Application Note

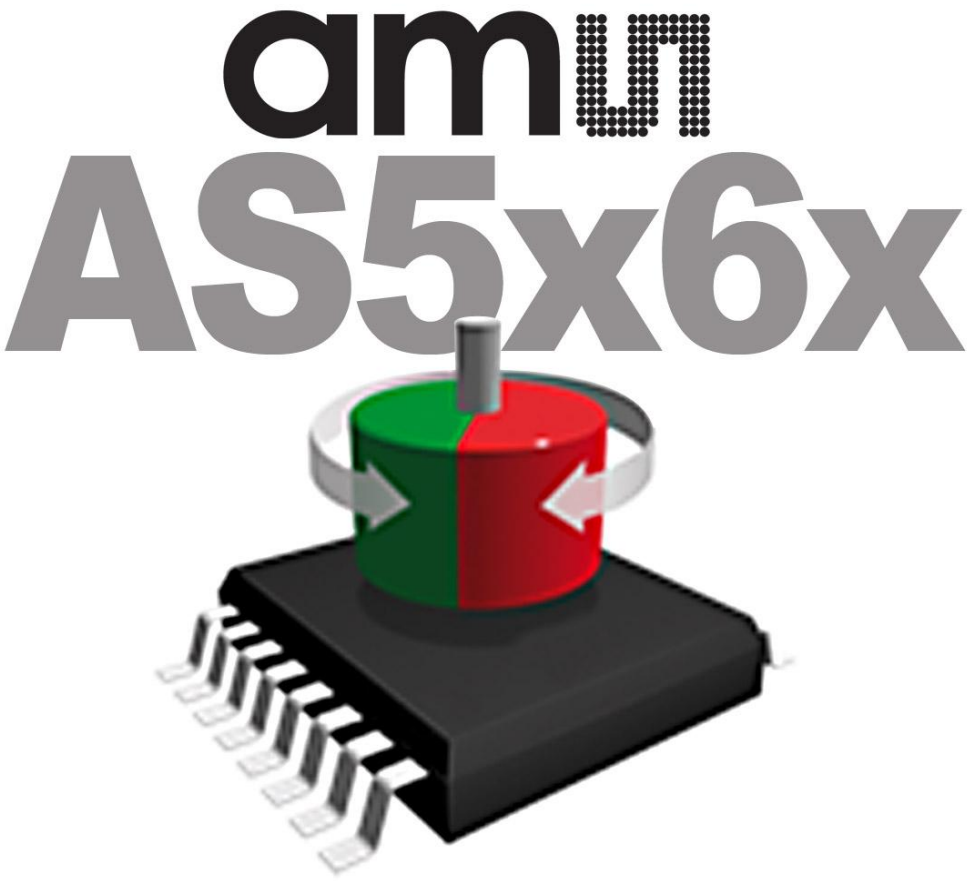
Report No:

AN136

Title:

In-System Programming (ISP) of the ams AS5x6x Rotary Position Sensor Devices family

Author:	Date:	Version Number:
John Marriott	19th May 2014	0.86



Contents

1.0 Introduction

This application note describes how to develop and implement In-System Programming (ISP) support for the ams AS5x6y family of magnetic encoder devices via the ***'Bi-directional UART'*** interface. This document explains how to control both single-sensor and dual-sensor devices using the ISPnano series of programmers.

1.1 Programmer cross-reference guide

The table below details the Equinox ISP programmers which are capable of supporting programming of the ams AS5x6x device family:

Programmer picture	Programmer model	Programming channels	Comments
	**ISPnano Series 3**	1 (network up to 32 programmers)	Standard ISPnano programmer
	**ISPnano GANG2 Programming System**	2	2-channel concurrent (parallel) programming system Ideal for programming dual-die AS5x6y devices
	**ISPnano Series 3 ATE**	1 (network up to 32 programmers)	Features additional relay module which isolates all programmer signals
	**ISPnano Series 4**	1 (network up to 32 programmers)	Features plug-in relay module which isolates all programmer signals
	**ISPnano-MUX2** 2-channel multiplexed programming system	1 (network up to 32 programmers)	Single programmer which can sequentially program up to 4 x DUTs

	**ISPnano-MUX4** 4-channel multiplexed programming system	4 sequential programming channels	Single programmer which can sequentially program up to 4 x DUTs
	**ISPnano-MUX8** 8-channel multiplexed programming system	8 sequential programming channels	Single programmer which can sequentially program up to 8 x DUTs

1.2 Overview of Equinox ISP Programmer range

The table below shows the full range of programmers supporting the ams AS5x6y device family.

	ISPnano Series III Single channel ISP programmer
	ISP nano gang Multi-Channel Gang Production ISP Programming Systems **ISPnano GANG2 Programming System** 2-channel concurrent (parallel) programming system
	ISPnano Series III-ATE Single channel ISP programmer with relay isolation
	ISPnano Series IV-ATE Single channel ISP programmer With plug-in connector module, relay signal isolation and opto-isolated TTL control port

	ISP nano mux2 2 channel Multiplexed programming system
	ISP nano mux4 4 channel Multiplexed programming system
	ISP nano mux8 8 channel Multiplexed programming system

1.3 Programmer firmware required

The minimum programmer firmware version required for programming the ams AS5x6x device family via the ***'Bi-directional UART'*** interface is detailed in the table below.

Programmer	Minimum Firmware version	Comment
ISPnano Series 3	6.74	Final production firmware release
ISPnano Series 3 ATE	6.74	Final production firmware release
ISPnano Series 4 ATE	6.74	Final production firmware release

Please note:

- It is very important that firmware 6.74 is used when programming the ams AS5x6y device family as many new commands such as ***FUSE*** and ***Pass2Func*** were added between version 6.62 and 6.74.
- Previous versions of firmware are not compatible with the Labview examples supplied by ams.
- Please see ***Application Note – AN112*** for instructions on updating your programmer firmware.

1.4 Equipment required for AMS AS5x6x support

The table below lists the equipment required to implement programming of ams AS5x6x devices via the UART interface.

Hardware:

- Equinox ISPnano ISP programmer (with the ***'ams - magnetic position encoder'*** device library enabled)
- AS5x6y Target Board (referred to as DUT in this document)
  ***** This board can either be your own custom Target Board or an AS5x6y Evaluation Board from ams. This board is NOT included with the programmer. *****

Software:

- Equinox ConsoleEDS console application (ConsoleEDS.exe)
  or
- Equinox ***'ActiveX control'*** + ***'Equinox Library'***

- ***ams AS5x6y - Labview demo*** - control software application (Labview VI)
- ams ***'Calculation dll'***

Miscellaneous

- Connection cable / interface board between the programmer and the DUT

1.5 ams AS5x6y - Device family - Device Support

The Equinox ***'ISPnano'*** family of ***'production ISP programmers'*** support In-System Programming (ISP) of ams AS5x6y devices. The programming is performed using the AMS ***'single-pin UART'*** interface.

The table below details the ams AS5x6y devices which are currently supported.....

ams device	Operating voltage	Output drive type	Sensors per IC	Device package	Programming interface	Number of programmers required
AS5162	5V	Analog voltage output	1 (single die)	SOIC 8	1 x Bi-directional UART	1
AS5161	5V	PWM output	1 (single die)	SOIC 8	1 x Bi-directional UART	1
AS5262	5V	Analog voltage output	2 (dual die)	QFN 16	2 x Bi-directional UART	2 x ISPnano Series 3 (ISPnanoS3-GANG2) or 2 x ISPnano Series 4 ATE or 1 x ISPnano-MUX2 programmer
AS5261	5V	PWM output	2 (dual die)	QFN 16	2 x Bi-directional UART	2 x ISPnano Series 3 (ISPnanoS3-GANG2) or 2 x ISPnano Series 4 ATE or 1 x ISPnano-MUX2 programmer

Important notes:

1. Only a single AS5x6x can be supported per programmer.
2. If both sensors inside a dual-die AS526x device are to be programmed, then this will require either 2 x ISPnano Series 3 (ISPnanoS3-GANG2 programming system) or 2 x ISPnano Series 4 ATE programmers or 1 x ISPnano-MUX2 programmer.
3. As the programming algorithm is identical for all devices, the AS5162 algorithm is used to program all devices.

1.6 AMS AS5x6x – Memory areas and memory mapping

The AMS AS5x6x device features the following ***'memory areas'*** which are accessible via an external programmer:

Memory area	Address range	Memory size	Equinox FLASH Page Size (bytes)	Equinox EEPROM Page Size (bytes)	Purpose / used for:
OTP	0x00 – 0x0F	16 bytes	16 bytes	2 bytes	Customer application-specific settings + AMS Factory Settings
Registers (SFRs)	0x10 – 0x2F	32 bytes	16 bytes	2 bytes	Special Function Registers

For an AS5x6x device, the entire 48 bytes (address 0x00 to 0x2F) of the device are mapped to **both** the EQTools **FLASH** and **EEPROM** areas.

As the EQTools FLASH and EEPROM areas are simply mapped onto the same physical memory area inside the AS5x6x, then this makes it possible to use either EQTools / ConsoleEDS FLASH or EEPROM commands to read from or write to the device.

Important notes:

- The FLASH commands support reading / writing in blocks of 16 bytes.
- The EEPROM commands support reading / writing in blocks of 2 bytes.
- The actual AS5x6x underlying UART protocol actually only supports reading / writing in blocks of 2 bytes but the programmer then performs 8 x reads or 8 x writes for the FLASH read / FLASH Write operation. This makes it possible to read / write a block of 16 bytes very quickly.

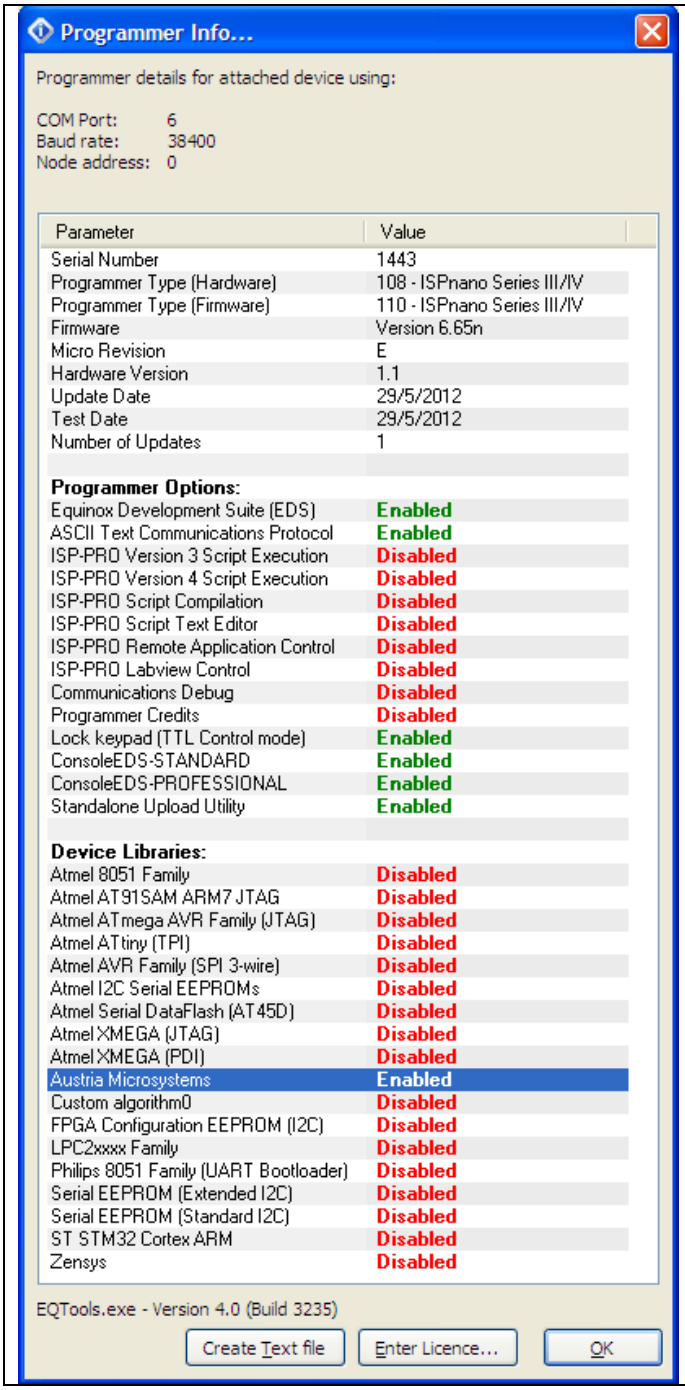
1.7 Enabling an Equinox programmer for ams device support

In order to be able to program an ***ams magnetic encoder*** device, it is necessary to purchase the ***'ams – Magnetic Encoder Device library'*** from Equinox. This library provides programming support for most available ***ams magnetic encoder*** device.

To check if your programmer is enabled for the ***'ams – Magnetic Encoder Device library'***, please follow these steps:

- Start EQTools
- Select Programmer → Programmer Info

→ The following screen should now be displayed…



Both the ***ConsoleEDS STANDARD*** and ***PROFESSIONAL*** options must be ENABLED. This allows the programmer to be controlled via the ***ConsoleEDS*** application.

The **'Austriamicrosystems'** library must be ENABLED.

2.0 ams 1-wire UART Programming Interface

2.1 Overview of the ams AS5x6y Bi-directional UART Interface

The programming of an ams AS5x6y device is performed over a proprietary ams ***'single-pin bi-directional UART Interface'***. This interface supports bi-directional communications with the target device (DUT) from the programmer using a single connecting wire. The ***'UART 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 (DUT) into a special ***'communications mode'*** in order to program the device.

Important note:

- A pull-up resistor is required between ***VDD*** and the ***AS5x6y DUT OUT*** pin in order for the ***'communications mode'*** to operate. This resistor is only required during programming.
- The pull-up could be connected within the probe bed of the test fixture or across the programmer ***'Target Vcc'*** and 0V pins.

2.2 Explanation of 'Communications' and 'Functional' modes

The ams AS5x6y devices can operate in one of two modes:

AS5x6y device mode	Control of the AS5x6y OUT pin	When is this mode used?	What is driving the AS5x6y OUT pin
Functional mode	AS5x6y DUT OUT pin will output either an analogue voltage or PWM waveform	During DUT output signal measurement phase	AS5x6y device
Communications Mode	The programmer is controlling / driving the AS5x6y DUT OUT pin	During the DUT programming / configuration process	Programmer

2.2.1. Communications mode

In **'Communications mode'**, the device is waiting for the programmer to send commands to it via the ***1-wire UART*** interface. As this interface is shared with the AS5x6x analogue / PWM ***OUT*** output pin, then the device must be placed into this mode by sending a special sequence of commands after power-up of the sensor.

2.2.2 Functional mode

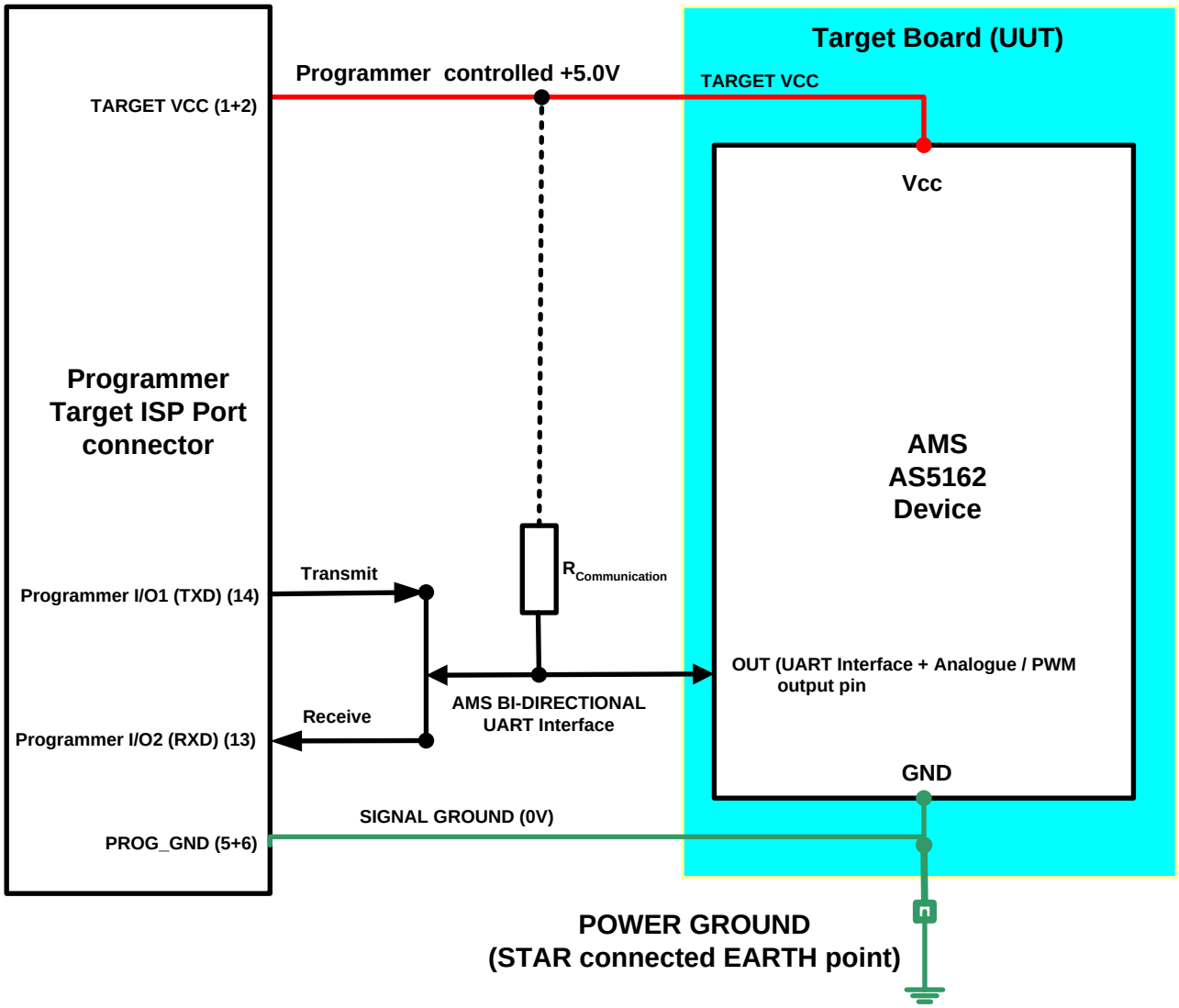
In **'Functional mode'**, the device is now operating in its final real-time operating state and so will output either an analogue voltage or a PWM waveform out of the ***OUT*** pin of the device (dependent on configuration). When the device is in this mode, it will not respond to an external programmer. The

only way to force it back into **'Communications mode'** is to power-cycle the device. This will only work if the device has not already been turned into an OTP device by sending the **'FUSE'** command.

If the device is already in the **'Communications mode'** then it can be forced into the '**Functional mode'** by sending the correct data and then sending the **'PASS2FUNC'** command,

2.3 Connecting a single AS5162 device to a programmer

The diagram below shows the connections required between any Equinox ISP programmer and a Target Board (DUT) for programming a single AMS AS5162 device via the ***'Bi-directional UART interface'***.



The pins required for programming the AS5162 device via the ***'Bi-directional UART interface'*** interface are detailed in section 2.7.

Notes:

- The maximum length of the interconnection cables between the programmer and the DUT depend on the choice of pull-up resistor (***Rcommunication***), communications baud rate and any capacitance or resistance placed on the DUT OUT pin. It is recommended that the wiring should be no more than 15cm.
- A pull-up resistor (***Rcommunication***) from the AS5162 OUT pin to the target VCC is required during the calibration / programming process of device in order for the ***'communications mode'*** to operate correctly.

- This pull-up resistor can be permanently tied to the target VCC on the target board / test fixture or it can be fixed on the 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 ***'Target Vcc'*** supply can be generated / switched from the programmer or generated from an external supply.

2.4 Rcommunication pull-up resistor value

A pull-up resistor (***Rcommunication***) from the AS5x6x OUT pin to the target VCC is required during the calibration / programming process of the device in order for the ***'communications mode'*** to operate correctly.

The value of the pull-up resistor (***Rcommunication***) required for an AS5x6y application depends on the following factors:

- Any capacitance which may already be connected to the OUT pin of the DUT on the Target Board.
- Any resistance which may be already connected to the OUT pin of the DUT on the Target Board.
- Length of connection cable between the programmer and DUT.

Equinox have tested an AS5162 device with a 15cm cable and no additional capacitance or resistance connected to the DUT OUT pin. The results are shown in the table below.

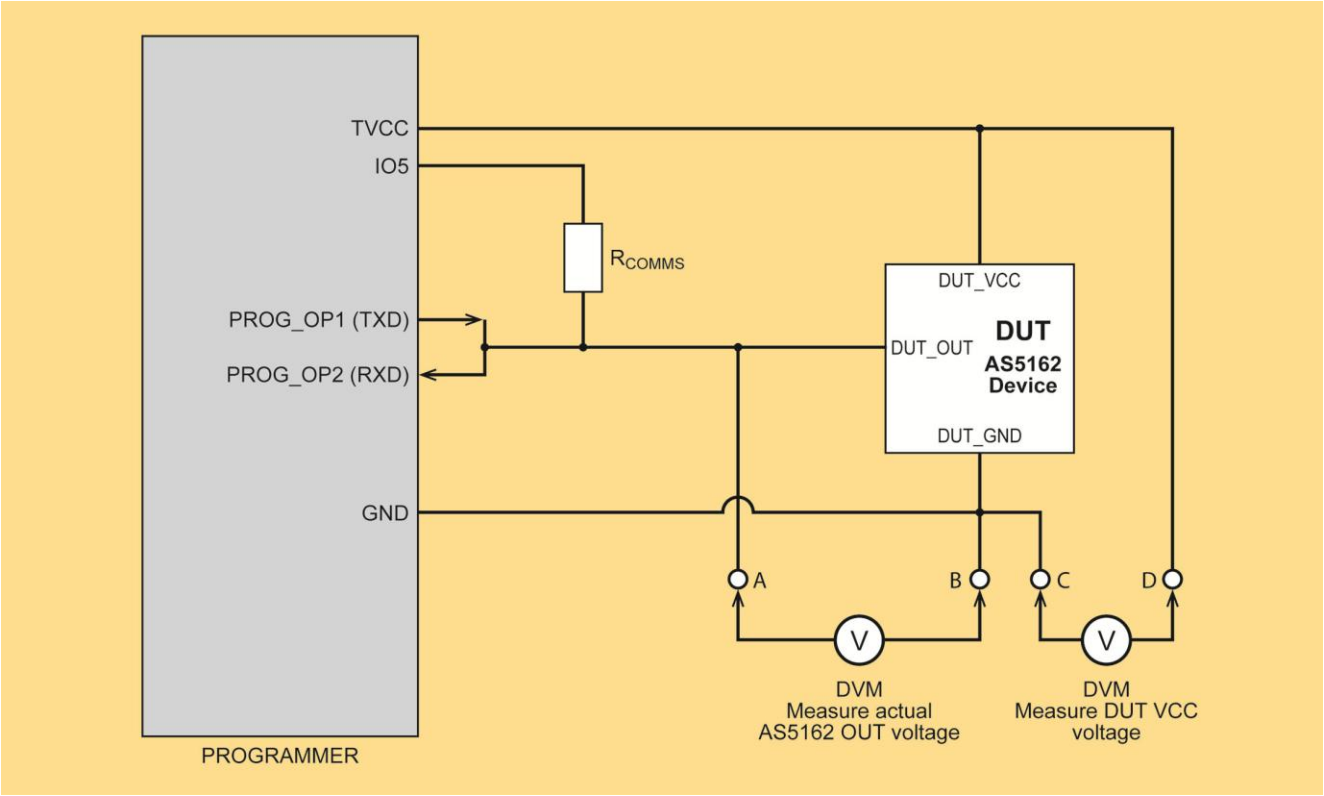
Application	**Cable length**	**Target communications BAUD rate**	**Rcommunications value**
Standard connection with no additional capacitance or resistance	10 – 15 cm	1,200 to 9,600	3.3 k Ohms

Important note:
The final value of the pull-up resistor (***Rcommunication***) depends on many factors and is target board (DUT) specific. Please contact ams or Equinox technical support if you need advice on selecting the correct value for your application.

2.5 Automatically switching the Rcomms resistor via OP5

It is possible to configure the programmer to automatically switch the ***'Communications pull-up resistor (Rcomms)'*** on and off under programmer control. This functionality allows the pull-up resistor to be automatically connected during the programming / calibration process and then automatically disconnected during the functional testing of the DUT when the target AS5x6y device goes into 'functional mode'. This ensures that the ***'Communications pull-up resistor'*** does not load the output of the target AS5x6y device when the device is in ***'functional mode'*** and outputting an analogue voltage.

The diagram below shows the connections required between the ISPnano programmer and a Target Board (DUT) for programming a single ams AS5162 device.



The ***'communications resistor' (Rcomms)'*** is connected between the ***OUT*** pin of the target AS5162 sensor and the programmer ***IO5*** pin.

The switching of the ***'communications resistor' (Rcomms)'*** operates as follows....

- During the programming / calibration process (AS5162 device is in 'communications mode'), the programmer switches the ***IO5*** pin HIGH so the ***Rcomms*** resistor is connected.

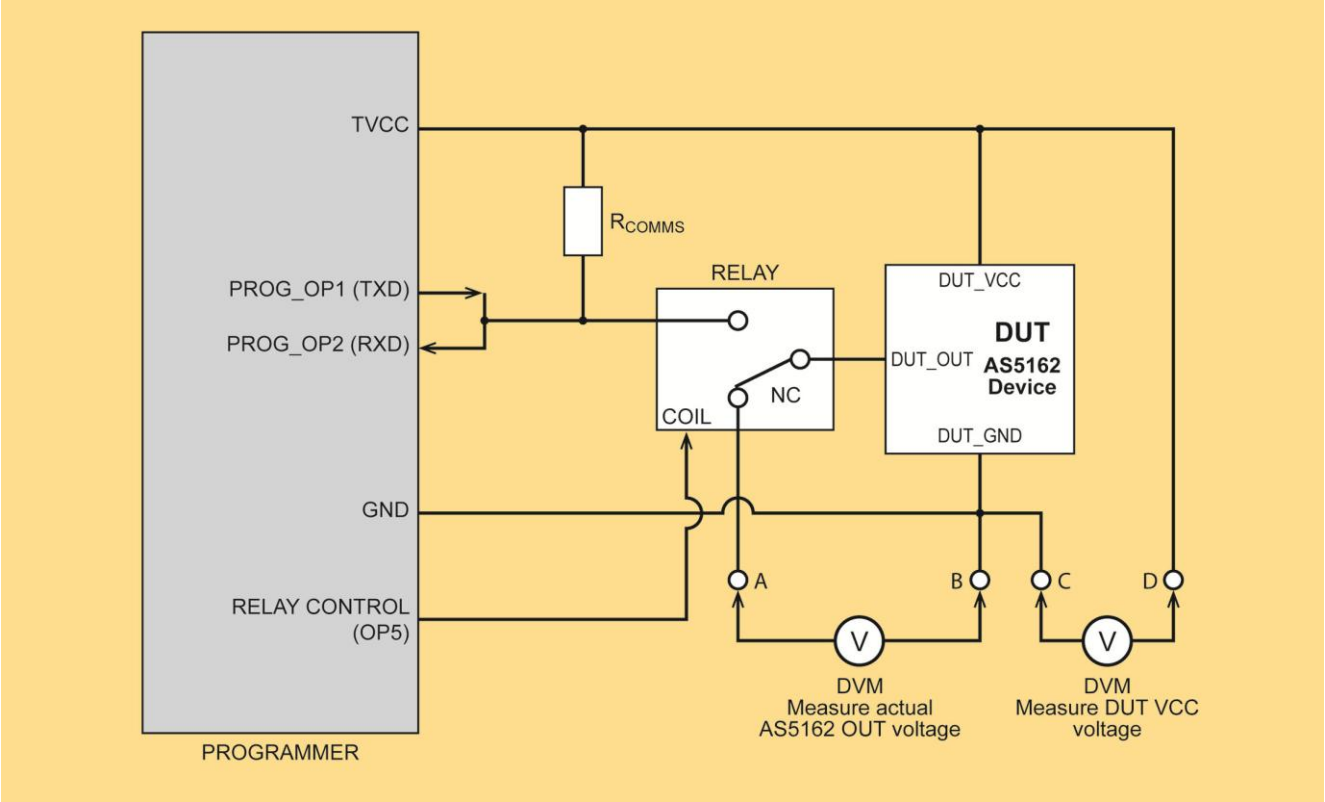
- When the PC control software wants to force the target AS5162 device into ***'functional mode'***, it must send a command to the programmer to switch the programmer ***IO5*** pin to tristate (not driving the pin) which disconnects the ***Rcomms*** resistor.

The table below details how the programmer controls whether the ***'communications resistor' (Rcomms)'*** is connected or not.

AS5x6y device mode	IO5 pin state	Rcomms Resistor	Programmer connected to AS5x6Y OUT pin	What is driving the AS5x6y OUT pin
Functional mode (during DUT output signal measurement)	TRI-STATE	DISCONNECTED	NO	AS5x6y device
Communications Mode (during programming process)	HIGH	CONNECTED	YES	Programmer

2.6 Isolating the DUT OUT pin / Rcomms resistor with a relay

It is possible to completely isolate the ***OUT*** pin of the target AS5x6y sensor device from the programmer by using the circuit detailed in the illustration below.



The ***'communications resistor' (Rcomms)'*** is permanently connected between the programmer ***OP1 (TXD) / OP2 (RXD)*** pins and the programmer ***TVCC*** supply rail. However, the programmer is only connected to the AS5x6y target device when the relay is closed. The relay can be controlled by the programmer ***OP5*** pin or by an external control pin (if required).

The relay isolation of the AS5x6y device operates as follows...

During the programming process.....

- The programmer switches the ***IO5*** pin HIGH which energises the relay which connects the AS5x6y OUT pin to the programmer.
- The ***Rcomms*** resistor is automatically connected at the same time.
- The programmer then sends the relevant command sequence to force the target device into ***'communications mode'***
- When the DUT is in ***'communications mode'***, the programmer is in control of the DUT ***OUT*** pin

During the DUT output measurement phase of the process.....

- When the PC control software wants to force the target AS5x6y device into ***'functional mode'***, it must send a command to the programmer to switch the programmer ***IO5*** pin to TRISTATE.
- This opens the relay contacts so the programmer and ***Rcomms*** resistor are now disconnected (isolated) from the AS5x6y OUT pin.
- It is now possible for an external DVM to measure the voltage output of the DUT OUT pin without any loading effects from the programmer.

The table below details how the programmer controls whether the ***'communications resistor' (Rcomms)'*** is connected or not.

AS5x6y device mode	IO5 pin state	Programmer / Rcomms Resistor	Relay status	What is driving the AS5x6y OUT pin
Functional mode (during DUT output signal measurement)	TRI-STATE	DISCONNECTED	OFF	AS5x6y device
Communications Mode (during programming process)	HIGH	CONNECTED	ON (energised)	Programmer

Please note:
The default state of the relay is OFF which completely isolates the programmer / from the target AS5x6y OUT pin.

2.7 Target communications BAUD Rate (speed)

The programmer communicates with the AS5x6x DUT via a single-wire UART interface. The speed of the communications between the programmer and the DUT is known as the ***'Target communications BAUD rate'***.

Important notes:

- The AS5x6x devices support BAUD rates between 1,200 and 9,600 baud.
- The slower the selected BAUD rate, the more reliable the communications between the programmer and the DUT will be.
- If a long cable is to be used between the programmer and the DUT, then a slower BAUD rate should be selected e.g. 1,200 baud.
- Selecting a slower BAUD rate will usually NOT slow the programming / calibration process down noticeably because of the minimal communications traffic between the programmer and the DUT.

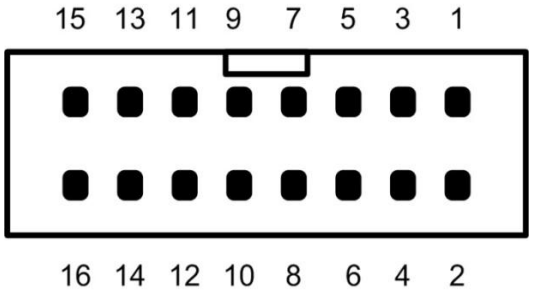
Recommendations:

- If your Target Board has any additional capacitance on the DUT OUT pin, then try using a lower BAUD rate of e.g. 1,200 baud.
- If the cable between the programmer and DUT is greater than 15cm long, then try using a lower BAUD rate of e.g. 1,200 baud.

2.8 Programmer Target ISP Port – ams pin-out

All Equinox ISP programmers feature a 16-way ***'Target ISP Connector'*** port. This connector features all the signals required to implement In-System Programming (ISP) of a target AS5x6x device using the ***'Bi-directional UART interface'***.

The illustration below shows the pin-out of the ***'Target ISP Connector'*** port:



'Target ISP Connector' port

The connector is a 16-pin bump-polarised IDC connector with 0.1" pin spacing.

Pin 1 is the top right pin as shown in the diagram opposite.

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 the AS5162 device 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 Allows analogue output of AS5162 device to be measured.	Input	OUT
GROUND (0V)	5+6	Target / Programmer Signal GROUND	Passive	GND
9	OP6_SPARE	Optional programmable pull-up on the target OUT pin	Output	OUT (optional)
10	Programmer I/O5	Optional programmable pull-up on the target OUT pin	Output	OUT (optional)
13	Programmer I/O2	- Programmer UART RXD (Receive) pin	Input	OUT
14	Programmer I/O1	- Programmer UART TXD (Transmit) pin	Output	OUT

Notes:

- For AS5x6y UART devices, the programmer pins 13 (RXD) + 14 (TXD) must be externally shorted together and then connected to the DUT OUT pin.
- A pull-up resistor is required on the DUT OUT pin otherwise the communication will not work.

2.9 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 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.

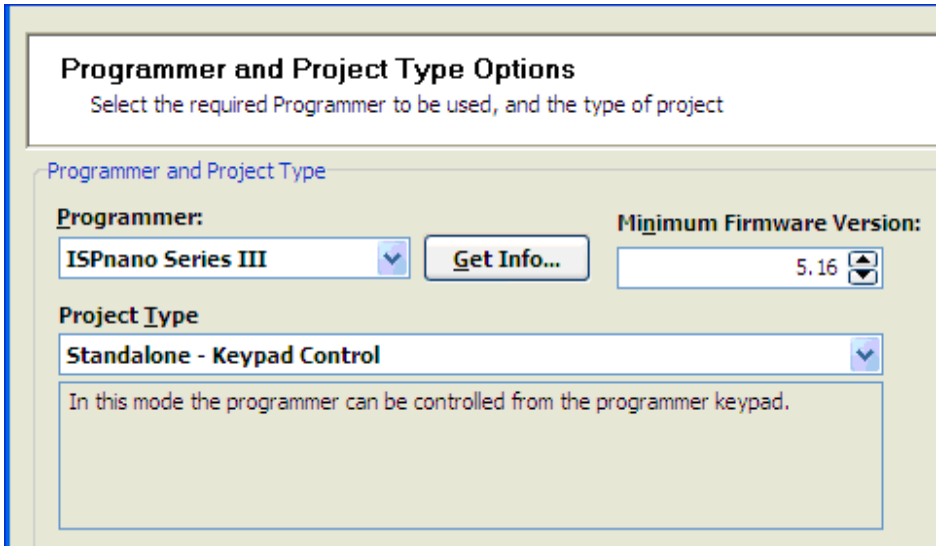
- Select ***the 'Development Project'*** icon and then click the <OK> button

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

3.3 Selecting 'Programmer and Project Type'

This screen allows you to select:

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



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.

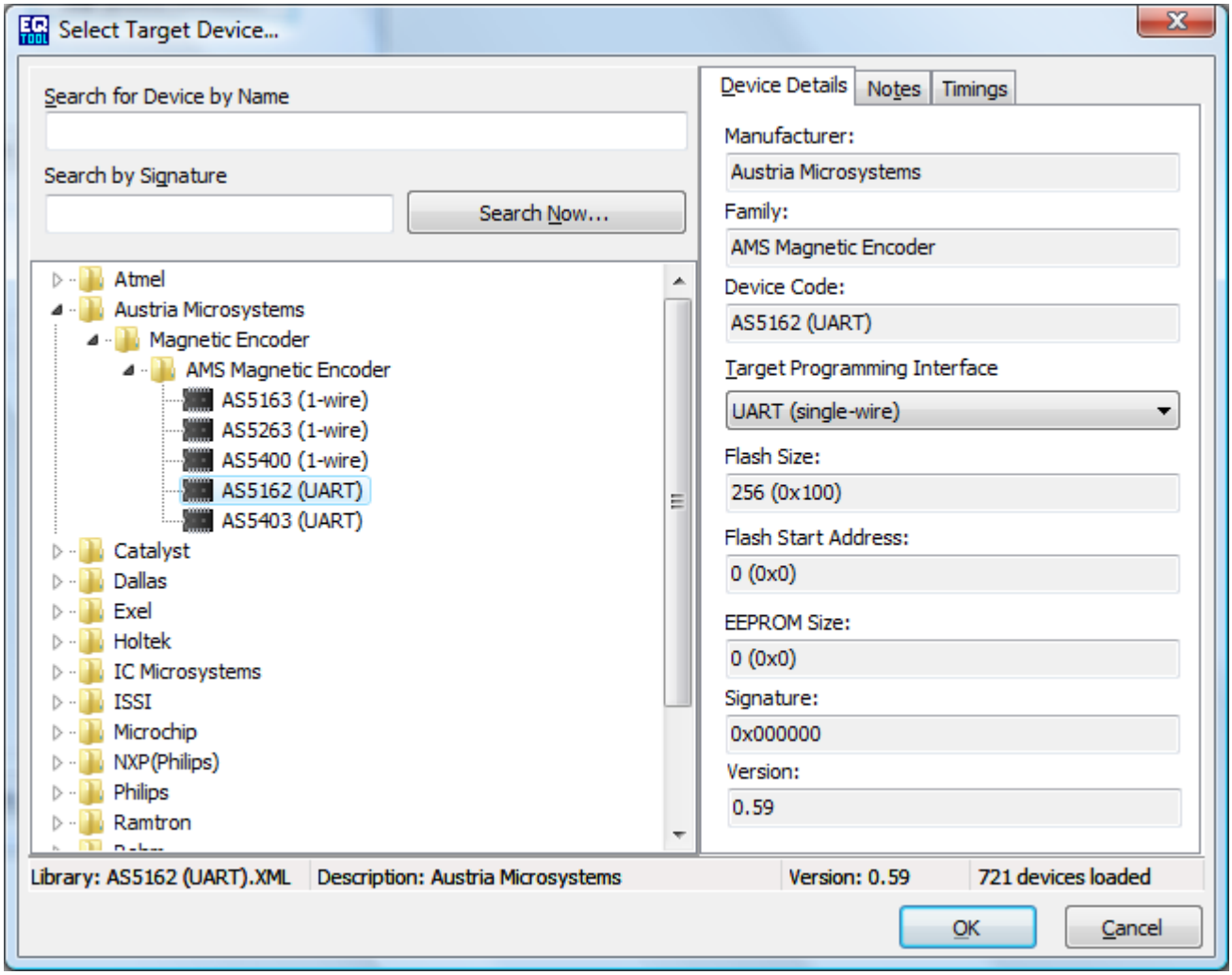
3.4 Selecting 'Target Device'

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. ***AS5162***
- Click the ***<Search now>*** button

→ All instances of the ***AS5162*** device are now displayed

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

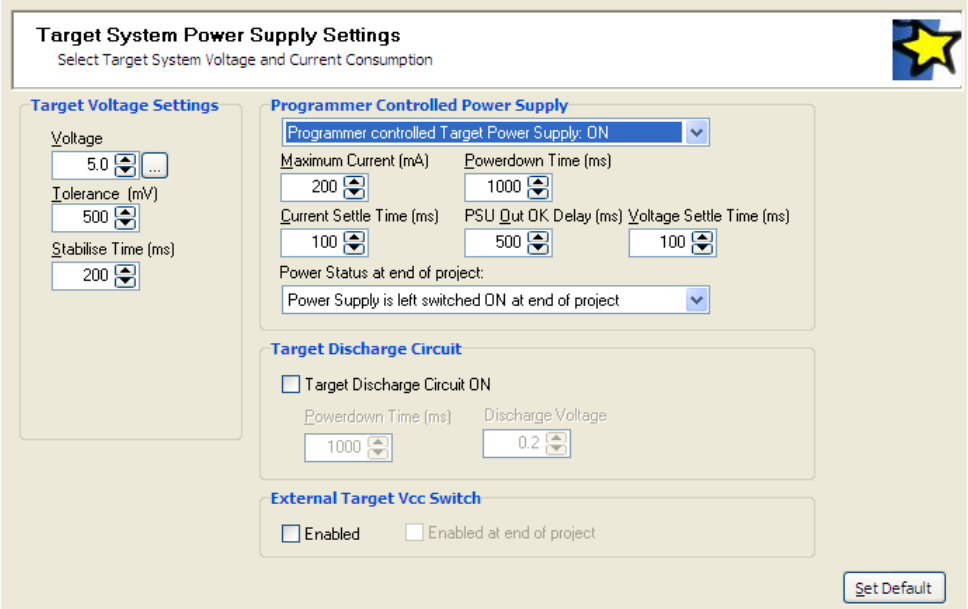


Please note:

- The AS5162 device is programmed using the single-wire UART interface.
- As the programming algorithm is identical for all AS516x devices, the AS5162 algorithm is used to program all devices.

3.5 Setting up the Target Power Supply

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



Power supply recommendations:

1. The programmer should control power to the Target System (DUT)

For most ams sensor applications it is recommended that the programmer should control power to the Target System (DUT). The ISPnano programmer features a very accurate ***'Programmer controlled Target Power Supply'*** which can generate 1.2 to 5.0V @300mA with +/-1% accuracy. This power supply also features pre-switch on load / short-circuit checking and a ***'Target Discharge Circuit'*** which can help to prevent damage to the DUT and programmer in a production environment.

To power the Target System (DUT) from the programmer:

- Set the ***'Programmer controlled Target Power Supply'*** to ***'ON'***
- Set the ***'Voltage'*** to e.g. 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.

2. The programmer should leave power on the DUT at the end of the project

For ams encoder devices, it is very important to keep power applied to the DUT at all times. If the DUT is power cycled for any reason then the programmer will lose synchronisation with the DUT and all user / calibration data resident in the DUT will be lost.

To keep power applied to the DUT after a project has finished:

- Set the ***'Power status at end of project'*** to ***'Power supply is left switched ON at end of project'***.

3. The programmer should discharge the Target System (DUT) after programming

The ISPnano programmer features a ***'Target Discharge Circuit'*** which automatically discharges any residual energy stored in the DUT (e.g. in electrolytic capacitors) at the end of every programmer operation. This can help to prevent ESD damage to the DUT and programmer in a production environment as there is no energy left in the DUT when it is disconnected from the programmer.

To discharge the Target System (DUT) after programming:

- Set the ***'Target Discharge Circuit'*** to ON
- Specify the ***'Discharge voltage'***. A value of 200mV (0.2V) is recommended.
- Specify the ***'Powerdown time'***. A value of 1s (1000ms) is recommended.
- After completing a programming operation, the programmer will switch off the target power supply, switch a resistive load across the target to provide a fast discharge path and then monitor the ***'Target voltage'*** until it has decayed down to the specified ***'Discharge voltage'***. The programmer will then report that the programming operation has completed.

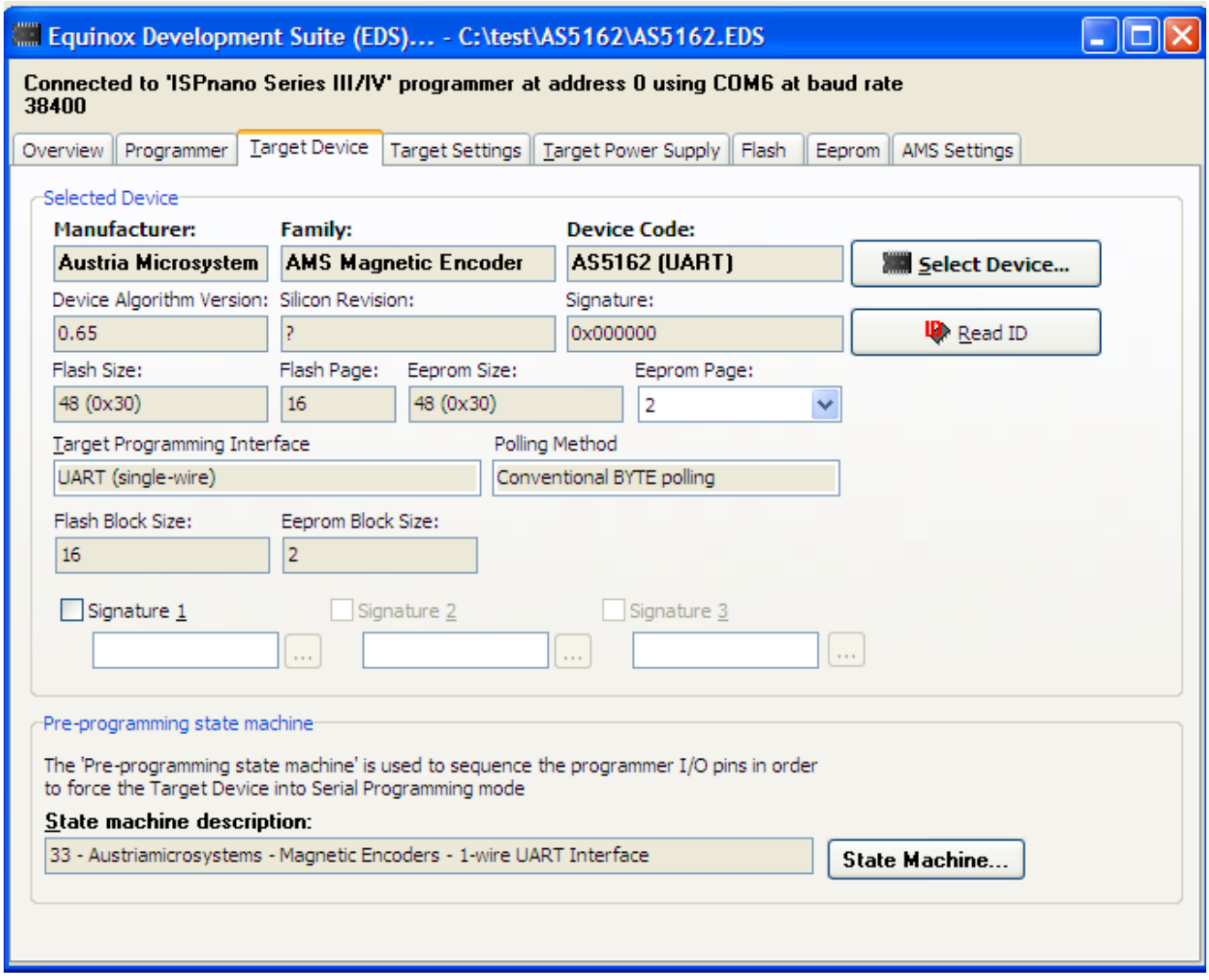
4.0 Testing an ams AS5162 device using EQTools – EDS (Development Mode)

4.1 Overview of EDS

The ***EQTools – EDS*** utility is a powerful ***'Development Mode'*** software utility which can be used to read / write / configure a target device under GUI control on a PC. It is possible to perform basic connectivity and read / write testing for an ams AS5162 device using the Equinox ***EQTools – EDS*** (Development Mode) utility. This ***'Development Mode'*** can be very useful when initially testing an ams DUT as it provides a straightforward method of testing that the programmer can read and write to the DUT without requiring any other software utilities to work.

EDS is an integral part of the Equinox EQTools toolsuite which is installed at the same time as EQTools.

The screenshot below shows a typical EDS session for the AS5162 device…



4.2 EDS – supported functionality for ams AS5x6x devices

At this current time, EDS only partially supports the programming functionality of ams AS516x devices. EDS is useful for initial ams device connectivity checks, read / write operations and reading the ***'Cordic value'***. EDS does not support any more complicated functionality such as device calibration etc.

EDS (Development mode) currently supports the following operations:

- Read / Check Device ID
- Read / Write 128-bit (16-byte) Configuration Memory
- Read / Write 2-byte SFR memory
- Read ***'Cordic Value'*** (single read operation)
- Read ***'Cordic Value'*** (continuous read operation)
- Read ams DUT output voltage (10-bit resolution)

EDS (Development mode) currently does NOT support the following operations:

- Pass2Func (switch to ***'functional mode'***)
- Fuse (permanent programming of OTP memory)
- Calculation of ***'calibration parameters'*** for final device calibration and programming
- Read ***'AGC Value'*** (This can be read indirectly with multiple EDS commands.)

4.3 Setting the 'Target BAUD Rate'

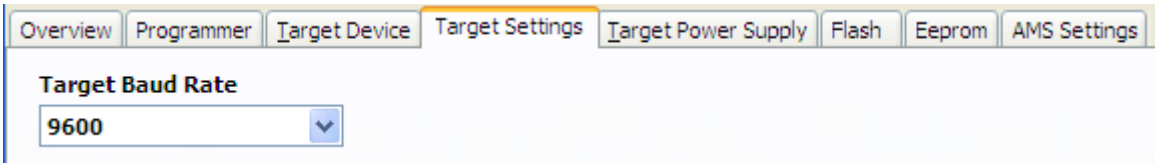
The programmer is capable of communicating with the target AS5x6x device at any BAUD rate between 1,200 and 115,200 baud. However, the actual AS5x6x devices will only work between 1,200 and 9,600 baud. The baud rate is set in EDS to 9,600 by default which should work for most applications where there is no capacitive load on the output pin of the DUT.

If your target board is connected to the programmer via long cables or it has a capacitive load / series resistor on the output pin of the sensor, then this may distort or skew the programming waveforms between the programmer and the AS5x6x DUT. A lower baud rate may be required before reliable communications between the programmer and DUT can be established.

To make communications more reliable, it may be necessary to reduce the baud rate to a lower speed.

Instructions:

- Make sure the programmer is connected to the PC
- Start an EDS session (if not already started)
- Select the ***'Target Settings'*** tab



- Change the ***'Target Baud Rate'*** down to the desired speed e.g. 2,400 baud
- Try any programming operation e.g. ***'Check ID'*** or a ***'FLASH Read'*** to see if the selected baud rate works OK.
- If the programmer fails to enter programming mode or the data read back is incorrect, try reducing the baud rate until a reliable connection is established between the programmer and the DUT.

4.4 Reading the Device ID / Signature

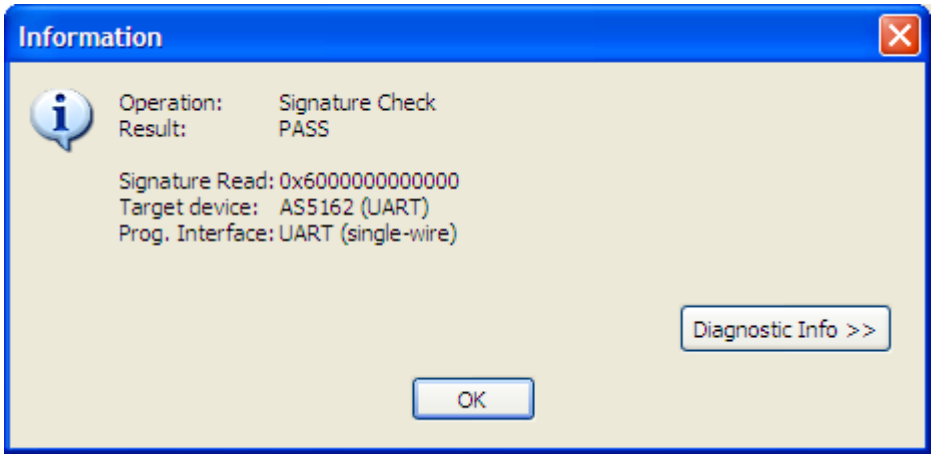
The ***'Read ID'*** function in EDS can be used to check whether the programmer can communicate to the AS5x6x DUT.

Instructions:

- Start an EDS session (if not already started)
- Select the ***'Target Device'*** tab
- Make sure the programmer is connected to the PC
- Make sure the AS5x6x DUT is connected to the programmer
- Click the ***'Read ID'*** button
- The programmer will now attempt to power up the DUT and communicate with the DUT via the 1-wire UART interface at the specified baud rate.

PASS condition

If the programmer is able to communicate with the DUT, then ***'Signature Check'*** operation will report a PASS as follows:



Please note:

- The ***'Signature Read'*** is simply dummy data as the AS5x6x devices do not return a valid ID.
- However, the fact that the ***'Signature Check'*** operation has passed means that the programmer can communicate with the DUT.

FAIL condition

If the programmer fails to communicate with the DUT, then ***'Signature Check'*** operation will report a FAIL.

The failure could be due to many reasons including:

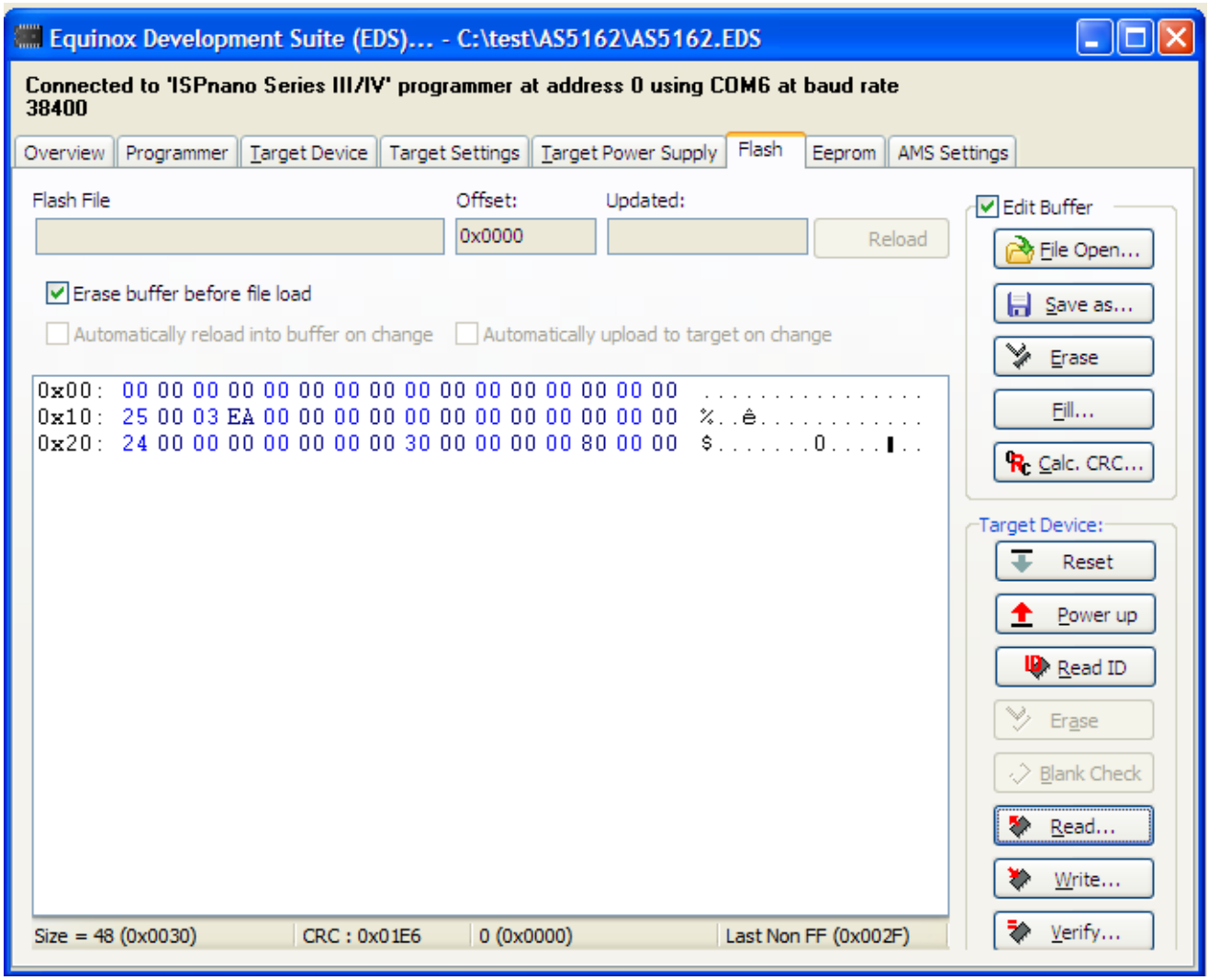
- Incorrect connections to DUT
- DUT not correctly powered
- Interconnect cables between the programmer and DUT are too long
- Incorrect **Rcommunications** pull-up resistor value
- ***'Target baud rate'*** (communications speed between programmer and DUT) is set too high

4.5 Reading / writing the OTP area (16-byte mode)

The ***'OTP area'*** of the AS5x6x device is mapped to the ***'FLASH'*** area in EDS mode.
In this mode, EDS supports reading / writing the AS5x6x device in blocks of 16 bytes at a time. This mode can be used to read / write the entire ***'OTP area'*** in one go. It can also be used to read / write the SFRs in the memory addresses above the ***'OTP area'***.

Instructions:

- Start an EDS session (if not already started)
- Select the ***'FLASH'*** tab
- Click the ***<Read>*** button to read the DUT memory back to the EDS ***'FLASH Buffer'*** area
- If the communications with the DUT is successful, then the read back data should be displayed in the ***'FLASH Buffer'*** - see screenshot below.



- Click the ***<Write>*** button to write the data in the EDS ***'FLASH Buffer'*** area into the target device.

4.6 Reading / writing the SFR area (2-byte mode)

The ***'SFR area'*** of the AS5x6x device is mapped to the ***'EEPROM'*** area in EDS mode.
In this mode, EDS supports reading / writing the AS5x6x device in blocks of 2 bytes (1 WORD) at a time. This mode can be used to read / write the individual 2-byte SFR or to read / write the entire SFR area.

Instructions:

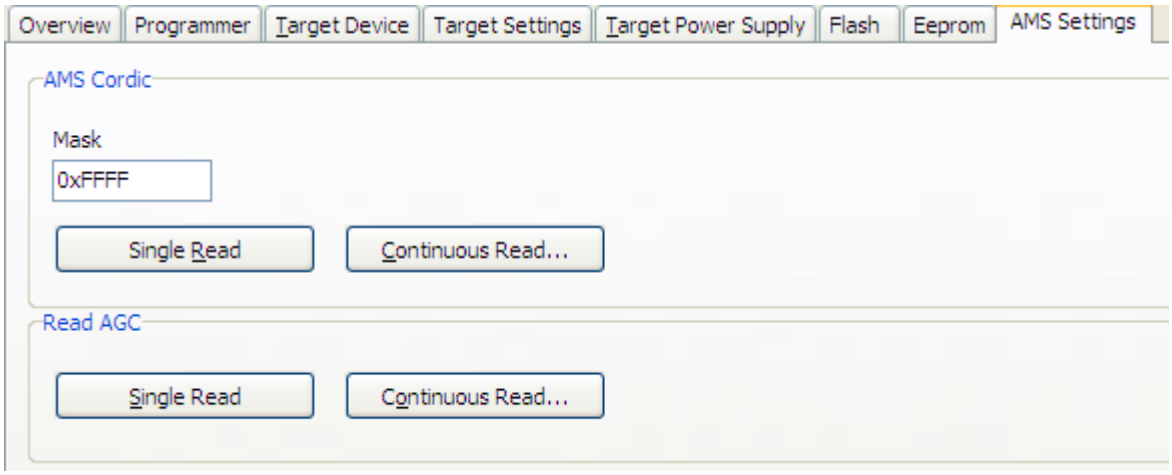
- Start an EDS session (if not already started)
- Select the ***'EEPROM'*** tab
- Click the ***<Read>*** button to read the DUT memory back to the EDS ***'EEPROM Buffer'*** area
- If you want to only read a single 2-byte SFR to the PC, set the ***'Start address'*** to the byte address of the required SFR and then set the number of bytes to read to 2 bytes.
- If the communications with the DUT is successful, then the read back data should be displayed in the ***'EEPROM Buffer'*** at the specified address.

4.7 Reading the Cordic value

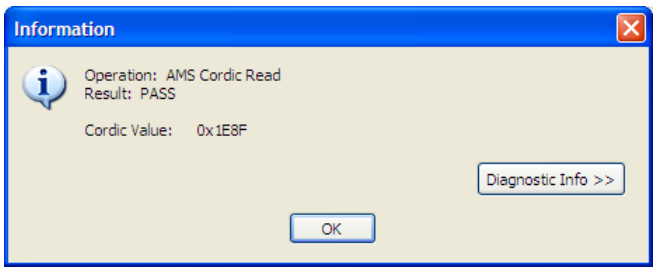
It is possible to read back the ***'Cordic value'*** (angular position) of the AS5x6x DUT using EDS.

Instructions:

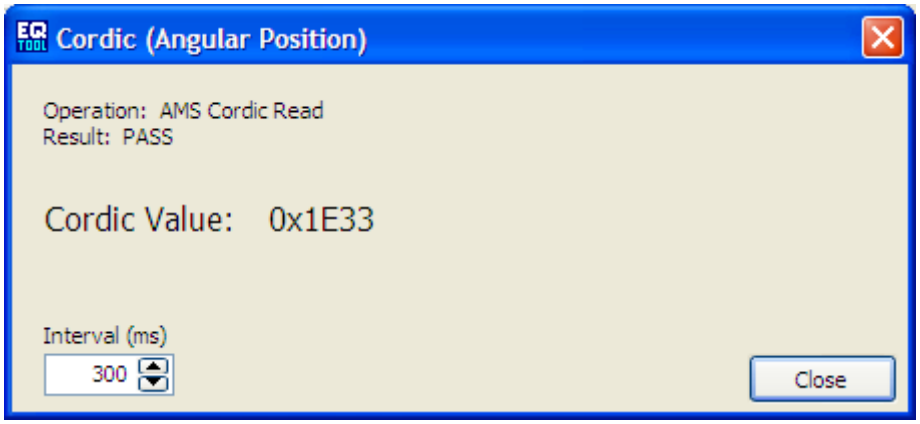
- Start an EDS session (if not already started)
- Select the ***'AMS Settings'*** tab
- The following screen should now be displayed….



- Make sure a suitable magnet is positioned above the AS5x6x DUT
- To make a single reading of the ***'Cordic value'***, press the **<Single Read>** button.



- To continuously read and display the ***'Cordic value'***, press the **<Continuous Read>** button.
- EDS will display the current ***'Cordic value'*** which should change value if the magnet is rotated above the AS5x6x device.



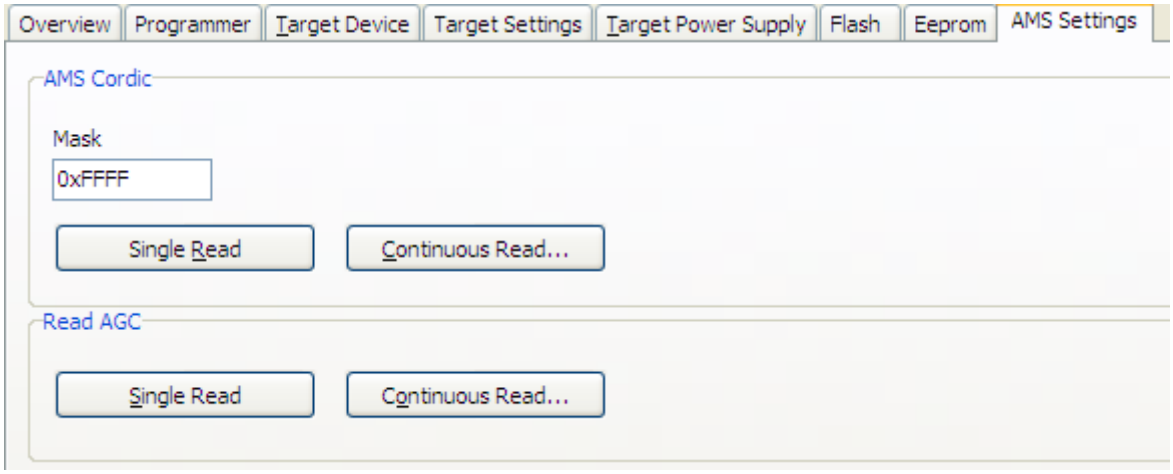
- To filter out noise from the ***'Cordic value'*** reading, it is possible to mask out e.g. the least significant 4 bits of the reading by setting the 'Mask' value to 0xFFF0.
- Altering the ***'Interval'*** parameter will change how often the ***'Cordic value'*** is read from the DUT and displayed.

4.8 Reading the AGC (Gain) value

It is possible to read back the ***'AGC value'*** (gain measurement) of the AS5x6x DUT using EDS. The ***'AGC value'*** gives an indication of whether a suitable magnet is positioned near the AS5x6x DUT and how far away the magnet is from the sensor. This parameter can therefore be used to double-check that a magnet is fitted the correct distance away from the sensor.

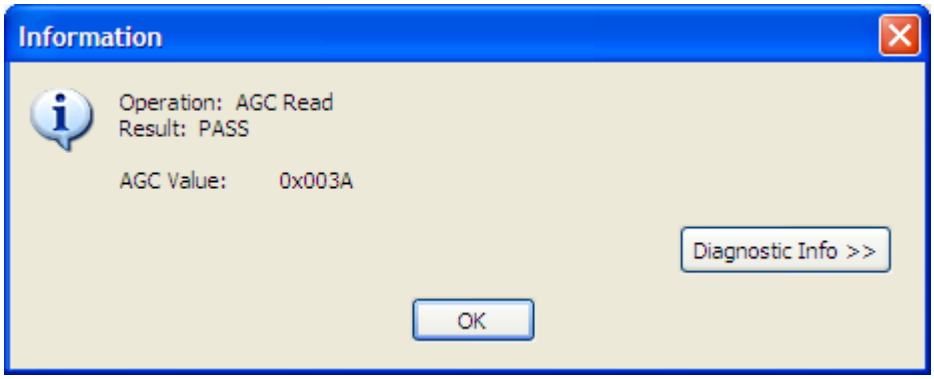
Instructions:

- Start an EDS session (if not already started)
- Select the ***'AMS Settings'*** tab
- The following screen should now be displayed….

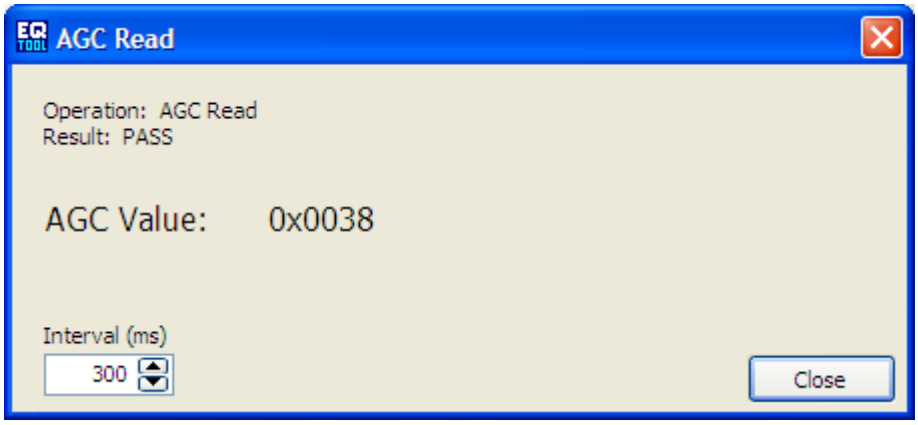


- Make sure a suitable magnet is positioned approximately 1mm above the AS5x6x DUT

- To make a single reading of the ***'AGC value'***, press the **<Single Read>** button.



- To continuously read and display the ***'AGC value'***, press the **<Continuous Read>** button.
- EDS will display the current ***'AGC value'*** which should only change value if the magnet is moved closer or further away from the AS5x6xDUT…



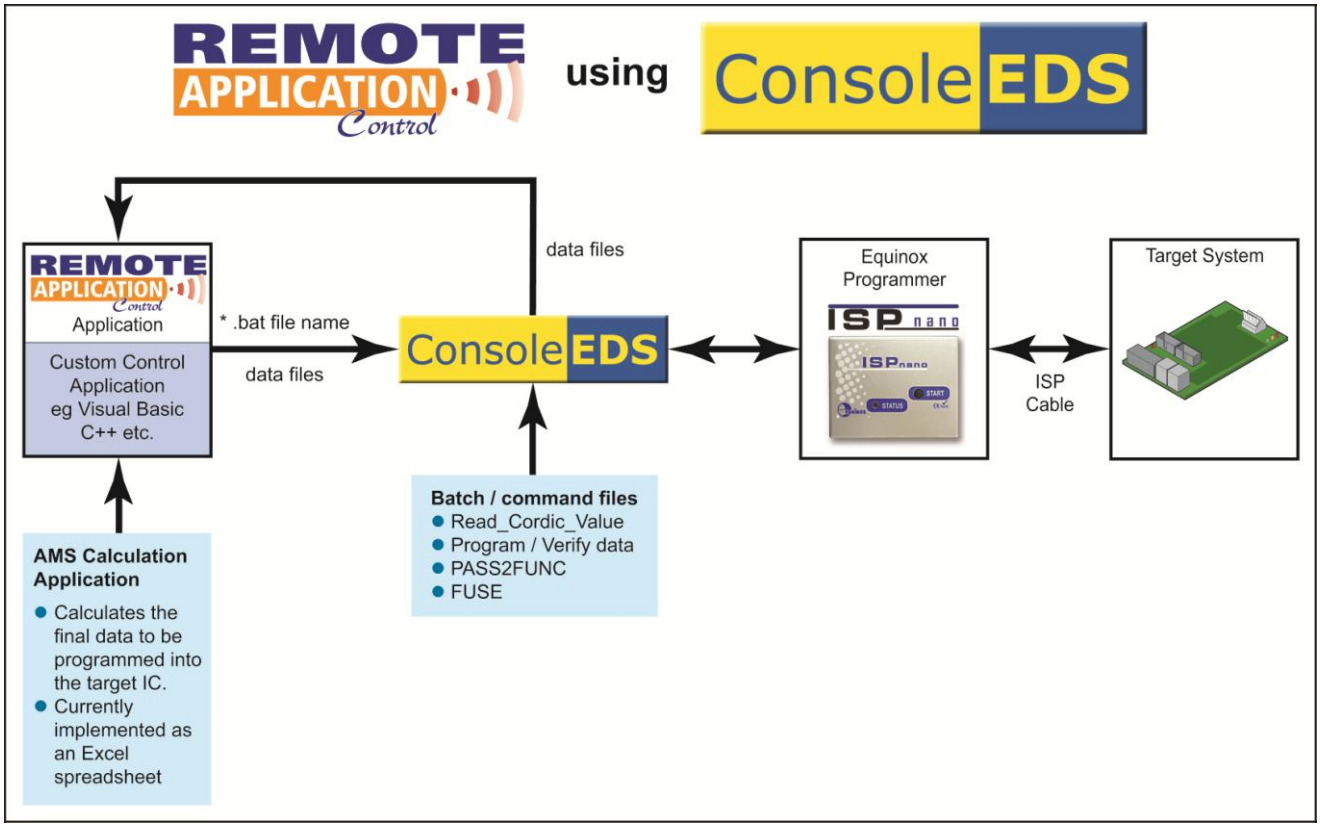
- Altering the ***'Interval'*** parameter will change how often the ***'AGC value'*** is read from the DUT and displayed.

Appendix 1 – Using ConsoleEDS to program a single AMS device

1.0 Overview

This appendix describes how to use the ConsoleEDS utility to calibrate / program a single AMS device. 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.



1.1 AMS AS5162 command set

The table below is taken from the ***'AMS AS5162 datasheet'***. It details all the possible AMS commands which the device supports.

Possible Interface commands	Description	AS5X63 Communication Mode	Address/ Command
WRITE	Write related to the address the user data	SLAVE	0x00-0x0F (OTP) 0x10-0x1F (SFR) 0x20-0xFF (Special Mode)
READ	Read related to the address the user data	SLAVE and MASTER	0x00-0x0F (OTP) 0x10-0x1F (SFR) 0x20-0xFF (Special Mode)
UPLOAD	Transfers the register content into the OTP memory	SLAVE	0x20 + key
DOWNLOAD	Transfers the OTP content to the register content	SLAVE	0x21+ key
FUSE	Command for permanent programming	SLAVE	0x22+ key
PASS2FUNC	Change operation mode from communication to operation	SLAVE	0x23+ key

Table 10 OTP commands and communication interface modes

Important notes:

1. The WRITE / READ commands can operate from address 0x00 to 0x7F so that both the 16-byte SFR area and SFR area can be accessed.
2. The UPLOAD / DOWNLOAD commands are not implemented in this device.
3. The 'Special mode' is not implemented in this device.

1.2 AMS command mapping to ConsoleEDS commands

This section describes how the ***'AMS AS5162 command set'*** has been mapped to the generic ConsoleEDS command set. This allows an AMS AS5162 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 performed automatically by ConsoleEDS as part of other ConsoleEDS commands.

AMS command	Equivalent ConsoleEDS command	Command function
WRITE (block)	***/FLASHWRITE= WriteFile.bin,Start_address,End_ address***	• Writes the data from the specified file to the target device. • This command will write in blocks of 16 bytes. • The Start_Address and End_address should be specified as 0x00 and 0x1F to read the OTP area of the device.
WRITE (WORD)	***/AMSWRITE= ADDRESS,DataLSB, DataMSB***	• Writes 2 bytes of specified data (MSB+LSB) to the specified address.
READ (block)	***/FLASHREAD=ReadFile.bin, Start_address,End_address***	• Reads the data from the target device and writes it to the specified file. • This command will read in blocks of 16 bytes. • The Start_Address and End_address should be specified as 0x00 and 0x1F to read the OTP area of the device.
READ (WORD)	***/EEPROMREADWORD= ByteAddress***	• Reads 2 bytes of data from the specified BYTE address. • This command allows the 16-bit SFR values e.g. AGC value to read out one at a time if required.
FUSE	***/AMSWRITE=16_data_bytes.bin***	• Command to permanently program the configuration data into the target device (OTP) • *** WARNING *** Once this command is sent, the device can then no longer be re-programmed!!! • This command requires programmer firmware 6.65 or above.
PASS2FUNC	***/AMSWRITE=0x00,0x00,0x00,0x0***	• Command to change the operating

	0, 0x00,0x00,0x00,0x000x00,0x00,0x00,0x00,0x000x00,0x00,0x00,0x00	mode of the target device from ***'communications mode'*** to ***'functional mode'***
Read Cordic value	***/AMSREAD=0X22,0X29,0XFFFF, CordicValue.bin***	• This command reads back the 14-bit Cordic value from the device, displays it as 2-byte hex number and also saves it to a file. • This command is a special case of the /AMSREAD command.
BAUDRATE	***/TARGETBAUDRATE=BaudRate***	• This command allows the communications baud rate between the programmer and DUT to be specified on a per ConsoleEDS session basis. The default is 9600 baud.
/SETSTARTUP BAUDRATE	***/SETSTARTUPBAUDRATE=Baud Rate***	• This command allows the communications baud rate between the programmer and DUT to be specified permanently so that it persists for all subsequent ConsoleEDS sessions. The default is 9600 baud.

Notes

- The WRITE and READ commands can operate on both the OTP and SFR memory areas of the device.
- To WRITE or READ a block of data, it is therefore necessary to specify the start and end address of the block. These addresses must land on a 2-byte boundary as the device can only written / read in packets of 2 bytes at a time.

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.
/FLASHVERIFYMASK	This command allows a 'Verify mask' to be applied when verifying a block of data with an area of memory in the target device.
/MEASUREVOLTAGE	This command will measure either the voltage either on the ***'Target Vcc'*** pin or the AMS device ***'OUT'*** pin.
/NODES	This command is used to specify how many programmers are connected to the PC via the RS485 network and at which 'node addresses' these programmers reside. This is a non-persistent command which must be specified on each ConsoleEDS session.
/SETSTARTUPNODES	This command is used to specify how many programmers are connected to the PC via the RS485 network and at which 'node addresses' these programmers reside. This is a persistent command which only needs to specified once and then the settings are permanently stored in the PC registry.
/TARGETIOWRITE=0x????	• This command sets the programmer IO5...IO5 I/O pins to auser-defined state. • It can be used to connect / disconnect the ***Rcomms*** communications resistor connected to the programmer IO5 pin.

1.4 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 (DUT) / target system:

- Target Device e.g. AS5162
- Target Programming Interface e.g. 1-wire UART 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

A suitable ***'Base Project'*** for use with the ams AS5162 device is available from Equinox.
It is called ***AS5162.PRJ***.

1.5 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. ***AS5162***
- 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. ***AS5162.prj***
- You have now created a ***'Base Project'.***

1.6 Applying power to the DUT

It is recommended that whenever possible the programmer should control / supply power to the Target Board (DUT). This allows the programmer to check for short-circuits when powering up of the DUT. Most importantly, it also allows the programmer to power-cycle the DUT after programming calibration data or after the ***'Pass2Func'*** command has been executed. A power-cycle of the DUT is required to force the device from ***'functional mode'*** to ***'communications mode'***.

The following command will power up the UUT at 5.0V but will **NOT** attempt to enter programming mode:
ConsoleEDS AS5162.PRJ /POWERUP=5.0 /NORESET

Important note:
For most applications, it is not necessary to use the **/POWERUP** command. Instead the first ConsoleEDS command will force a power-up condition.

1.7 Setting the 'Target Baud Rate'

It is possible to set the ***'Target Baud rate'*** (communications speed between the programmer and DUT) on a one-off basis at the start of the programming / calibration process.

To permanently set the ***'Target Baud rate'***, the following command should be used:
ConsoleEDS AS5162.PRJ /SETSTARTUPBAUDRATE=BaudRate

Where the ***BaudRate*** parameter can be specified as 1200, 2400, 4800 or 9600 baud.
Any higher baud rates will not work with an AS5x6x device.

ConsoleEDS will remember this ***'Baud rate'*** setting (persistent setting) and all subsequent ConsoleEDS sessions will then use the specified baud rate.

1.8 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 different data depending on which ams chip is being used.

Typical command useage:
ConsoleEDS AS5162.PRJ /READSIG /NORESET

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.

1.9 Reading the 'Cordic value' from a target device

The ***'Cordic value'*** is the digital representation of the angle of the hall sensor inside the AS5x6x device. It is a 14-bit value and is stored in register address 0x29 – 0x2A. The top 2 bits of register 0x2A always read back as '0' so the absolute ***'Cordic value'*** is left in the lower 14-bits of the 2-byte register.

Read/Write area	DATA1	0x29	0	CORDICOUT<0>	0	Cordic Output
			1	CORDICOUT<1>	0	
			2	CORDICOUT<2>	0	
			3	CORDICOUT<3>	0	
			4	CORDICOUT<4>	0	
			5	CORDICOUT<5>	0	
			6	CORDICOUT<6>	0	
			7	CORDICOUT<7>	0	
	DATA2	0x2A	0	CORDICOUT<8>	0	
			1	CORDICOUT<9>	0	
			2	CORDICOUT<10>	0	
			3	CORDICOUT<11>	0	
			4	CORDICOUT<12>	0	
			5	CORDICOUT<13>	0	
			6	Not used	0	a read command returns all data bits at 0
			7	Not used	0	

The ***'Cordic value'*** can be read back from the target AS5x6x devices using the following command:
ConsoleEDS AS5162.PRJ /AMSREAD=0X22,0X29,0XFFFF,CordicValue.bin /NORESET

Where:

- 0x22 is the address of the DSPRN bit (bit 7).
- 0x29 is the BYTE address of the ***'Cordic Output'*** register (16 bits with top 2 bits always reading back as 0)
- 0xFFFF is the bit-mask used to mask out any unwanted bits before saving the result in the specified file.
- ***Cordic.bin*** is the name of the file into which the 2-byte Cordic value is saved.

Here is the debug information from ConsoleEDS when this command is executed.....

Console EDS - Version 4.0 (Build 3127)
00:01.828 Console EDS - AMS Read Value of **0x3B17**
00:01.828 Console EDS - AMS Apply Mask of 0xFFFF gives **0x3B17**
00:01.828 Console EDS - AMS Read, write to file 'CORDICVALUE.BIN'

Result:

- The ***'Cordic value'*** is read back and displayed as: **0x3B17**
- The ***'Cordic value'*** is also written to a 2-byte file called ***'CordicValue.bin'***.

Important note:

In order to read the ***'Cordic value'***, the programmer automatically enables the DSP by writing 0x80 to Address 22 (and 0x00 to address 23) when this command sequence is sent from ConsoleEDS.

1.10 Reading the 'AGC (gain) value' from a target device

The ***'AGC value'*** is a simple method of establishing whether a magnet is applied to the DUT and approximately how far this magnet is away from the sensor. It can be used as an indication / double-check to make sure the magnet is fitted and is the correct distance from the sensor.

The command sequence to read the ***'AGC value'*** is as follows:

- Write 0x0008 to address 0x22 (which sets DSPRN=1 and enables the DSP. Remember it's a 2 byte write.)
- Read the ***'AGC value'*** from address 0x2C --> returns 2 bytes. The AGC value is in the lower byte. Write 0x0000 to address 0x22 (which sets DSPRN=0 and disables the DSP. Remember it's a 2 byte write.)

To read the ***'AGC value'*** using ConsoleEDS, the following command sequence should be executed:

ConsoleEDS AS5162.PRJ /EEPROMWRITEWORD=0x0080,0x22 /EEPROMREADWORD=0x2C /EEPROMWRITEWORD=0x0000,0x22 /NORESET

This command sequence will return 2 bytes of data. The lower byte is the ***'AGC value'***.

1.11 Reading the entire device OTP area to a file

The AS5162 ***'OTP area'*** is located from address 0x00 to 0x0F and is 16 bytes in size.

The following command will read back the contents of the OTP area to a file:
ConsoleEDS AS5162.PRJ /FLASHREAD=ReadFile.bin,0x00,0x0F

Note:
Because the ***'OTP area'*** is mapped as 48 bytes, it is necessary to specify:

- Start address: 0x00
- End address: 0x0F

This will then read back only 16 bytes in one 16 byte block.

1.12 Writing the entire device OTP area from a file

The AS5162 ***'OTP area'*** is located from address 0x00 to 0x0F and is 16 bytes in size. The top address range from 0x0D bit 3 up to 0x0F contains ***'AMS Factory Data'*** which means that this area cannot be written by the programmer.

To write data to the ***'OTP area'*** it is necessary to perform the following steps:

- Read back the entire 16 bytes of the ***'OTP area'*** to a file
- Mask out the top bytes from address 0x0D bit 3 up to 0x0F.
- This leaves just the ***'Customer data'***
- Insert the actual data to be programmed into the file
- Write the file into the OTP area
- Perform a verify of the ***'OTP area'*** but mask out the ***'AMS Factory Data'***

This sequence can be achieved in ConsoleEDS as follows:

ConsoleEDS AS5162.PRJ
/FLASHWRITE=WriteData.bin,0x00,0x1F
/FLASHVERIFYMASK=0x00000FFFFFFFFFFFFFFFFFFFFFFFFF
/FLASHVERIFY=WriteData.bin,0x00,0x1F
/NORESET

Where:

- ***AS5162.PRJ*** is the ***'Base Project'***
- ***WRITE_DATA.BIN*** is a 16-byte long binary file containing the ***'configuration data'*** to be programmed.
- The ***/FLASHVERIFYMASK*** is a data mask used mask out the ***'AMS Factory Data'.***
- This example will program the internal RAM area of the device with the contents of the binary file.

1.13 Reading a 2-byte register (SFR) from a target device

The AS5162 features a number of 2-byte ***'Special Function Registers' (SFRs)*** which allow functions such as the AGC to be read back from the target device. The ***'SFR area'*** is mapped to address range 0x10 – 0x2F.

It is possible to read back any of the 2-byte SFR registers in the AS5162 device using the ***/EEPROMREADWORD*** command.

The EEPROM commands operate in 2-byte chunks so they will always read or write 2 bytes.
So to read back a 2-byte SFR value from an AS5162 device, you would use the following command…

ConsoleEDS AS5162.PRJ /EEPROMREADWORD=ByteAddress

Example:
To read back the AGC value from the AS5162, a read of byte address 0x2D is required.

The read of the AGC SFR is implemented in ConsoleEDS as follows:
ConsoleEDS AS5162.PRJ /EEPROMREADWORD=0x2D /NORESET

Where 0x2D is the BYTE address of the SFR to be read back.

The programmer must always read back 2 bytes as the underlying device protocol only supports reading back of 2 bytes at a time.

ConsoleEDS will output the following debug text:

```
00:01.578 Console EDS - Read Target EEPROM
00:01.578 Console EDS - Reading from address 0x00002D, 2 bytes
00:01.578 Console EDS - MISP_EEPROM_READ
00:01.625 Console EDS - Read back 0x00FF
```

So the read back AGC value is contained in the lower byte of the data read back 0x00FF so the AGC value is 0xFF. (The upper byte at address 0x2E is not used and can be ignored.)

Please note:

- The ‘read back’ SFR value is displayed a 2-byte hex number.
- In this example the lower byte corresponds to address 0x2D which is the AGC value and 0x2E is 0x00.

1.14 Changing to ‘functional mode’ using Pass2Func command

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 AS5162.PRJ /AMSWRITE=0x00,0x00,0x00,0x00,
0x00,0x00,0x00,0x000x00,0x00,0x00,0x000x00,0x00,0x00,0x00

(It is necessary to specify 16 x 0x00 on the command line or to specify a binary file which contains 16 zeros.)

→ When this command is executed, the programmer sends the ***‘PASS2FUNC’*** command to the target device.
→ The target device should now go into ***‘functional mode’*** and start outputting a voltage or a PWM signal on the VOUT pin.
→ The voltage / PWM pulse ratio on the Vout pin of the target device should then change as the magnet is rotated. The exact behaviour depends on how the device has been set up.

Important note:
The programming of the calibration settings using the ***‘PASS2FUNC’*** command are **NOT** permanent. It is possible to reset the device back to factory default values by simply power cycling the device. All calibration settings will then be lost.

1.15 Automatically disconnecting the Rcomms resistor

If you want to change the AS5162 DUT from ***'communications mode'*** to ***'functional mode'***, then it is also necessary to send a command to the programmer to tell it to switch programmer output OP5 to TRISTATE which will disconnect / disable the ***'communications resistor'*** from the DUT OUT pin.

To instruct the programmer to switch off / isolate the ***'communications resistor'***, the new ***TargetIOWrite*** method should be used.

ConsoleEDS AS5162.PRJ /TARGETIOWRITE=0x0DC0;

The value to send to set programmer output OP5 (and all other programmers outputs) to TRISTATE is: **0x0DC0**

For this scheme to work, the programmer must be already either in ***'programming / communications mode'*** or the 'Target controlled programmer power supply' must be ON. The programmer will switch OP5 to TRISTATE which should either disconnect the pull-up from OP5 or switch the relay isolating the DUT output pin, depending on which hardware configuration you are using to switch the ***'communications resistor'***. This command would normally be sent

immediately after the ***'Pass2Func'*** command so the DUT power stays on, but the ***'communications resistor'*** is disconnected.

1.16 Measuring the DUT OUT pin voltage

Once the target device is in ***'functional mode'***, it is possible to measure the voltage at the OUT pin of the device. 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

1.17 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 to power cycle the device.

The following command is used to switch power off to the UUT:
ConsoleEDS AS5162.PRJ /RESET

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

1.17 FUSE command (permanently OTP programming)

1.17.1 Sending the FUSE command

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 AS5162.PRJ /AMSWRITE=16_byte_data_file.bin /NORESET

where:
16_byte_data_file.bin is a binary file containing the 16-bytes of ***'calibration data'*** you want to permanently program into the device.

→ This command executes a complicated sequence of write and verify commands to the target device in order to permanently write the data into the device memory cells.
→ If the OTP programming / verify sequence is successful, then the programmer will automatically 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. The device is now permanently programmed!

1.17.2 Explanation of how FUSE command works

To send the ***'FUSE'*** command, the following ConsoleEDS command sequence is required:

ConsoleEDS AS5162.PRJ /FASTSTART /AMSWRITE=16_byte_data_file.bin /NORESET

where
- ***16_byte_data_file.bin*** is the 16-byte data file to be written to the device.
- For this example, the ***'Customer lock bit'*** in the data sent is set to 0, so the OTP fuses are not burned.

The programmer then executes the following sequence:

1. At the start of target programming and powering up the target device it is initialised to 0xFF

2. At the start of the ***/AMSWRITE*** command the diagnostic Byte DebugData[24] is set to 0x00

3. First Fuse write / verify operation

- The programmer writes the 14 Bytes of data and then writes the Fuse Write command.
- The programmer reads the 14 Bytes of data from the target device. If the physical data read is succesful DebugData[24] =0x01
- The programmer then verifies the 14 Bytes of data read back are the same as that written if OK DebugData[24] = 0x02

4. The R1K Tests are then performed

- The 1st R1K_R10K is written with a data value of 0x0060 and verified OK DebugData[24] = 0x03

- The 2nd R1K_R10K is written with a data value of 0x0060 and verified OK DebugData[24] = 0x04

5. Second read back and verify operation

- The programmer then again reads the 14 Bytes of data from the target device. If the physical data read is succesful DebugData[24] = 0x05
- The programmer then verifies for the 2nd time the 14 Bytes of data are the same as that written if OK DebugData[24] = 0x06

6. The R10K Tests are then performed

- The 1st R1K_R10K is written with a data value of 0x0040 and verified OK DebugData[24] = 0x07
- The 2nd R1K_R10K is written with a data value of 0x0040 and verified OK DebugData[24] = 0x08

7. Second read back and verify operation

- The programmer again reads the 14 Bytes of data from the target device. If the physical data read is succesful DebugData[24] = 0x09
- The programmer then verifies for the 3rd time the 14 Bytes of data are the same as that written if OK DebugData[24] = 0x0A == Decimal 10

8. ConsoleEDS will then return a PASS or FAIL result

1.17.3 How to check the FUSE command has executed correctly

It is possible to read back the ***'AMS Debug Byte'*** from the programmer after executing the ***/AMSWRITE*** command by using the ***/READDIAGNOSTICS*** command.

ConsoleEDS /READDIAGNOSTICS

The programmer will then return 55 ***'debug bytes'*** in ASCII format separated by square brackets.

00:00.640 Diagnostics:
00:00.640 55 diagnostic bytes returned
[0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x00][0x1C][0x01][0xFF][0x00][0x06][0x00][0x00][0x00][0x00][0x00][0x00][0x00] **[0x0A]**
[0xFF][0xFF][0xFF][0xFF][0xFF][0xFF][0xFF][0xFF][0xFF][0xFF][0xFF][0xFF][0xFF][0x60][0x0D][0xC0][0x0D][0xC0][0x00][0x02][0x00][0x18][0x02][0x00][0x08][0x00][0x00][0x00][0x00]

The 25th byte in this list is the ***'AMS Debug Byte'*** ie DebugData[24] in the array of 55 bytes.
The byte value should be **0x0A (decimal 10)** if the entire ***'FUSE'*** sequence executed OK.

So looking at the data returned from the programmer in the example above, we can see that the value returned was **[0x0A]** so the ***'FUSE'*** sequence worked OK.

1.18 Optimising ConsoleEDS operations

1.18.1 Commands to execute once at startup

The following ConsoleEDS commands should be executed **once only** in order to setup the programmer ready to program a batch of products. The commands could be placed in a separate batch file which is only executed once before the main programming operation commences.

ConsoleEDS /BAUDRATE=BaudRate
ConsoleEDS AS5162.PRJ /SETSTARTUPBAUDRATE=BaudRate
ConsoleEDS /SETSTARTUPUSECACHE

Please see the points 1 – 3 below for a full explanation of each command.

1. Set up the PC to programmer Baud Rate

This command permanently sets up the ***'Baud Rate'*** (communications speed) between the PC and the programmer. The default baud rate is 38400 but this can be increased to 115200 for RS232 COM port connections and 230400 for USB connections. Increasing the baud rate can speed up programming interactions if there is a lot of data going between the programmer and PC.

The following command will change the baud rate used by ConsoleEDS to communicate with the programmer:

ConsoleEDS /BAUDRATE=BaudRate

Where

- ***BaudRate*** parameter can be specified as e.g. 38400, 115200, 230400

Please note:
For ams device programming / calibration it is recommended to keep the baud rate at 38,400. Using a faster baud rate does not speed up the overall ConsoleEDS operation due to the small amounts of data being sent.

2. Set up the *'Target Baud Rate'*

This command permanently sets up the ***'Target Baud Rate'*** between the programmer and the AS5x6x DUT.

ConsoleEDS AS5162.PRJ /SETSTARTUPBAUDRATE=BaudRate

Where

- ***BaudRate*** parameter can be specified as 1200, 2400, 4800 or 9600 baud.
- Any higher baud rates will not work with an AS5x6x device.
- ConsoleEDS will remember this ***'Baud rate'*** setting (persistent setting) and all subsequent ConsoleEDS sessions will then use the specified baud rate.

3. Use Programmer license caching

This command switches on programmer license caching which will dramatically speed up the operation of ConsoleEDS.

ConsoleEDS /SETSTARTUPUSECACHE

1.18.2 Commands to execute for every ConsoleEDS session

The following ConsoleEDS command should be executed at the start of every ConsoleEDS session.

1. Forcing the programmer not to re-enter programming mode

As ams devices lose all their programmed settings as soon as power is removed from the DUT, it is therefore important to keep the DUT powered up and in ***'programming mode'***. The ***'Base Project'*** must be set to ***'Keep target powered up at end'*** and ConsoleEDS must be told on a per session basis not to re-enter programming mode every time.

This is achieved using the ***/FASTSTART*** command which needs to be specified at the start of each ConsoleEDS command line as follows….

ConsoleEDS AS5162.PRJ /FASTSTART /ProgrammingAction1 / ProgrammingAction2

If the ***/FASTSTART*** command is not used, then the programmer will exit programming mode and power down the DUT at the start of each ConsoleEDS session which will add e.g. 400 – 500ms execution time per ConsoleEDS session.

1.18.3 Optimising ConsoleEDS PASS / FAIL detection

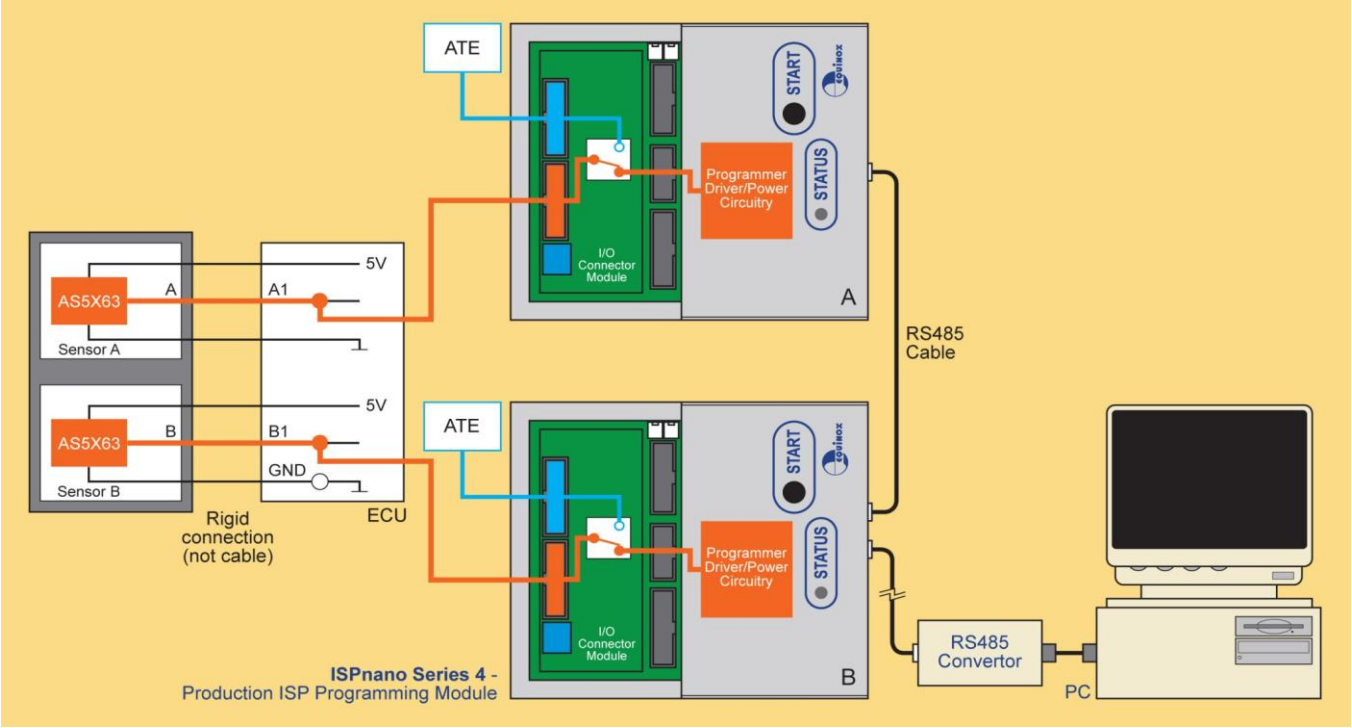
When a ***'Remote Application'*** executes a ConsoleEDS console session, ConsoleEDS will execute and then return an ***'Error code'*** to the calling application. This ***'Error code'*** is returned via Windows as part of the operation of calling a console application. It is much quicker for the ***'Remote Application'*** to check the returned ***'Error code'*** than it is to manually parse through the re-directed ConsoleEDS debug text to find the PASS / FAIL result.

For full instructions on how to handle the ConsoleEDS ***'Error code'***, please refer to the ConsoleEDS application note AN111.

Appendix 2 – Using ConsoleEDS to program two ams devices concurrently

1.0 Overview

This appendix describes how to use the ConsoleEDS utility to calibrate / program both sensor devices inside an AS5262 concurrently. In this scenario, two independent ISPnano programmers are used – one to program ***#DeviceA*** and the other to program ***#DeviceB***. The illustration below shows how the two programmers are connected to the two target devices.



The two programmers are connected via an RS485 network to an ***'RS485 to RS232 converter'*** which allows both programmers to be controlled from a single PC RS232 COM port.

2.0 Equipment required

The following equipment is required for this scenario:

- 1 x AS5262 – dual sensor device
- 2 x ISPnano Series 3 or Series 4 programmer
- 1 x Equinox RS485 to RS232 converter
- 1 x ISPnano RS485 interconnect cable

3.0 Hardware setup

Programmer communications setup….

- Connect an RS232 serial cable between the ***'RS485 Converter'*** and a spare COM port on the PC
- Connect an RS485 network cable between the ***'RS485 Converter'*** and ***Programmer#A***
- Connect an RS485 network cable between ***Programmer#A*** and ***Programmer#B***

Programmer to target device setup…

- Connect ***Programmer#A*** to DUT #A
- Connect ***Programmer#B*** to DUT #B

4.0 Selecting AMS device A or B in ConsoleEDS

It is possible to select which AMS device to program inside an AS5162 IC by selecting either ***'Programmer#A'*** or ***'Programmer#B'***. The programmers must be at e.g. node address #0 and node address #1.

Here is how you select an individual programmer on the RS485 network….

i. To select ***'Programmer#A'*** at address #0, use the following ConsoleEDS command….
/SETSTARTUPNODES=1,0

ii. To select ***'Programmer#B'*** at address #1, use the following ConsoleEDS command….
/SETSTARTUPNODES=1,1

This setting is stored in the PC registry so you only need to execute this command once before you execute any other command. All commands executed after this command will automatically used the programmer(s) which was specified with the ***/SETSTARTUPNODES*** command.

How is the device selection integrated into the Control Application?

In the control application, you simply need to have a ***'list box'*** which allows selection of:

- Device#A
- Device#B

When the user makes / changes the selection, the control application simply needs to send the the ***/SETSTARTUPNODES*** command once and then all following commands will use the specified programmer node.

This technique means that the batch files can remain identical for either Device#A or Device#B.

Appendix 4 - ActiveX control implementation

1.0 Overview

This appendix describes how to use the Equinox ***'ActiveX control'*** to control an Equinox programmer and to calibrate / program both sensor devices inside an ams AS5x6x device..

1.1 Software modules and licenses required

In order to use the Equinox ***'ActiveX control'*** to control a programmer, it is necessary to implement the following 'software modules' and 'programmer licenses.

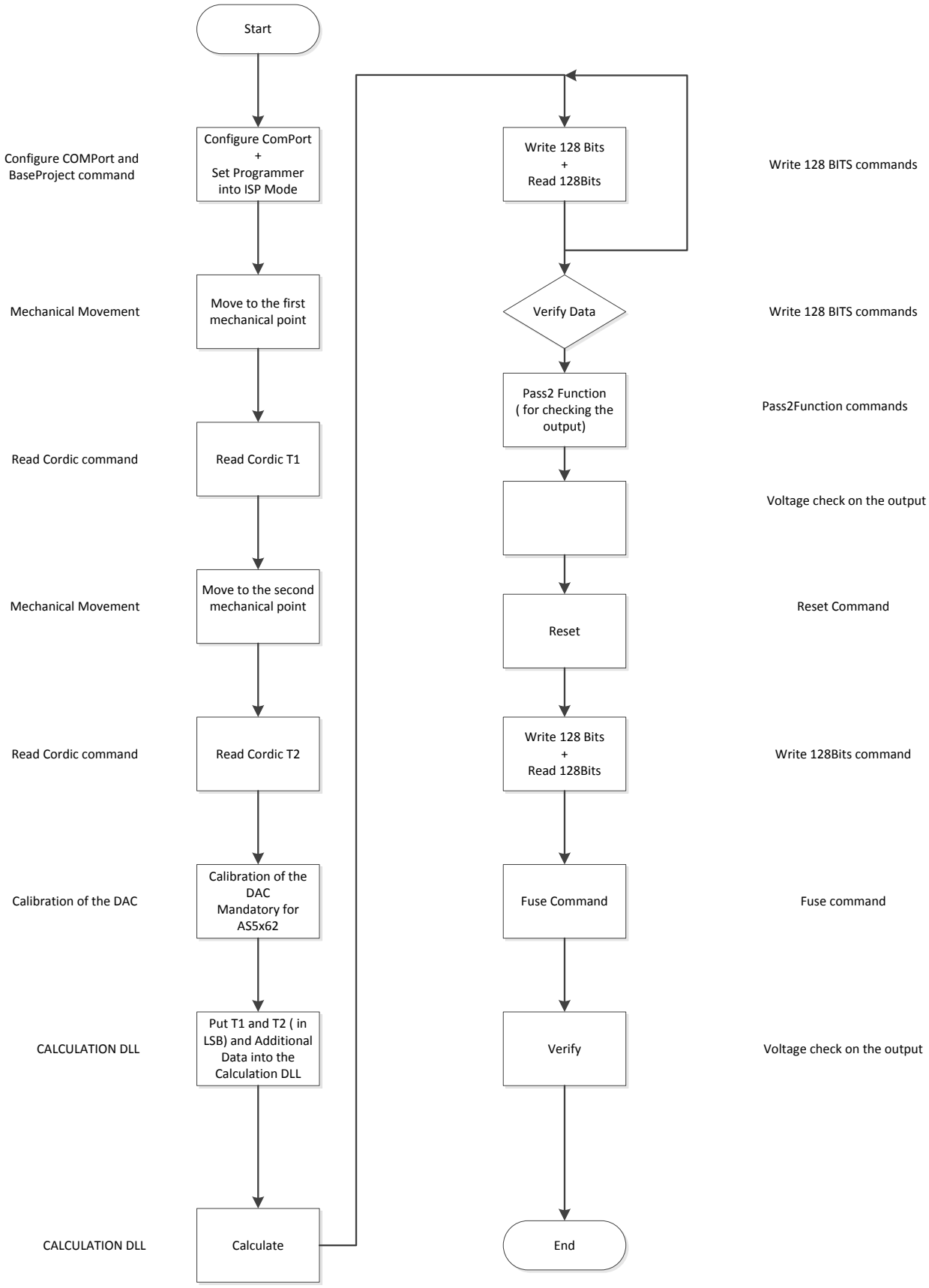
Install and register Equinox ***'ActiveX control'***
Purchase and install an Equinox ***'ActiveX control license'***

1.2 Additional information about the ActiveX control

Please refer to Application Note 141 for additional general information about how to use the Equinox ***'ActiveX control'*** to control an Equinox programmer.

1.3 Typical ActiveX control sequence flowchart

The flowchart below shows a typical calibration / programming flow use to program an ams AS5x6x device.



1.4 Typical ActiveX control sequence - step-by-step guide

The guide below gives an example of how to set up the ActiveX 'methods' for each stage of the sequence.

1.4.1 Configure ComPort and Base Project

Properties and Methods	
Set ComPort	
TestComPort	
Test ISP Nano	
WordAddresses	True - Setting
BaseProject	AS5162PRJ
Address	(for Programmer 0 or Programmer 1)
Put Into MISP Mode	

1.4.2 Mechanical movement

The ***'control application'*** should instruct an actuator to physically move the DUT to the next position,

1.4.3 Read Cordic Command

The 'Read Cordic' command is used to read the position of the sensor.

Properties and Methods	
Address	(for Programmer 0 or Programmer 1)
AMS Read Sensor	

1.4.4 Mechanical movement

The ***'control application'*** should instruct an actuator to physically move the DUT to the next position,

1.4.5 Read Cordic Command

Properties and Methods	
Address	(for Programmer 0 or Programmer 1)
AMS Read Sensor	

1.4.6 Calibration DAC Command

Properties and Methods	
Address	(for Programmer 0 or Programmer 1)
AMS Read Sensor	
Address	(for Programmer 0 or Programmer 1)
SwapWordBytes (True)	
EepromWriteWord	(0x11 for Address and 0xXXXX for New Value)
EepromReadWord	(0x11for Address)
LastErrorMessage	
AMSWrite	b1 = 0 ; b2= 0 ; b3 = 0
SwapWordByte	(false)
MEASURING VOLTAGE ON THE SENSOR	
Address	
ForceReset	
WordAddresses	
BaseProject	(true)
Base Project	(AS5162.PRJ)
PutIntoMISPMode	
Address	(for Programmer 0 or Programmer 1)
SwapWordBytes	(True)
EepromWriteWord	(0x11 for Address and 0xXXXX for New Value)
EepromReadWord	(0x11for Address)
LastErrorMessage	
AMSWrite	b1 = 0 ; b2= 0 ; b3 = 0
SwapWordBytes	(false)
“ MEASURING VOLTAGE ON THE SENSOR”	
Address	
ForceReset	
WordAddresses	
BaseProject	(true)
Base Project	(AS5162.PRJ)
PutIntoMISPMode	

1.4.7 Calculation DLL

➔ Information and CodeSample in Dokument: DLL_Explanation

For Calculation the following inputs are necessary:

Clamping high level
Clamping low level
Voltage on Position 1
Voltage on Position2
Voltage measured on Position1 (during DAC Calibration)
Voltage measured on Position2 (during DAC Calibration
Cordic Value on Position 1
Cordic Value on Position 2
Quadrant Mode (1 – 4)
Hysteresis Level
Trimming Direction
BP_T1 (default 0)
BP_T2 (default 0)
Trimming mode : Default 0

After the calculation the follwing information is ready:
ScaleFactor; offsetoutput;breakpoint; clamping low, clamping high, hystsel, quadEN and sel Scaling.

This information has to save in the BIN file for the Write 128Bits together with the settings as written and descripted in the AS5162 Datasheet.

1.4.8 Write 128 Bits

Properties and Methods	
Address	for Programmer 0 or Programmer 1)
Write programming Data as BIN	
FlashWriteFrom File: From 0 to 3	Writing of the BIN File
FlashVerifyFromFile: From0 to 3	Verify BIN File with MASK 0x00000000FFFCFFFFFFFFFFFFFFFFFFFF

1.4.9 Pass 2 Function command

Properties and Methods	
Address	
SwapWordBytes	(true)
AMSWrite	b1 = 0 ; b2= 0 ; b3 = 0
SwapWordBytes	(false)
LastErrorMessage	

1.4.10 Verify

Voltage Check

1.4.11 Reset command

Properties and Methods	
Address	
ForceReset	(true)
WordAddresses	b1 = 0 ; b2= 0 ; b3 = 0
BaseProject	(true)
Base Project	(AS5162.PRJ)
PutIntoMISPMode	

1.4.12 Write 128 Bits

Properties and Methods	
Address	(for Programmer 0 or Programmer 1)
Write programming Data as BIN	
FlashWriteFrom File: From 0 to 3	Writing of the BIN File
FlashVerifyFromFile: From0 to 3	Verify BIN File with MASK 0x00000000FFFCFFFFFFFFFFFFFFFFFFFF

1.4.13 Fuse command

Properties and Methods	
Address	
Write programming Data as BIN	(16bytesFuseData.BIN)
AMSWriteFromFile	Writing the 16bytesFuseData.bin
CheckDiagnostic	Bytenumber:18 ; Bytecheck 0xA
Check Diagnostic	MANDATORY
LastErrorMessage	

1.4.14 Verify

Voltage Check

1.4.15 Switching the 'communications resistor' in-circuit

If the correct AS5162.prj ***'Base Project'*** is used which enables OP5 on power-up, then the programmer will automatically set the programmer OP5 pin HIGH when it first powers up the DUT and enters 'programming mode'. There is therefore no need for the ActiveX to perform this action. The ***'communications resistor'*** will be automatically enabled / connected when the programmer enters 'programming mode'.

However, if you do want / need to manually instruct the programmer to switch on / connect the ***'communications resistor'***, then the new ***TargetIOWrite*** method should be used.

public virtual bool TargetIOWrite(short write);

The value to send to set programmer output OP5 HIGH is: **0x0D70**

For this scheme to work, the programmer must be already either in 'programming mode' and the 'Target controlled programmer power supply' must be ON. The programmer will switch OP5 HIGH which should either connect the pull-up from OP5 or switch the relay on thereby connecting the DUT output pin to the programmer (depending on which hardware configuration you are using to switch the ***'communications resistor'***.)

1.4.16 Isolating the 'communications resistor'

If you want to change the AS5162 DUT from ***'communications mode'*** to ***'functional mode'***, then it is necessary to send a command to the programmer to tell it to switch programmer output OP5 to TRISTATE which will disconnect / disable the ***'communications resistor'*** from the DUT OUT pin.

To instruct the programmer to switch off / isolate the ***'communications resistor'***, the new ***TargetIOWrite*** method should be used.

public virtual bool TargetIOWrite(short write);

The value to send to set programmer output OP5 (and all other programmers outputs) to TRISTATE is: **0x0DC0**

For this scheme to work, the programmer must be already either in ***'programming / communications mode'*** or the 'Target controlled programmer power supply' must be ON. The programmer will switch OP5 to TRISTATE which should either disconnect the pull-up from OP5 or switch the relay isolating the DUT output pin, depending on which hardware configuration you are using to switch the ***'communications resistor'***. This command would normally be sent immediately after the ***'Pass2Func'*** command so the DUT power stays on, but the ***'communications resistor'*** is disconnected.

Appendix 4 – AS5x6x – Programmer Error codes

1.0 Overview

This section details all the possible error codes which can be generated by EQTools and ConsoleEDS when programming ams AS5x6x devices.

1.1 Error 40 / 3040 – Failed to enter programming mode

The Error 40 / 3040 simply means that the programmer could not communicate with the target AS5x6x device.

This could be for many reasons including:

1. Incorrect connections to the DUT
    - Make sure that pins 13 + 14 of the 16-way Target ISP Port are connected together - see connection diagram.

2. No power on the DUT
    - Check that there is +5V on the Vcc pin of the DUT when the programmer is trying to communicate with the DUT.

3. There is no ***'Rcommunications'*** pull-up resistor fitted to the DUT output pin
    - Check the relevant datasheet and the schematic of the DUT for the required pull-up resistor value.

4. Programmer or DUT cannot drive the ***'Rcommunications'*** pull-up resistor
    - Try increasing the value of the pull-up resistor until communications starts to work correctly.
    - Equinox used a 3k3 pull-up for our testing with no other components fitted to the DUT output pin.

5. Cables between the programmer and DUT are too long
    - Try reducing the cable length between the programmer and the DUT to e.g. 10cm.
    - Try reducing the ***'Target Baud Rate'*** between the programmer and the DUT.

6. The AS5x6x device may already be in ***'functional mode'***
    - Try resetting power to the DUT (power cycle).
    - If the device is not already 'burned' then it should return to ***'communications mode'*** after a power cycle.
    - Once in ***'communications mode'*** then the programmer should be able to communicate with the DUT again.

7. The AS5x6x device may have already been ‘burned’ and so will no longer respond to the programmer.

- Check that the DUT is a ‘new’ ie virgin device and that the OTP area has no already been programmed.

8. Try reducing the ***‘Target Baud Rate’***

- If all other suggestions fail, it is worth reducing the ***‘Target Baud Rate’*** to 2400 or 4800 baud.
- If there are line drive issues with the AS5x6x UART output pin due to either long cables or capacitance / resistance on the output pin of the DUT, then using a slower baud rate should make communications more reliable.

1.2 Error 4064 / 57 - FUSE command failed

When the AS5x6x **'FUSE'** command is executed....
ConsoleEDS AS5162.PRJ /FASTSTART /AMSWRITE=16_byte_data_file.bin /NORESET

...it is possible for this command to fail during one of the verification stages which are performed as part of the execution of this command.

Error 4064
If the actual verification of the data sent down with the FUSE command fails, then this will be reported as follows:

Channel	*Result*	*Error Condition*
CH0	*FAIL*	*Error 4064 - AMS Write failed!*

ConsoleEDS will then exit with error code 57.

It is important to send the ***/AMSWRITE*** command as this returned diagnostics information contains the 'ams Debug byte' which should tell you exactly where in the **'FUSE'** command sequence the programmer actually failed.

Important
If the **'FUSE'** fails for any reason, then this usually means that there is a problem with the data retention inside the ams DUT. The DUT should therefore be rejected as it may not have been programmed correctly.