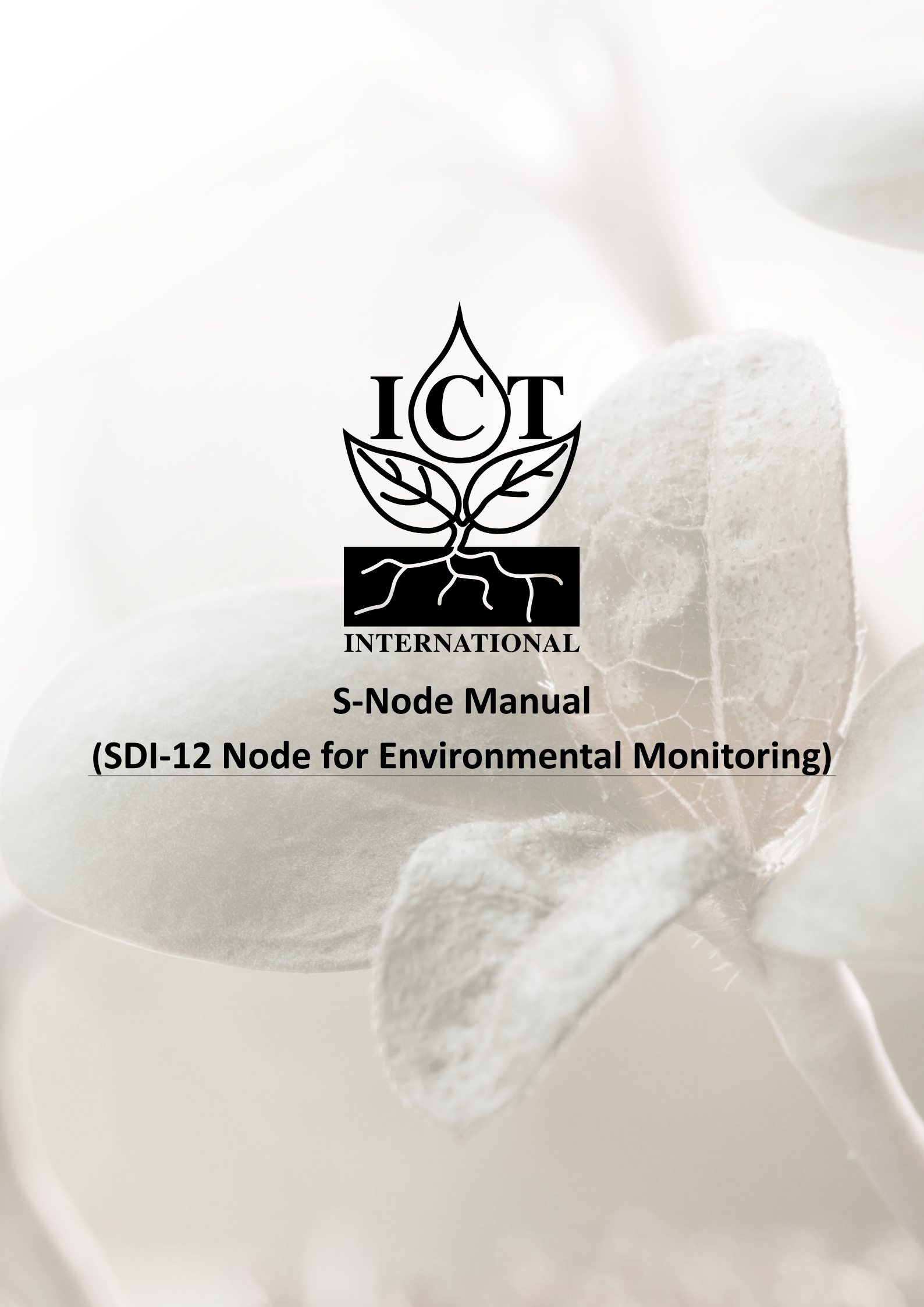


S-Node Manual

(SDI-12 Node for Environmental Monitoring)

Contents

1. Introduction

The ICT International S-Node is a low power SDI-12 LoRaWAN data transmission device.

The device is powered by a lithium-ion battery pack (6.7Ah or 13.4Ah) and is charged by external 12-24V DC input – typically a 10W or 20W solar panel.

Long-range low-power communications are achieved through use of LoRaWAN™ networks via an onboard LoRa® radio.

It is designed to easily connect to 4 x SDI-12 sensors. The Node can be customised to accommodate more sensors upon request.

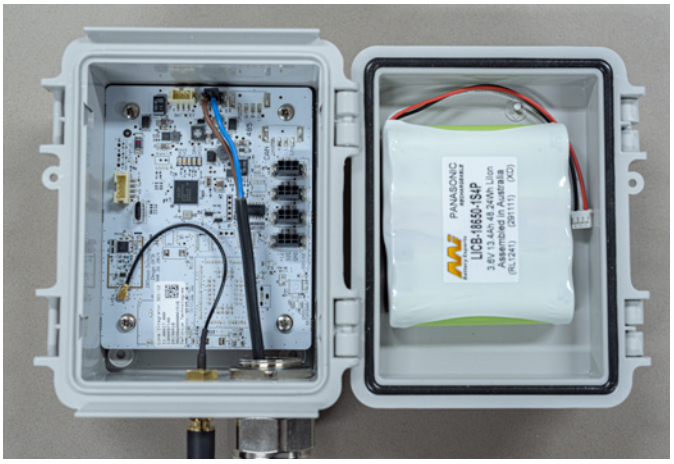


Figure 1. Inside The S-Node Box - The S-Node Board

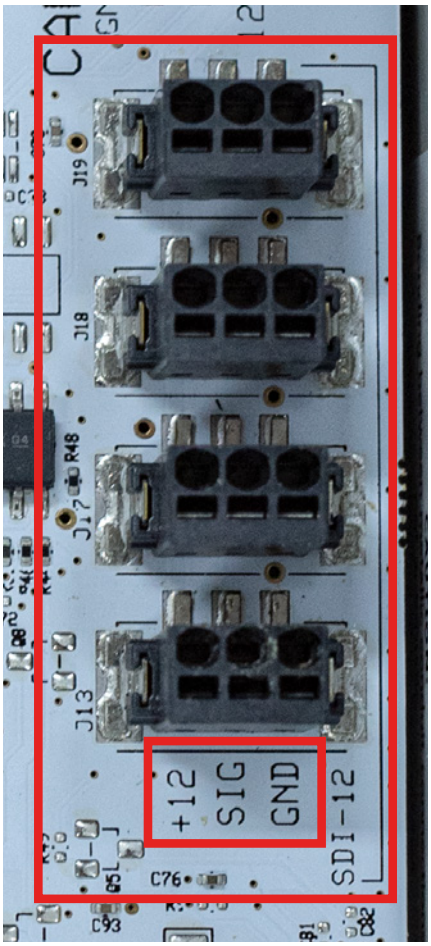
Figure 2. The S-Node Box - With Antenna

2. Connecting SDI-12 Sensors

A connector is available on the right-hand side of the board for each type of sensor. See instructions below for removing and inserting sensor wires to these sockets.

For all four SDI-12 Sensor ports, the left sockets are labelled +12 (Excitation Power), the middle sockets are SIG (SDI-12 Signal) and the right sockets are labelled GND (Ground). Match each wire core to their correct respective sockets accordingly.

Please refer to each Sensor Manufacturers Manual for technical information on wire colouring and identification.



2.1 Connecting and Removing The Wires

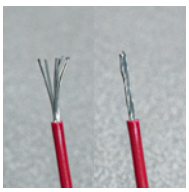
For each device and their sensor wires follow the instructions below. The following steps also apply to wires connecting to an external solar power source. The images below and right also correspond to their respective steps.

2.1.1 Insert and Connect The Wires

1. Prepare the wire - Strip the outer insulation of the inner conductors to reveal the inner conductor exposed wire. If stranded, lightly twist before inserting in (1.a). Make sure the final amount of exposed wire is a maximum length of 7mm (1. b).
2. Use a pair of tweezers or suitable flathead screwdriver to push into the square hole beside the desired socket to temporarily release the spring clamp.
3. Hold the tweezers down and insert the prepared wire (3.a). Push the wire in, down all the way (3.b).
4. Remove the tweezers and the wire should be clamped in.
5. After removing tweezers, do a simple gentle tug test to each wire to be sure it is clamped and secured correctly.

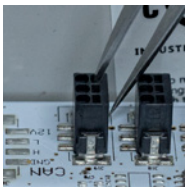
2.1.2 Remove or Reposition Wires

2. Use a pair of tweezers or suitable flathead screwdriver to push into the square hole beside the desired socket to temporarily release the spring clamp.
3. Hold the tweezers down and gently pull out the inner wire to remove it - (3.b) then (3.a). Remove the tweezers. The wire is now free to remove or reposition. To reinsert, follow the order of instructions above in 4.1.1.

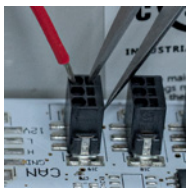


1.a

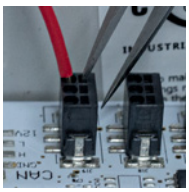
1.b



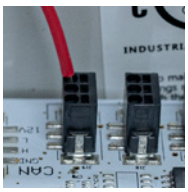
2.



3.a



3.b



4.

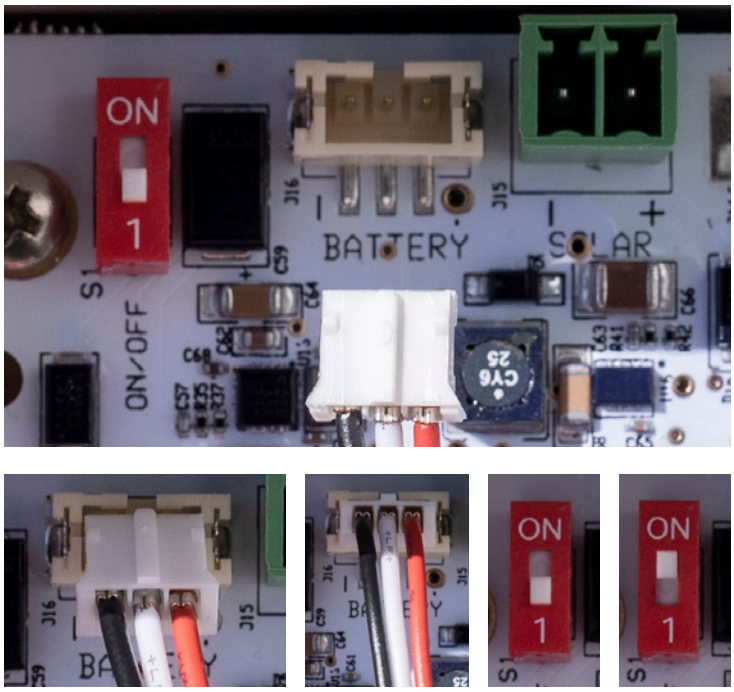
3. Connect The Internal Battery

To commence measurement and communication operations you need to power the node:

1. Ensure the internal battery is plugged in to the battery port.
2. Turn the power switch to the “ON” position.

After the node is powered via the internal battery an external DC power source can be connected (refer to section 4)

If de-powering the node, ensure that external DC power source has been removed, and turn the power switch to the “1” position.

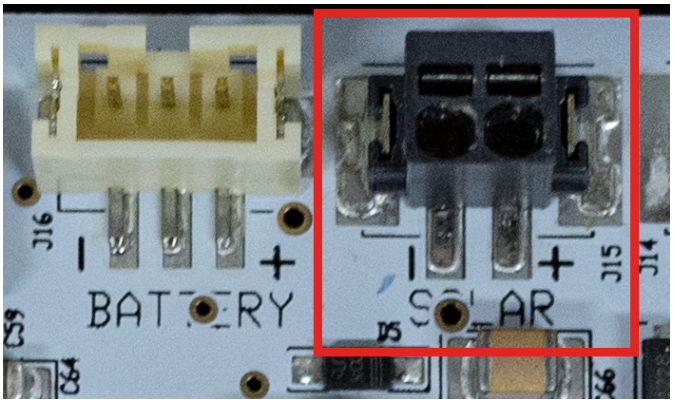


S-NODE-LA2 board version showing battery connection and power switch.

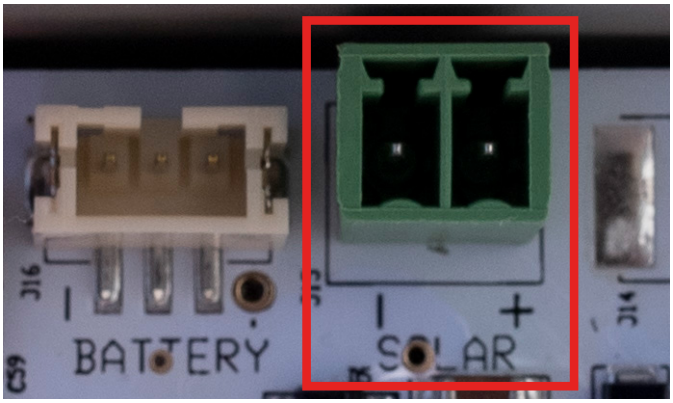
6. Connect External DC Power

The input for external power is located at the top, on the right of the board, labelled SOLAR.

Please ensure that the S-Node’s internal battery is plugged in and the device power is turned on before connecting the solar or external power.



S-NODE-LA1 Board version of the solar port - requiring the power to be connected and disconnected via the instructions on page 5.



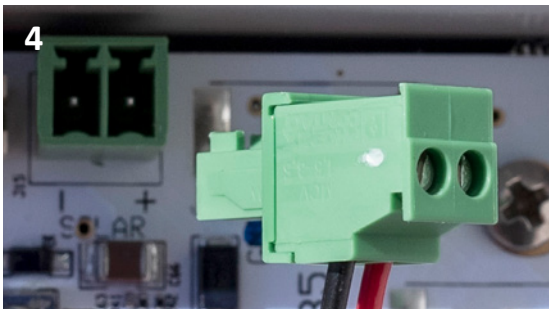
S-NODE-LA2 Board version of the exernal/solar power port (pheonix connector).

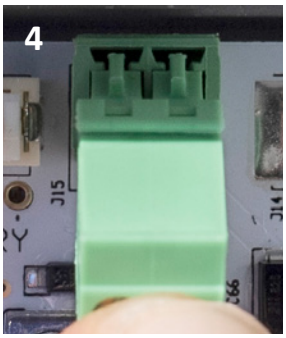
6.1 S-NODE-LA1 And -LA2 Board Versions

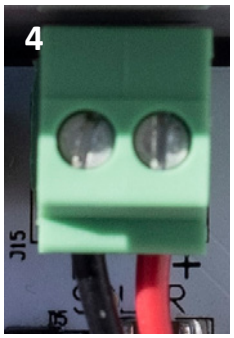
Please follow the instructions on page 5 for connecting and disconnecting power wires on S-NODE-LA1 boards.

This input is polarised, please ensure that positive (+) is inserted in the (+) terminal and negative (-) in the (-) terminal. It's advised to use a multi-meter to test the polarity before connecting.

The S-NODE has an on-board solar charge controller and can be directly connected to a 12V or 24V solar panel. Alternatively, a 12V to 24V mains DC power supply can be connected for indoor use.



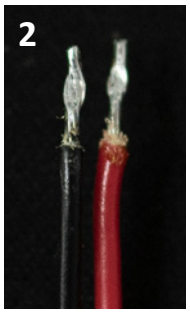


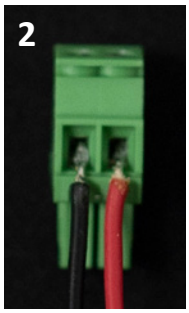


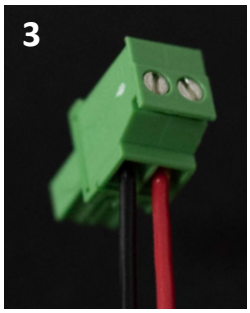
S-NODE-LA2 Board version of the solar port (pheonix connector).

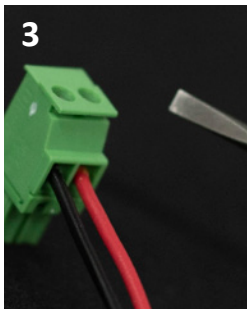
6.3 Integrations with -LA2 Pheonix Connector

1. Take the removable green pheonix connector out.
2. Prepare and insert soldered wires into the correct terminal.
3. Insert the wires and use a flathead screw driver to secure the wires in.
4. Plug in the pheonix connector in.





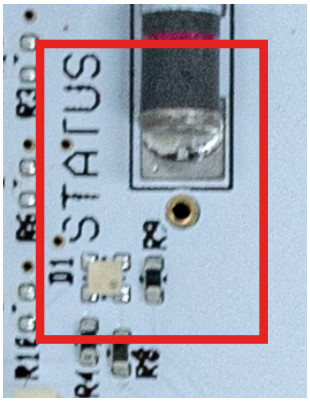




7. LED Status

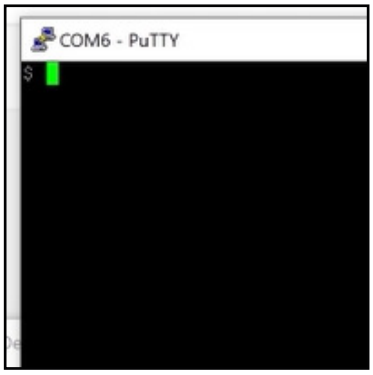
The status led is used to indicate the following:

- **LIGHT BLUE**: Joining Network
- **DARK BLUE**: Network Joined / Taking measurement
- **ORANGE**: Transmitting sensor data
- **PURPLE**: Measurement Complete
- **GREEN**: USB Idle
- **RED**: Failed to Join Network



5. Device Operation

5.1 SDI-12 Configuration



Configuration Program Example

The S-Node's SDI-12 communications can be customised to allow for interfacing with any SDI-12 sensor. See section 8.5 for a detailed description of the configuration interface of the device.

Adding or modifying an SDI-12 command for taking a measurement from a sensor uses the `sdi12 add` command. Commands can be disabled/enabled using the `sdi12 activate` command – by default, commands are enabled when they are added or modified. Other commands such as diagnostic and calibration requests can be sent directly to the sensor using the `sdi12 send` command.

For more information on SDI-12 related commands, see section 8.5.

7. Decoder Notes

Please contact ICT International for the S-NODE decoder applicable to your order and suitable for TTN (https://www.thethingsnetwork.org/). For Integrators, the following is a base decoder for customising:

```
/* TTN S-Node Flat Decoder Base
      Includes Base Diagnostic
*/
/*Class Buffer
      Purpose: A psuedo buffer class for accessing packet data,
            allows for uniformity between decoder types
*/
function Buf(buf){this.pl=buf}
Buf.prototype.readUInt8=function(ofs){return this.pl[ofs]};
Buf.prototype.readUInt16BE=function(ofs){return this.pl[ofs++]<<8|this.pl[ofs++]};
Buf.prototype.readUInt32BE=function(ofs){return this.pl[ofs++]<<24|this.pl[ofs++]<<16|this.
pl[ofs++]<<8|this.pl[ofs++]};
Buf.prototype.readFloatBE=function(ofs){return B2Fl(this.readUInt32BE(ofs))};
Buf.prototype.slice=function(s,e){return this.pl.slice(s,e)};
Buf.prototype.length=function(){return this.pl.length};

/*Function Bytes2Float32(bytes)
      Purpose: Decodes an array of bytes(len 4(32 bit) into a float.
      Args:  bytes - an array of bytes, 4 bytes long
      Returns: 32bit Float representation
*/
function B2Fl(b){
      var sign=(b&0x80000000)?-1:1;
      var exp=((b>>23)&0xFF)-127;
      var sig=(b&~(-1<<23));
      if(exp==128) return sign*((sig)?Number.NaN:Number.POSITIVE_INFINITY);
      if(exp==-127){
            if(sig===0) return sign*0.0;
            exp=-126;
            sig/=(1<<22);
      } else sig=(sig|(1<<23))/(1<<23);
      return sign*sig*Math.pow(2,exp);}

/*Function Decoder(bytes, port)
      Purpose: Main Entry point of TTN Console Decoder
      Args:  bytes - An array of bytes from LoRaWan raw payload(Hex Represented)
                  port - LoRaWan Port that the message came through(set by Definium
firmware)
```

```
	Returns: decoded - An object with data fields as decoded parameter values
*/
function Decoder(b,p){
	var byte = 0;
	var decoded = {};
	var buf = new Buf(b);

	// Data Packet Received
	if(p === 1){
		decoded['packet-type'] = "DATA_PACKET";
		decoded["uptime-s"] = +(buf.readUInt32BE(byte));
		decoded["battery-voltage-V"] = +(buf.readUInt16BE(byte = byte+4)/1000).
toFixed(3);
		decoded["solar-voltage-V"] = +(buf.readUInt16BE(byte = byte+2)/1000).toFixed(3);
		var charge_fault = buf.readUInt8(byte=byte+2);
		decoded["charging-state"] = +(charge_fault & 1);
		decoded["fault"] = +((charge_fault & 2) >> 1);

		var com = buf.readUInt8(byte = byte+1);
		decoded["command"] = com;
		byte = 10;

		// SDI From Command 0
		if(com == 0){
			//
		}
		// End SDI-12 Section
	}else if(p === 10){
		//Device Info Packet Recieved
		decoded["packet-type"] = "DEVICE_INFO";
		decoded["product-id"] = buf.readUInt32BE(byte);
		decoded["batch-number"] = buf.readUInt32BE(byte=byte+4);
		decoded["software-version"] = buf.readUInt32BE(byte=byte+4);
	}else if(p === 100){
		//Downlink Response Packet Recieved
		decoded["packet-type"] = "DOWNLINK_RESPONSE";
		decoded["downlink-response"] = String.fromCharCode.apply(String, buf.slice(0,
buf.length));
	}else{
		// Unknown Response Recieved
		decoded["packet-type"] = "UNKNOWN_RESPONSE";
		decoded["raw-payload"] = buf.slice(0, buf.length);
	}

	return decoded;		//Return 'decoded' object
}
```

8. Device Configuration

The ICT International S-Node is configured (i.e. identified and authenticated) over USB serial console using a terminal or terminal emulator. It is compatible with Windows 10, Mac OS and Linux.

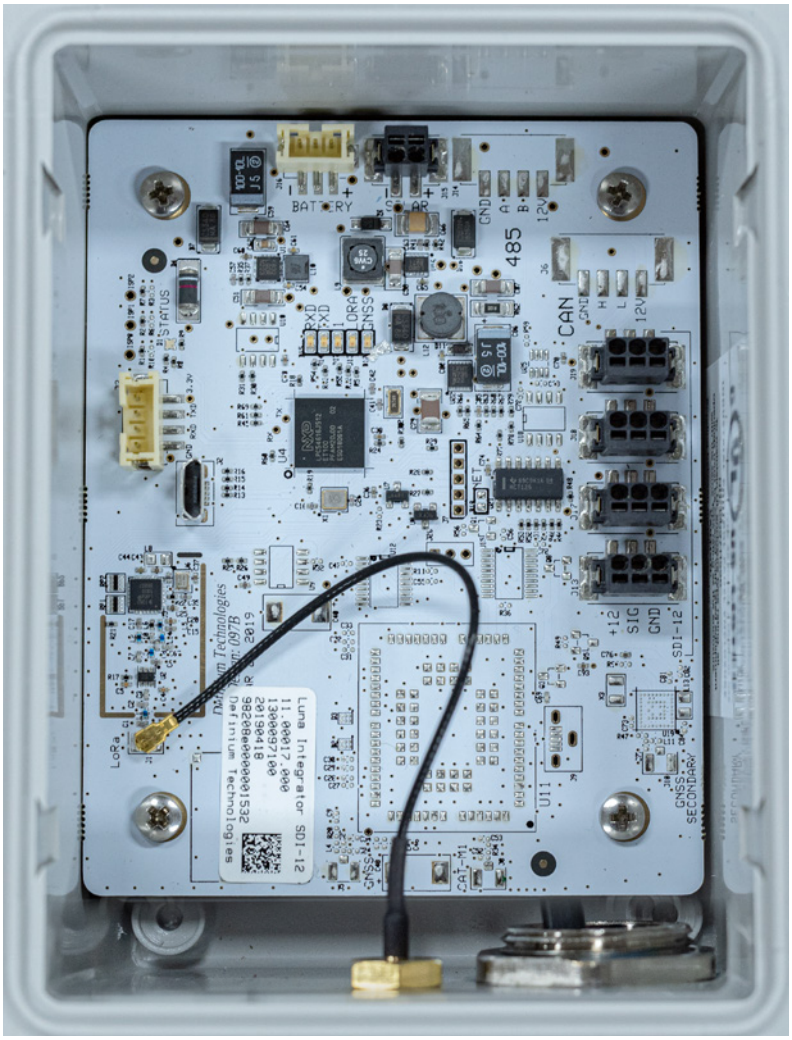
One terminal emulator we recommend is Putty, which can be downloaded from https://www.putty.org/.

All commands are entered as ASCII text and will return any response as ASCII text.

Connecting a S-Node to a computer via the Micro USB port will provide a serial port for configuration.

Recommended settings are as follows:

- Baud Rate: 115200 baud
- Bits: 8
- Parity: None
- Stop Bits: 1
- Flow Control: Disabled.



8.1 General Node Configuration Commands

These commands are entered into a terminal or terminal emulator such as Putty, https://www.putty.org/, to action several types of commands to the S-Node.

8.1.1 Firmware Version – version

Command Input: `version`

Compatible: Device Firmware Versions > 1.2

Command Description: Returns information about the device firmware version and configured frequency.

Command Type	Syntax Used	Response Type	Example Result of Command
Get	`version`	<String>	`Definium Technologies Pty Ltd LoRaWAN Class-A Sensor 235c0e89-dirty Luna Station 4000096001-"AS923"`

8.1.2 Save Current Device Configuration – save

Command Input: `save`

Compatible: All Device Firmware Versions

Command Description: Saves the running configuration to permanent storage.

Command Type	Syntax Used	Response Type	Example Result of Command
Action	`save`	Saved config	`Saved config`

8.1.3 Reload Saved Configuration – load

Command Input: `load`

Compatible: All Device Firmware Versions

Command Description: Saves the running configuration to permanent storage.

Command Type	Syntax Used	Response Type	Example Result of Command
Action	`load`	Loaded config	`Loaded config`

8.1.4 Firmware Update Mode – bootloader

Command Input: `bootloader`

Compatible: All Device Firmware Versions

Command Description: Puts the device into firmware update mode. MFR-Node Firmware can be downloaded from: http://ictinternational.com/support/software/

To firmware update the node:

Install Python (make sure to add to path when prompted) - https://www.python.org/downloads/

Then run the following commands in a cmd window:

```
python -m pip install -U pip
pip install pyserial
```

Connect to the node using a terminal emulator (e.g: putty - https://www.putty.org/)

Disable the node by typing in: `enable 0`
Run command: bootloader

The device will stop flashing LED and appear to disconnect via USB.

Open windows cmd, type in: `cd (directory where you saved the firmware)`
Then (in cmd): `python windows_loader.py fw-4000097003-<frequency>.bin`
It will take 1-2 minutes, then once the firmware flash is done the USB will reconnect.

Unplug the node for ~30 seconds, then you can reconnect and reprogram the node.

Command Type	Syntax Used	Response Type
Action	`bootloader`	Node disconnects from serial interface

8.1.5 Set low battery mode threshold – battery threshold

Command Input: `battery threshold`

Compatible: All Device Firmware Versions

Command Description: Sets the battery threshold, below which the device will enter low power mode and cease regular transmission until the battery has charged above the threshold. By default, this is set to 3.4 volts or 3400 milivolts.

Command Type	Syntax Used	Response Type	Example Result of Command
Get	`battery threshold`	<voltage>	`battery threshold` `3400`
Set	`battery threshold <voltage>`	<voltage>	`battery threshold 3400` `3400`

Parameter	Type	Description
<voltage>	Number	Low power cutoff in mV.

8.1.6 Command List – help

Command Input: `help`

Compatible: All Device Firmware Versions

Command Description: Lists all available commands with brief descriptions of their functions.

Command Type	Syntax Used	Response Type
Get	`help`	List of commands

8.1.7 Reset to Factory Defaults – config reset

Command Input: `config reset`

Compatible: Device Firmware Versions > 1.2

Command Description: Resets the running configuration factory defaults.

Command Type	Syntax Used	Response Type	Example Result of Command
Get	`config reset`	Reset app config to defaults	`Reset app config to defaults`

8.1.8 Logging interval / Period between reports – report period

Command Input: `report period`

Compatible: Device Firmware Versions > 1.2

Command Description: Command for managing the device's state transition timings. Initial wait time on failed communications before retrying. Doubles each failure until it reaches backoff max.

Command Type	Syntax Used	Example Command	Response Type	Example Result of Command
Get	`report period`	`report period`	<string>	`Report Period: 900 sec Current: 12 Last: 0 Next:0`
Set	`report period <period>`	`report period 600`	<string>	`report period 600 Report Period: 600 sec Current: 26 Last: 0 Next:0`

Parameter	Type	Description
<period>	Number	Time in seconds between reports.
<time>	Number	Current device timestamp
<last>	Number	Timestamp that state last triggered at
<next>	Number	Timestamp of next state trigger

8.2 General LoRaWAN™ Configuration

These commands are entered into a terminal or terminal emulator such as Putty, https://www.putty.org/, to action several types of commands to the S-Node.

8.2.1 LoRaWAN EUIs – lora eui

Command Input: `lora eui`

Compatible: All Device Firmware Versions

Command Description: Manage the device's LoRaWAN EUIs (unique identifier), both device and application/join (dependent on LoRaWAN version).

Command Type	Syntax Used	Example Command	Response Type	Example Result of Command

Get	`lora eui <type>`	`lora eui app`
		`1122334455667788`
			<eui>	`lora eui app`
				`0x11 0x22 0x33 0x44`
				`0x55 0x66 0x77 0x88`

Set	`lora eui <type>`
	`<eui>`
		`lora eui dev`
		`1122334455667788`
			<eui>	`lora eui app`
				`0x11 0x22 0x33 0x44`
				`0x55 0x66 0x77 0x88`
				`0x11 0x22 0x33 0x44`
				`0x55 0x66 0x77 0x88`

Parameter	Type	Description
<type>	String	Type of EUI, options are: `dev`: Device EUI `app`: Application/Join EUI
<eui>	String	16 character Hexadecimal string representing an 8-byte EUI. Example: `0x11 0x22 0x33 0x44 0x55 0x66 0x77 0x88`

8.2.2 LoRaWAN Application/Join Key – lora key app

Command Input: `lora key app`

Compatible: All Device Firmware Versions

Description: Manage the device's LoRaWAN application/join key used for Over-the-Air Activation (OTAA).

Command Type	Syntax Used	Example Command	Response Type	Example Result of Command
Get	`lora key app`	`lora eui app 1122334455667788`	<key>	`lora key app 00112233445566778899AABBCCDDEEFF`
Set	`lora key app <key>`	`lora key app`	<key>	`lora eui app 1122334455667788AABBCCDDEEFFGGHH 1122334455667788AABBCCDDEEFFGGHH`

Parameter	Type	Description
<key>	String	32 character Hexadecimal string representing an 16-byte EUI. Example: `0x11 0x22 0x33 0x44 0x55 0x66 0x77 0x88 0xAA 0xBB 0xCC 0xDD 0xEE 0xFF 0xGG 0xHH`

8.2.3 LoRaWAN Network Joined Status – lora joined

Command Input: `lora joined`

Compatible: All Device Firmware Versions

Description: Manage the device's LoRaWAN application/join key used for Over-the-Air Activation (OTAA).

Command Type	Syntax Used	Example Command	Response Type	Example Result of Command
Get	`lora joined`	`lora joined`	<status>	`enabled`
Set	`lora joined <bool>`	`lora joined 0`	<status>	`lora joined 0 disabled`

Parameter	Type	Description
<bool>	Number	Network joined status to be set: `0`: Unjoined `1`: Joined
<status>	String	Network joined status string. Options: `enabled`: Joined `disabled`: Unjoined / Not Joined

8.2.4 LoRaWAN Public Network Mode – lora net public

Command Input: `lora net public`

Compatible: All Device Firmware Versions

Description: Enable/Disable public network mode. Change the sync word used by the LoRa radio between the public and private settings.

Command Type	Syntax Used	Example Command	Response Type	Example Result of Command
Get	`lora net public`	`lora net public`	<status>	`lora net public enabled`
Set	`lora net public <bool>`	`lora net public 0`	<status>	`lora net public 0 disabled`

Parameter	Type	Description
<bool>	Number	Network joined status to be set: `0`: Disable `1`: Enabled

<status>	String	Public network status string. Options:
		`enabled`
		`disabled`

8.2.5 LoRaWAN Adaptive Data Rate – lora net adr

Command Input: `lora net adr`

Compatible: All Device Firmware Versions

Description: Devices using adaptive data rate will query the server periodically to determine the highest data rate they can transmit at and still be heard. The LoRaWAN specification states that stationary devices should use ADR, however use on mobile devise is discouraged as it is unlikely to work correctly.

Command Type	Syntax Used	Example Command	Response Type	Example Result of Command
Get	`lora net adr`	`lora net adr`	<status>	`disabled`
Set	`lora net adr <bool>`	`lora net adr 1`	<status>	`lora net adr 1 disabled`

Parameter	Type	Description
<bool>	Number	Network joined status to be set: `0`: Disable `1`: Enabled

<status>	String	Public network status string. Options:
		`enabled`
		`disabled`

8.2.6 ADR Link Check Period– linkcheck period

Command Input: `linkcheck period`

Compatible: Device Firmware Versions > 1.2

Description: Determines the number of unconfirmed packets between link checks when using ADR. Typically, this is set automatically and should not need to be changed.

Command Type	Syntax Used	Example Command	Response Type	Example Result of Command
Get	`linkcheck period`	`linkcheck period`	<period>	`64`
Set	`linkcheck period <period>`	`linkcheck period 0`	<period>	`linkcheck period 200 200`

Parameter	Type	Description
<period>	Number	Number of unconfirmed packets between link checks.

8.2.7 LoRaWAN Default Data Rate – lora data rate

Command Input: `lora data rate`

Compatible: All Device Firmware Versions

Description: Manage the device's default data rate. Used when ADR is not enabled.

Command Type	Syntax Used	Example Command	Response Type	Example Result of Command
Get	`lora data rate`	`lora data rate`	<data rate>	`2`
Set	`lora data rate <data rate>`	`lora data rate 0`	<data rate>	`lora data rate 1 1`

Parameter	Type	Description
<data rate>	Number	LoRaWAN Data Rate, where 0 is minimum (LoRa spreading factor 12). Maximum value is region specific.

8.2.8 LoRaWAN Maximum TX Power – lora tx power

Command Input: `lora tx power`

Compatible: All Device Firmware Versions

Description: Manage the device's default maximum transmit power. This is relative to the maximum EIRP, which can be changed by the network server after the device is joined to the network.

The actual maximum transmit power will be the maximum EIRP minus 2 times this setting, in dBm, unless that value is above 20 dBm, as that is the maximum power of the device's radio.

Command Type	Syntax Used	Example Command	Response Type	Example Result of Command
Get	`lora tx power`	`lora tx power`	<power>	`0`
Set	`lora tx power <power>`	`lora tx power 2`	<power>	`lora tx power 2 2`

Parameter	Type	Description
<power>	Number	Maximum transmit power, power below MAX_EIRP. TX_PWR = MAX_EIRP – 2 x power (dBm)

8.2.9 Confirmed Messaging – lora confirmed

Command Input: `lora confirmed`

Compatible: All Device Firmware Versions

Description: Enables/Disables LoRaWAN confirmed messaging.

Command Type	Syntax Used	Example Command	Response Type	Example Result of Command
Get	`lora confirmed`	`lora confirmed`	<string>	`disabled`
Set	`lora confirmed <bool>`	`lora confirmed 0`	<status>	`lora net adr 0 disabled`

Parameter	Type	Description
<bool>	Number	Indicates whether to enable or disable confirmed messaging: `0`: Disable `1`: Enabled

8.2.10 LoRa Band

Command Input: `lora band`

Compatible: Device Firmware Versions

Description: Sets the LoRaWAN sub-band and default channel mask. Note that band should be set to 2 for AU915 and US915. Band selection is not applicable to AS923.

Command Type	Syntax Used	Example Command	Response Type	Example Result of Command
Get	`lora band`	`lora band`	<string>	`lora band` `Band set to: 1`
Set	`lora band <band>`	`lora band 2`	<string>	`lora band 2` `Band set to: 2`

Parameter	Type	Description
<band>	Number	LoRaWAN Sub-band selection. Frequency plan dependant.

8.3 LoRaWAN Activation-by-Personalisation Mode

These commands are entered into a terminal or terminal emulator such as Putty, https://www.putty.org/, to action several types of commands to the S-Node. In Activation-by-Personalisation mode the device is pre-supplied with all keys and identifiers such that it does not need to undergo an over-the-air join process.

All settings in this section are required to be manually set when using Activation-by-Personalisation (ABP) and will be retrieved during the join process for OTAA devices. Do not change these if using OTAA on the device.

8.3.1 LoRaWAN Activation-by-personalisation – lora net abp

Command Input: `lora net abp`

Compatible: All Device Firmware Versions

Command Description: Enable/Disable Activation-by-Personalisation Mode.

Command Type	Syntax Used	Example Command	Response Type	Example Result of Command
Get	`lora net abp`	`lora net abp`	<status>	`disabled`
Set	`lora net abp`	`lora net abp 1`	<status>	`lora net abp 1 enabled`

Parameter	Type	Description
<bool>	Number	Indicates whether to enable or disable abp mode: `0`: Disable / `1`: Enabled
<status>	String	ABP mode status string. Options: `enabled` / `disabled`

8.3.2 LoRaWAN Session Keys – lora key session

Command Input: `lora key session`

Compatible: All Device Firmware Versions

Command Description: Manage the device's application and network session keys.

Type	Syntax Used	Response Type	Example Result of Command
Get	`lora key session <type>`	<key>	`1122334455667788AABBCCDDEEFFGGHH`
Set	`lora key session <type> <key>`	<key>	`lora key session app 1122334455667788AABBCCDDEEFFGGHH` `1122334455667788AABBCCDDEEFFGGHH`

Parameter	Type	Description
<key>	String	32 character Hexadecimal string representing an 16-byte EUI. Example: `0x11 0x22 0x33 0x44 0x55 0x66 0x77 0x88 0xAA 0xBB 0xCC 0xDD 0xEE 0xFF 0xGG 0xHH`
<type>	String	Key Types include: `app` (Application Session Key) `nwk` (Network Session Key)

8.3.3 LoRaWAN Device Address – lora net dev addr

Command Input: `lora net dev addr`

Compatible: Device Firmware Versions

Command Description: Get/Set the device's LoRaWAN network address.

Command Type	Syntax Used	Example Command	Response Type	Example Result of Command
Get	`lora net dev addr`	`lora net dev addr`	<eui>	`11223344`
Set	`lora net dev addr <eui>`	`lora net dev addr`	<eui>	`lora net dev addr 11223344 11223344`

Parameter	Type	Description
<eui>	String	8 character Hexadecimal string representing an 4-byte EUI. Examples: `0x11 0x22 0x33 0x44` `11223344`

8.3.4 LoRaWAN Network Indentifier – lora net id

Command Input: `lora net id`

Compatible: Device Firmware Versions

Command Description: Get/Set the network identifier of the connected LoRaWAN network.

Command Type	Syntax Used	Example Command	Response Type	Example Result of Command
Get	`lora net id`	`lora net dev addr`	<eui>	`11223344`
Set	`lora net id <eui>`	`lora net dev addr`	<eui>	`lora net id 11223344 11223344`

Parameter	Type	Description
<eui>	String	8 character Hexadecimal string representing an 4-byte EUI. Examples: `0x11 0x22 0x33 0x44` `11223344`

8.4 LoRaWAN Timing

Commands for managing the device's state transition timings.

8.4.1 Time to retransmit – backoff initial

Command Input: `backoff initial`

Compatible: Device Firmware Versions > 1.2

Command Description: Initial wait time on failed communications before retrying. Doubles each failure until it reaches backoff max.

Command Type	Syntax Used	Example Command	Response Type	Example Result of Command
Get	`backoff initial`	`backoff initial`	<string>	`Initial Backoff Time: 60 sec`
Set	`backoff initial <time>`	`backoff initial 90`	<string>	`backoff initial 90 Initial Backoff Time: 90 sec`

Parameter	Type	Description
<time>	Number	Wait time in seconds.

8.4.2 Maximum retransmit time after failed communication – backoff max

Command Input: `backoff max`

Compatible: Device Firmware Versions > 1.2

Command Description: Maximum wait time on failed communications before retrying.

Command Type	Syntax Used	Example Command	Response Type	Example Result of Command
Get	`backoff max`	`backoff initial`	<string>	`Maximum Backoff Time: 86400 sec`
Set	`backoff max <time>`	`backoff initial 90`	<string>	`backoff initial 90000 Initial Backoff Time: 90000 sec`

Parameter	Type	Description
<time>	Number	Wait time in seconds.

8.5 SDI-12

8.5.1 SDI-12 add command – sdi12 add

Command Input: `sdi12 add`

Compatible: Device Firmware Versions

Command Description: Add or modify SDI-12 command in slot. Available SDI-12 commands are: Measurement (M!), Concurrent (C!) and Result (R!).

Measurement and Concurrent must use the Measure type command. Result must use the Data command

Older Firmware Examples:

```
sdi12 add 0 M 0C0! 3 3 0D0! 111
```

Use SDI-12 command id/slot 0 to send a Concurrent measurement command to SDI-12 sensor address 0, delay 3 seconds, value length 3, send a data command to sensor address 0, get 3 parameters. Older firmware models explicit delay and value lengths.

Newest Firmware Examples:

```
sdi12 add 0 M 0C0! 0D0! 111
```

Use SDI-12 command id/slot 0 to send a Concurrent measurement command to SDI-12 sensor address 0.

The sensor will return a response in the form of `atttnn` where a = the sensor address, ttt = the specified time in seconds until the sensor will have the measurements ready, and nn = the number of measurement values.

After the specified wait time, the node will send a data command `0D0!` to sensor address 0, it will return all available results but the `111` sensor measurement masking will only prepare and transmit the first three parameters. Newest firmware models don't explicit delay and value lengths.

Command Type	Syntax Used	Response Type
Measure/Data	`sdi12 add <id> M <measure command> <data command> <mask>`	<string>

Parameter	Type	Description
<id>	Number	0-9 Slot ID to add or modify.
<measure command>	String	SDI-12 measure command to execute on specified address.
<data command>	String	SDI-12 command to return data on specified address.
<mask>	Binary	Sensor measurement masking. Length of the mask can be equal to the number of readings. E.g: the mask to select the first and fourth reading for an 8 reading data command is: 10010000, where 1 is on and 0 is off.

8.5.2 Send SDI-12 Command – sdi12 send

Command Input: `sdi12 send`

Compatible: All Device Firmware Versions
Command Description: Send an SDI-12 command. Can be used for identifying sensors on a bus, configuring SDI-12 addresses, or any other sensor specific functions.

Type	Syntax Used	Example Command	Response Type	Example Result of Command
Action	`sdi12 send <command>`	`sdi12 send ?!`	<response>	`1`

Parameter	Type	Description
<command>	String	SDI-12 command to execute.
<response>	String	Response from SDI-12 command, command dependant

Commands	Description
`aI!`	Sends sensor identification request for the sensor at address `a`
`aAb!`	Change sensor address from `a` to `b`
`?!`	Query sensor address, can only be done when a single sensor is connected

8.5.3 Delete all SDI-12 Commands – sdi12 remove all commands

Command Input: `sdi12 remove all commands`

Compatible: All Device Firmware Versions. Immediately after using this command, you will need to enter the save command.
Command Description: Removes all configured SDI-12 command slots.

Type	Syntax Used	Response Type	Example
Action	`sdi12 remove all commands`	<string>	`sdi12 remove all commands` `SDI12 Commands Erased`

Enabling better global research outcomes in soil, plant & environmental monitoring.

www.ictinternational.com sales@ictinternational.com.au
Ph: +61 2 6772 6770 | 211 Mann St., Armidale, NSW 2350, Australia