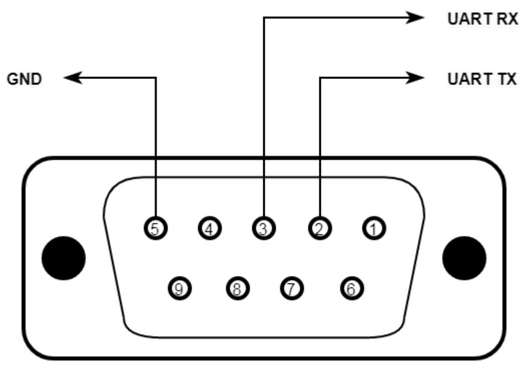
The following is the full list of the SCU10's commands:

Command	Description	Example
HELLO	Say hello to the device.	@11_HELLO;
SYSID	Get the software version of the device.	@11_SYSID;
UNIT	Control the state of the device (ON or OFF).	@11_UNIT=OFF;
ETH	Set and get the ethernet's configuration.	@11_ETH;
RTC	Set and get the real time clock parameters of the deviec.	@11_RTC;
STOR-AGE	A set of commands to manage the SD Card.	@11_STORAGE=SIZE;
CALBRT	Set and get the calibration parameters (Scale, Off-set).	@11_CALBRT=VIN,1, FS,1.453;
CONFIG	Configure the parameters of voltage channels.	@11_CONFIG=SAMPLING, CHAN-NEL,V,2,1;
TSTRT	Validate the configuration.	@11_TSTRT;
TSTOP	Reset the configuration.	@11_TSTOP;
GET-VOLT	Get the analog measurement channel's voltage value.	@11_GETVOLT=2;
SAM-PLING	Start and stop the voltage aquisitions.	@11_SAMPLING=STOP,V,2;
CAN	Send and receive a CAN message.	@11_CAN=1,STD,0X11,0X34FF;
SETDIG	Change the state of the digital output to high.	@11_SETDIG=1;
CLRDIG	Change the state of the digital output to low.	@11_CLRDIG=5;
GETDIG	Get the state of the digital input.	@11_GETDIG=2;

1.5 Communication Interface

A control host can communicate with the SCU10 via the **USB-C** 2.0 connection, or over an IP network via **TCP** with the 10/100 Mbps Ethernet link.

1.5.1 RS-232 Communication



The SCU10 has one RS-232 port that can be used for the communication with the device. The default parameters of the UART are the following:

Baud
921600

Data Bits
8

Stop Bit
1

Parity
None

1.5.2 USB-C Communication

The SCU10 has one USB Port 2.0 type C that can be used for the communication with the device. It is detected as an USB COM Port and can be used with any serial terminal.

Note: Install the provided device driver before you connect the SCU10.

1.5.3 Ethernet TCP Communication

If the Ethernet interface is used for the communication, the default parameters are the following:

Protocol
TCP

IP address
192.168.0.11

IP mask
255.255.255.0

Port number
6025

These parameters are not fixed, and can be changed using commands.

Set the IP Addresses

To change the source, gateway or mask IP address of the device, the user should send the following command.

- **Syntax**:

```
@n_ETH=<IP type>,<IP Address>;
```

- **Description**:
  - <**IP type**>: the type of the IP address to change. A valid IP type is: SOURCE, GATEWAY and MASK.
  - <**IP Adress**>: the new IP address to be set. The format is four decimal values seperated by ..

  The command changes the source IP address of the device to 192.168.0.131.

- **Response**:

```
#n_ETH=<IP type>,<IP Address>;
```

- **Example**:

```
@11_ETH=SOURCE,192.168.0.131;
#11_ETH=SOURCE,192.168.0.131;
```

Get the IP Addresses

To see the current Ethernet configuration of the device with the IP addresses, the user should send the following command.

- **Syntax**:

```
@n_ETH;
```

- **Description**:

  The response of the command contains the type of IP address followed by the current IP address set for the source, gateway and mask addresses.

- **Response**:

```
#n_ETH=SOURCE,<Source IP Address>,GATEWAY,<IP Address>,MASK,<IP Address>;
```

- **Example**:

```
@11_ETH;
#11_ETH=SOURCE,192.168.0.11,GATEWAY,192.168.0.210,MASK,255.255.255.0;
```

1.6 Real Time Clock

The **SCU10** integrates a real time clock that can keep the date and time values even after powering off the device. This clock is used mainly to set the timestamp of every device's action and include it in the header of every response. For example, the timestamp of a CAN message reception or a voltage measurement.

1.6.1 Set RTC parameters

To set new parameters of the RTC, the user needs to send this command:

- **Syntax**:

```
@11XX_RTC=SET,<Year>,<Month>,<Day>,<WeekDay>,<Hours>,<Minutes>,<Seconds>;
```

- **Acknowledge**:

```
#11XX_RTC=<Year>,<Month>,<Day>,<WeekDay>,<Hours>,<Minutes>,<Seconds>;
```

- **Description**:
  - <**Year**>: Value less than 99.
  - <**Month**>: Value between 1 and 12.
  - <**Day**>: Value between 1 and 31.
  - <**WeekDay**>: Value between 1 and 7.
  - <**Hours**>: Value between 0 and 23.
  - <**Minutes**>: Value between 0 and 59.
  - <**Seconds**>: Value between 0 and 59.
- **Example**:

```
@1111_RTC=SET,23,8,30,3,8,21,1;
#1111_RTC=23,8,30,3,8,21,1;
```

After sending this command, the board RTC is set to 23/08/30 08:21:01.

1.6.2 Get RTC parameters

The user can get the parameters of the RTC currently used by the device.

- **Syntax**:

```
@11XX_RTC=GET;
```

- **Acknowledge**:

```
#11XX_RTC=<Year>,<Month>,<Day>,<WeekDay>,<Hours>,<Minutes>,<Seconds>;
```

- **Description**:

  The parameters are the same as the Set RTC command.

- **Example**:

```
@1111_RTC=GET;
#1111_RTC=23,3,2,6,21,24,19;
```

  The response of the board indicates the RTC parameters: 23/03/02 21:24:19.

1.7 Resources

The **SCU10** offers 2 CAN channels 16 voltage aquisition channels as resources to be used and controlled in the test sequences defined by the user.

Resource	Number of channels
Voltage Input	16
CAN	2
Digital Input	5
Digital Output	5

1.7.1 CAN bus

The **SCU10** has 2 High-Speed CAN connections (ISO 11898-2), compliant with the CAN specifications ISO 11898-1.

Using the defined commands, the user can configure the baudrate of the bus, send and receive CAN messages.

The CAN termination can be activated through the jumper **JP901** for the CAN1, and the jumper **JP900** for the CAN2.

Configure the Baudrate

To select the baudrate of the CAN bus, the user needs to send the following command:

- **Syntax**:

```
@11_CONFIG=CAN<channel>,BAUDRATE,<baudrate>;
```

- **Acknowledge**:

```
#11_CONFIG=CAN<channel>,BAUDRATE,<baudrate>;
```

- **Description**:
  - <**channel**>: This parameter defines the CAN bus to configure. Valid value is from **1** to **2**.
  - <**baudrate**>: The value of the baudrate to set. The supported baudrate values are: 10K, 20K, 33.3K, 40K, 83.3K, 100K, 125K, 250K, 500K and 1000K (or 1M).
- **Example**:

```
@11_CONFIG=CAN1,BAUDRATE,500K;
#11_CONFIG=CAN1,BAUDRATE,500K;
```

  This command sets the baudrate **500 kbit/s** for the **CAN channel 1**.

Send a CAN Message

The following command is used to send a CAN message in the specified CAN channel:

- **Syntax**:

```
@11_CAN=<channel>,<ID type>,<ID>,<message>;
```

- **Acknowledge**:

```
#11_CAN=<channel>,<ID type>,<ID>,<message>;
```

- **Description**:
  - <**channel**>: This parameter defines the CAN bus used to send the message. Valid value is from **1** to **2**.
  - <**ID type**>: The standard or extended mode of the CAN message ID. **STD** (11 bits) or **EXT** (29 bits).
  - <**ID**>: The ID of the CAN message to send. A hexadecimal number preceded by *0X* from 0X00 to 0X7FF in STD and from 0X00 to 0X1FFFFFFF for EXT mode.
  - <**message**>: the data bytes to send preceded by *0X* (in hexadecimal). The maximum length is 8 bytes.
- **Example**:

```
@11_CAN=1,STD,0X13,0X401100FE;
@11_CAN=1,STD,0X13,0X401100FE;
```

  This command sends a CAN message **0x401100FE** with the Standard CAN ID **0x13** on the CAN channel **1**.

Receive a CAN Message

The device automatically send the messages received to the host with the following format:

- **Message Format**:

```
#11_CAN=<channel>,<ID type>,<ID>,<message>;
```

- **Description**:
  - <**channel**>: This parameter defines the CAN bus that received the message. Valid value is from **1** to **2**.
  - <**ID type**>: The standard or extended mode of the CAN message ID. **STD** (11 bits) or **EXT** (29 bits).
  - <**ID**>: The ID of the CAN message received. A hexadecimal number preceded by *0X* from 0X00 to 0X7FF in STD and from 0X00 to 0X1FFFFFFF for EXT mode.
  - <**message**>: the data bytes received preceded by *0X* (in hexadecimal). The maximum length is 8 bytes.
- **Example**:

```
#1111_CAN=1,STD,0XF0,0X3FEE45;
```

  This command sends back the CAN message received with the standard CAN ID with 0xF0 on the CAN channel **1**.

Quick Validation Test

To validate the opration of the CAN channels of the **SCU10**, the user can connect the two channels together and perform a quick send and receive test.

- Connect the CAN1_L with the CAN2_L.
- Connect the CAN1_H with the CAN2_H.
- Make sure that the CAN termination with the jumper is the same for both channels.
- Configure the same baudrate for CAN1 and CAN2:

```
@11_CONFIG=CAN1,BAUDRATE,500K;
@11_CONFIG=CAN2,BAUDRATE,500K;
```

- Send a message from CAN1 to CAN2, and from CAN2 to CAN1:

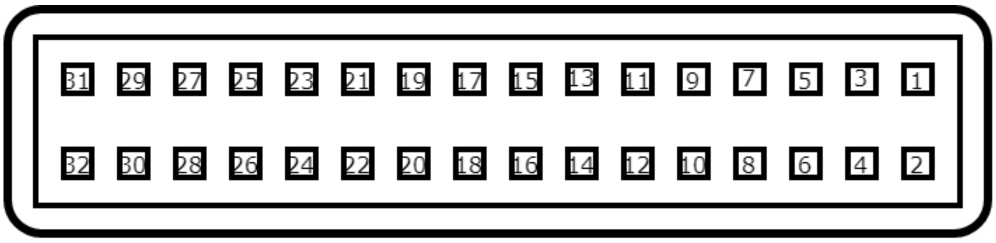
```
@11_CAN=1,STD,0X11,0X0102030405060708;
@11_CAN=2,STD,0X22,0X1122334455667788;
```

- Check the reception of the same messages from CAN1 and CAN2:

```
#11_CAN=1,STD,0X22,0X1122334455667788;
#11_CAN=2,STD,0X11,0X0102030405060708;
```

1.7.2 Voltage Acquisition

The SCU10 offers 16 channels of differential voltage inputs, with a precision of 16 bits, accessible from the 32 pins connector on the front panel. The channels support two modes of operation, the **Single Mode** and the **Sampling Mode**.



01: VIN1+	**09**: VIN5+	**17**: VIN9+	**25**: VIN13+
02: VIN1-	**10**: VIN5-	**18**: VIN9-	**26**: VIN13-
03: VIN2+	**11**: VIN6+	**19**: VIN10+	**27**: VIN14+
04: VIN2-	**12**: VIN6-	**20**: VIN10-	**28**: VIN14-
05: VIN3+	**13**: VIN7+	**21**: VIN11+	**29**: VIN15+
06: VIN3-	**14**: VIN7-	**22**: VIN11-	**30**: VIN15-
07: VIN4+	**15**: VIN8+	**23**: VIN12+	**31**: VIN16+
08: VIN4-	**16**: VIN8-	**24**: VIN12-	**32**: VIN16-

Single Mode

The single mode is used to get one single voltage measurement from a channel by sending the GETVOLT command.

- **Syntax**:

```
@n_GETVOLT=<channel>;
```

- **Description**:
  - <**channel**>: defines the voltage input channel number in the SCU10. Valid value is 1~16.
- **Response**:

```
#n_GETVOLT=<channel>,<voltage value>;
```

  The response of the command returns the voltage value measured at the moment the command was received. Valid value is a decimal number in the range -/+ 10 V or -/+ 40V depending on the range programmed.

- **Example**:

```
@11_GETVOLT=11;
#11_GETVOLT=11,6.094217;
```

  The voltage measured on the channel number 11 is 6.094217 V.

Sampling Mode

The sampling mode of the acquisition is used to measure the voltage channels continuously or for a time window, with a sampling rate up to 100 kHz. To use this mode, the user needs to configure first the acquisition parameters and once the configuration is set and validated, the sampling operation can be started and stopped, with the voltage data values being pushed automatically to the host or saved in the SD Card in the stand alone mode.

Channel Configuration

To set the acquisition parameters of the channels, the CONFIG commands is used.

- **Syntax**:

```
@n_CONFIG=SAMPLING,CHANNEL,V,<channel>,<mode>,<windows count>,<acq period>,
<time unit>,<sampling period>,<filter>,<amplitude>,<offset>,<window logging>,
<graph logging>,<Compression>[TRIGGER,<trigger source>,<trigger edge>,
<precision>,<level>,<setup time>,<Holdoff>,<filter flag>];
```

- **Description**:
  - <**channel**>: The voltage input channel number in the SCU10. Valid value is 1~16.
  - <**mode**>: The acquisition mode for this channel. Up to 16 different modes can be defined on the same channel. Different modes can be started one after the other but never in the same time for one channel. Valid value is 1~16.
  - <**windows count**>: The number of windows for the selected channel in the selected mode. Up to 16 different windows can be defined in one mode. They can be started one after the other or in the same time. One window is associated with an acquisition calculation mode. Valid value is 1~16.
  - <**Acq period**>: The time period of the acquisition, starting after the trigger or directly after the SAMPLING=START command if no trigger is defined on the specified channel. Valid value is a multiple of 10 starting from 0,10 to 4294967290. The value 0 defines a continuous operation until a SAMPLING=STOP is received.
  - <**time unit**>: Token specifying the acquisition period unit. Valid values are US for microseconds, MS for milliseconds and S for seconds.
  - <**sampling period**>: the time between each acquisition point in microseconds. Valid value from 10 us to 1000 us with a 10 us stepping.
  - <**filter**>: The filter applied on the Signal, valid value is one of the following token: SA|NONE. SA stands for Sliding Average, performed over an optimized number of points given the acquisition period and speed, not configurable, default value is 3. NONE is used for the No Filter option.
  - <**amplitude**>: Amplitude of the signal in Volts. This parameter is useful when performing a RLE (Run Length Encoding) signal compression to determine the Epsilon value for the compression as 1% of the <amplitude> value. Valid value is a float.
  - <**offset**>: This parameter is useful when performing "level" operation, it fixes the histogram calculation on the values present in the range specified with <amplitude> and <offset> parameters. Valid value s a float.
  - <**window logging**>: A token to define the logging condition of window's operation data.
    - NEVER: only Failure Table are returned, no Data.
    - ONFAIL: Failure Table are returned with Data of failed windows.
    - ALWAYS: Failure Table and all data returned.
  - <**graph logging**>: A token specifying if a graph of points is sent back along with the windows results.
    - NEVER: no graph returned.
    - ONFAIL: graph returned if any window fails.

    - ALWAYS: graph always returned.
  - <**compression**>: A token to specify the compression method used to return graph points, it can be: NONE, SUBS8, SUBS16, RLE.
    - NONE: no compression. All points are returned up to the maximum allowed.
    - SUBS8: oversampling, one point over 8 is returned.
    - SUBS16: high oversampling, one point over 16 is returned.
    - RLE: RLE operation, consecutive data points in an epsilon range are returned as one point. Epsilon is fixed at 1% of <amplitude> parameter.

  The following parameters are optional, if a trigger event will start the acquisition.

  - **TRIGGER**: Token specifying to start the trigger definition optional parameters
  - <**trigger source**>: Trigger source type, INT for internal and EXT for external.
  - <**trigger edge**>: Trigger to define the trigger type, RISING for rising edge, FALLING for falling edge.
  - <**precision**>: The number of acquisition points used to calculate the trigger edge. Valid value is from 2 to 256 in decimal. Only power of 2 values will be internally used for calculations. If another value is used, the closest lower power of 2 values will be chosen.
  - <**level**>: The level of the trigger, in channel type unit. Value is a float, for example 5.0 for a trigger level at 5 Volts.
  - <**setup time**>: The time to wait after trigger event before starting the acquisition, in acquisition unit. Valid value from 0 to 65535, in microseconds.
  - <**holdoff**>: The time from trigger event during which no new trigger event can occur. Valid value is from 0 to 4294967290, in microseconds.
  - <**filter flag**>: A token specifying if the trigger calculation system using the filter previously defined or if it is calculated over the raw acquisition points. Valid values are FILTERED and UNFILTERED.

- **Response**:

  The device sends back the configuration set in case it is valid. Otherwise, the response will contain an error message.

- **Example**:

```
@11_CONFIG=SAMPLING,CHANNEL,V,2,1,1,10,S,10,0,SA,10,DATA:NEVER,GRAPH:ALWAYS,NONE;
#11_CONFIG=SAMPLING,CHANNEL,V,2,1,1,10,S,10,0,SA,10,DATA:NEVER,GRAPH:ALWAYS,NONE;
```

  This command configures mode number 1 for voltage input 1 with 1 window for a 10s acquisition at 100 kHz using the Sliding Average filter over a signal from 0V to 10V, without compression and without trigger.

Configuration Validation

After sending the configuration of all the channels to be used, the user needs to send the following command to validate the configuration:

- **Syntax**:

```
@n_TSTRT;
```

- **Description**:

  The TSTRT command validates the configuration of the channels set previously.

- **Response**:

```
#n_TSTRT;
```

- **Example**:

```
@11_TSTRT;
#11_TSTRT;
```

Configuration Deletion

In case the user wants to delete the configuration of the channels, the TSTOP can be used for this purpose.

- **Syntax**:

```
@n_TSTOP;
```

- **Description**:

  This commands deletes all the configuration sent previously. The channels should be stopped before sending this command.

- **Response**:

```
#n_TSTOP;
```

- **Example**:

```
@11_TSTOP;
#11_TSTOP;
```

Starting the Acquisition

After sending the configuration of the channels to be used and validating it, the command to start the aquisition of the voltage inputs is the following.

- **Syntax**:

```
@n_SAMPLING=START,V,<channel>,<mode>;
```

- **Description**:

  This commands starts the voltage acquisitions immediately upon reception. In case a trigger was set, the device waits for the trigger to happen to start the acquisitions.

  – **channel**: The voltage input channel number in the SCU10. Valid value is 1~16.

  – <**mode**>: The acquisition mode for the channel to start.

- **Response**:

```
#n_SAMPLING=START,V,<channel>,<mode>;
```

- **Example**:

```
@11_SAMPLING=START,V,10,1;
#11_SAMPLING=START,V,10,1;
```

Stopping the Acquisition

The acquisition of the channels is stopped automatically upon reaching the time aquisition period defined in the configuration. Also, the user can stop it by sending the STOP command.

- **Syntax**:

```
@n_SAMPLING=STOP,V,<channel>;
```

- **Description**:

  This commands stops the voltage acquisitions immediately upon reception.

  – **channel**: The voltage input channel number in the SCU10. Valid value is 1~16.

- **Response**:

```
#n_SAMPLING=STOP,V,<channel>;
```

- **Example**:

```
@11_SAMPLING=STOP,V,16;
#11_SAMPLING=START,V,16;
```

1.7.3 Digital Input

The **SCU10** offers **5 Digital Inputs** as resources.

Parameters

- **Input impedance**: 100 kOhms.
- **Input voltage range**: -50 V to 50 V.
- **Threshold**: 2.5 V.
- **Hysteresis**: 17 mV typical.

All the digital inputs share the same ground return.

Reading the Digital State

To get the state of the digital input, the user needs to send the following command:

- **Syntax**:

```
@11_GETDIG=<channel>;
```

- **Acknowledge**:

```
#11_GETDIG=<channel>,<state>;
```

- **Description**:

  – <**channel**>: This parameter defines the digital input channel in the SCU10 to access for reading. Valid value is from **1** to **5**.

  – <**state**>: The acknowledge returns the state of the digital input requested. **1** means high state and **0** means low state.

- **Example**:

```
@1111_GETDIG=3;
#1111_GETDIG=3,1;
```

The digital input channel **3** is in **high** state.

1.7.4 Digital Output

The **SCU10** offers **5 Digital Onputs** as resources.

Parameters

The digital output is an open drain driver, and it is protected for short circuit to positive battery and over temperature.

- **Output impedance**: 60 mOhms.
- **Off leakage current**: 75 µA.
- **Maximum voltage**: 35 VDC.
- **Maximum current**: 2.0 A.

All the digital outputs share the same ground return.

Set the Digital Output State

To set the state of the digital output to **high**, the user needs to send the following command:

- **Syntax**:

```
@11_SETDIG=<channel>;
```

- **Acknowledge**:

```
#11_SETDIG=<state mask>;
```

- **Description**:
  - <**channel**>: This parameter defines the digital output channel in the SCU10 to set to **high**. Valid value is from **1** to **5**.
  - <**state mask**>: The acknowledge returns the state of all digital outputs in hexadecimal format starting with '0X'. The bit 0 is corresponding to the channel 1, and the bit 4 is corresponding to the channel 5; with **1** for a **high** state and **0** for a **low** state.
- **Example**:

```
@11_SETDIG=2;
#11_SETDIG=0X13;
```

The value returned *0X13* in hexadecimal format, which is *10011* in binary format indicates:

Channel	State
1	high
2	high
3	low
4	low
5	high

Clear the Digital Output State

To set the state of the digital output to **low**, the user needs to send the following command:

- **Syntax**:

```
@11_CLRDIG=<channel>;
```

- **Acknowledge**:

```
#11_CLRDIG=<state mask>;
```

- **Description**:
  - <**channel**>: This parameter defines the digital output channel in the SCU10 to set to **low**. Valid value is from **1** to **5**.
  - <**state mask**>: The acknowledge returns the state of all digital outputs in hexadecimal format starting with '0X'. The bit 0 is corresponding to the channel 1, and the bit 4 is corresponding to the channel 5; with **1** for a **high** state and **0** for a **low** state.

- **Example**:

```
@1111_CLRDIG=5;
#1111_SETDIG=0X0F;
```

The value returned *0X0F* in hexadecimal format, which is *01111* in binary format indicates that the channel **5** is in **low** state, and other channels are **high**.

1.8 Synchronization

In many cases, there is a need to use and work with more than one SCU10 device to extend the voltage channels number. Therefore, these devices need to be synchronized by receiving a clock signal of **100 kHz** from the same source. This signal should be applied on the **Sync** pin 8 and **GND** pin 9, of the DB9 on the front panel.

1.8.1 Synchronization Control

To start or stop the synchronization function on the SCU10, the following command is used.

- **Syntax**:

```
@n_SYNCHRO=<action>;
```

- **Description**:
  - <**action**>: START to enable the synchronization function, and STOP to disable it.

- **Response**:

```
#n_SYNCHRO=<action>;
```

- **Example**:

```
@11_SYNCHRO=START;
#11_SYNCHRO=START;
```

The synchronization function will be actived if the clock signal is applied.

1.8.2 Synchronization Status

The following command is used to check the status of the synchronization function.

- **Syntax**:

```
@n_SYNCHRO=STATE;
```

- **Description**:

  The response of the command gives back the status of the synchronization function. It returns **ACTIVE** or **INACTIVE**.

- **Response**:

```
#n_SYNCHRO=<status>;
```

  - <**status**>: ACTIVE if enabled or INACTIVE if disabled.

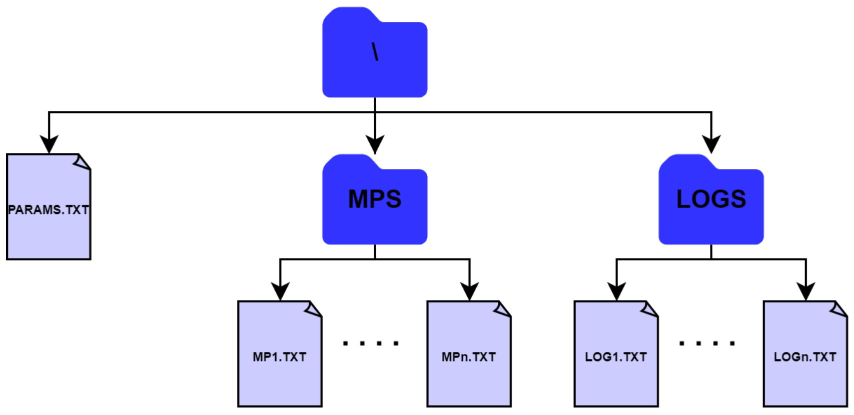
- **Example**:

```
@11_SYNCHRO=STATE;
#11_SYNCHRO=ACTIVE;
```

1.9 Standalone Mode

the SCU10 is able to run in an independant mode without the intervention of a host controller connected to the device. In this Standalone Mode, the SCU10 downloads the configuration of the input channels automatically from the SD Card at the startup and saves the results in the form of logs.

The SD Card file system is organized as a file **PARAMS.TXT** for the global parameters, a folder **MP** that contains the aquisition configurations and a folder **LOGS** that contains the saved results.



1.9.1 Standalone Parameters

The parameters of the Standalone Mode using the SD Card are stored in the **PARAMS.TXT** file. These are the configurable parameters:

- **LOGGING**: Enable (=1) or disable (=0) the logging of the results in the **LOGS** folder.
- **AUTO_READ**: Enable (=1) or disable (=0) the automatic loading the configurations at the startup of the SCU10.
- **AUTO_PUSH**: Enable (=1) or disable (=0) the board to push the results on the communication interface.

1.9.2 Test Configuration

The configuration of the channels are saved inside the **MP** folder. The user can manually connect the SD Card on the computer and save the configuration, or instruct the SCU10 to save the commands:

```
@n_STORAGE=DLSTART;
<Configurations>
@n_STORAGE=DLSTOP;
```

Inside the folder **MP**, we can find two types of files:

- **CONFIG.TXT**: contains the commands CONFIG shared by all processes.
- **MP<n>.TXT**: contains the configuration of the process with the identifier n.

1.9.3 Saving the Results

The SCU10 saves the results inside the folder **LOGS**, if the parameter **LOGGING** is enabled in the **PARAMS.TXT** file.

Each result is saved in one line following the same format: **<Timestamp>,<Result>\r\n**

1.10 Copyright

1.11 Contact

Tel: +86(21) 6107 5469

Cell: +86 139 1860 5295

Website: www.art-logics.com

Support: support@art-logics.com