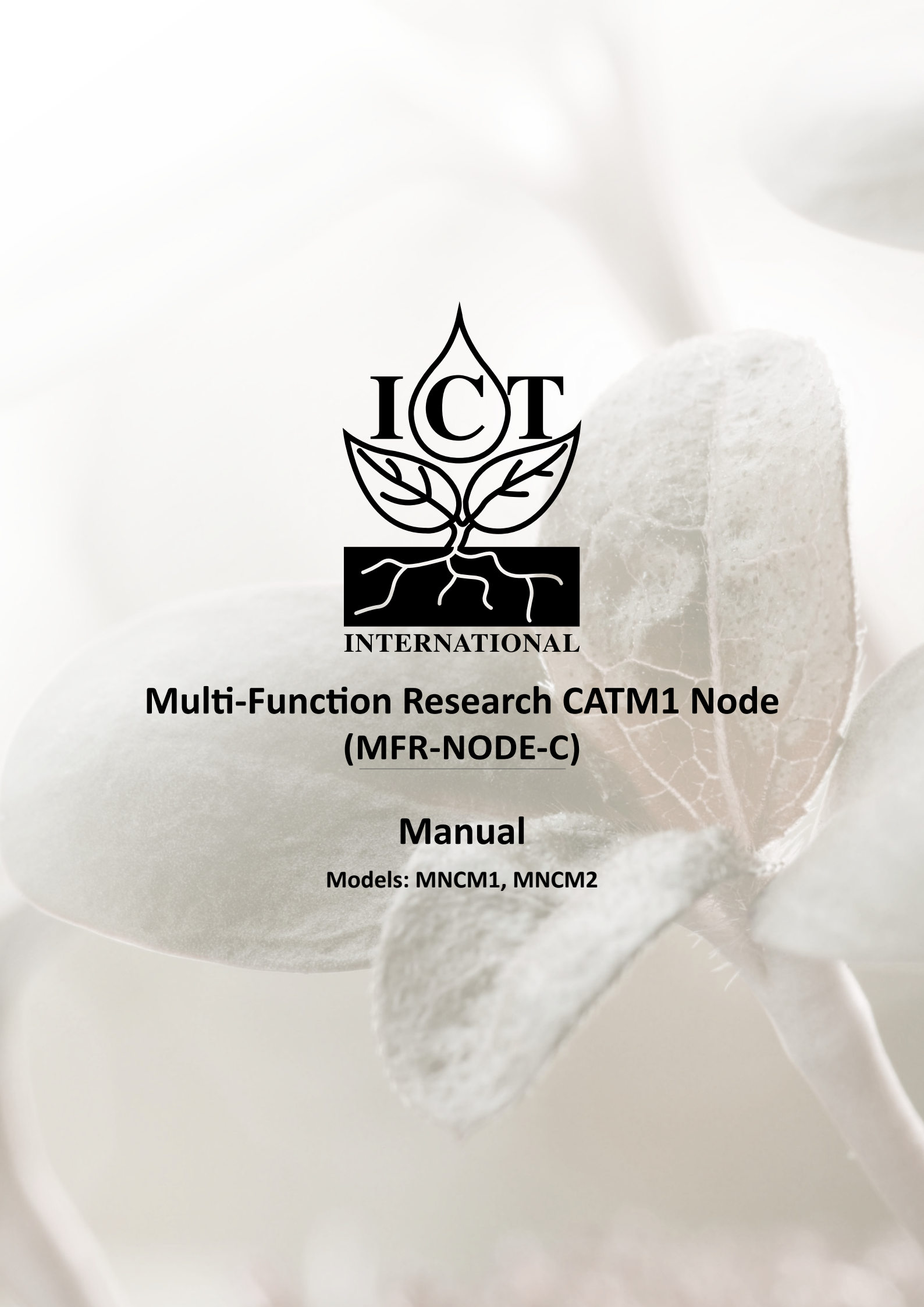


Multi-Function Research CATM1 Node (MFR-NODE-C)

Manual

Models: MNCM1, MNCM2

Contents

1. Introduction

The ICT International MFR-NODE-C is a CAT-M1/NB-IoT data storage and transmission device. SD Card Data is stored in CSV format. 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. Available sensor inputs are:

- SDI-12: Default 2x, up to 6 sensors on request.
- Analogue (4 single-ended or 2 differential, 3V, 5V or 12V selectable excitation),
- and 4 digital pulse inputs.

MFR-NODE-C Version MNCM1 (Legacy Model) includes: Direct Battery connection and Solar connection.

MFR-NODE-C Version MNCM2 (Improves on MNCM1) with the inclusion of:

- Extra Power and Grounds connections for all sensor/bus inputs for easier wiring.
- Adds Green Connector for isolation of Solar Panel/DC Input, which in turn allows larger range of wiring sizes.
- Adds Power Switch to isolate the Battery, allowing the Battery cable to stay connected for shipping/integration.

Figure 1. The MFR-NODE-C Box

Figure 2. Inside The MFR-NODE Box - Legacy Version MNCM1

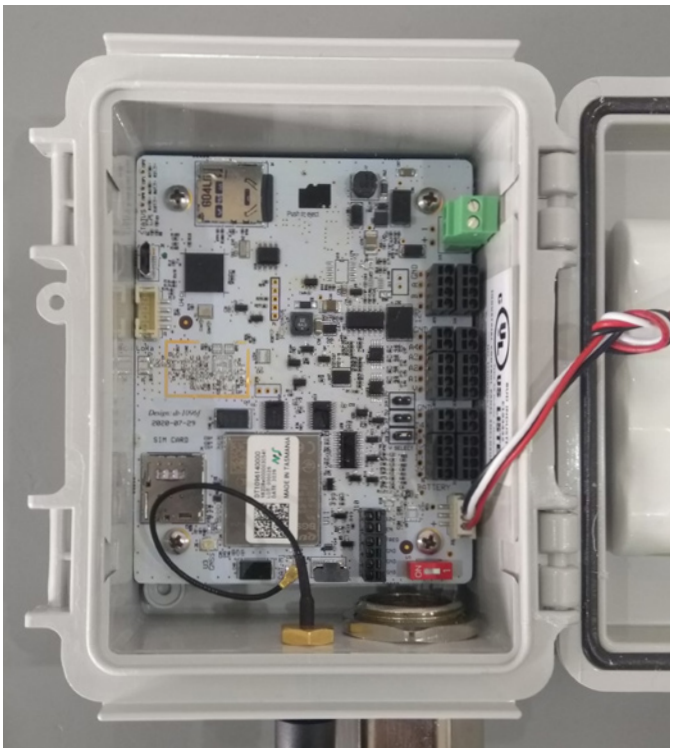


Figure 3. Inside The MFR-NODE Box - Version MNCM2

2. Connecting Sensors

2.1 Sensor Connector Locations on the MFR-Node board

A connector is available on the right-hand side of the board for each type of sensor. SDI-12 is below the solar input. Analog is below SDI-12 (A1 to A4). Digital is D1 to D4 .

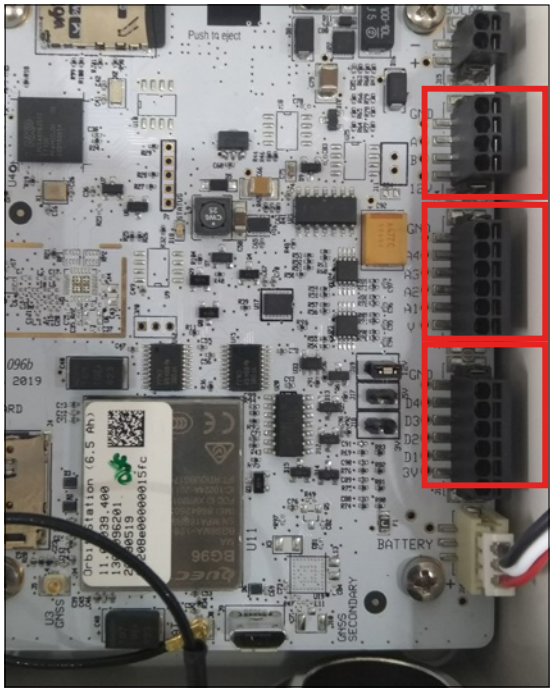


Figure 4. MFR-NODE-C Hardware Version MNCM1

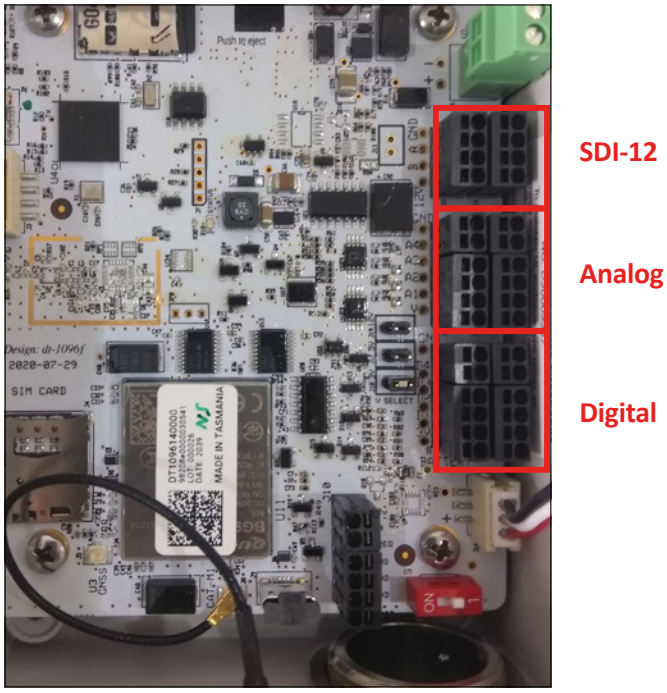


Figure 5. MFR-NODE-C Hardware Version MNCM2

2.2 SDI-12

To connect an SDI-12 sensor, insert the Ground Wire of the sensor into the connector labelled GND. Insert the Data line into the connector labelled B. Insert the Power wire into the connector labelled 12V. Hardware variant 2 physically supports the connection of 2x SDI-12 sensors. For both hardware variants 1 & 2 additional SDI-12 sensors can be bussed off the board.

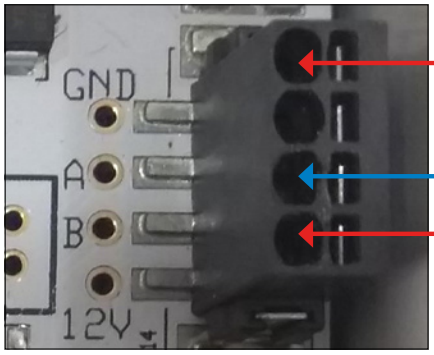


Figure 6. MFR-NODE-C Version MNCM1

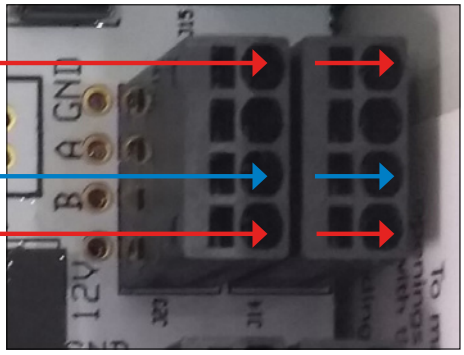


Figure 7. MFR-NODE-C VERSION MNCM2

2.3 Analog Sensors

2.3.1 Analog Excitation Voltage Selection

To connect analog sensors, first ensure that the excitation voltage is set correctly. Available voltages are 12V (top), 5V (middle) and 3V (bottom). Put the jumper on the pins for the excitation required (shown below on 5V). Wire sensors according to the sensor manual. The V inputs supply the selected Excitation to the sensor; GND is ground. A1 to A4 are the analog channels. If using differential sensors, use A1 and A2 or A3 and A4.

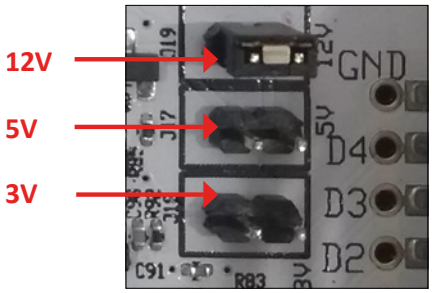


Figure 8. Voltage Selection

2.3.2 Wiring Analog Sensors

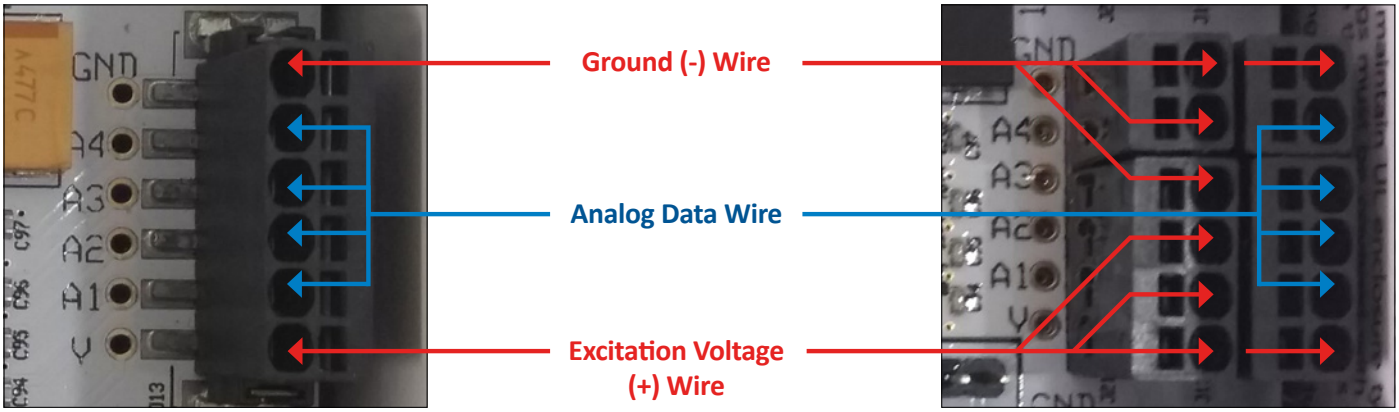


Figure 9. MNCM1 Analog Wiring

Figure 10. MNCM2 Analog Wiring

2.4 Digital (Pulse) Sensors

Wire the sensor according to the sensor manual. Most Digital Pulse sensors (rain gauges, anemometers, etc.) are passive and should be wired between an input (D1 to D4) and ground. If excitation is required, use the 3V pin.

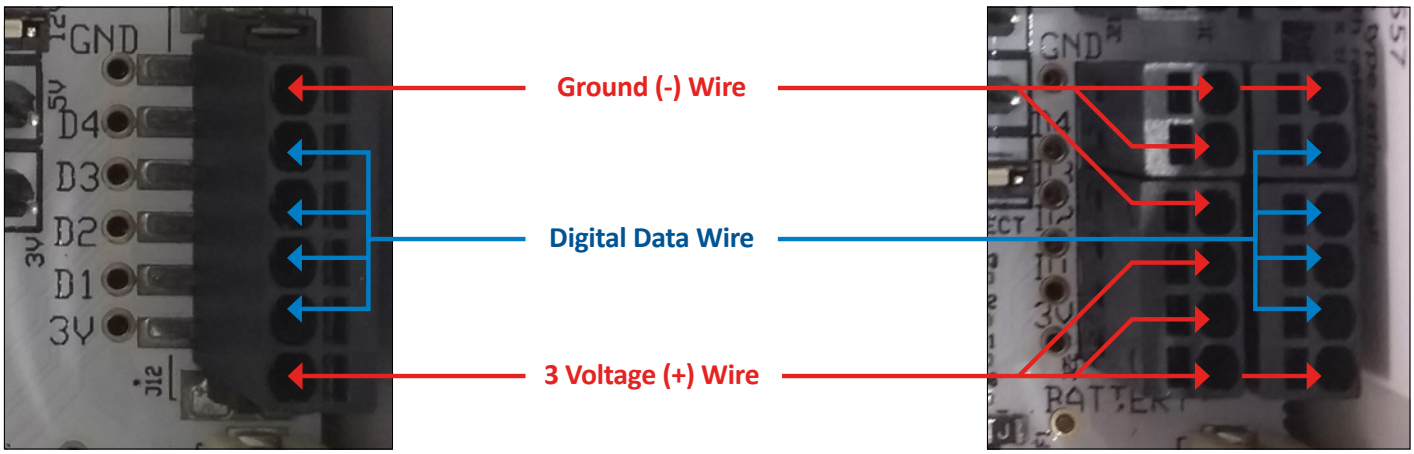


Figure 11. MNCM1 Digital Wiring

Figure 12. MNCM2 Digital Wiring

3. Connect The Internal Battery

When the node is mounted and ready, it is important to ensure that the battery is connected to the board first. Next the power switch can be turned on. After the node is booted for the first time, then external power can be connected - such as a 12V or 24V solar panel.

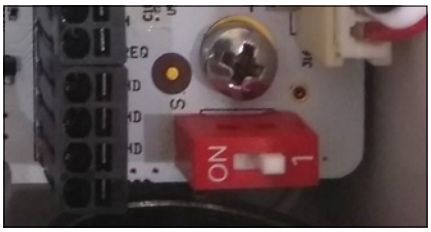


Figure 13. Battery and Power Switch (shown turned off)

4. Connect External DC Power

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

This input is polarised, please ensure that positive is inserted in the + terminal and negative in the -.

The MFR 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 indoors use.

Please ensure that the MFR Node's internal battery is plugged in before (and when) using solar power.

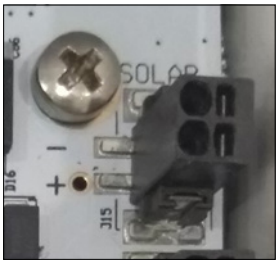


Figure 14. MNCM1

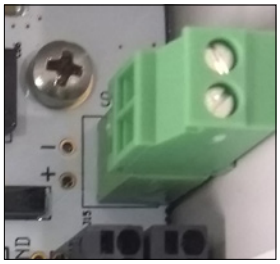


Figure 15. MNCM2

5. LED Status

The main status LED is used to indicate the following:

- **LIGHT BLUE**: Registering to the cellular network
- **DARK BLUE**: Collecting sensor data (should usually be too quick to notice)
- **YELLOW/ORANGE**: Transmitting data
- **PURPLE/PINK**: Diagnostic during start-up or sleep; error if permanently on
- **GREEN**: USB Idle or Node not enabled/configured
- **WHITE**: Not used in most cases

Figure 16. Main Status LED

The modem status LED has different flash rates, used to indicate the following:

- **FLICKER SLOWLY**: (200ms High/1800ms Low) Network Searching
- **FLICKER SLOWLY**: (1800ms High/200ms Low) Idle
- **FLICKER QUICKLY**: (125ms High/125ms Low) Data transfer is ongoing
- **ALWAYS ON**: Voice Calling (Issues/Unused)

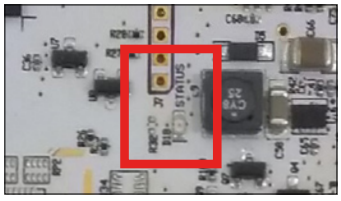
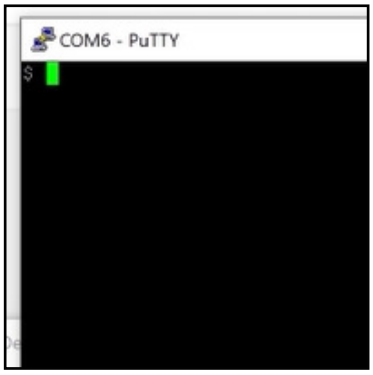


Figure 17. Modem Status LED

6. Device Operation

6.1 SDI-12 Configuration



Configuration Program Example

The MFR Node's SDI-12 communications can be customised to allow for interfacing with any sensor implementing the SDI-12 protocol. See section 7.5 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 can be sent directly to the sensor using the sdi12 send command.

For more information on SDI-12 related commands, see section 7.5.

6.3 Analog Input Configuration

The MFR Node's analog inputs can be configured as 4 single ended, 2 differential, or a combination. This is done using the adc ch config command. A differential channel is a pair of single ended channels, as such, only channel 1 and 3 can be configured as differential.

For more information on analogue configuration, see section 7.6.

6.4 SD Card Logging

SD card logging is enabled by sd enable. The MicroSD card will be formatted by the MFR Node. Data will be logged at the report interval in standard CSV format. The data is timestamped by the MFR-Node's RTC, which is synced daily to the cellular network time.

For more information on SD Card Logging, see section 7.8.

7. Device Configuration

The ICT International MFR Node is configured (i.e. identified and authenticated) over USB serial console using a terminal/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 an MFR Node to a computer via the Micro USB port (top right, fig. 1) 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.

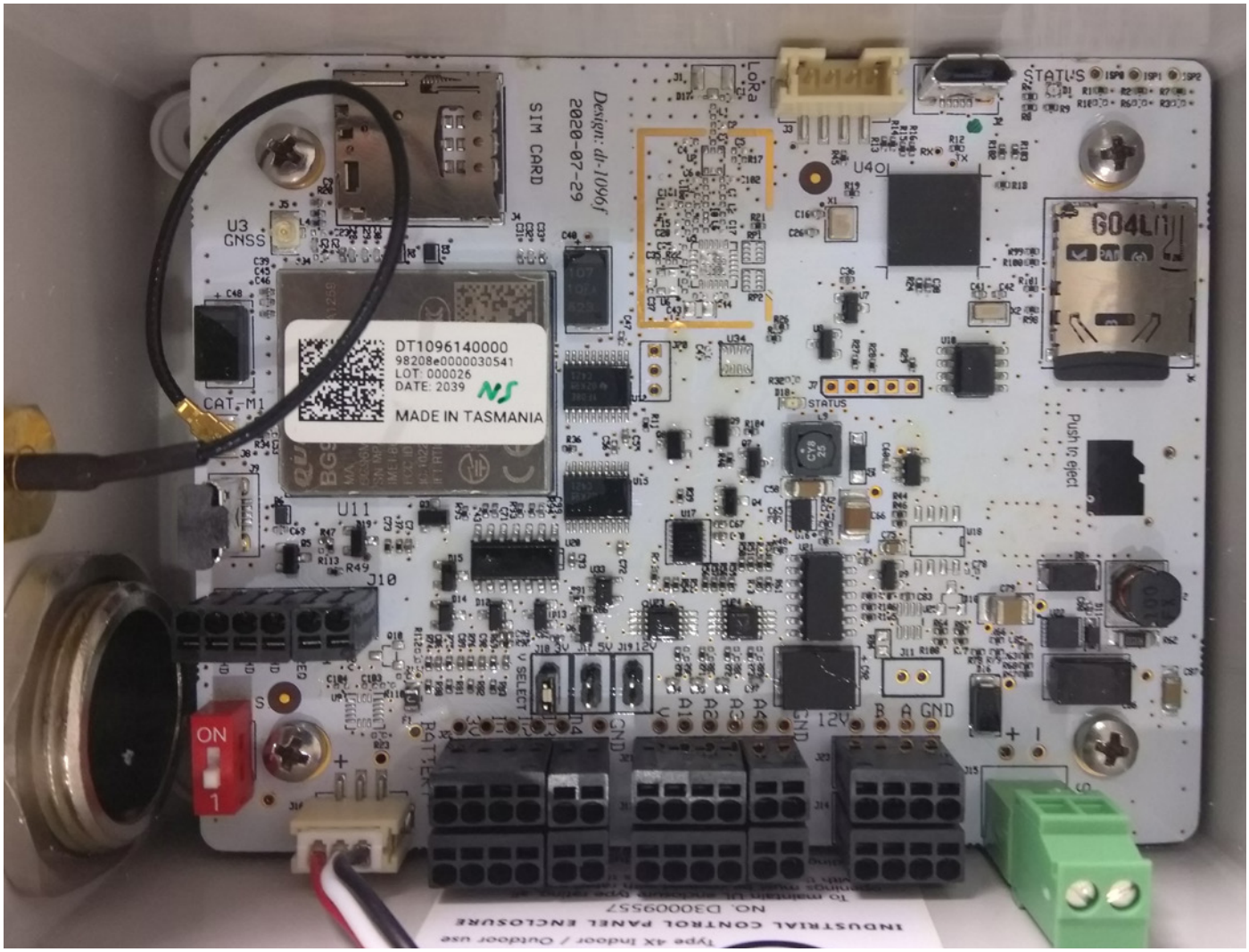


Figure 17. Version MNCM2

7.1 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 MFR-Node.

7.1.1 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"`

7.1.2 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`

7.1.3 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`

7.1.4 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

7.1.5 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.

7.1.6 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

7.1.7 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`

7.1.8 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

7.2 LtE™ Communication Configurations - Modem Settings

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 MFR-Node.

7.2.1 Modem Access Point (APN) – modem apn

Command Input: `modem apn`

Command Description: Sets the modem access point (APN) specific to the SIM card provider.

Command Type	Syntax Used	Example Command	Response Type	Example Result of Command
Get	`modem apn`	`modem apn`	<string>	`modem apn telstra.m2m`
Set	`modem apn <string>`	`modem apn telstra.m2m`	<string>	`modem apn telstra.m2m`

7.2.2 APN Username – modem usr

Command Input: `modem usr`
Command Description: Set the Modem username if required by the SIM card provider you are using.

Command Type	Syntax Used	Example Command	Response Type	Example Result of Command
Get	`modem usr`	`modem usr`	<string>	`Modem username:`
Set	`modem usr <string>`	`modem usr yourMSID@uscc.net`	<string>	`Modem username: yourMSID@uscc.net`

7.2.3 APN Password – modem pwd

Command Input: `modem pwd`
Command Description: Set the Modem password if required by the SIM card provider you are using.

Command Type	Syntax Used	Example Command	Response Type	Example Result of Command
Get	`modem pwd`	`modem pwd`	<string>	`Modem password:;`
Set	`modem pwd <string>`	`modem pwd your MSID`	<string>	`Modem password: your MSID`

7.2.4 APN Authentication – modem auth

Command Input: `modem auth`
Command Description: Enable Password Authentication Protocol (PAP) and Challenge Handshake Authentication Protocol (CHAP).

Command Type	Syntax Used	Example Command	Response Type	Example Result of Command
Get	`modem auth`	`modem auth`	<status>	`Modem auth method: 0`
Set	`modem auth <bool>`	`modem auth 1`	<status>	`Modem auth method: 1`

Parameter	Type	Description
<bool>	Number	Password Authentication Protocol (PAP) and/or Challenge Handshake Authentication Protocol (CHAP) status to be set: `0`: none `1`: pap `2`: chap `3`: pap or chap

7.3 MQTT Communication Configurations - Settings

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 MFR-Node.

7.3.1 MQTT Hostname – mqtt host

Command Input: `mqtt host`
Command Description: Enter in the hostname of your MQTT broker.

Command Type	Syntax Used	Example Command	Response Type	Example Result of Command
Get	`mqtt host`	`mqtt host`	<string>	`MQTT Hostname:`
Set	`mqtt host <string>`	`mqtt host ictcatm1.com`	<string>	`MQTT Hostname: ictcatm1.com`

7.3.2 MQTT Port – mqtt port

Command Input: `mqtt port`
Command Description: Set network port to connect to (optional, 0 = default based on TLS setting).

Command Type	Syntax Used	Example Command	Response Type	Example Result of Command
Get	`mqtt port`	`mqtt port`	<int>	`mqtt port 0`
Set	`mqtt port <int>`	`mqtt port 1883`	<int>	`mqtt port 1883`

7.3.3 MQTT Username – mqtt user

Command Input: `mqtt user`
Command Description: The username to use with your MQTT broker.

Command Type	Syntax Used	Example Command	Response Type	Example Result of Command
Get	`mqtt user`	`mqtt user`	<string>	`MQTT User:`
Set	`mqtt user <string>`	`mqtt user ictinternational`	<string>	`MQTT User: ictinternational`

7.3.4 MQTT Password – mqtt pass

Command Input: `mqtt pass`
Command Description: The corresponding password for the username to use with your MQTT broker.

Command Type	Syntax Used	Example Command	Response Type	Example Result of Command
Get	`mqtt pass`	`mqtt pass`	<string>	`MQTT Password:`
Set	`mqtt pass <string>`	`mqtt pass 1234`	<string>	`MQTT Password: 1234`

7.3.5 MQTT Topic – mqtt topic

Command Input: `mqtt topic`
Command Description: UTF-8 string that the broker uses to filter messages for each connected client. The topic consists of one or more topic levels. Each topic level is separated by a forward slash (topic level separator).

Command Type	Syntax Used	Example Command	Response Type	Example Result of Command
Get	`mqtt topic`	`mqtt topic`	<string>	`MQTT Topic:`
Set	`mqtt topic <string>`	`mqtt topic`	<string>	`MQTT Topic: ict/data/mfr/1K704`

7.3.6 MQTT QoS Level – mqtt qos

Command Input: `mqtt user`
Command Description: Quality of Service (QoS) level is an agreement between the sender of a message and the rceiver of a message that defines the guarantee of delivery for a specific message. There are 2 QoS levels: At most once (0); At least once (1).

Command Type	Syntax Used	Example Command	Response Type	Example Result of Command
Get	`mqtt qos`	`mqtt qos`	<bool>	`0`
Set	`mqtt qos <bool>`	`mqtt qos 1`	<bool>	`1`

7.3.7 Use MQTT over TLS (MQTTS) – mqtt tls

Command Input: `mqtt tls`
Command Description: Use MQTT over TLS (MQTTS): disabled (0); enabled (1).

Command Type	Syntax Used	Example Command	Response Type	Example Result of Command
Get	`mqtt tls`	`mqtt tls`	<bool>	`disabled`
Set	`mqtt tls <bool>`	`mqtt tls 1`	<bool>	`enabled`

7.3.8 Report in JSON rather than CSV – mqtt json

Command Input: `mqtt json`
Command Description: Send data in json format rather than csv format (default): disabled (0); enabled (1).

Command Type	Syntax Used	Example Command	Response Type	Example Result of Command
Get	`mqtt json`	`mqtt json`	<bool>	`disabled`
Set	`mqtt json <bool>`	`mqtt json 1`	<bool>	`enabled`

7.3.9 MQTT Connection is for Azure IoT Hub– mqtt azure

Command Input: `mqtt azure`
Command Description: Azure Mode settings reporting data in .json, and will expect a Azure string as per section 7.3.10.

Command Type	Syntax Used	Example Command	Response Type	Example Result of Command
Get	`mqtt azure`	`mqtt azure`	<status>	`disabled`
Set	`mqtt azure <bool>`	`mqtt azure`	<status>	`mqtt azure 1 enabled`

Parameter	Type	Description
<bool>	Number	mqtt azure to be set: `0`: Unjoined `1`: Joined
<status>	String	mqtt azure status string. Options: `enabled`: mqtt azure enabled `disabled`: mqtt azure disabled

7.3.10 MQTT Azure Configuration String – mqtt azureconn

Command Input: `mqtt azureconn`
Command Description: To configure the node to communicate to the Azure Hub it must contain in a string organised below to include the following: HostName, Device ID and Shared Access Key, please note the Shared Access Key will end in an equals (=) sign:.

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

Get	`mqtt azureconn`	`mqtt azureconn`	<status>	`Azure IoT Hub Hostname:`
				`Azure IoT Hub Device ID:`
				`Azure IoT Hub Device Key:`

Set	`mqtt azureconn <string>`	`see parameter description below`	<status>	`Azure IoT Hub Hostname: ictcatm1.com`
				`Azure IoT Hub Device ID:`
				`Azure IoT Hub Device Key:`

Parameter	Type	Description
<status>	String	Example Layout of the Command String to enter will be: mqtt azureconn HostName=xxxxxxxx.azure-devices.net; DeviceId=yyyyyyyyyyyyy;SharedAccessKey=Base64EncodedAccessKeyGoesHere=

7.3.11 Enable MQTT Last Will Testament (LWT) – mqtt will enable

Command Input: `mqtt will enable`
Command Description: Enable a LWT message to be sent to other clients if device doesn't connect to broker.

Command Type	Syntax Used	Example Command	Response Type	Example Result of Command
Get	`mqtt will enable`	`mqtt will enable`	<status>	`disabled`
Set	`mqtt will enable <bool>`	`mqtt will enable 1`	<status>	`enabled`

Parameter	Type	Description
<bool>	Number	Enable a LWT message to be sent to other clients if device doesn't connect to broker: `0`: disable `1`: enable

7.3.12 Enable MQTT Last Will Testament (LWT) Topic – mqtt will topic

Command Input: `mqtt will topic`
Command Description: The LWT topic that the broker will publish to.

Command Type	Syntax Used	Example Command	Response Type	Example Result of Command
Get	`mqtt will topic`	`mqtt will topic`	<string>	`MQTT LWT Topic:`
Set	`mqtt will topic <string>`	`mqtt will topic device mlna1902/ status`	<string>	`MQTT LWT Topic: device mlna1902/status`

7.3.13 Enable MQTT Last Will Testament (LWT) Message– mqtt will msg

Command Input: `mqtt will msg`
Command Description: LWT Message posted to the designated LWT topic.

Command Type	Syntax Used	Example Command	Response Type	Example Result of Command
Get	`mqtt will msg`	`mqtt will msg`	<string>	`MQTT LWT Message:`
Set	`mqtt will msg <string>`	`mqtt will msg device mlna1902 offline`	<string>	`MQTT LWT Message: device mlna1902 offline`

7.5 SDI-12

7.5.1 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.

7.5.2 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>	`sdi12 send 0A1! 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

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

Command Input: `sdi12 remove all commands`

Compatible: All Device Firmware Versions
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`

7.6 Analog Commands

7.6.1 Analog Channel Configuration – adc ch config

Command Input: `adc ch config`

Compatible: Device Firmware Versions > 1.2

Command Description: Enables and sets the configuration of the analog channels.

Command Type	Syntax Used	Example Result of Command
Set	`adc ch config <mask>`	`adc ch config DOSS`

Parameter	Type	Description
<mask>	String	State of each analog channel: `S`: Single Ended, available for channels 1-4 `D`: Differential, available for channels 1 and 3, channels 2 and 4 must be set to off respectively `O`: Off – disables channel

7.6.2 Analog Single Ended Test – adc single test

Command Input: `adc single test`

Compatible: All Device Firmware Versions

Command Description: Display readings for the 4 single-ended analog channels, in µV.

Command Type	Syntax Used	Response Type	Example Result of Command
Action	`adc single test`	<string>	`adc single test` `ADC SING=160,129,175,121`

7.6.3 Analog Differential Test – adc diff test

Command Input: `adc diff test`

Compatible: All Device Firmware Versions

Command Description: Display readings for the 2 differential channels, in uV.

Command Type	Syntax Used	Response Type	Example Result of Command
Action	`adc diff test`	<string>	`adc diff test` `ADC DIFF=39,50`

7.6.4 Disable Constant Excitation – persistent pwr

Command Input: `persistent pwr`

Compatible: Device Firmware Versions > 1.2

Command Description: Enables/disables constant sensor excitation for both SDI-12 and analog. Please contact ICT International to confirm that this is suitable for your application.

Command Type	Syntax Used	Response Type	Example Result of Command
Get	`persistent power`	<string>	`persistent power enabled`
Set	`persistent power <enable>`	<string>	`persistent power 0 disabled`

Parameter	Type	Description
<enable>	Number	Enable (1)/Disable (0)

7.6.5 Calibrate Analog Channel – adc ch calibrate

Command Input: `adc ch calibrate`

Compatible: Device Firmware Versions > 1.2

Command Description: Used to precisely calibrate analog channels. This is generally not necessary – Contact ICT International to confirm that this is applicable to your application. Calibration requires a power source that can cover 1mV to 1V accurately.

Procedure: (Repeat on all channels necessary.)

Set the power supply to 1mV and measure the voltage into the connector. Enter the following command:
`adc ch calibrate <channel> 0 <voltage measured as µV>`

For example: `adc ch calibrate 1 0 1024`

Set the power supply to 1V and measure the voltage into the connector. Enter the following command:
`adc ch calibrate <channel> 1 <voltage measured as µV>`

For example: `adc ch calibrate 1 1 1001024`

7.6.6 Disable voltage divider in ADC calculation

Command Input: `adc div`

Compatible: Device Firmware Versions > 1.2

Command Description: Disables the resistor divider in the calculation and calibration of the ADC. Contact ICT International to see if this is applicable to your application. Requires hardware modification and can only be used with sensors with < 3V output.

Command Type	Syntax Used	Response Type	Example Result of Command
Get	`adc div`	<string>	`adc div` `enabled`
Set	`adc div <enable>`	<string>	`adc div 0` `disabled`

Parameter	Type	Description
<enable>	Number	Enable (1)/Disable (0)

7.7 Digital Input Commands

7.7.1 Enable Digital Inputs - counter enable

Command Input: `counter enable`

Compatible: All Device Firmware Versions

Command Description: Enable or disable logging and upload of digital inputs.

Command Type	Syntax Used	Response Type	Example Result of Command
Get	`counter enable`	<string>	`counter enable` `enabled`
Set	`counter enable <enable>`	<string>	`counter enable 1` `disabled`

Parameter	Type	Description
<enable>	Number	Enable (1)/Disable (0)

7.7.2 Test Digital Inputs – counter test

Command Input: `counter test`

Compatible: All Device Firmware Versions

Command Description: Displays current values of the digital inputs

7.8 MFR-Node SD Card Commands

7.8.1 Enable SD Card Logging – sd enable

Command Input: `sd enable`

Compatible: All Device Firmware Versions

Command Description: Enable or disable logging of data to the onboard MicroSD card. Note that the log file uses instrument RTC time unless an offset has been set. See section 10.8.3.

Command Type	Syntax Used	Response Type	Example Result of Command
Get	`sd enable`	<string>	`sd enable` `enabled`
Set	`sd enable <enable>`	<string>	`sd enable 1` `enabled`

Parameter	Type	Description
<enable>	Number	Enable (1)/Disable (0)

7.8.2 Test SD Card – sd test

Command Input: `sd test`

Compatible: All Device Firmware Versions

Command Description: Confirms that the device is able to save data to the MicroSD card.

Command Type	Syntax Used	Response Type	Example Result of Command
Action	`sd test`	<string>	`sd test` `SD CARD=WRITE=OKAY,READ=OKAY`

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