Application Note

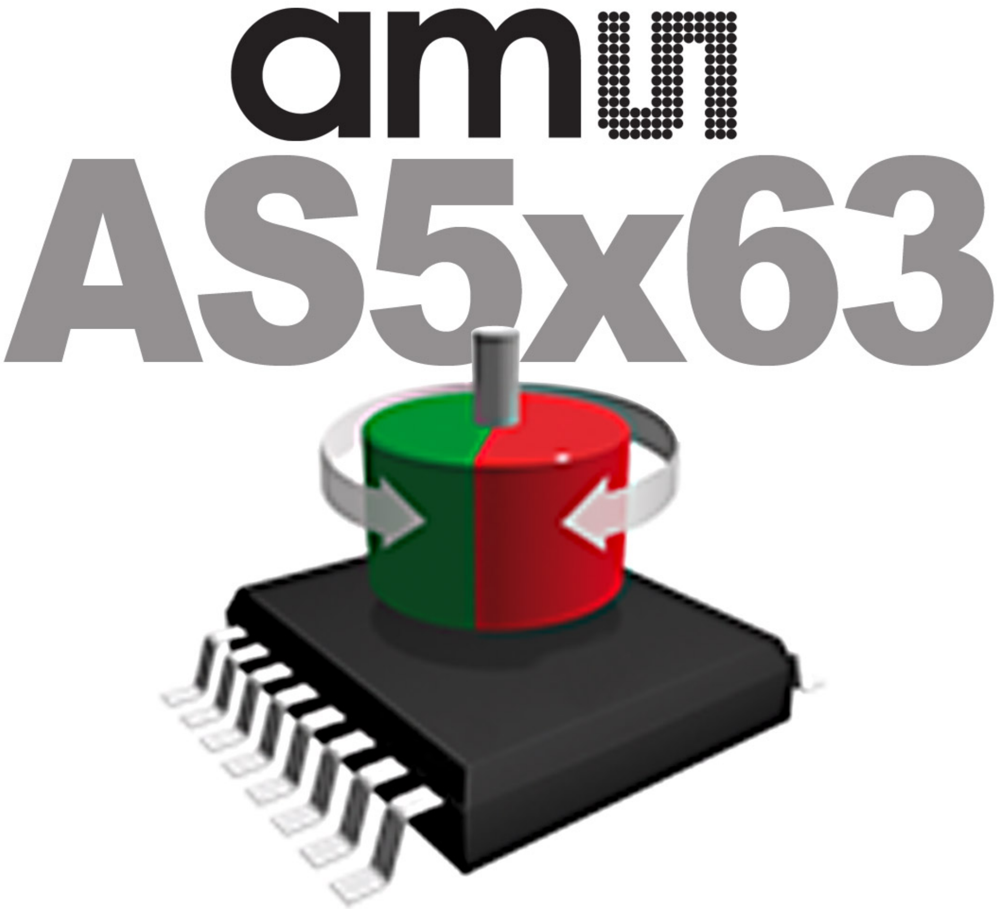
Application note:

AN134

Title:

# In-System Programming (ISP) and calibration of the ams AS5x63 family of Rotary Position Sensor devices

| Author: | Date: | Version Number: |
|---|---|---|
| John Marriott | 7th May 2013 | 0.67 |



Rotary Position Sensor Devices

## Contents

# 1.0 Introduction

This application note describes how to develop and implement In-System Programming (ISP) and device calibration for the ams AS5x63 family of ***'Rotary Position Sensor'*** devices. The document covers the AS5163 device which features a single on-chip sensor and also the AS5263 which features two identical on-chip sensors.

## 1.1 ams AS5x63 Rotary Position Sensor - Device Support

The Equinox ***'ISPnano Programmer family'*** and ***'ISPjuno'*** programmers support In-System Programming (ISP) and calibration of the AMS AS5163 and AS5263 ***'Rotary Position Sensor'*** devices.

The table below details the devices in the AS5x63 family which are currently supported.

| **Ams device** | **On-chip memorySize** | **Page Size (bytes)** | **Sensors per IC** | **Programming interface** |
|---|---|---|---|---|
| | | | | |
| AS5163 | 128 bits (16 bytes) | 128 bits (16 bytes) | 1 | 1-wire ams interface (1 channel) |
| AS5263 | 128 bits (16 bytes) | 128 bits (16 bytes) | 2 | 1-wire ams interface (2 channels) |

The programming / calibration is performed using the ***'ams 1-wire'*** programming interface.

# 2.0 Programmer and control method selection

## 2.1 Introduction

This section introduces the range of Equinox programmers which support programming and calibration of the AS5x63 family of sensor devices.

## 2.2 Programmers supporting ams 'position sensor' devices

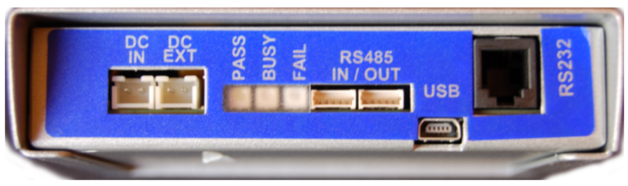
The table below details the Equinox ISP programmers which are capable of supporting programming of the ams AS5x63 family of sensor devices.

| Programmer picture | Programmer model | Programming channels | Supports 2 x AMS devices sequentially | Comment |
|---|---|---|---|---|
| | **ISPnano Series 3** | 1 (network up to 32 programmers) | Yes - can program 2 x AS5x63 devices sequentially | Standard programmer |
| | **ISPnano Series 3 ATE** | 1 (network up to 32 programmers) | Yes - can program 2 x AS5x63 devices sequentially | Features additional relay module which isolates all programmer signals |
| | **ISPnano Series 4** | 1 (network up to 32 programmers) | Yes - can program 2 x AS5x63 devices sequentially | Features plug-in relay module which isolates all programmer signals |
| | **ISPnano-MUX2** 2-channel multiplexed programming system | 1 (network up to 32 programmers) | Yes - can program 4 x AS5x63 devices sequentially | Single programmer which can sequentially program up to 4 x DUTs |
| | **ISPnano-MUX4** 4-channel multiplexed programming system | 4 sequential programming channels | Yes - can program 8 x AS5x63 devices sequentially | Single programmer which can sequentially |

| | | | | |
|---|---|---|---|---|
| | | | | program up to 4 x DUTs |
| | **ISPnano-MUX8** 8-channel multiplexed programming system | 8 sequential programming channels | Yes - can program 16 x AS5x63 devices sequentially | Single programmer which can sequentially program up to 8 x DUTs |
| | **ISPjuno** Portable ISP Programmer | 1 | Yes - can program 2 x AS5x63 devices sequentially | Portable ISP programmer. Ideal for field service use. |

**Please note:**

- All of the above programmers supports uploading of up to 64 independent ***'Standalone Programming Projects'***.

| | |
|---|---|
| | |
|  | Series III<br>Single channel ISP programmer |
| | Series III - ATE<br>Single channel ISP programmer with relay isolation |

| | |
|---|---|
| | ISP nano Series IV<br>Single channel ISP programmer<br>With plug-in connector module, relay signal isolation and opto-isolated TTL control port |
| | ISP nano mux 2<br>2 channel multiplexed Production ISP Programmer<br>2 channel<br>Multiplexed programming system |
| | ISP nano mux 4<br>4 channel multiplexed Production ISP Programmer<br>4 channel<br>Multiplexed programming system |
| | ISP nano mux 8<br>8 channel multiplexed Production ISP Programmer<br>8 channel<br>Multiplexed programming system |

## 2.3 Programmer control methods – Overview

There are four possible methods of controlling an Equinox programmer in order to program an ams AS5x63 sensor device. An overview of the different control methods is shown in the table below.

| Control method: | |
|---|---|
| EQTools<br>**EDS**<br>**Development Mode** | **EQTools – EDS – Development Mode**<br>• Simple development mode IDE<br>• Supports reading / writing of sensor memory, reading Cordic value, switching between the sensor 'communications' and 'functional' modes.<br>• Only suitable for evaluation of devices as it does NOT support actual calibration of the sensor. |
| ConsoleEDS | **ConsoleEDS – console application**<br>• This software utility allows any Equinox programmer to be controlled via simple Command Line instructions from a Command Window within Windows. |
| ISPPRO | **ISP Pro – Production Programming standalone application**<br>• This software is used to control the programmer in a production environment. It is not supplied as standard with this programmer. |
| ActiveX | **ActiveX control**<br>• Fully featured ***'ActiveX Control'*** supporting all aspects of programming / calibration of an ams AS5x63 sensor device.<br>• Currently only supports control of a single programmer |

## 2.4 Programmer control methods – selection guide

The table below gives an overview of the capabilities of the different programmer control methods.

| | EDS Development Mode | ConsoleEDS | ISP-PRO | ActiveX Control |
|---|---|---|---|---|
| **Type of control** | Simple but powerful Graphical User Interface (GUI) | Direct command-line control or Execution of commands via batch files | Customised 'programming scripts' are used to control programming execution | An ActiveX Control and ams-specific control library are used to control programmer execution |
| **Standalone application** | Yes (No other external application or software is needed) | Yes (Only requires external batch files) | No Requires Remote Application | No Requires Remote Application |
| **Possible to integrate with Remote Application?** | No | Yes | Yes | Yes |
| **Number of concurrent programming channels supported** | 1 | 32 | 32 | 1 |
| **Full programming and calibration possible** | No EDS does not integratation with ams 'Calculation algorithm' so full calibration not possible. | Yes The Remote Application must perform the calibration algorithm. | Yes The Remote Application must perform the calibration algorithm. | Yes The Remote Application must perform the calibration algorithm. |
| **Recommended use** | Development use or sample testing or evaluation | Production control of programmer | Production control of programmer | Production control of programmer |
| **User knowledge required** | None | Minimal Need to know how to run a Console application. | Very involved. Requires in-depth knowledge of programmer scripting. | Requires a working programming knowledge of Windows control applications and ActiveX technology. |

## 2.5 Programmer control methods – supported device operations

The table below gives an overview of the which ams device-specific operations are supported by the different programmer control methods.

| AS5x63 Supported operations | EDS Development Mode | ConsoleEDS | ISP-PRO | ActiveX Control |
|---|---|---|---|---|
| **Read / Write memory areas** | Yes | Yes | Yes | Yes |
| **Read Cordic (angular position) value** | Yes Displayed on-screen in real-time | Yes Returned as a parameter | Yes Returned as a parameter | Yes Returned as a parameter |
| **Read Gain value** | Yes | Yes | Yes | Yes |
| **Calculate calibration parameters** | No | Yes* | Yes* | Yes* |
| **PASS2FUNC** | Under development | Yes | Yes | Under development |
| **FUSE** | No | Yes | Yes | Under development |
| **AS5263 Device Selection** | Yes | Yes | Yes | Yes |

*The calculation of the ***'AS5x63 Calibration parameters'*** for the ams AS5x63 devices requires a separate ***'Calculation dll'*** which is available directly from ams.

## 2.6 ams AS5x63 – Labview - Demo control application

ams have produced a comprehensive ‘demo application’ written for Labview which demonstrates the salient aspect of programming and calibration of the ams AS5x53 family of sensor devices.

| | |
|---|---|
| | • **ams Labview – Demo application**<br>Demo application written for Labview (from National Instruments)<br>• Demonstrates all the required programmer interactions required to program and calibrate an ams AS5x63 sensor device.<br>• Uses ConsoleEDS to actually control the programmer(s). |

This application is available directly from ams.

## 2.7 Programmer firmware required

The minimum programmer firmware version required for programming the AMS AS5163 and AS5263 devices is detailed in the table below.

| Programmer | Minimum Firmware version | Comment |
|---|---|---|
| ISPnano Series 3 | 6.74 | Release version |
| ISPnano Series 3 ATE | 6.74 | Release version |
| ISPnano Series 4 | 6.74 | Release version |
| ISPjuno | TBA | Not released yet |

**Please note:**

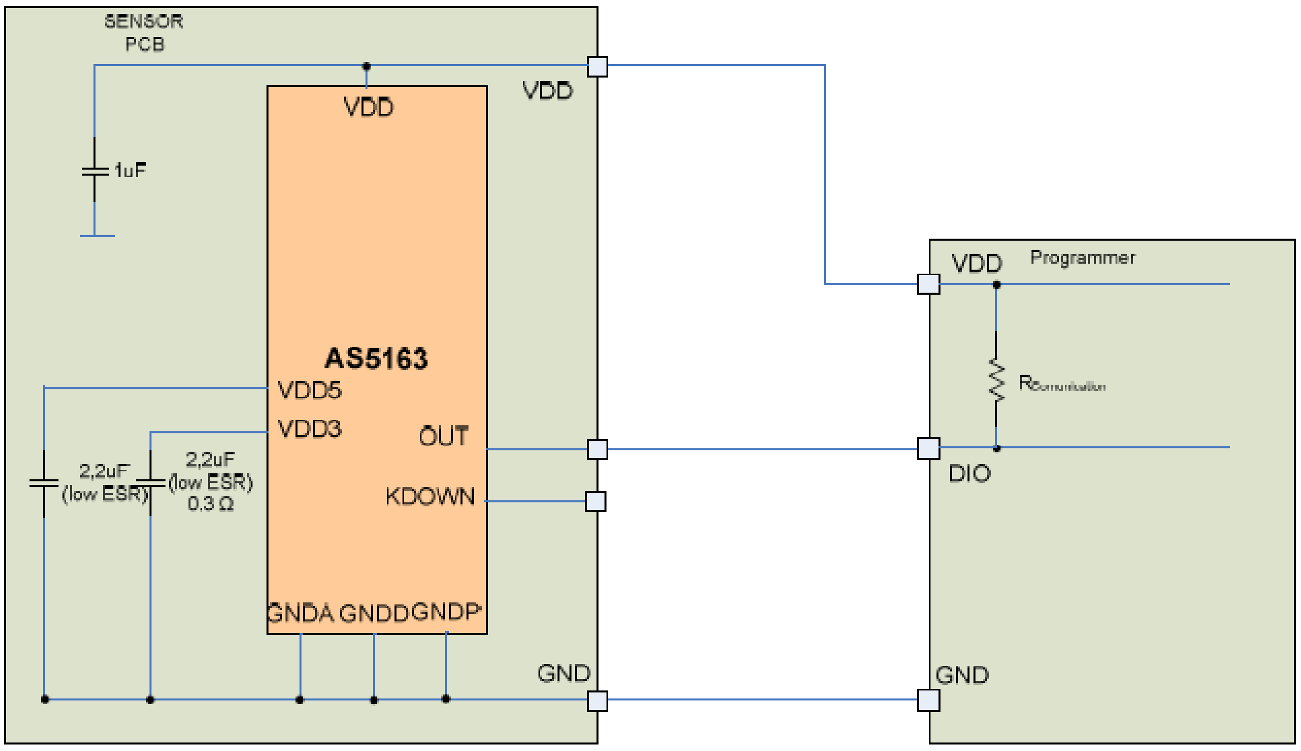
- Please see ***Application Note – AN112*** for instructions on updating your programmer firmware.

# 2.0 AMS 1-WIRE - Programming Interface

## 2.1 Overview of the AMS 1-WIRE Programming Interface

The programming of the AMS AS5163 and AS5263 devices is performed over a proprietary AMS ***'single-pin Interface'***. This interface supports bi-directional communications with the target device from the programmer. The ***'single-pin Interface'*** is shared with the analogue output pin (OUT pin) of the AMS device which means that the external programmer must place the target device in a special ***'communications mode'*** in order to program the device.

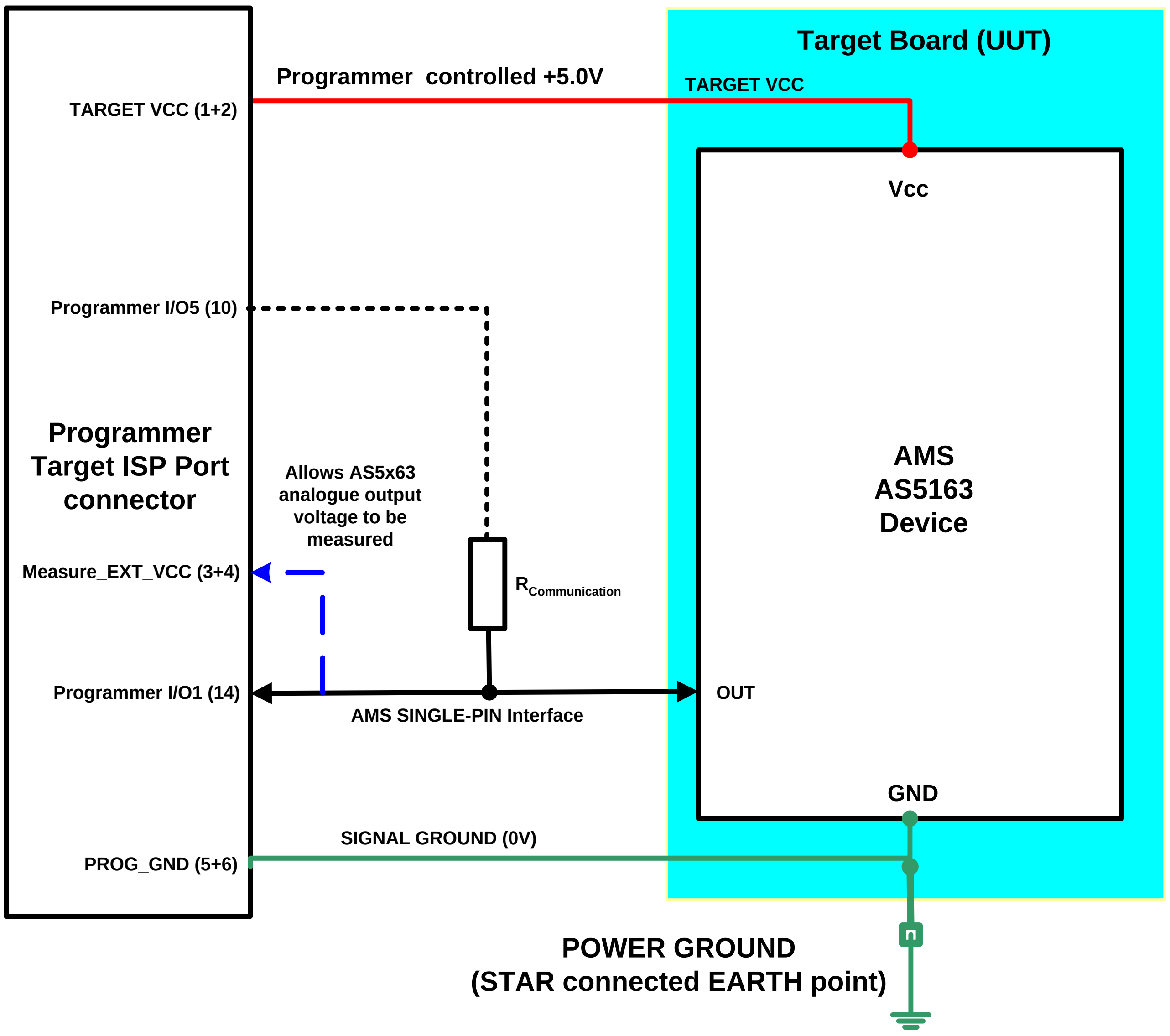
The schematic below is taken from the datasheet for the AS5163 device and shows the connections between a 'programmer' and a typical AS5163 target board (UUT).



**Important note:**
A pull-up resistor is required between VDD and the AMS OUT pin in order for the 'communications mode' to operate. This resistor is only required during programming. It could be connected within the probe bed of the test fixture or across the programmer 'Target Vcc' and 0V pins.

## 2.2 Connecting a single AS5163 to an ISPnano programmer

The diagram below shows the connections required between an Equinox ***'ISPnano Series 3 or 4'*** programmer and a Target Board (UUT) for programming a single AMS AS5163 device via the AMS ***'single-pin interface'***.



The pins required for programming the AMS devices via the ***'single-pin interface'*** interface are detailed in section 2.4.

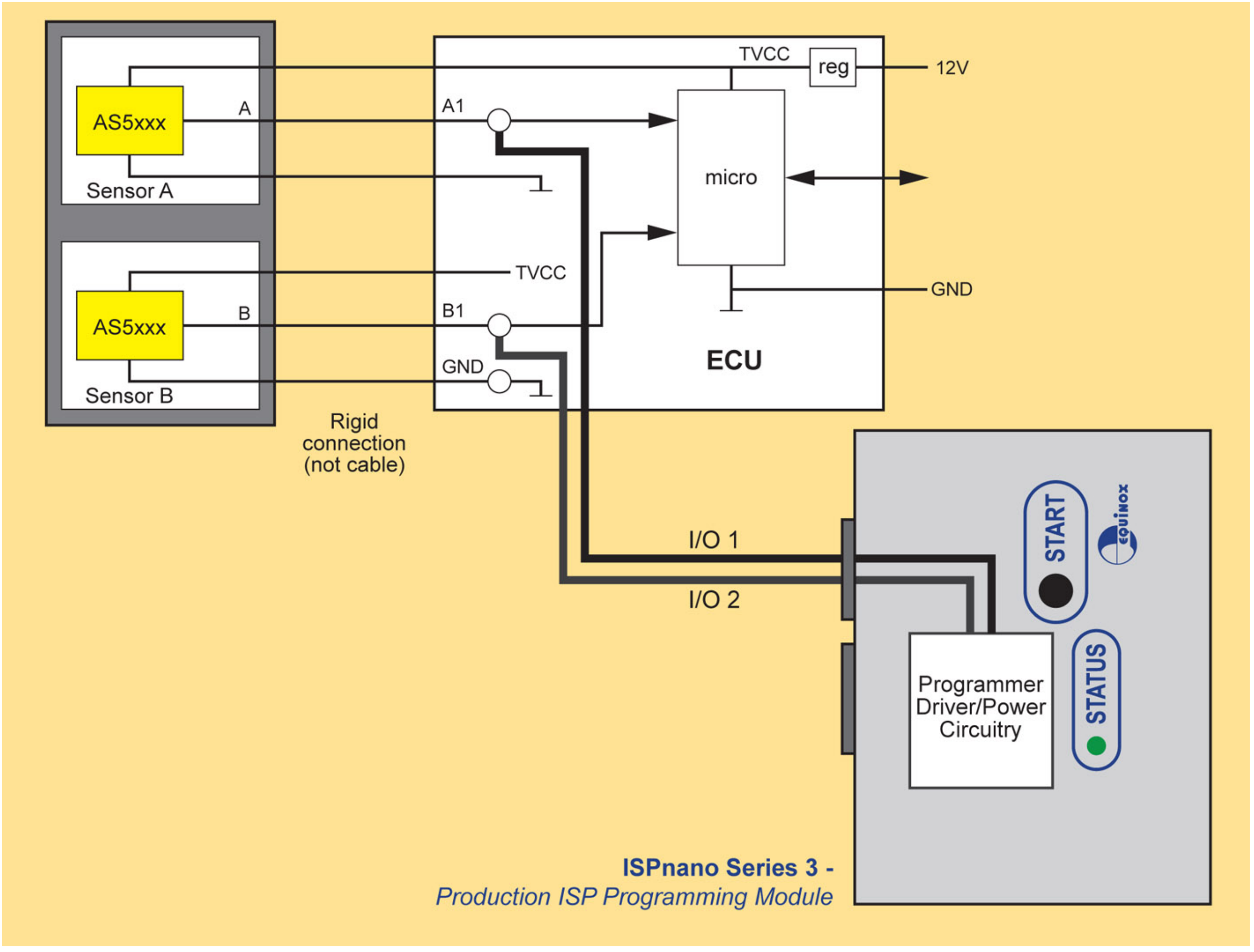
**Notes:**
A pull up resistor ***Rcommunication*** to VCC is required during the programming process in order for the ***'communications mode'*** to operate correctly. This pull-up resistor can be either permanently tied to VCC on the target board / test fixture / programmer connector. Alternatively, if the pull-up resistor could interfere with run-time measurements, it is possible to use one of the spare programmer I/O pins to create a ***'programmable pull-up'*** which is only enabled during programming.

## 2.3 Connecting a 2-sensor AS5263 to a single ISPnano programmer

The AMS AS5263 device features TWO completely independent sensors in a single package. Each sensor features its own independent connections and must be programmed / calibrated independently of the other sensor. It is possible to perform calibration of both sensors using a single ISPnano programmer programming each device sequentially.

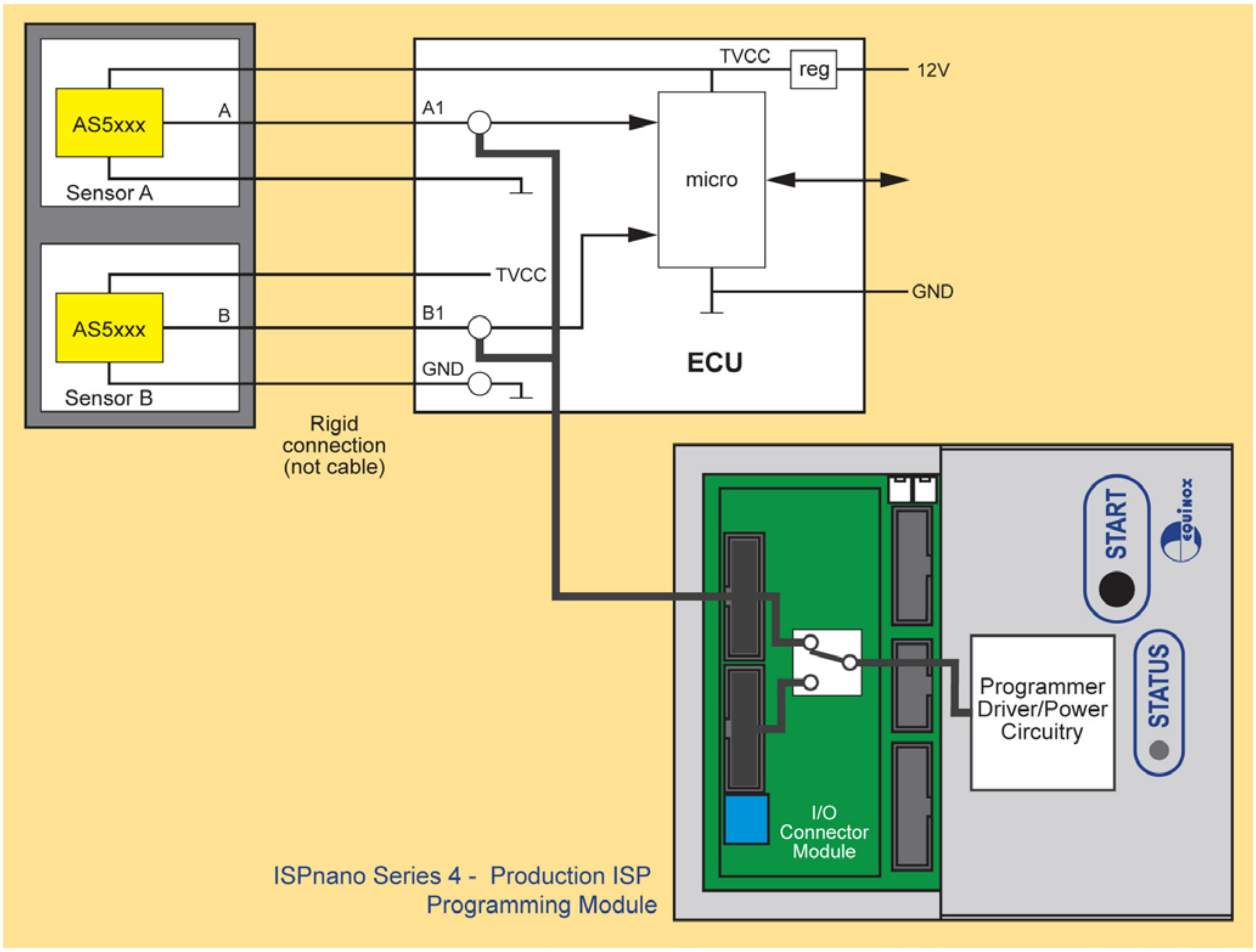
### *2.3.1 Single ISPnano Series III programmer connected to AS5263 device*

The diagram below shows the connections required between an Equinox ***'ISPnano Series 3'*** programmer and a Target Board (UUT) for programming both sensor devices inside an AMS AS5263 device via the AMS ***'single-pin interface'***.



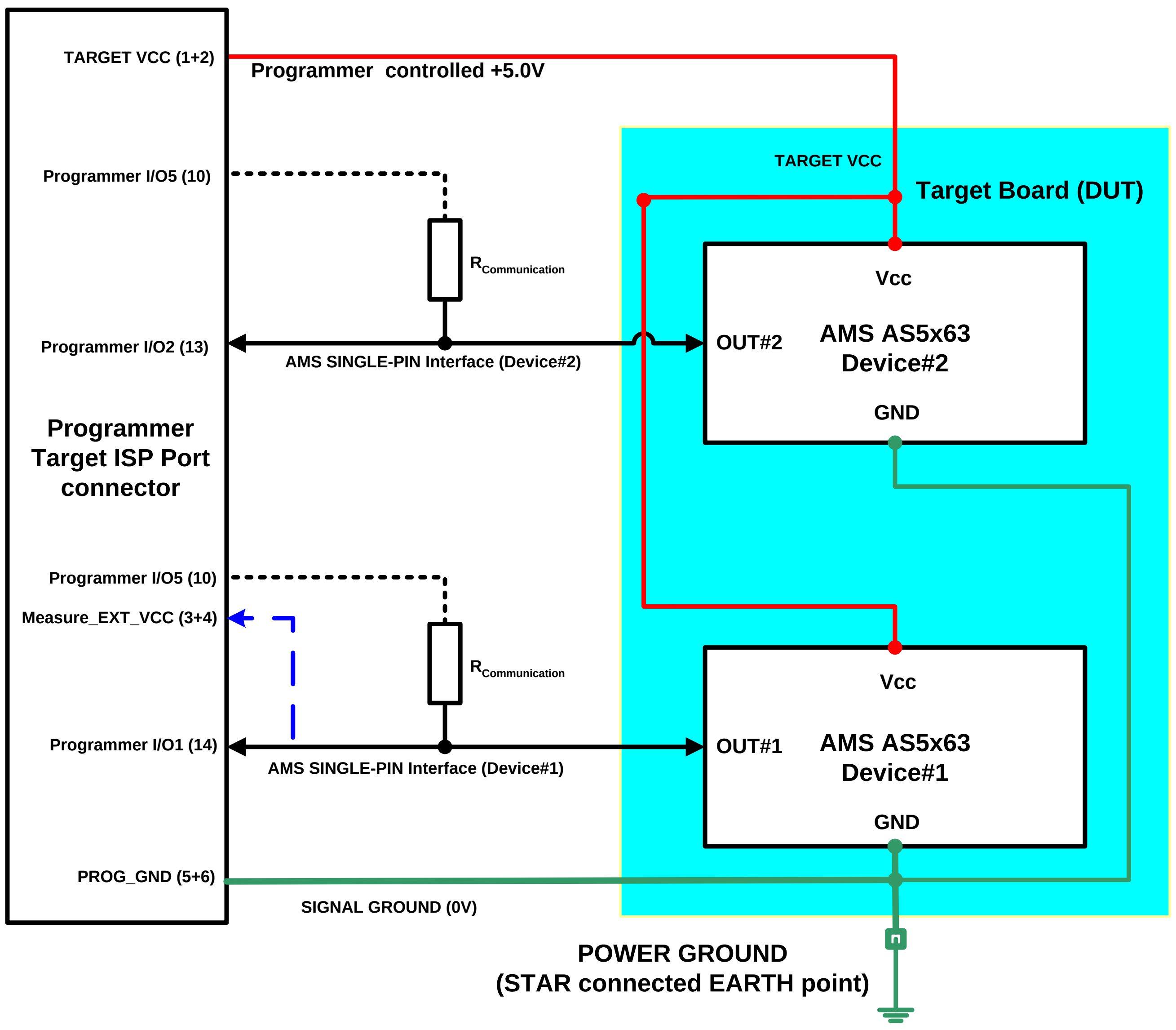
## *2.3.2 Single ISPnano Series IV ATE programmer connected to AS5263 device*

The diagram below shows the connections required between an Equinox ***'ISPnano Series IV ATE'*** programmer and a Target Board (UUT) for programming both sensor devices inside an AMS AS5263 device via the AMS ***'single-pin interface'***.


AS5xxx
A
Sensor A
AS5xxx
B
Sensor B
TVCC
reg
12V
A1
micro
TVCC
GND
B1
GND
ECU
Rigid connection (not cable)
START
STATUS
Programmer Driver/Power Circuitry
I/O Connector Module
ISPnano Series 4 - Production ISP Programming Module

### 2.3.3 Connection schematic for programmer connected to AS5263 device

The diagram below shows the connections between the programmer ***'Target ISP Port'*** of either an ***'ISPnano Series 3'*** or ***'ISPnano Series 4 ATE'*** programmer and the two sensors inside the AS5263 device.



The pins required for programming the AMS devices via the ***'single-pin interface'*** interface are detailed in section 2.4.

**Notes:**

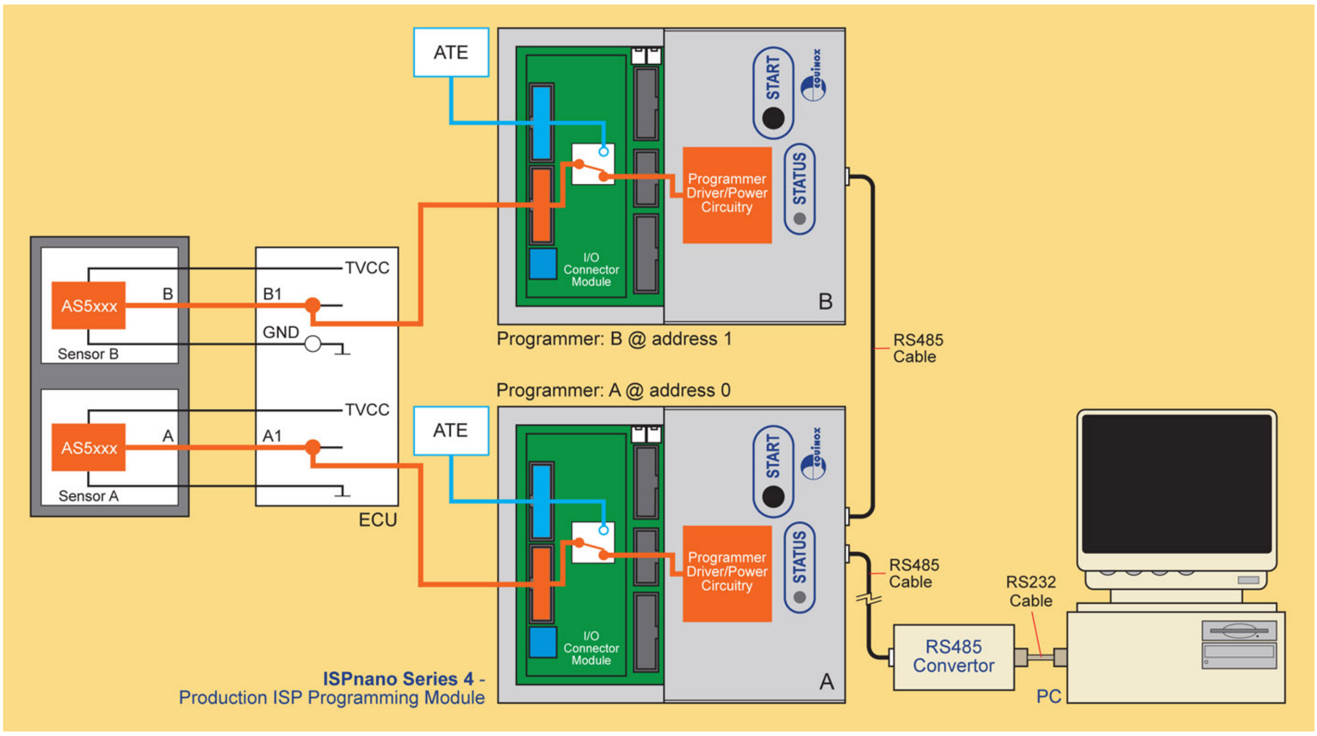
- ***Sensor Device#1*** is connected to the programmer ***I/O1*** pin.
- ***Sensor Device#2*** is connected to the programmer ***I/O2*** pin.
- The power supply to each sensor is common.
- A pull up resistor ***Rcommunication*** to VCC is required on the OUUT pin of each sensor device during the programming process in order for the ***'communications mode'*** to operate correctly. This pull-up resistor can be either permanently tied to VCC on the target board / test fixture / programmer connector. Alternatively, if the pull-up resistor could interfere with run-

time measurements, it is possible to use one of the spare programmer I/O pins to create a ***'programmable pull-up'*** which is only enabled during programming.
- The programmer switches between ***Sensor Device#1*** and ***Sensor Device#2*** using a either a setting in the 'programming project' or a special ConsoleEDS command.

## 2.4 Connecting a 2-sensor AS5263 to two independent ISPnano programmers

If concurrent programming of both sensors inside an AS5263 device is required, then it is possible to use 2 x ISPnano programmers to perform the programming. Each programmer connects to a different sensor so programming of each sensor is completely independent of the other sensor.

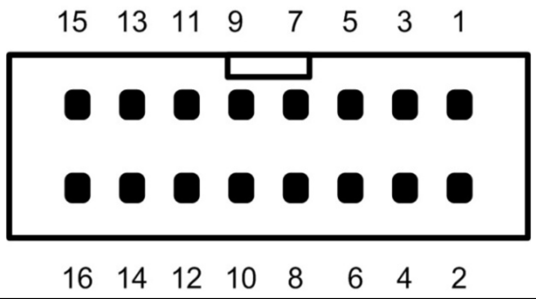
The illustration below shows how 2 x ***'ISPnano Series IV ATE'*** programmers are connected to ***Device#1*** and ***Device#2*** inside the AS5263 device.....



## 2.5 ISPnano Series 3 - Target ISP Port – AMS pin-out

The ***'Target ISP Connector'*** ports of the ***'ISPnano Series III'*** and ***'ISPnano Series IV ATE'*** programmers features all the signals required to implement In-System Programming (ISP) of a target AMS device using the ***'AMS single-pin interface'***.

The illustration below shows the location of the ***'Target ISP Connector'*** port on the rear panel of the programmer.

<table>
<tr><td> </td><td><b><i>'Target ISP Connector'</i></b> port<br><br>The connector is a 16-pin bump-polarised IDC connector with 0.1" pin spacing.<br><br>Pin 1 is the top right pin as shown in the diagram opposite.</td></tr>
</table>

This connector also features the programmable ***"Target Vcc"*** and ***"Target Vpp"*** voltages plus a switched ***"EXTERNAL Vcc"*** supply.

The pins on this connector which are used for programming AMS AS5163 / AS5263 devices are detailed in the table below.

| **Programmer Signal name (16-way IDC)** | **IDC pin** | **Signal description** | **Direction from programmer** | **Pin name on AMS device** |
|---|---|---|---|---|
| TARGET_VCC | 1+2 | Target Vcc Supply | Passive | VCC |
| 3 + 4 | TARGET_ EXT_VCC | External Target Vcc<br>Allows analogue output of AS5x63 device to be measured. | Input | AS5163 – OUT<br>AS5263 – OUT_B |
| GROUND (0V) | 5+6 | Target / Programmer Signal GROUND | Passive | GND |
| 9 | OP6_SPARE | Optional programmable pull-up on the target OUT pin | Output | AS5263 – OUT_T |
| 10 | Programmer I/O5 | Optional programmable pull-up on the target OUT pin | Output | AS5163 – OUT |
| 13 | Programmer I/O2 | - AS5163 – 1-wire programming interface<br>- AS5263 – 1-wire programming interface (DIE2 – TOP) | Bidirectional | AS5263 – OUT_T |
| 14 | Programmer I/O1 | - AS5163 – 1-wire programming interface<br>- AS5263 – 1-wire programming interface (DIE1 – BOTTOM) | Bidirectional | AS5163 – OUT<br>AS5263 – OUT_B |

**Notes:**

1. The AS5163 features a single sensor in the device package.
   - Programmer pin I/O1 (IDC pin 14) should be connected to the AS5163 ‘OUT’ pin.

2. The AS5263 features a TWO separate sensor devices arranged as ‘stacked dies’ in a single device package.
   - Programmer pin ***I/O1 (IDC pin 14)*** should be connected to the ***AS5263 ‘OUT_B’*** pin which connects to the ***‘BOTTOM’*** die inside the device.
   - Programmer pin ***I/O1 (IDC pin 14)*** should be connected to the ***AS5263 ‘OUT_T’*** pin which connects to the ***‘TOP’*** die inside the device.

## 2.6 Signal / Power GROUND (0V) connections

It is very important that both the programmer and Target System (UUT) are earthed correctly. Incorrect grounding can lead to current flowing in the 0V signal back to the PC which could cause ESD damage to either the programmer or the UUT. The ISPnano programmer features a ***'Signal GROUND'*** which is a specially filtered (cleaned) 0V signal which is used only for the programming signals. The UUT should then use its own dedicated ***'Power GROUND'*** as this will be much noisier the programmer 0V.

**Signal GROUND (0V)**
The ***'Signal GROUND'*** is the 0V to which the programming signals (PDI, SPI, JTAG etc) are referenced to. This is a specially filtered 0V signal line which is used only for the programming signals. The ***'Signal GROUND'*** should be connected from the GROUND pins on the 'Target ISP Port' connector directly to the main GROUND on the UUT. The minimum cable length should be used for this connection.

**Power GROUND (0V)**
The ***'Power GROUND'*** is the 0V to which the Target Board (UUT) uses as its 0V reference. The ***'Power GROUND'*** should be connected from the main GROUND (0V) point on the UUT to the 'Star connected GROUND of the overall programming fixture. This is usually the point where all 0V GROUND connections are made for the power supplies in the fixture.

# 3.0 Creating an EDS Development Project

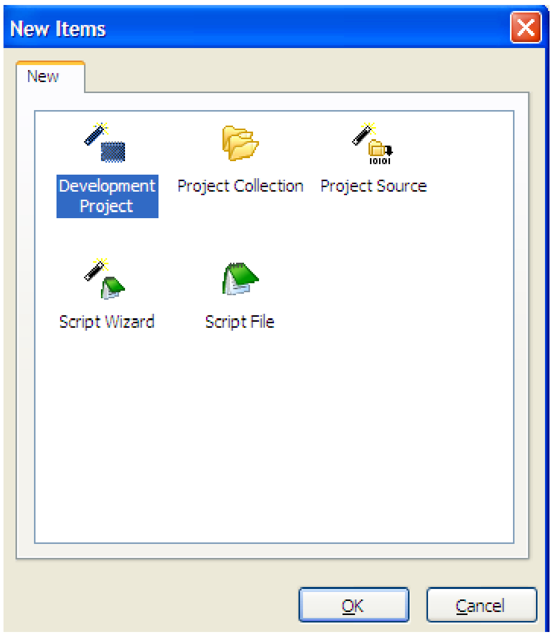
## 3.1 Overview

This section describes the steps required to create an ***'EDS – Development Project'*** which can be used to program an AMS device under PC control.

## 3.2 Selecting a Development Project

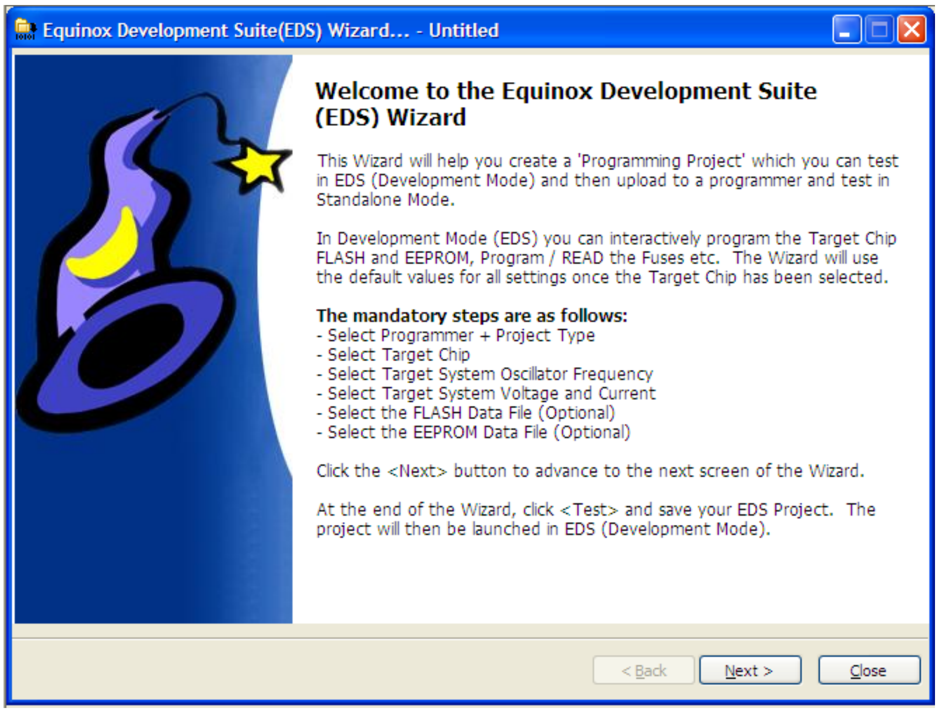
To create a ***'Development Project'***…

- Launch EQTools
- Click the ***'New'*** icon
- The ***'New items'*** screen is displayed.



- Select ***the 'Development Project'*** icon and then click the <OK> button

→ The ***'Equinox Development Suite (EDS)'*** wizard will now start.


Equinox Development Suite(EDS) Wizard... - Untitled

**Welcome to the Equinox Development Suite (EDS) Wizard**

This Wizard will help you create a 'Programming Project' which you can test in EDS (Development Mode) and then upload to a programmer and test in Standalone Mode.

In Development Mode (EDS) you can interactively program the Target Chip FLASH and EEPROM, Program / READ the Fuses etc. The Wizard will use the default values for all settings once the Target Chip has been selected.

**The mandatory steps are as follows:**
- Select Programmer + Project Type
- Select Target Chip
- Select Target System Oscillator Frequency
- Select Target System Voltage and Current
- Select the FLASH Data File (Optional)
- Select the EEPROM Data File (Optional)

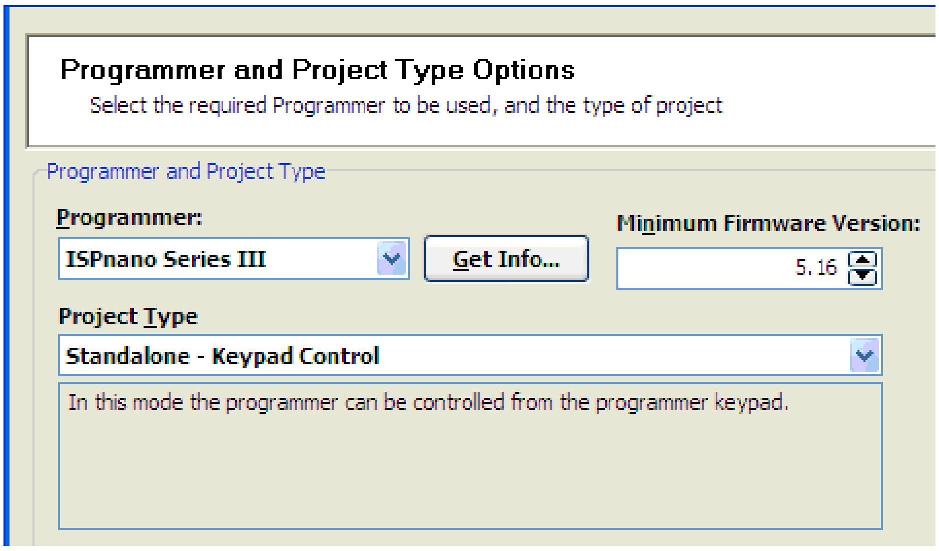
Click the <Next> button to advance to the next screen of the Wizard.

At the end of the Wizard, click <Test> and save your EDS Project. The project will then be launched in EDS (Development Mode).

## 3.3 Selecting 'Programmer and Project Type'

This screen allows you to select:

- The ***'Programmer'***
- The ***'Project Type'***


**Programmer and Project Type Options**
Select the required Programmer to be used, and the type of project

Programmer and Project Type

Programmer: ISPnano Series III

Minimum Firmware Version: 5.16

Project Type: Standalone - Keypad Control

In this mode the programmer can be controlled from the programmer keypad.

### i. Programmer

- Select the ***'Programmer'*** for which the project is to be compiled for.
- If the programmer is attached to the PC, you can simply click the ***<Get Info>*** button to automatically select the correct programmer.

### ii. Project Type

- The ***'Project Type'*** should be set to ***'Standalone – keypad control'*** for all standalone projects and ConsoleEDS projects.

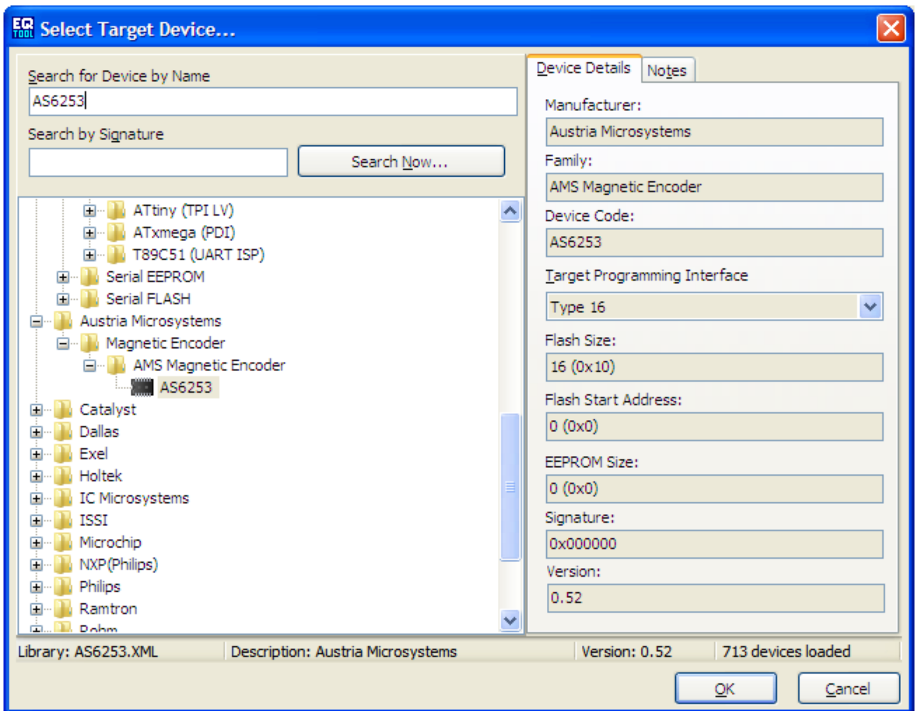
## 3.4 Selecting ‘Target Device’

This screen allows you to select the ***‘Target Device’*** to be programmed.
The simplest way to find a particular device is as follows:

- Type the ***‘Device Name / code’*** into the ***‘Search for device’*** field e.g. ***AS5163***
- Click the ***<Search now>*** button

→ All instances of the ***AS5163*** device are now displayed

- Select the required device from the drop-down list and then click the ***<OK>*** button.


Select Target Device...

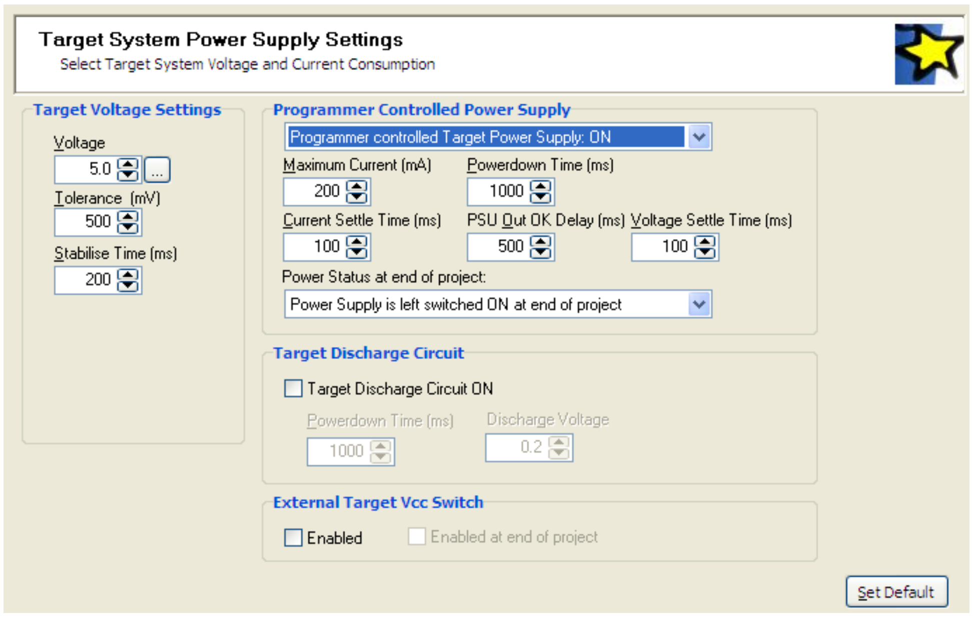
Search for Device by Name: AS6253
Search by Signature:

Manufacturer: Austria Microsystems
Family: AMS Magnetic Encoder
Device Code: AS6253
Target Programming Interface: Type 16
Flash Size: 16 (0x10)
Flash Start Address: 0 (0x0)
EEPROM Size: 0 (0x0)
Signature: 0x000000
Version: 0.52

Library: AS6253.XML | Description: Austria Microsystems | Version: 0.52 | 713 devices loaded

## 3.5 Setting up the Target Power Supply

This screen allows you to set up how the programmer powers the Target System.



Target System Power Supply Settings
Select Target System Voltage and Current Consumption

Target Voltage Settings
Voltage: 5.0
Tolerance (mV): 500
Stabilise Time (ms): 200

Programmer Controlled Power Supply
Programmer controlled Target Power Supply: ON
Maximum Current (mA): 200
Powerdown Time (ms): 1000
Current Settle Time (ms): 100
PSU Out OK Delay (ms): 500
Voltage Settle Time (ms): 100
Power Status at end of project: Power Supply is left switched ON at end of project

Target Discharge Circuit
☐ Target Discharge Circuit ON
Powerdown Time (ms): 1000
Discharge Voltage: 0.2

External Target Vcc Switch
☐ Enabled
☐ Enabled at end of project

For most AMS applications, the programmer should power the ***'Target System'*** at e.g. 5.0V

To power the Target System from the programmer:

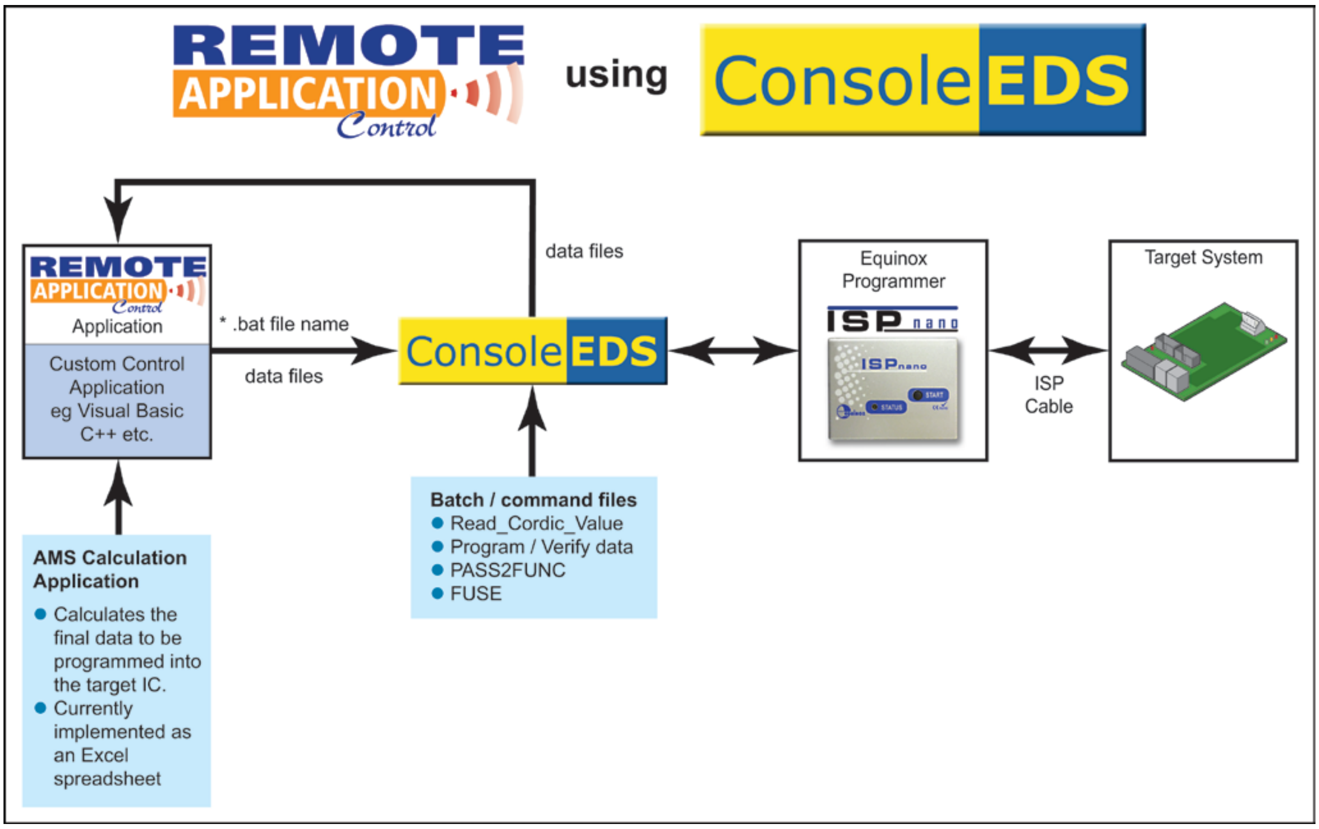
- Set the ***'Programmer controlled Target Power Supply'*** to ***'ON'***
- Set the 'Voltage' to 5.0V
- Set the ***'Power status at end of project'*** to ***'Power supply is left switched ON at end of project'***.
- Leave all other options as defaults.

# Appendix 1 – Using ConsoleEDS to program AMS devices

## 1.0 Overview

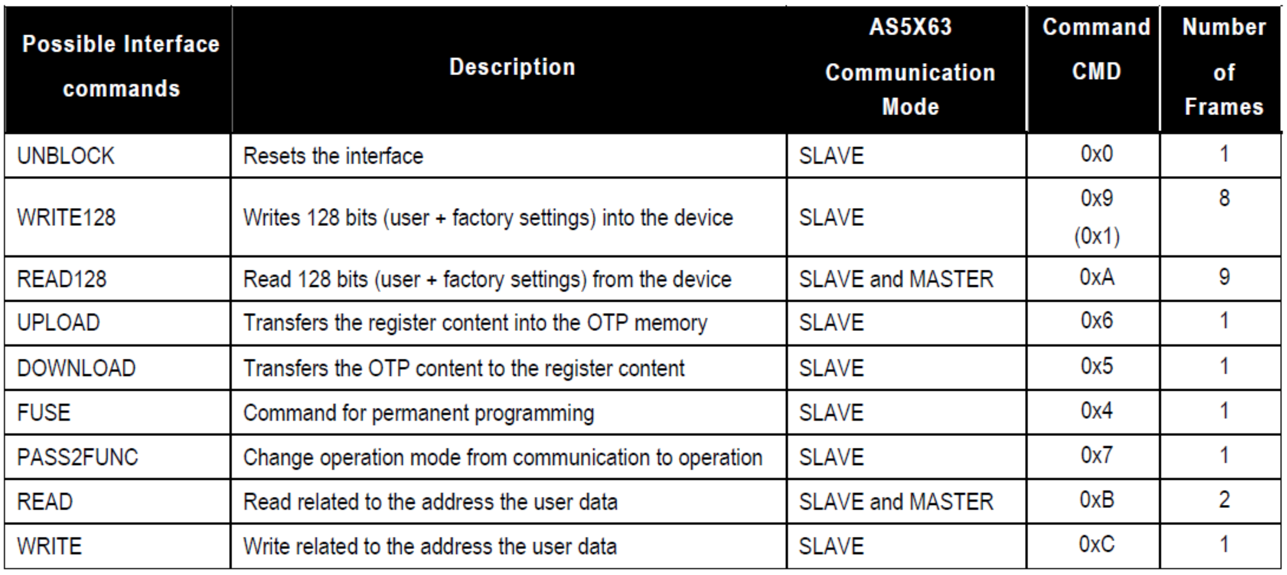
This appendix describes how to use the ConsoleEDS utility to program AMS devices. It is possible to use ConsoleEDS to program the configuration memory of an AMS device, switch the device from ***'communications mode'*** to ***'functional mode'***. Once the output voltage has been measured and validated, then ConsoleEDS can set program the device so that the configuration is permanently stored in the target device.

**Please note:**
This section provides specific instructions of how to use ConsoleEDS to program an AMS device. For further information about how to use the ConsoleEDS application in general, please refer to the separate Application Note AN111.


REMOTE APPLICATION Control
using
ConsoleEDS
data files
REMOTE APPLICATION Control Application
Custom Control Application eg Visual Basic C++ etc.
* .bat file name
data files
ConsoleEDS
Equinox Programmer
ISP nano
ISP Cable
Target System
AMS Calculation Application
Calculates the final data to be programmed into the target IC.
Currently implemented as an Excel spreadsheet
Batch / command files
Read_Cordic_Value
Program / Verify data
PASS2FUNC
FUSE

## 1.1 AMS AS5x63 command set

The table below is taken from the ***'AMS AS5x63 datasheet'***. It details all the possible AMS commands which the AS5x63 devices support.



| Possible Interface commands | Description | AS5X63 Communication Mode | Command CMD | Number of Frames |
|---|---|---|---|---|
| UNBLOCK | Resets the interface | SLAVE | 0x0 | 1 |
| WRITE128 | Writes 128 bits (user + factory settings) into the device | SLAVE | 0x9 (0x1) | 8 |
| READ128 | Read 128 bits (user + factory settings) from the device | SLAVE and MASTER | 0xA | 9 |
| UPLOAD | Transfers the register content into the OTP memory | SLAVE | 0x6 | 1 |
| DOWNLOAD | Transfers the OTP content to the register content | SLAVE | 0x5 | 1 |
| FUSE | Command for permanent programming | SLAVE | 0x4 | 1 |
| PASS2FUNC | Change operation mode from communication to operation | SLAVE | 0x7 | 1 |
| READ | Read related to the address the user data | SLAVE and MASTER | 0xB | 2 |
| WRITE | Write related to the address the user data | SLAVE | 0xC | 1 |

## 1.2 AMS command mapping to ConsoleEDS commands

This section describes how the ***'AMS AS5x63 command set'*** has been mapped to the generic ConsoleEDS command set. This allows an AMS AS5x63 device to be programmed / read back / forced into functional mode under the control of the ConsoleEDS application.

The table below lists all the available ***'AMS commands'*** with the corresponding ***'ConsoleEDS command'*** shown alongside the AMS command.

**Please note:**
Not all ***'AMS commands'*** are directly supported by ConsoleEDS. This is because these commands are automatically performed by ConsoleEDS as part of other ConsoleEDS commands.

| AMS command | Equivalent ConsoleEDS command | Command function |
|---|---|---|
| | | |
| UNBLOCK | Not available *See Note 1 | Resets the AMS 1-wire programming interface |
| WRITE128 | ***/FLASHWRITE=WriteFile.bin*** | Writes 16 bytes (128 bits) of data from the specified file to the target device. |
| READ128 | ***/FLASHREAD=ReadFile.bin*** | Reads 16 bytes (128 bits) of data from the target device and writes it to the specified file. |
| UPLOAD | ***/AMSWRITE=0x00,0x00,0x06*** | Transfers the ***'register content'*** into the OTP memory. |
| DOWNLOAD | **/AMSWRITE=0x00,0x00,0x05** | Transfers the OTP content to the ***'register content'*** |
| FUSE | ***/AMSWRITE=0x00,0x00,0x04*** | Command to permanently program the configuration data into the target device (OTP) |
| PASS2FUNC | ***/AMSWRITE=0x00,0x00,0x07*** | Command to change the operating mode of the target device from ***'communications mode'*** to ***'functional mode'*** |
| READ | ***/AMSREAD=ADDRESS1,ADDRESS2*** | Reads 2 data bytes back from the specified address in the target device |
| WRITE | ***/AMSWRITE=DATA,ADDRESS,0x0C*** | Writes a single data byte to the specified address in the target device |
| Sensor device select | ***/AMSSELECTDEVICE*** | Selects either ***Device#1*** or ***Device#2*** in a 2-sensor AS5263 device. Selection is only for a single ConsoleEDS session. |
| Sensor device select | ***/SETSTARTUPAMSSELECTDEVICE*** | Selects either ***Device#1*** or ***Device#2*** in a 2-sensor AS5263 device. Selection is permanently stored (persistent) in PC registry |

**Notes**
*Note 1 – The ***'UNBLOCK'*** command is automatically sent by the programmer in order to enter programming mode if the target device is not already in programming mode.

## 1.3 Additional ConsoleEDS commands

The table below lists additional ConsoleEDS commands which may be useful when programming AMS devices.

| ConsoleEDS command | Command function |
|---|---|
| | |
| ***/POWERUP*** | Applies power to the UUT using the voltage / current parameters specified in the 'Base project'. |
| ***/READSIG*** | This command will read back the ***'Chip ID'*** from the target AMS device. It is returned as a 3-byte hex number. The ***'Chip ID'*** is not validated / checked by ConsoleEDS. |
| ***/RESET*** | This command will exit programming mode, switch off the programmer controlled target power supply (TVCC) and tri-state all programmer I/O lines. |
| ***/NORESET*** | This command prevents ConsoleEDS from resetting the programmer / target power supply at the end of each ConsoleEDS session. |
| ***/FLASHVERIFY*** | This command will read back the contents of the entire device (16 bytes) and then verify this data against the data stored in the specified file. |
| ***/MEASUREVOLTAGE*** | This command will measure either the voltage either on the ***'Target Vcc'*** pin or the AMS device ***'OUT'*** pin. |

## 1.4 Typical programming sequence

A typical programming sequence for an AS5163 device (as described in the datasheet) is detailed below:

- Apply power to the UUT
- Write the 'trimming' bits to the target device
- Change the target device to ***'Functional Mode'***
- Measure the Vout voltage of the target device
- Change the target device back to ***'Communications Mode'***
- Program the configuration into target device OTP memory
- Remove power from the UUT

The next sections detail the ConsoleEDS commands which can be used to implement the above programming sequence.

### 1.4.1 Applying power to the UUT

- Apply power to the UUT
- Wait the ***'Startup time'*** of the target device

→ The device enters ***'Communications mode'*** ready to communicate with the programmer.

The following command will power up the UUT at 5.0V but will **NOT** attempt to enter programming mode:
***ConsoleEDS AS5163.PRJ /POWERUP=5.0 /NORESET***

### 1.4.2 Writing the 'trimming bits' into the target device

The ***'AMS Remote Application'*** must create a binary file called e.g. ***WRITE_DATA.BIN*** which is 16 bytes long and which contains all the ***'trimming data'*** required for the target device.

The following sequence to write / read-back and verify / upload to SFR memory….

- Write128 command: the trimming bits are written in the SFR memory
- Read128 command: the trimming bits are read back and verified
- Upload command: the SFR memory is transferred into the OTP RAM

This sequence can be achieved with the following ConsoleEDS as follows:

1. AMS application writes the ***'trimming data'*** into the 16-byte file called ***WRITE_DATA.BIN***

2. AMS application sends:
***ConsoleEDS AS5163.PRJ /FLASHWRITE=WRITE_DATA.BIN /NORESET***
→ The entire 16 bytes (128 bits) of data contained in the ***WRITE_DATA.BIN*** file are written to the target device.

3. AMS application sends:
***ConsoleEDS AS5163.PRJ /FLASHREAD=READ_DATA.BIN /NORESET***
→ The entire 16 bytes (128 bits) are read back from the device to the ***READ_DATA.BIN*** file.

4. AMS application must now perform a manual verify of the 'read back data' to the 'written data' to make sure the ***'trimming data'*** has been loaded correctly into the device.
AMS application would need to:

- Take ***READ_DATA.BIN*** file
- Isolate just the data bits which were written by masking out any bits which are '0' in the ***WRITE_DATA.BIN file*** and then use this mask to isolate these bits in the ***READ_DATA.BIN*** file.

5. If the verify operation in step (4) was successful, then the programmer can load the data into the OTP area using the following command:
***ConsoleEDS AS5163.PRJ /FUSEWRITE=0x00,0x00,0x06 /NORESET***
→ The entire 16 bytes (128 bits) of data are now transferred to the ***'OTP memory'***.

### *1.4.3 Changing the target device to ‘functional mode’*

Once the correct configuration data has been written to the target device, the programmer can then instruct the target device to go into ***‘functional mode’***.

This can be achieved with the following ConsoleEDS command:
***ConsoleEDS AS5163.PRJ /FUSEWRITE=0x00,0x00,0x07 /NORESET***
→ This sends the ***‘PASS2FUNC’*** command to the target device.
→ The target device should now go into ***‘functional mode’*** and start outputting a voltage on the VOUT pin.

When this command is executed, the target device should then go into ***‘functional mode’***.
→ The Vout pin of the target device will then change as the magnet is rotated.

### *1.4.4 Measuring the Vout voltage*

Once the target device is in ***‘functional mode’***, it is possible to measure the voltage at the Vout pin. This is best done using an external DVM with 12 – 14 bit measurement accuracy.

For a general non-accurate voltage check, it is also possible to use the programmer to measure the Vout pin voltage to an accuracy of eg. 8 bits using the following command:
***ConsoleEDS /MEASUREVOLTAGE=4***

### *1.4.5 Changing the target device back to ‘Communications Mode’*

Once the target device is in ***‘functional mode’***, the only way to revert back to ***‘Communications Mode’*** is power cycle the device.

The following command is used to switch power off to the UUT:
***ConsoleEDS AS5163.PRJ /RESET***

This will switch off power to the UUT. The next ConsoleEDS command will then automatically re-enter ***‘Communications Mode’***.

### *1.4.6 Reading the Cordic Value*

The ***‘Cordic value’*** represents the 14 bit absolute angular position data from the target device. It is 14-bits long and is stored across 2 bytes at address 0x10 and 0x11.

| Area Region | Address | Address | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |
|---|---|---|---|---|---|---|---|---|---|---|
| R/W USER DATA | 0x10 | 16 | CORDIC_OUT[7:0] | | | | | | | |
| | 0x11 | 17 | 0 | 0 | CORDIC_OUT[13:8] | | | | | |
| | 0x12 | 18 | OCF | COF | 0 | 0 | 0 | 0 | DSP_RES | R1K_10K |
| | 0x17 | 23 | AGC_VALUE[7:0] | | | | | | | |

| | |
|---|---|
| 🟩 | Read only |
| 🟨 | Read and Write |

To read the Cordic value back from a target device, the following AMS procedure is recommended:

**1. interaction : value 0x02 ; address 0x12 ; command 0xC : write DATA**
With this command you are setting the DSP_RES to high ( resetting the DSP)

The equivalent ConsoleEDS command is:
***ConsoleEDS AS5163.PRJ /FUSEWRITE=0x02,0x12,0x0C /NORESET***
→ This command writes the value 0x02 into address 0x12.

**2. interaction : address: 0x11; address 0x10 ; command 0xB : read DATA**
With this command you set the reading of the address 0x11 and 0x10 which contains the 14-bit Cordic value.

The equivalent ConsoleEDS command is:
***ConsoleEDS AS5163.PRJ /FUSEREAD=0x11,0x10 /NORESET***
→ This command would read the values of address 0x11 and 0x10 and display them.
→ These two values must be combined by the AMS application to make up the 14-bit Cordic value.

**3. interaction : value 0x00 ; address 0x12 ; command 0xC : write DATA**
With this command you are setting the DSP_RES to low

The equivalent ConsoleEDS command is:
***ConsoleEDS AS5163.PRJ /FUSEWRITE=0x00,0x12,0x0C /NORESET***

### *1.4.6 Programming the configuration into target device OTP memory*

Once you are happy that the ***'configuration data'*** is correct, it is then possible to program this data permanently into the ***'OTP memory'*** of the target device.

The command to permanently program the ***'configuration data'*** into the ***'OTP memory'*** is as follows:
***ConsoleEDS AS5163.PRJ /FUSEWRITE=0x00,0x00,0x04***
→ This command will send the AMS ***'FUSE'*** command to the target device which will cause the device to go into ***'functional mode'*** with the settings which have already been programmed.

**!!! Warning !!!**
Once this operation has been performed, it is no longer possible to re-enter or to change any of the device settings.

## 1.5 Explanation of ConsoleEDS 'Base Projects'

Most ConsoleEDS commands require that a ***'Base Project'*** is created for each device to the programmed. The ***'Base Project'*** is used by ConsoleEDS to define the following parameters about the target device / target system:

- Target Device e.g. AS5263
- Device selection: Allows selection of 'Device1' or 'Device2' in an AS5263 device
- Target Programming Interface e.g. 1-wire interface
- Target Vcc Voltage e.g. +5V
- Target Vpp Voltage: 0V
- Target Power Supply characteristics e.g. current
- Target Power Supply settings: e.g. 'Leave power supply ON at end of project'
- Target Programming speed: fixed
- Device Signature / Device ID

The ***'Base Project'*** is declared on the ConsoleEDS command line as follows:
***ConsoleEDS BaseProject.prj /FLASHWRITE=FlashData.bin***

## 1.6 Setting up a ConsoleEDS 'Base Project'

The simplest way to set up a ConsoleEDS ***'Base Project'*** is to use the EDS ***'Development Wizard'***.

Here is an overview of how to set up a ***'Base Project'***:

- Launch the EDS Development Wizard
- Select the required device eg. ***AS5163***
- Make sure the ***'Fuse task'*** is enabled in the project. It doesn't matter what fuse values are selected, only that the ***'Fuse task'*** is enabled.
- Make sure the ***'Security task'*** is enabled in the project. It doesn't matter what fuse values are selected, only that the ***'Security task'*** is enabled.
- You should not select any FLASH file in the project.
- Compile the project to make a *.prj project eg. ***AS5163.prj***
- You have now created a ***'Base Project'.***

## 1.7 Reading the 'Device Signature / ID'

It is possible to read the ***'Chip ID'*** from the target device using the ***/READSIG*** command.
The command will return 6 bytes of data which is the contents of DATABYTE10… DATABYTE15 of the device. The returned data includes the ***'Chip ID'*** , ***'Kickdown threshold'*** and ***'Clamping low'*** data. It is up to the remote AMS application to unpack this data and make sense of it.

**Typical command useage:**
***ConsoleEDS AS5163.PRJ /READSIG***

**Typical response:**

***Console EDS - Signature read back: 0x123456789ABC***

**Important note:**
The ***'Chip ID'*** for AMS devices is not actually unique from device to device. It contains information about the device wafer number, X/Y position on the wafer etc. This means that the ***'Chip ID'*** value cannot be used to check which AMS device is currently connected to the programmer. The ***'Chip ID'*** validation must be switched off in the programming project otherwise the project will fail if the read back ***'Chip ID'*** is different from the ***'Chip ID'*** specified in the project.

## 1.8 Programming the memory using the /FLASHWRITE command

The ***/FLASHWRITE*** command is used to program data from a file on the PC hard disk to the RAM area of the target device. For AMS devices, the data must always be programmed in a single block of 16 bytes (128 bits) as this is the 'page size' of the memory.

**Typical example:**
***ConsoleEDS AS5163.PRJ /FLASHWRITE=WRITE_DATA.BIN***

**Where:**
***AS5163.PRJ*** is the 'Base Project'
***WRITE_DATA.BIN*** is a 16-byte long binary file containing the 'configuration data' to be programmed.

This example will program the internal RAM area of the device with the contents of the binary file.

## 1.9 Selecting Device#1 or Device#2 in an AS5263 sensor

The AMS AS5263 device features TWO independent sensor devices within the same IC package. A single ISPnano programmer is capable of programming both sensors sequentially (one after the other) by connecting one sensor devices to the programmer I/O1 pin and the other sensor device to the programmer I/O2 pin.

To instruct ConsoleEDS / the programmer to communicate to Device#1, use the following command once only at the start-up of the programming sequence.

***ConsoleEDS /SETSTARTUPAMSSELECTDEVICE=0 or 1***

Where:

- 0 = Select to program Device#1
- 1 = Select to program Device#2

This setting will be permanently stored in the PC registry and so all subsequent ConsoleEDS sessions will use this setting.

If you want to just change the device selection for a single session of ConsoleEDS, then the following command should be used:

***ConsoleEDS /AMSSELECTDEVICE=0 or 1***

Where:

- 0 = Select to program Device#1
- 1 = Select to program Device#2

This setting will only be used for a single session of ConsoleEDS and then will be "forgotten".