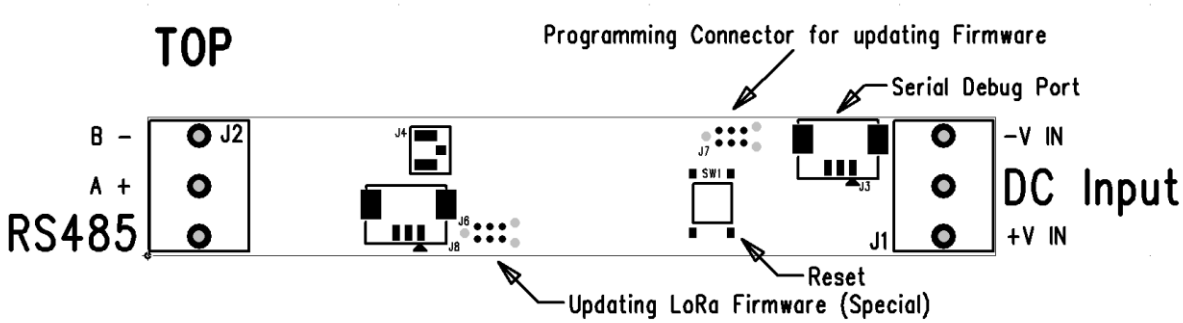
WISka Wire RS485 LoRa Sensor

Item Specifications

Go-IoT Item Id:	WISka Wire RS485 LoRa
Wireless Technology	LoRaWAN® 1.0.3
Wireless Security	LoRaWAN® End-to-End encryption (AES-CTR), Data Integrity Protection (AES-CMAC)
LoRaWAN Device Type	Class A,C
Supported LoRaWAN® features	OTAA, ABP, ADR, Adaptive Channel Setup
Supported LoRaWAN® regions	EU433, CN470, IN865, EU868, AU915, US915, KR920, AS923 (Other Regions are optional)
Link Budget	131 dB (SF5) to 151 dB (SF12)
RF Transmit Power	14 dB / 22 dB (Region specific)
Antenna	Internal
DC Supply	12V – 24V AC/DC
Blue/ Green LED Status	
Serial Interfaces	
RS485	300 -115200 Baud - 2 Wire Interface
RS232 (on request)	300 -115200 Baud - 2 Wire Interface
Modbus Energy Meters	Carlo Gavazzi , Eastron , Schneider , Hiking , Rayleigh , Others
Supported	Others available to request
Case Style Hex	90mm x 57mm x 16mm
Case Material	UL94-V0 polycarbonate
Case Mounting	DIN Rail or Wall
Case Color	Grey

2. Connections

The connections are marked on the case for clarity and are as follow:-



DC Input = 12V – 24V DC INPUT

Gavazzi Wiring EM540

WISka Wire J2 B connect to Gavazzi Pin A-
WISka Wire J2 A connect to Gavazzi Pin B+

3 LoRa Payload Decoder

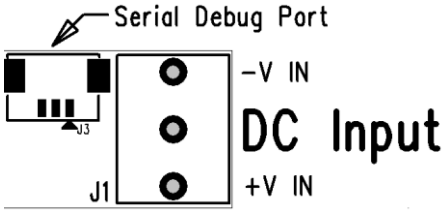
This will be available via an attachment on the email

```
//  __          _____ ____  _ __    _
//  \ \        / /_ _/ ___|| |/ /   / \
//   \ \ /\ / / | |\___ \| ' /   / _ \
//    \ V  V /  | | ___) | . \  / ___ \
//     \_/\_/  |___|____/|_|\_\/_/   \_\
//
//
//  WISKA simple payload decoder.
//  www.go-iot.io

//  August 31st 2023
//  V1.9
//  Added
//  var TYPE_MODBUS_HZ       = 0xEA;  //2 bytes value Hz
//  var TYPE_MODBUS_KWH      = 0xEB;  //4 bytes value kWh
//  var TYPE_MODBUS_KVARH    = 0xEC;  //4 bytes value kvarh
//


var TYPE_MODBUS_HZ       = 0xEA;  //2 bytes value Hz
var TYPE_MODBUS_KWH      = 0xEB;  //4 bytes value kWh
var TYPE_MODBUS_KVARH    = 0xEC;  //4 bytes value kvarh
var TYPE_MODEL           = 0xED;  //12 bytes - Sensor Model and Firmware
var TYPE_SENSORS         = 0xEE;  //20 bytes - Installed Sensors
var TYPE_DEBUG           = 0xEF;  //4  bytes debug EE



function bin16dec(bin) {
    var num=bin&0xFFFF;
    if (0x8000 & num)
        num = - (0x010000 - num);
    return num;
}
function bin8dec(bin) {
    var num1=bin&0xFF;
    if (0x80 & num1)
        num1 = - (0x0100 - num1);
    return num1;
}
function hexToBytes(hex) {
    for (var bytes = [], c = 0; c < hex.length; c += 2)
        bytes.push(parseInt(hex.substr(c, 2), 16));
    return bytes;
}
var decodeWiskaPayload=function(data,port){
    var obj = new Object();
    for(i=0;i<data.length;i++){
        //console.log(data[i]);
        switch(data[i]){

            case TYPE_MODBUS_HZ:                        //Modbus Hz
                obj.modbus_hz=(data[i+1]);
                i+=2;
                break
            case TYPE_MODBUS_KWH:                       //Modbus kWh
                obj.modbus_kwh=(data[i+1]);
                i+=4;
                break
            case TYPE_MODBUS_KVARH:                     //Modbus Kvarh
                obj.modbus_kvarh=(data[i+1]);
                i+=4;
                break
            case TYPE_MODEL:
                obj.model = [];
                for(var j = 0; j < 20; j++)  {
                    obj.model[j] = (data[j]);           //Model Type
                }
                i += 20;
                break
            case TYPE_SENSORS:
                obj.sensors = [];
                for(var j = 0; j < 20; j++)  {
                    obj.sensors[j] = (data[j]);         //Sensor Type
                }
                i += 20;
                break
            default:                                    //somthing is wrong with data
                i=data.length;
                break
        }
    }
    return obj;
}



exports.decode = function(bytes,port)
{
    return decodeWiskaPayload(bytes,port);
}
```

4 Programming the WISka Sensor with USB Cable

Note: That a special serial interface cable is required for the WISka Wire

1. Plug the Special Serial Cable (micro) into WISka Wire PCB and PC (Type A)
   Note: The cable only fits one way



2. Open you favorite Serial Terminal – Tera Term, Putty, YAT
3. Set the COM Port correctly and the baud rate to 115200, n, 8 , 1
4. Push reset button on WISka Wire
5. Observe the output as below.
6. There is a 10 second delay to press a key or the WISka will quit the menu and enter operational mode. If this happens just push the reset button again.

```
01 DevEui      :AC1F09FFFE0A1003
03.AppEui      :0000000000000001
04.AppKey          :3272357538782F413F4428472B4B6250
07.OTA         :1   ... OTA Mode
08.Port        :5   ... 2 - 9
09.ACK         :0   ... OFF
10.ADR         :1   ... ON
12.DR          :5   ... SF7
13.DR MAX      :5
14.DR MIN      :5
15.SubBand     :0
16.Region      :4   ... EU868
17.Class       :1   ... A
18.Lock        :0
20.Spl Period  :900
21.Send Period  :1
52.MODbus Period          :1
53.MODbus Config          :80  ... Gavazzi EM540
54.Baud Rate      :4   ... 115200

Model No          :3423-0100-0100
Sensor Type       :WISka Wire  V1.0

S : Sensor Settings
W : WISka Wire RS485
R : Read RAK3172
P : Program RAK3172
X : Reset WISka
U : Reset RAK
F : Factory Defaults
Q : Quit Setup
?
```

4.1 Setting Uplink Period

The base Uplink Period in seconds = Spl Period (20) x Send Period (21)

= 900 x 1
= 900 seconds = 15minutes

To change the base Uplink Period from 900 seconds to 600 seconds

S20=600 <CR>

This changes 20 – The SPL Period # from 900 to 600.

The base Uplink Period in seconds = Spl Period (20) x Send Period (21)
= 600 x 1
= 600 seconds = 15minutes

4.2 Setting MODbus Uplink Period

The MODbus Uplink Period in seconds = Spl Period (20) x Send Period (21) x CO Period (62)

= 900 x 1 x 1
= 900 seconds (15 minutes)

To change the CO Uplink Period from 900 seconds to 1800 seconds

S52=2 <CR>

The CO Uplink Period in seconds = Spl Period (20) x Send Period (21) x CO Period (62)

= 900 x 1 x 2
= 1800 seconds (30 minutes)

4.3 Setting MODbus Config

The following MODbus Meters are supported

- 79 = PC Test Mode
- 80 = Carlos Gavazzi EM540 Series
- 81 = Carlos Gavazzi EM210 Series

S53=80 <CR>

This will set the RS485 to communicate with a Carlos Gavazz EM540 Series

4.4 Setting MODbus Baud rate

The following bauds can be set :-

0 = 9600

1 = 19200

2 = 38400

3 = 57600

4 = 115200

To change the baudrate to 9600

S54=1 <CR>

4.5 Testing the RS485 Connection to the MODbus Meter

Connect the MODbus meter to the WISka Wire

Setup the correct MODbus Meter and Baud Rate

NOTE: The Address of the MODbus Meter must be set to 1

Press “W” on menu

MODbus Phy Address 0x0033 Hz*10

<Data received from MODbus Meter>

MODbus Phy Address 0x0034 kWh*10

<Data received from MODbus Meter>

MODbus Phy Address 0x0036 kvarh*10

<Data received from MODbus Meter>

5 WISka Wire downlink payload

Downlink payloads are sent on the configured LoRa port + 1.

If configured port is the default (5) then downlink settings should be sent on port 6.

Header byte (0xA8)	**Payload length**	**Settings data**	**Default**	**Settings data**
1 byte	**1 bytes**	n bytes		n bytes

0x41	Sensor Time Base	2 byte time (seconds)	900	4 Bytes HEX
0x42	Send Time	2 byte uplink period	1	4 Bytes HEX
0x49	MODbus Data	2 byte uplink period	1	4 Bytes HEX
0x57	MODBus Config	2 byte Config	80	4 Bytes HEX 80 = Carlos Gavazzi 540 81 = Carlos Gavazzi 210
0x60	MODBus BaudRate	2 byte Baud Rate	4	4 Bytes HEX 0 = 9600 baud 1 = 19200 baud 2 = 38400 baud 3 = 57600 baud 4 = 115200 baud

Example 1

The default uplink time is 900 seconds – 15 minutes.

The LoRa Uplink Port = 05

Therefore, to change the uplink time to 1200 seconds – 20 minutes

Send a Downlink to Port 6

A803410480

Where A8 = Header Byte
03 = Number of bytes following in settings data
41 = Sensor Time Base
0480 = 1200 in hex

Example 2

The default setting for the uplink period for the MODbus data is

1
X (times)
Sensor Time Base (900 seconds) X Send Time Period (1) = 900seconds

Therefore, a MODbus data is uplinked every 900 seconds (15minutes)

If this was required to be changed from 1 to 2 (every 1800 seconds – 30minutes)

Send a Downlink to Port 6

A803490002

Where A8 = Header Byte
03 = Number of bytes following in settings data
49 = Modbus Uplink Period
0002 = 2

Example 3

Set the Modbus Meter Type to 81 = Carlos Gavazzi 210

Send a Downlink to Port 6

A803570002

Where A8 = Header Byte
03 = Number of bytes following in settings data
57 = Modbus Config
0051 = 81 (decimal)

Example 4

Set the Modbus Baud Rate to 9600

Send a Downlink to Port 6

A803600000

Where A8 = Header Byte
03 = Number of bytes following in settings data
60 = Modbus Baud Rate
0000 = 0 (decimal) =9600

6 Programming the WISka Wire Sensor Firmware

Note: The required Hex file(s) and processor type will be emailed as requested

6.1 Programmer Required

Part Number: ATATMEL-ICE-BASIC - Atmel-ICE Basic Kit - Cost £116.00

https://www.microchipdirect.com/dev-tools/ATATMEL-ICE-BASIC?dfw_tracker=92744-ATATMEL-ICE-BASIC&allDevTools=true&gad=1&gclid=Cj0KCQjwyLGjBhDKARIsAFRNgW_uLncC5ZnaRwQqL8flHOrc49LcKGDmzdoAqEy_UQEbM1FGlZz86T4aAqPeEALw_wcB

Using the AVR Port (not the SAM port on the Programmer) – 10way connector

- Pin 2 – GND
- Pin 3 – UPDI
- Pin 4 – VTG
- Pin 6 – nSRST

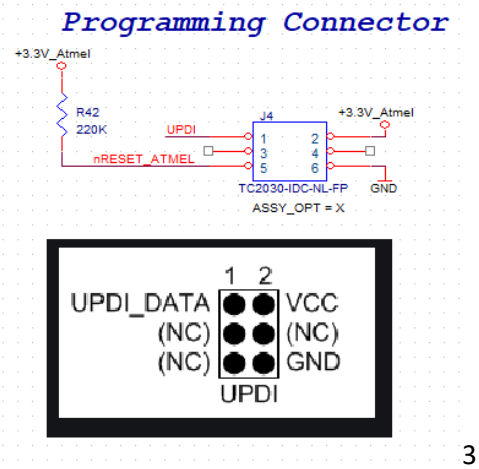
6.2 Software Required

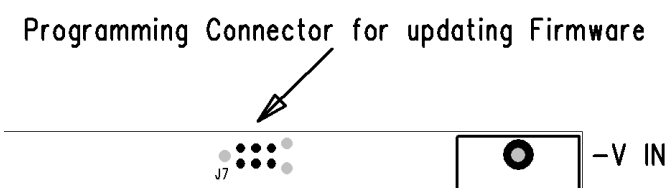
Install the following software for Windows 10 or Windows 11

Microchip Studio for AVR and SAM Devices

https://www.microchip.com/en-us/tools-resources/develop/microchip-studio

6.3 Programming Connection on WISka





3

6.4 Programming Cable Required

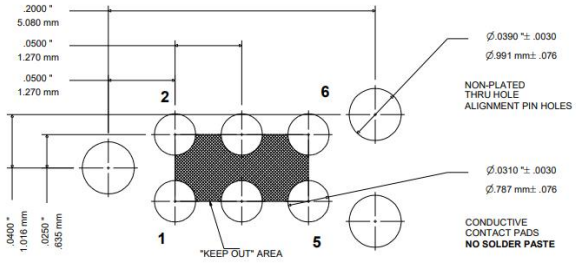
TC2030-ICESPI-NL No Leg Cable for use with Atmel-ICE - $59

https://www.tag-connect.com/product/tc2030-icespi-nl-no-leg-cable-for-use-with-atmel-ice

Information Only

TC2030-ICESPI-NL Connections

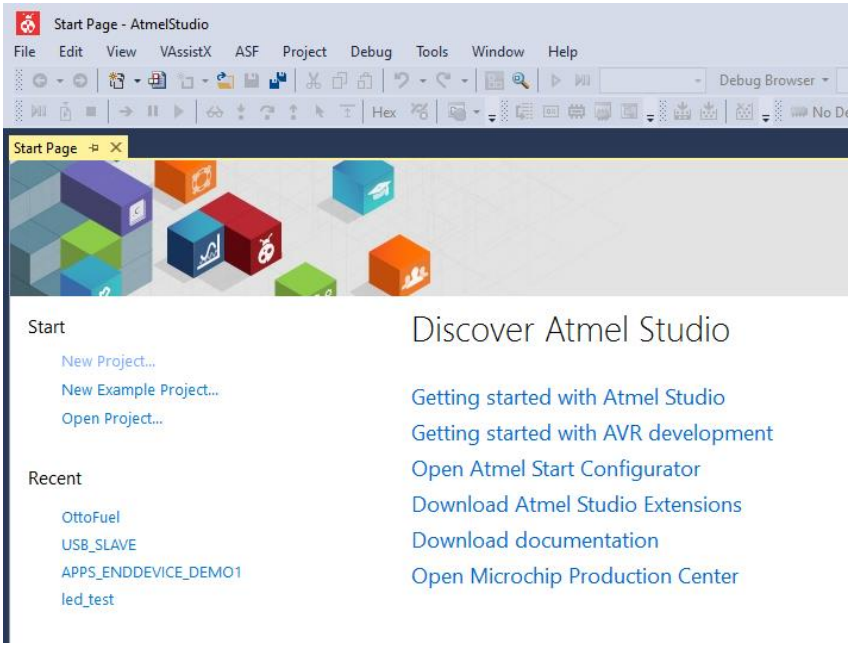
TC2030	SPI	PDI	aWire	10-pin IDC
1	MISO	PDI_DATA	DATA	3 TDO
2	VTGS	VTref	VTG	4 VTG
3	SCK			1 TCK
4	MOSI			9 TDI
5	/RESET	PDI_CLK		6 nSRST
6	GND	GND	GND	2 GND



PCB TOP LAYER

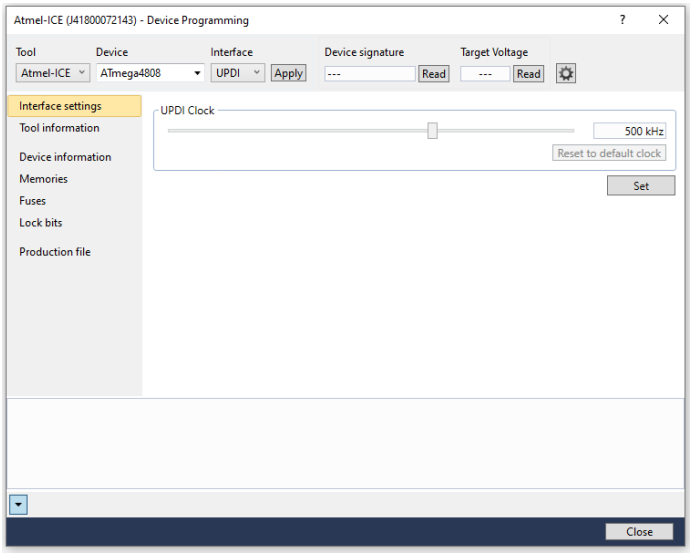
6.5 Connect the Atmel-ICE to the USB Port on the Windows PC

Start Microchip Studio (Note: Startup screens will differ)



6.6 Select Tools – Device Programming

Note: Ensure the ATmega4808 is selected

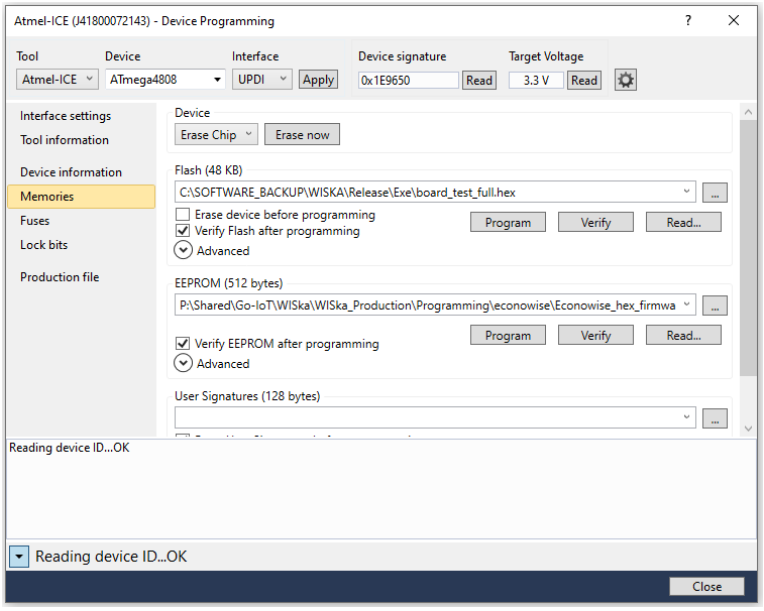


Note: The **Tool** dropdown will display the information on the connected programmer

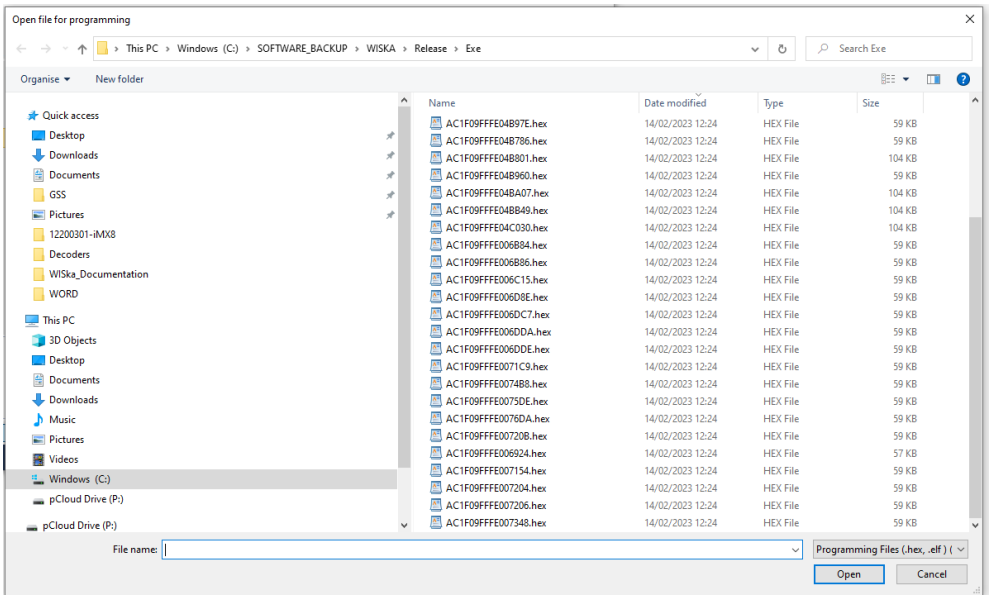
The **Device** dropdown will display the processor type to be programmed.

Place programming header onto the programming pins on the WISka Sensor. The programmed header is polarised by 3 x Fixed Pins. Push Down Gently on the 6 x spring loaded pins

6.7 Select Apply – then Select Memories



6.6 Flash (48 KB) Browse the PC for the supplied hex file


Open file for programming
This PC › Windows (C:) › SOFTWARE_BACKUP › WISKA › Release › Exe
Search Exe
Organise
New folder
Quick access
Desktop
Downloads
Documents
GSS
Pictures
12200301-iMX8
Decoders
WISka_Documentation
WORD
This PC
3D Objects
Desktop
Documents
Downloads
Music
Pictures
Videos
Windows (C:)
pCloud Drive (P:)
pCloud Drive (P:)
Name
Date modified
Type
Size
AC1F09FFFE04B97E.hex 14/02/2023 12:24 HEX File 59 KB
AC1F09FFFE04B786.hex 14/02/2023 12:24 HEX File 59 KB
AC1F09FFFE04B801.hex 14/02/2023 12:24 HEX File 104 KB
AC1F09FFFE04B960.hex 14/02/2023 12:24 HEX File 59 KB
AC1F09FFFE04BA07.hex 14/02/2023 12:24 HEX File 104 KB
AC1F09FFFE04BB49.hex 14/02/2023 12:24 HEX File 104 KB
AC1F09FFFE04C030.hex 14/02/2023 12:24 HEX File 104 KB
AC1F09FFFE006B84.hex 14/02/2023 12:24 HEX File 59 KB
AC1F09FFFE006B86.hex 14/02/2023 12:24 HEX File 59 KB
AC1F09FFFE006C15.hex 14/02/2023 12:24 HEX File 59 KB
AC1F09FFFE006D8E.hex 14/02/2023 12:24 HEX File 59 KB
AC1F09FFFE006DC7.hex 14/02/2023 12:24 HEX File 59 KB
AC1F09FFFE006DDA.hex 14/02/2023 12:24 HEX File 59 KB
AC1F09FFFE006DDE.hex 14/02/2023 12:24 HEX File 59 KB
AC1F09FFFE0071C9.hex 14/02/2023 12:24 HEX File 59 KB
AC1F09FFFE0074B8.hex 14/02/2023 12:24 HEX File 59 KB
AC1F09FFFE0075DE.hex 14/02/2023 12:24 HEX File 59 KB
AC1F09FFFE0076DA.hex 14/02/2023 12:24 HEX File 59 KB
AC1F09FFFE00720B.hex 14/02/2023 12:24 HEX File 59 KB
AC1F09FFFE006924.hex 14/02/2023 12:24 HEX File 57 KB
AC1F09FFFE007154.hex 14/02/2023 12:24 HEX File 59 KB
AC1F09FFFE007204.hex 14/02/2023 12:24 HEX File 59 KB
AC1F09FFFE007206.hex 14/02/2023 12:24 HEX File 59 KB
AC1F09FFFE007348.hex 14/02/2023 12:24 HEX File 59 KB
File name:
Programming Files (.hex, .elf) (
Open
Cancel

6.8 Flash (48 KB) – Select Program

The progress will be displayed on the screen. Ensure the programming header is held correct in place during programming.

Connector the WISka PCB to USB or Battery for power

1. The printed number on the TAG Connect Grey Cable from the programmer should be TC2030-ICESPI-NL-LEMTA ?

2. The same cable is plugged into the AVR Connector - There is a bump on the black connector, so it only fits in one way

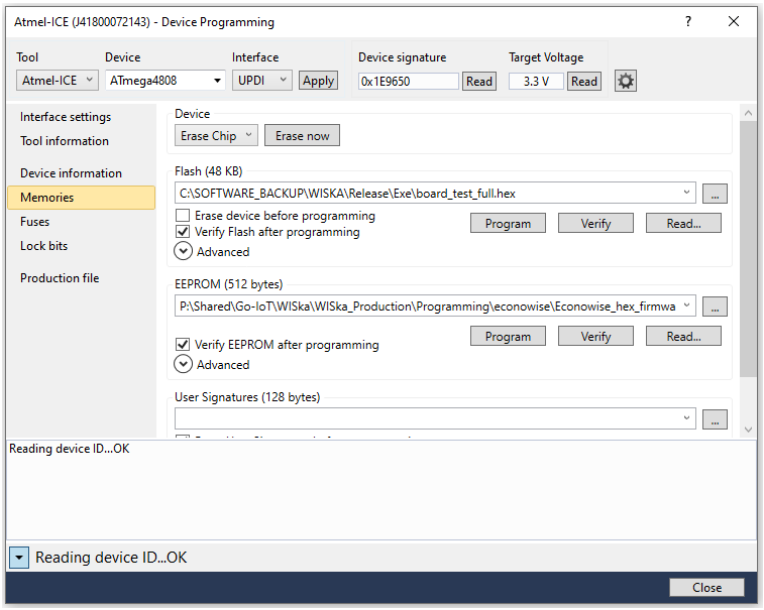
3. Start the program on the PC - Microchip Studio - or Atmel Studio

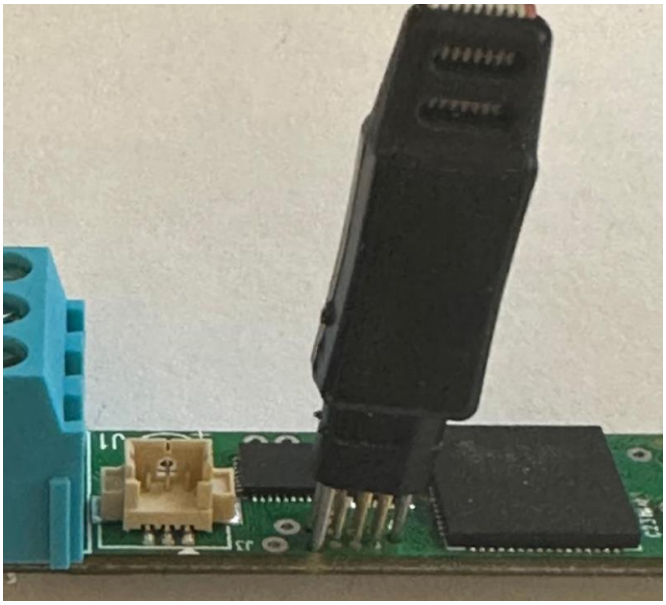
4. Click on "Interface" - Apply button

5. Connect the Spring loaded connector in to the connector as shown below - Be careful as there are three locating hole in the PCB to polarise the programming connector corrector - see below

6. Click on "Device Signature" - Read button

7. You should see the device signature - 0x1E9650 and Target Voltage of 3.3V - See below





7 MODbus Meter Registers

7.1 Carlos Gavazzi EM540 Data Format

The variables are represented by integers or floating numbers, with 2's complement notation in case of "signed" format, using the following:

Format	IEC data type	Description	Bits	Range
INT16	INT	Integer	16	-32768 .. 32767
UINT16	UINT	Unsigned integer	16	0 .. 65535
INT32	DINT	Double integer	32	-2^{31} .. 2^{31}
UINT32	UDINT	Unsigned double integer	32	0 .. 2^{32}-1
UINT64	ULINT	Unsigned long integer	64	0 .. 2^{64}-1
IEEE754 SP		Single-precision floating-point	32	$-(1+[1-2^{-23}])x2^{127}$ .. 2^{128}

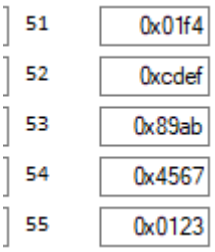
For all the formats the byte order (inside the single word) is MSB->LSB. In INT32, UINT32 and UINT64 formats, the word order is LSW-> MSW.

7.2 Carlos Gavazzi EM540 function code 03 & 04

Modicom address	Physical address	Length (words)	VARIABLE ENG. UNIT	Data Format	Notes
300052	0033h	1	Hz	INT16	Value weight: Hz*10
300053	0034h	2	kWh (+) TOT	INT32	Value weight: kWh*10
300055	0036h	2	Kvarh (+) TOT	INT32	Value weight: kvarh*10

7.3 Carlos Gavazzi EM540 simulation with MODbus slave

- 51 = 0x33x = Hz
- 52(LSW) & 53(MSW) = 0x34/0x35 = kWh (+) TOT
- 54(LSW) & 55(MSW) = 0x36/0x37 =Kvarh (+) TOT



MODbus Phy Address 0x0033 Hz*10
*** CASE 3 ***
Data RXed :01F4

MODbus Phy Address 0x0034 kWh*10
*** CASE 3 ***
Data RXed :CDEF89AB

MODbus Phy Address 0x0036 kvarh*10
*** CASE 3 ***
Data RXed :45670123 AT
OK

AT+SEND=5:EA01f4 EB89abcdef EC01234567 // Showing words reversed
OK