Geosense Smart Mux Quick Guide – Connection & Communication

Smart Mux Variants

There are two types of SmartMux:

- Analogue – E.g. Vibrating Wire, 4-20mA, Volt
- Digital – E.g. Tilt Meter, Tilt Beam, IPI,

This documentation details how to connect to and communicate with each type of SmartMux.

Analog Smartmux

The Analog SmartMux can be controlled via RS485 half-duplex or RS232 via the communication ports marked on the device. Firstly, configure the Serial Port Settings: -

Baud Rate: 9600

Data Bits: 8

Stop Bits: 1

Parity: None

Sending Commands

To instruct the SmartMux to take a reading, construct the HEX command using the following format: -

$$$$,AAAA,BB,CC,DD,EE,FF,GGGG,HH + Char(10)

$$$$(4 char): preamble – always constant

AAAA(4 char): not used

BB (2 char): GMux Address defined by dip switches on mux

CC (2 char): channel to read

DD (2 char): Channel A Type

EE (2 char): Channel B Type

FF (2 char): Warming Time (in seconds)

GGGG (4 char) : Not used

HH (2 char): Checksum

Char 10 - The ASCII character code 10 is sometimes written as \n and is called a New Line or NL. ASCII character 10 is also called a Line Feed or LF.

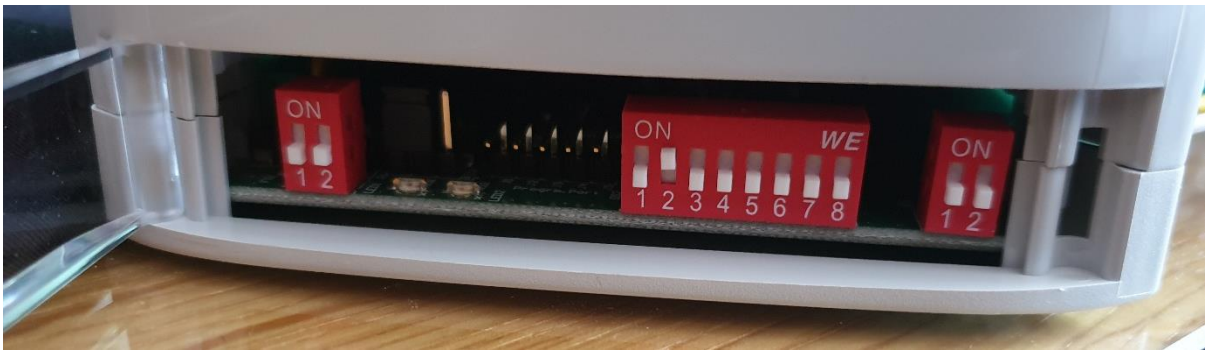
GMux Address

The GMux Address (BB) is defined by the dip switch positions on the SmartMux. It is possible to set the GMux ID to a value between 1 and 256. The dip switches represent binary values whereby: -

- Switch 1 = 1
- Switch 2 = 2
- Switch 3 = 4
- Switch 4 = 8
- Switch 5 = 16
- Switch 6 = 32
- Switch 7 = 64
- Switch 8 = 128

The switches can only be activated by first setting them all to the UP/1 position, then the unit must be powered and both green and red LEDS should flash, during this time the dip switches can be set to their correct positions and ID will be set.

To set the GMux ID to 2 enable switch 2 as shown: -



To set the GMux ID to 57, enable switches 6,5,4 and 1 to sum 57.

6 5 4 1

32 + 16 + 8 + 1 = 57

Channels, Sensor Types and Warming times

The channel to read (CC) represents the channel on the SmartMux to be read. These are marked on the SmartMux unit and range from 1 to 16. A separate serial command needs to be sent per channel regardless of whether they all have the same type of sensor attached.

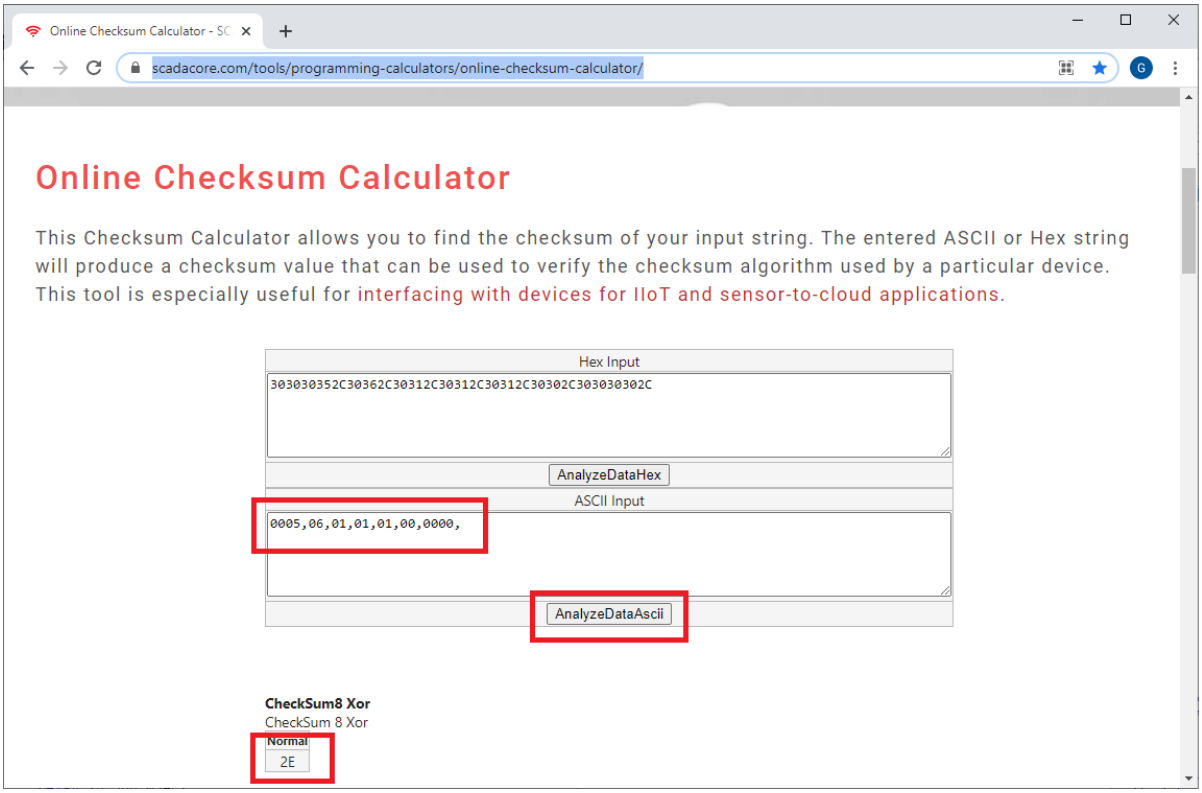
Each channel (CC) is split into 2 types A (DD) and B (EE) and has a warming time. These will need to be set depending on the type of sensor connected using the following table: -

Sensor Type	Type A	Type B	Warming Time
Vibrating Wire	01	01	00
4/20mA 2 Wires	03	00	01
4/20mA 3 Wires	06	00	01
+20V single	09	00	01
+20V dual	0a	00	01
mV/V single	07	00	01
mV/V double	08	00	01
NTC	04	00	01
PT100	05	00	01
+/- 12V dual	0b	00	01

Checksum

All the commands end with a checksum which is a logic xor of all the ascii format of the previous transmitted characters, except for the preamble and the first comma. The following online tool can be used to calculate the checksum manually:

https://www.scadacore.com/tools/programming-calculators/online-checksum-calculator/



Examples

Below there is an example for each sensor type with a common Mux ID and channel number to show how they should be formatted: -

Sensor Type	Example with checksum
Vibrating Wire	$$$$,0005,06,01,01,01,00,0000,2e
4/20mA 2 Wires	$$$$,0005,06,01,03,00,01,0000,2c
4/20mA 3 Wires	$$$$,0005,06,01,06,00,01,0000,29
+20V single	$$$$,0005,06,01,09,00,01,0000,26
+20V dual	$$$$,0005,06,01,0a,00,01,0000,7e
mV/V single	$$$$,0005,06,01,07,00,01,0000,28
mV/V double	$$$$,0005,06,01,08,00,01,0000,27
NTC	$$$$,0005,06,01,04,00,01,0000,2b
PT100	$$$$,0005,06,01,05,00,01,0000,2a
+/- 12V dual	$$$$,0005,06,01,0b,00,01,0000,7d

Receiving a Response

Upon sending a successful command the SmartMux will show a solid green and red for approximately 1 second and make an audible clicking sound. It will then return a response using the following format: -

####,AAAA,BB,CC,Str1,Str2,FFFF,GG + Char(10)

(4 char) : preamble

AAAA (4 char) : not used

BB (2 char) : GMux Address as defined in send request

CC (2 char) : Channel number as defined in send request

Str1 : Sensor reading for type A of channel (max 30 chars)

Str2: Sensor reading for type B of channel (max 30 chars)

FFFF (4 char) : not used

GG (2 char) : checksum

Example Response for a Vibrating Wire with VW in Type A and the Thermistor in Type B:

####,0000,02,10,3026.5,3118.8,00002e

Analogue Sample Code using CRBasic with a Campbell Scientific CR6

Below is some sample code designed to read a Vibrating Wire with Thermistor connected to an Analog SmartMux using a Campbell Scientific CR6 Data logger via RS485: -

```
Public PTemp, Batt_volt, Status As String * 100
Public MiddlePart(1) As String * 100
Public Result(1) As String * 100
Public ResultSplit(7)
Public Retry
Public VWSMBUnits(1)
Public VWSMTemp(1)

SequentialMode

'Main Program
BeginProg
 SerialOpen (ComC1,9600,0,0,64,4) ' 9600
        Scan (500,mSec,0,0)
                PanelTemp (PTemp,15000)
                Battery (Batt_volt)
        NextScan
 SlowSequence
  Scan (5, sec, 1, 0)
   SerialFlush (ComC1)
   Delay (0,300,mSec)
   ' GMux Id = 2 and Channel is 1
   Retry = 0
   Result(1) = ""
   Sprintf(MiddlePart(1), "$$$$,003a,%02x,%02x,%02x,%02x,%02x,%04x,7A" + CHR(10),2,1,2,4,1,0)
   Status = SerialOut (ComC1,MiddlePart(1),"",1,0)
   Do While Result(1) = ""
    If SerialInChk(ComC1)>0 Then
     SerialIn(Result(1),ComC1,100,13,100)
     SplitStr (ResultSplit,Result(1),",",7,0)
     If ResultSplit(4) > 80000 Then ' Nothing connected
      VWSMBUnits(1) = 0
      VWSMTemp(1) = 0
     Else
      VWSMBUnits(1) = (ResultSplit(4)^2)/1000
```

```
        VWSMTemp(1) = (1/(0.0014051+(0.0002369*LOG(ResultSplit(5)))+(0.0000001019*((LOG(ResultSplit(5))^3)))))-
273.15
       EndIf
       ExitDo
       'Delay (0,300,mSec)
      EndIf
      Retry = Retry + 1
      Delay (0,500,mSec)
      If Retry >20 AND Result(1) = "" Then
        Result(1) = "Serial comms timeout - moving to next sensor"
      EndIf
    Loop
    SerialFlush (ComC1)
  NextScan
EndProg
```

Digital Smartmux

The Digital SmartMux can be controlled via RS485 half-duplex or RS232 via the communication ports marked on the device. Firstly, configure the Serial Port Settings: -

Baud Rate: 9600

Data Bits: 8

Stop Bits: 1

Parity: None

Sending Commands

The digital SmartMux works differently to the Analogue version, as a specific text command will need to be sent to the digital sensor connected to a channel.

To communicate with a digital sensor connected to the digital Mux firstly instruct the Mux which channel to send the command to, this will open the channel to send one or more text commands. The channel will remain open for a short period of time whilst sending commands as strings or receiving data but will close the channel when there is nothing being transferred.

To instruct the Digital SmartMux to open a channel, construct the HEX command using the following format: -

$$$$,DMUX,AAAA,BB + Char(10)

$$$$(4 char): preamble – always constant

DMUX(4 char): not used

AAAA (4 char): channel to read

BB (2 char): checksum

Char 10 - The ASCII character code 10 is sometimes written as \n and is called a New Line or NL. ASCII character 10 is also called a Line Feed or LF.

For information about the checksum please refer to the *Analogue version* of this document

Example

Below is any example of the comms being sent to channel 1. These commands are used to read a Geosense Tilt Meter.

First open channel 1
Send -> **$$$$,DMUX,0001,05 + char(10)**
Get a response from the logger to confirm channel 1 is open
Receive -> **####,ON01**
Command to digital tilt sensor to take a reading
Send -> **@@65535 tr + char(10) + char(13)**
Sensor returns TR as string to tell the user it has taken a reading
Receive -> TR
Ask the sensor to show the reading
Send -> **@@65535 sr + char(10) + char(13)**
Reading is returned
Receive -> **SR OR,OR,30.3**
Comms stop so the channel will close within a few seconds.

Digital Sample Code using CRBasic with a Campbell Scientific CR6

Below is some sample code designed to read a Geosense Tiltmeter connected to an Digital SmartMux using a Campbell Scientific CR6 Data logger via RS485: -

```
Public PTemp, Batt_volt, Status As String * 100
Public MiddlePart(1) As String * 100
Public Result(1) As String * 100
Public ResultSplit(7)
Public Retry

SequentialMode

'EndSub

'Main Program
BeginProg
 SerialOpen (ComC1,9600,0,0,64,4) ' 9600
        Scan (500,mSec,0,0)
                PanelTemp (PTemp,15000)
```

```
            Battery (Batt_volt)
        NextScan

  SlowSequence
    Scan (20, sec, 1, 0)
      SerialFlush (ComC1)
      Delay (0,2000,mSec)
      Retry = 0
      Result(1) = ""
      Sprintf(MiddlePart(1), "$$$$,DMUX,%04x,05" + CHR(10),1)
      Status = SerialOut (ComC1,MiddlePart(1),"",1,1)
      Delay (0,3000,mSec)
      SerialFlush (ComC1)
      Status = SerialOut (ComC1,"@@65535 tr" + CHR(10) + CHR(13),"",1,1)
      Delay (0,2000,mSec)
      Status = SerialOut (ComC1,"@@65535 sr" + CHR(10) + CHR(13),"",1,1)
      Do While Result(1) = ""
        If SerialInChk(ComC1)>0 Then
          SerialIn(Result(1),ComC1,100,13,100)
          SplitStr (ResultSplit,Result(1),",",7,0)

          ExitDo
          'Delay (0,300,mSec)
        EndIf
        Retry = Retry + 1
        Delay (0,500,mSec)
        If Retry >20 AND Result(1) = "" Then
          Result(1) = "Serial comms timeout - moving to next sensor"
        EndIf
      Loop
      SerialFlush (ComC1)
    NextScan
EndProg
```