UProg2 – User Manual

Document name: UProg2_User_Manual_V0.21_2018_02_10

SmartDesign4you Ltd.
Neuseiersberger Strasse 115
8055, Graz

1 Table of Contents

2 General description

2.1 Description

The second generation SD4Y programmer “UProg2” offers a universal production programming solution for all magnetic position sensors from AMS. It supports following interfaces: SPI, I2C, UART, 1WIRE and PSI5. It is designed for integration in fully automated EoL (End-of-Line) production lines and offers ESD and over-voltage protected I/Os. Software integration is implemented by using the provided LabVIEW drivers or DLL (Dynamic Link Library).

2.2 Scope/Purpose

This document describes the connectors and mechanics of the programmer, which interfaces are available and how to integrate the programmer using LabVIEW and the DLL.

2.3 Key features

- Supports all major AMS magnetic position sensors
- Real-time parallel programming for dual die devices
- ESD and Overvoltage protected I/Os (galvanic isolation)
- 4 user configurable output supplies (V_{VDD}, V_{VIO}, V_{PROG}, V_{PSI5})
- 3 user configurable current limits (V_{VDD}, V_{VIO}, V_{PROG}/V_{PSI5})´
- Min. 8 user configurable GPIOs for production line integration
- Self diagnostic function

2.4 Supported AMS sensors

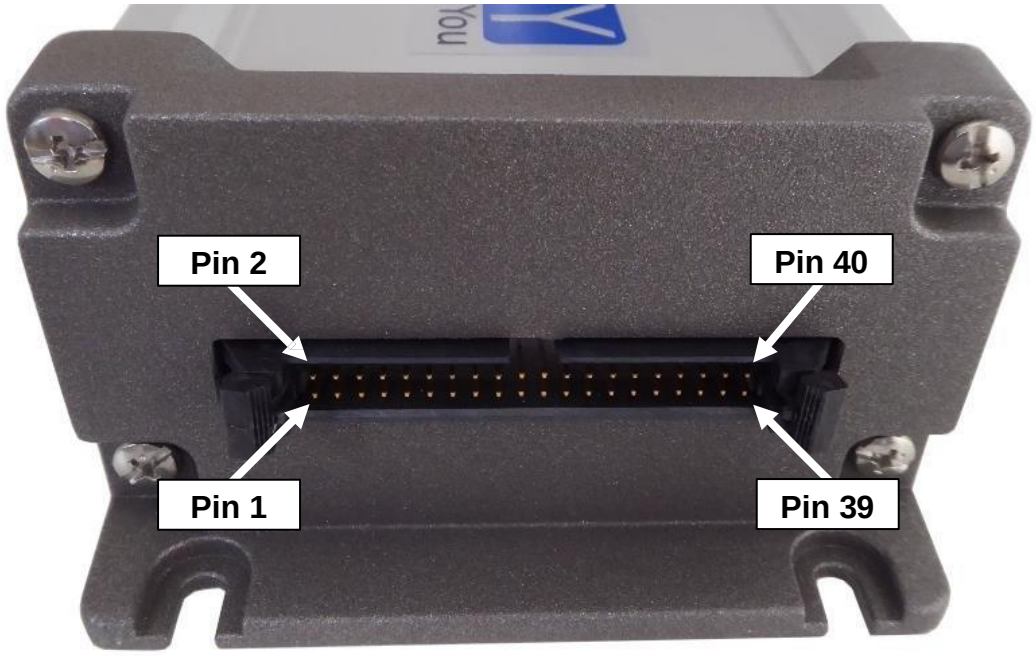
The following list shows all AMS magnetic position sensors supported by the SD4Y programmer “UProg2”:

AS5170, AS5171, AS5270
AS5172

2.5 Front View

The front side of the programmer offers a 40-pin connector for connection to the sensor.
Manufacturer: SAMTEC
Manufacturer part number: EJH-120-01-F-D-RA-22.

Figure 1: Programmer front view



2.6 Connector pinout

The general pinout of the 40-pin connector can be found in Figure 2: 40-pin connector pinout and Table 2-1: 40-pin connector pinout. For device specific pinouts please refer to each device section itself.

Figure 2: 40-pin connector pinout

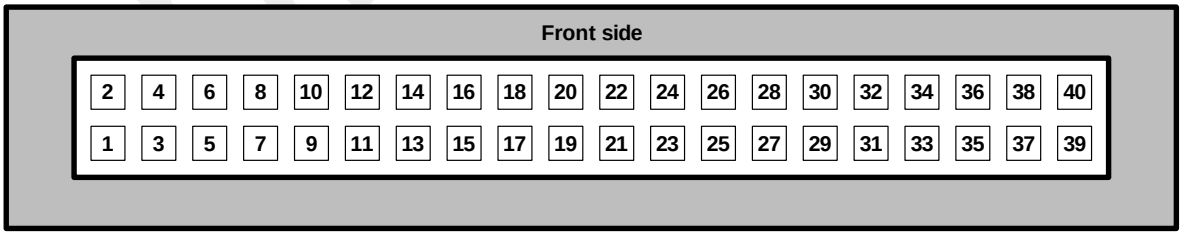


Table 2-1: 40-pin connector pinout

40 Pin Connector		
Pin No.	Mnemonic	Description/Function
1	VDD	ext. device supply
2	VPROG	analog prog supply
3	GND	device ground
4	GND	device ground
5	DIG PIN7	
6	VIO	ext. I/O voltage for dig.bus
7	DIG PIN8	
8	GND	device ground
9	DIG PIN11	
10	DIG PIN12	
11	PROGIO_0	analog prog pin
12	PROGIO_1	analog prog pin
13	NC	
14	DIG PIN13	
15	DIG PIN4	
16	DIG PIN14	
17	DIG PIN9	
18	DIG PIN5	
19	DIG PIN10	
20	DIG PIN6	
21	VDD	ext. device supply
22	VDD	ext. device supply
23	GND	device ground
24	GND	device ground
25	DIG PIN3	
26	NC	
27	DIG PIN1	
28	DIG PIN17	
29	GND	device ground
30	DIG PIN2	
31	PSI5_PIN	PSI5 interface pin
32	DIG PIN16	
33	DIG PIN18	
34	GND	device ground
35	DIG PIN15	
36	GPAD2	analog input
37	GND	device ground
38	GND	device ground
39	GPAD0	analog input
40	GPAD1	analog input

2.7 Rear View

The back side of the programmer offers the USB or RS232 connector for connection to the PC and the DC jack for external power supply. These three connectors are described in Table 2-2: Connector overview.

An external power supply is always mandatory (USB and RS232 mode).

Figure 3: Programmer rear view

Table 2-2: Connector overview

Connector Type	Description/Function
USB	USB 2.0 Type B jack
RS232	9-pin female DSUB (not active in FW < 1.7)
Power Supply	5.5/2.1mm DC jack

3 Electrical Specification

3.1 ESD protection and operating temperature

Table 3-1: ESD protection and operating temperature

Parameter	Min	Typ	Max	Unit	Comments
ESD_{HBM}	2000			V	Human Body Model
ESD_{MM}	200			V	Machine Model
ESD_{CDM}	1000			V	Charge-Device Model
T_{OP}	10	25	35	degC	

3.2 Supply inputs

Table 3-2: Supply inputs

Parameter	Min	Typ	Max	Unit	Comments
VDD_{USB}	4.6	5	5.5	V	
VDD_{RS232}	8	12	14	V	
IDD_{USB}	0.2		0.5	A	
IDD_{RS232}	0.1		1.5	A	

3.3 Supply outputs

Table 3-3: Supply outputs

Parameter	Min	Typ	Max	Unit	Comments
V_{VDD}	1.8		5.5	V	
V_{VIO}	1.8		5.5	V	
V_{PROG}	3		9.5	V	
V_{PSI5}	3		16	V	
I_{VDD}	0		400	mA	
I_{VIO}	0		400	mA	
I_{PROG}	0		400	mA	
I_{PSI5}	0		200	mA	

4 Mechanical dimensions and drawings

4.1 Housing CAD drawing

Figure 4: Housing CAD drawing

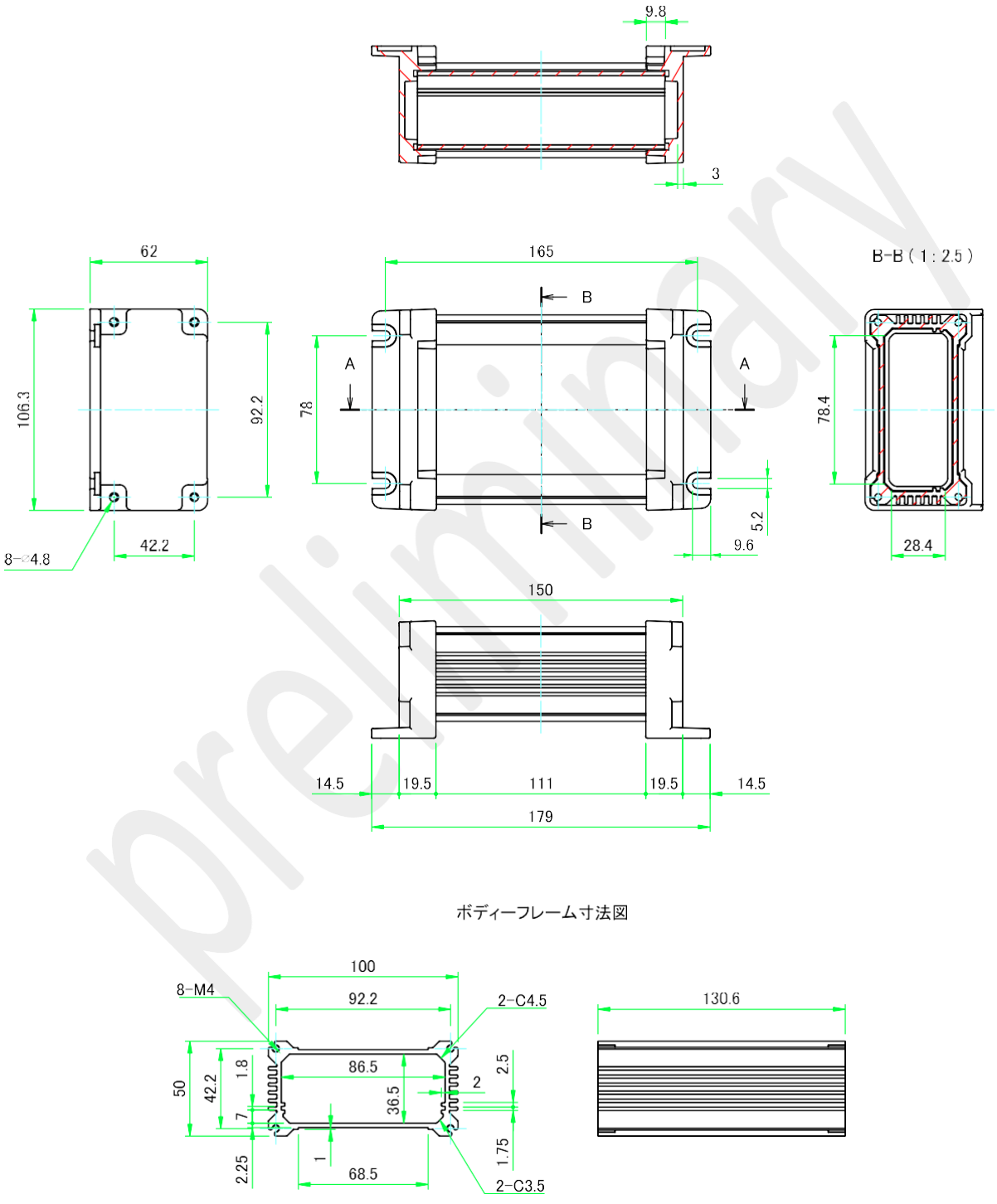


Figure 5: Housing CAD model (3D)

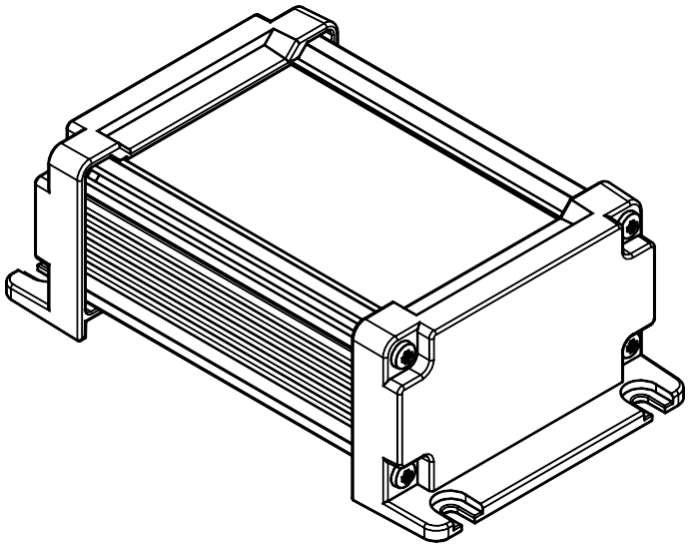
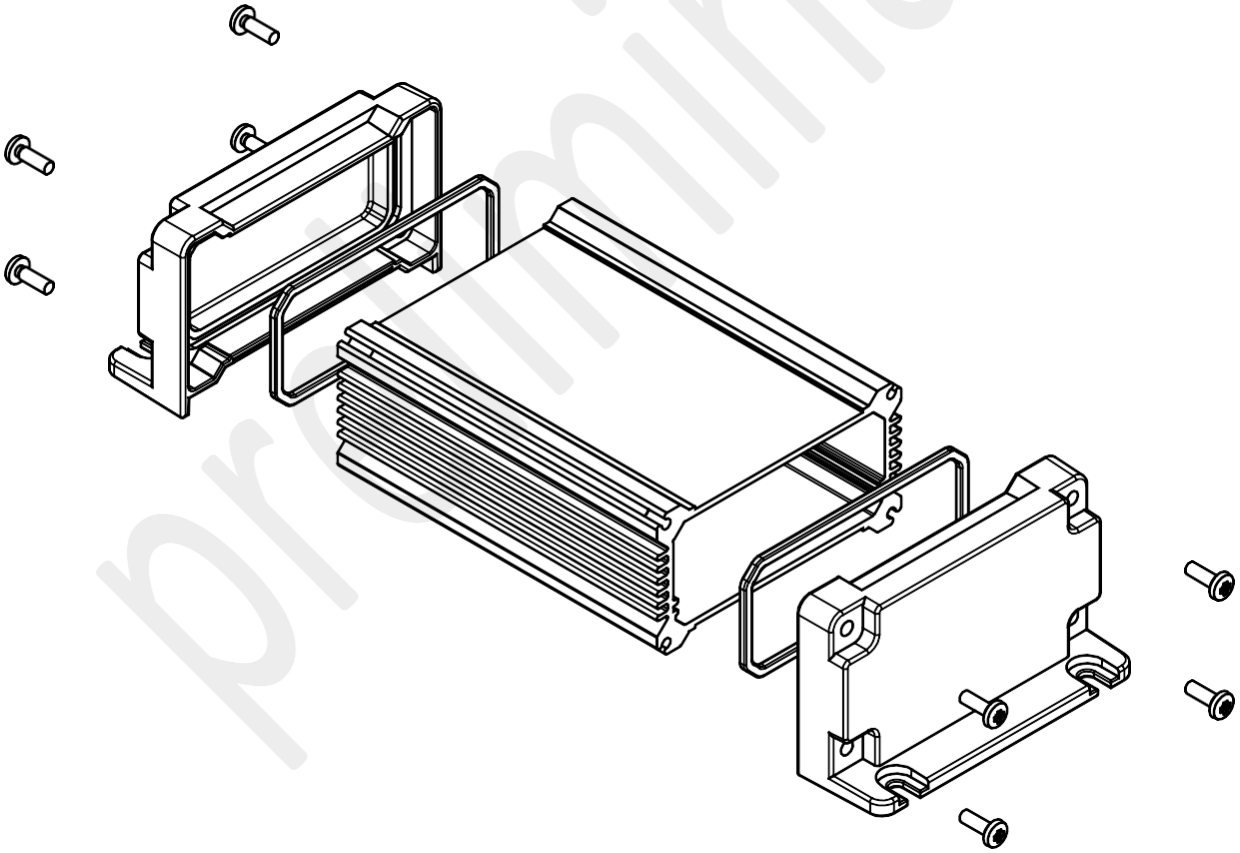


Figure 6: Housing CAD model open (3D)



5 Host interfaces

5.1 USB / RS232

The SD4Y programmer “UProg2” uses a serial communication via USB or RS232 connection with typical baudrates as rated in Table 5-1: Host interfaces.

To initialize the serial connection between programmer and PC the VI or function ***“uprogOpen”*** has to be called first. Only after successful call of this function further functions may be called. Please refer to chapter **Error! Reference source not found. Error! Reference source not found.**.

Table 5-1: Host interfaces

Parameter	Min	Typ	Max	Unit	Comments
USB_{HOST}	115.2		460.8	kBaud	via USB connector
$RS232_{HOST}$	115.2		115.2	kBaud	via RS232 connector

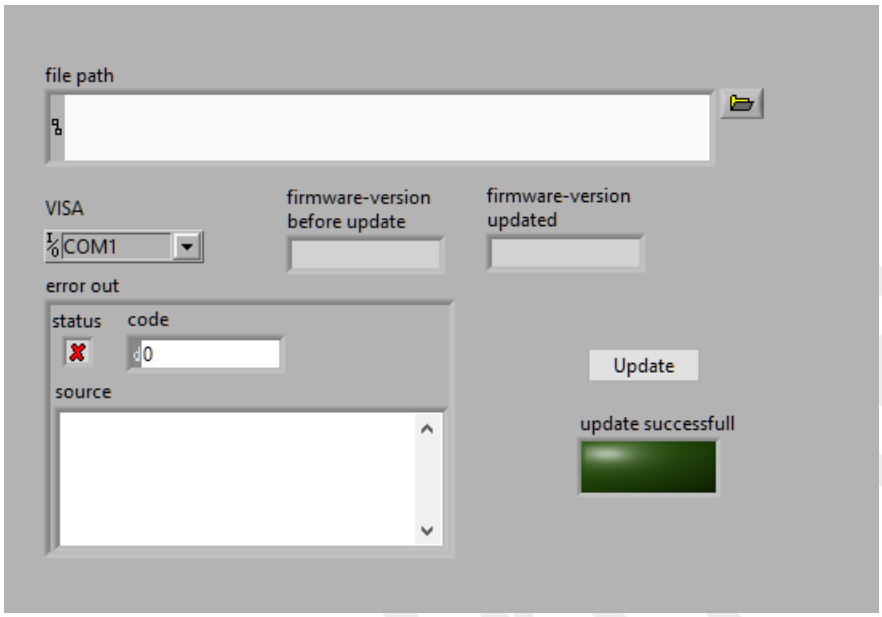
To be able to use the programmer via the USB interface the VCP (Virtual COM port) driver from FTDI is necessary. The VCP drivers cause the USB device to appear as an additional COM port available to the PC. Application software can access the USB device in the same way as it would access a standard COM port.

The drivers can be found in the software package delivered with the programmer or can be downloaded from the FTDI website using following link: http://www.ftdichip.com/Drivers/VCP.htm

5.2 Updating the firmware

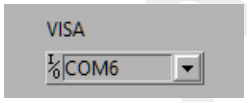
The SD4Y programmer “UProg2” always has to be used with the latest available firmware. To update the firmware please use the ***“Uprog2Bootloader”*** which can be found in the software package delivered with the programmer.

1. Start the ***“UProg2_Bootloader.exe”.***

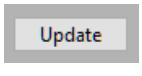


2. Select the file path for the latest firmware .hex file.

3. Select the correct VISA port for the Programmer.

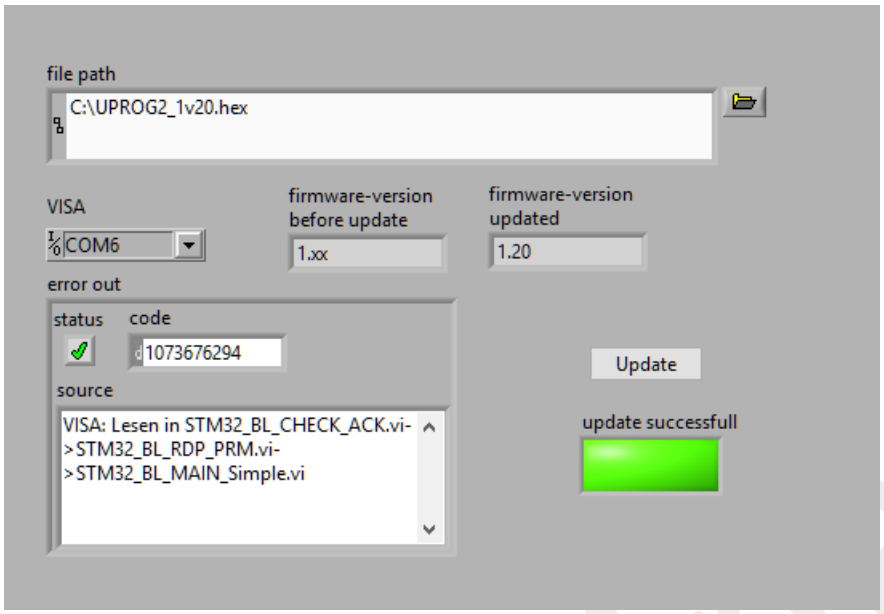


4. Click the “Update” button to execute the firmware update.



5. A progress bar will indicate the actual progress of the firmware update. During the update please do not disconnect the USB or power cable from the programmer.

6. A green indicator will show if the firmware update was successful. In addition the firmware version before and after the update will be shown.



6 Device interfaces

This chapter lists all device interfaces available in the SD4Y “UProg2” programmer.

6.1 UART

The AS5x7y is equipped with a UART interface, which allows reading and writing the registers as well as permanently programming the non-volatile memory (OTP).

6.2 PSI5

Peripheral Sensor Interface 5 is a two wire interface used to connect sensors to the ECU. It uses a current modulation over the supply line with a Manchester coding. More detailed description can be found in the datasheet of the AS5172.

6.3 UART-Over-PSI5

The AS5172 is equipped with a one-wire UART-over-PSI5 interface based on a “Tooth Gap” similar method according to PSI5 specification, which allows reading and writing the registers as well as permanent programming of the non-volatile OTP.

7 Integration options

This chapter gives advice how to connect the specific device to the programmer and how to use the LabVIEW drivers and DLL functions for integration into full automated EOL production lines.

7.1 Hardware integration

7.1.1 Device connection and pinout

7.1.1.1 AS5170/AS5171

This chapter shows the specific pinout of AS5170/AS5171 on the 40-pin connector.

Figure 7: 40-pin connector pinout

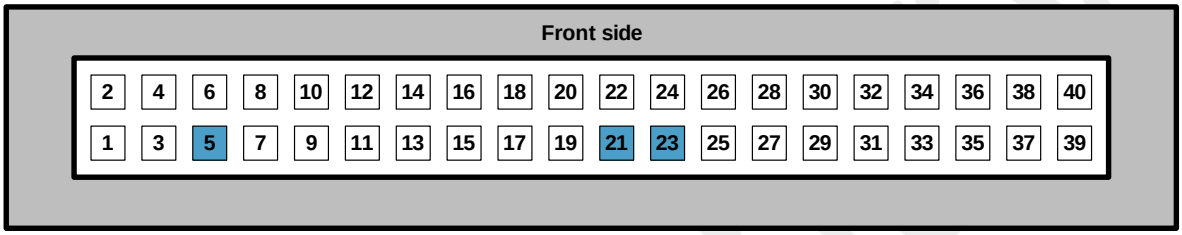


Table 7-1: AS5170/AS5171 pinout

40 Pin Connector			AS5170	AS5171
Pin No.	Mnemonic	Description/Function	SOIC8	SIP
5	DIG PIN7		5 / OUT	3 / OUT
21	VDD	ext. device supply	1 / VDD	1 / VDD
23	GND	device ground	8 / GND	2 / GND

7.1.1.2 AS5172

This chapter shows the specific pinout of AS5172 on the 40-pin connector.

Figure 8: 40-pin connector pinout

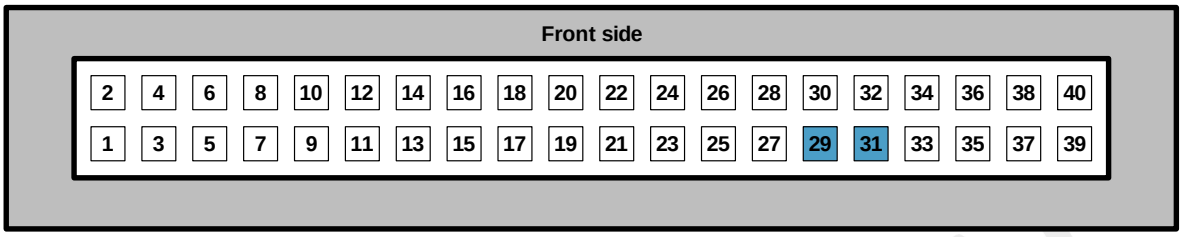


Table 7-2: AS5172 pinout

40 Pin Connector			AS5172A	AS5172B
Pin No.	Mnemonic	Description/Function	SIP	TSSOP14
29	GND	device ground	2 / GND	12 / GND
31	PSI5_PIN	PSI5 interface pin	1 / VDD	14 / VDD

7.1.1.3 AS5270

This chapter shows the specific pinout of AS5270 on the 40-pin connector.

Figure 9: 40-pin connector pinout

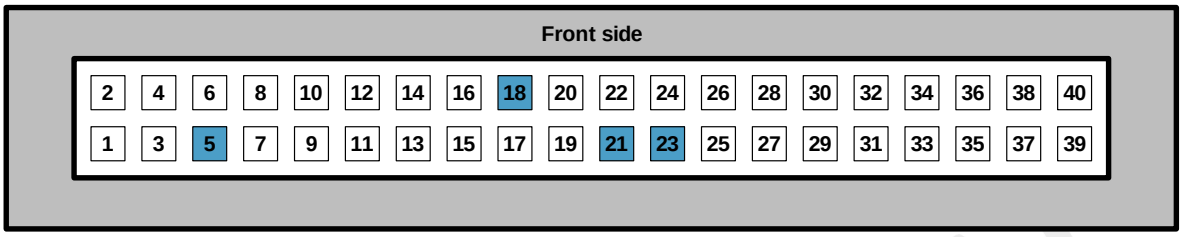


Table 7-3: AS5270 pinout

40 Pin Connector			AS5270A	AS5270B
Pin No.	Mnemonic	Description/Function	MLF-16	MLF-16
5	DIG PIN7		8 / OUT_B	8 / OUT_B
18	DIG PIN5		7 / OUT_T	7 / OUT_T
21	VDD	ext. device supply	15 / VDD_T 16 / VDD_B	15 / VDD_T 16 / VDD_B
23	GND	device ground	13 / GND_T 14 / GND_B	13 / GND_T 14 / GND_B

7.2 Software integration

7.2.1 General software functions

7.2.1.1 LabVIEW

All general LabVIEW drivers can be found in the ***"UProg2_system.llb"*** delivered in the software package of the programmer.

7.2.1.1.1 uprogOpen

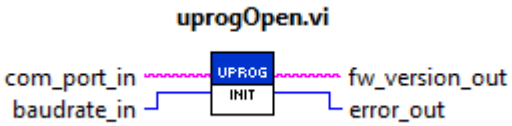


Table 7-4: Parameter list

Description			
This VI initializes the serial connection between programmer and PC and has to be called prior to any other VI call.			
Parameter name	**Data type**	**Type**	**Description**
com_port_in	char	in	Serial COM port of the programmer (e.g. COM2)
baudrate_in	uint32	in	Baudrate for serial connection to PC (typ. 115200 baud)
fw_version_out	char	out	Current firmware version
error_out	uint8	out	Error bus output

7.2.1.1.2 uprogReset

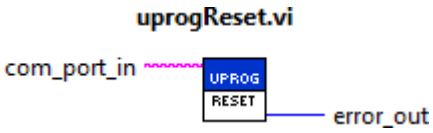


Table 7-5: Parameter list

Description			
This VI sends a soft reset to the programmer.			
Parameter name	**Data type**	**Type**	**Description**
com_port_in	char	in	Serial COM port of the programmer (e.g. COM2)
error_out	uint8	out	Error bus output

7.2.1.1.3 uprogClose

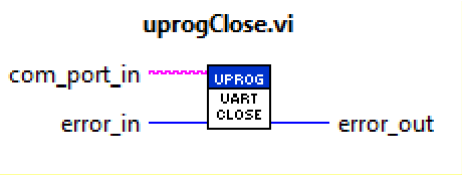


Table 7-6: Parameter list

Description			
This VI closes the serial connection between programmer and PC.			
Parameter name	**Data type**	**Type**	**Description**
com_port_in	char	in	Serial COM port of the programmer (e.g. COM2)
error_in	uint8	in	Error bus input
error_out	uint8	out	Error bus output

7.2.1.2 DLL

The DLL functions described hereafter can be found in the ***"UProg2Driver.dll"*** and ***"UProg2Driver.h"*** files delivered in the software package of the programmer.

7.2.1.2.1 uprogOpen

Table 7-7: Function list

Function	
This function initializes the serial connection between programmer and PC and has to be called prior to any other function call.	
Function call	
`uint8_t uprogOpen(char com_port_in[], uint32_t baudrate_in, char fw_version_out[], uint8_t *error_out, int32_t fw_version_out_len)`	
Transfer values	
Parameter name	**Description**
`char com_port_in[]`	Serial COM port of the programmer (e.g. COM2)
`uint32_t baudrate_in`	Baudrate for serial connection to PC (typ. 115200 baud)
`char fw_version_out[]`	Current firmware version
`int32_t fw_version_out_len`	Length of the array
Return value	
`uint8_t error_out`	Status: For detailed error information please refer to chapter 7.2.4 Error codes

7.2.1.2.2 uprogReset

Table 7-8: Function list

Function	
This function sends a soft reset to the programmer.	
Function call	
`uint8_t uprogReset(char com_port_in[])`	
Transfer values	
Parameter name	**Description**
`char com_port_in[])`	Serial COM port of the programmer (e.g. COM2)
Return value	
`uint8_t error_out`	Status: For detailed error information please refer to chapter 7.2.4 Error codes

7.2.1.2.3 uprogClose

Table 7-9: Function list

Function	
This function closes the serial connection to the programmer.	
Function call	
`uint8_t uprogClose(char com_port_in[], uint8_t error_in)`	
Transfer values	
Parameter name	**Description**
`char com_port_in[]`	Serial COM port of the programmer (e.g. COM2)
`uint8_t error_in`	Error In
Return value	
`uint8_t error_out`	Status: For detailed error information please refer to chapter 7.2.4 Error codes

7.2.2 Generic interface functions

7.2.2.1 LabVIEW

All generic device interface LabVIEW drivers can be found in the ***"Uprog_StandardSerial.llb"*** delivered in the software package of the programmer.

7.2.2.2 DLL

The DLL functions described hereafter can be found in the ***"UProg2Driver.dll"*** and ***"UProg2Driver.h"*** files delivered in the software package of the programmer.

7.2.3 Device interface functions

7.2.3.1 AS5170/AS5171

7.2.3.1.1 LabVIEW

All LabVIEW drivers for AS5170/AS5171 can be found in the AS5x7y.llb delivered in the software package of the programmer. Before using the specific AS5170/AS5171 VIs the UART interface needs to be initialized. The necessary ***"uartInit.vi"*** can be found in the ***"Uprog2_StandardSerial.llb"***.

7.2.3.1.1.1 uartInit

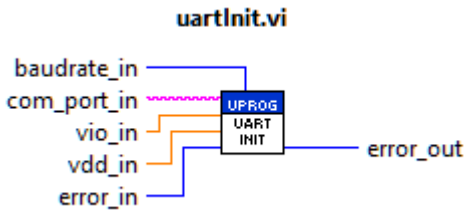


Table 7-10: Parameter list

Description			
This VI initializes the UART interface for AS5170/AS5171.			
Parameter name	**Data type**	**Type**	**Description**
com_port_in	char	in	Serial COM port of the programmer (e.g. COM2)
baudrate_in	uint32	in	Baudrate for the UART (typ. 9600 baud)
vio_in	double	in	Voltage for IO of the UART (typ. 5V)
vdd_in	double	in	Voltage for sensor supply (typ. 5V)
error_in	uint8	in	Error bus input
error_out	uint8	out	Error bus output

7.2.3.1.1.2 as5x7yRead

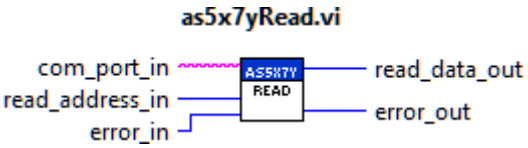


Table 7-11: Parameter list

Description			
This VI reads two consecutive registers starting at a specific address.			
Parameter name	**Data type**	**Type**	**Description**
com_port_in	char	in	Serial COM port of the programmer (e.g. COM2)
read_address_in	uint8	in	Read register address
error_in	uint8	in	Error bus input
read_data_out	uint16	out	Register data of address and address+1
error_out	uint8	out	Error bus output

7.2.3.1.1.3 as5x7yWrite

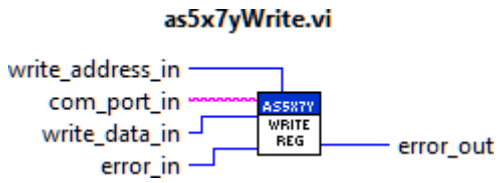


Table 7-12: Parameter list

Description			
This VI writes two consecutive registers starting at a specific address.			
Parameter name	**Data type**	**Type**	**Description**
com_port_in	char	in	Serial COM port of the programmer (e.g. COM2)
write_address_in	uint8	in	Write register address
write_data_in	uint16	in	Register data for address and address+1
error_in	uint8	in	Error bus input
error_out	uint8	out	Error bus output

7.2.3.1.1.4 as5x7yReadArray

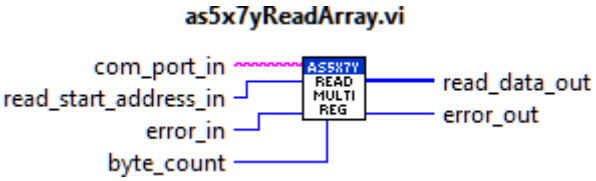


Table 7-13: Parameter list

Description			
This VI reads n number of registers starting at a specific address.			
Parameter name	**Data type**	**Type**	**Description**
com_port_in	char	in	Serial COM port of the programmer (e.g. COM2)
read_start_address_in	uint8	in	Read start address
byte_read_count	uint8	in	Number of bytes to be read
error_in	uint8	in	Error bus input
read_data_out	uint8	out	Read data array
error_out	uint8	out	Error bus output

7.2.3.1.1.5 as5x7yWriteArray

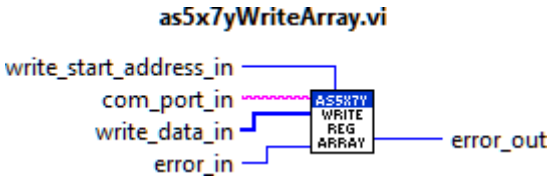


Table 7-14: Parameter list

Description			
This VI writes a data array of n values starting at a specific register address.			
Parameter name	**Data type**	**Type**	**Description**
com_port_in	char	in	Serial COM port of the programmer (e.g. COM2)
write_start_address_in	uint8	in	Write start address for the data array
write_data_in	uint8	in	Data array (n bytes)
error_in	uint8	in	Error bus input
error_out	uint8	out	Error bus output

7.2.3.1.1.6 as5x7yReadCordic

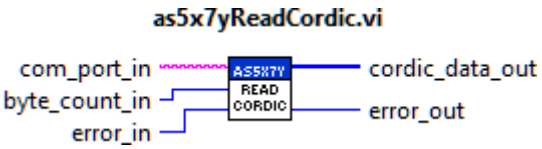


Table 7-15: Parameter list

Description			
This VI reads the CORDIC (14-bit angle) register of the AS5170/AS5171.			
Parameter name	**Data type**	**Type**	**Description**
com_port_in	char	in	Serial COM port of the programmer (e.g. COM2)
byte_count_in	uint8	in	Number of CORDIC reads
error_in	uint8	in	Error bus input
cordic_data_out	uint16	out	Read CORDIC data array
error_out	uint8	out	Error bus output

7.2.3.1.1.7 as5x7yReadAgcMag

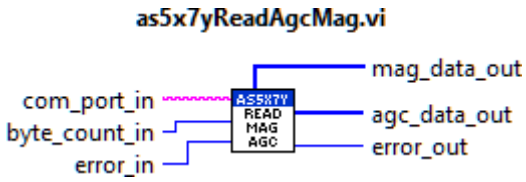


Table 7-16: Parameter list

Description			
This VI reads the AGC (8-bit) and Magnitude (8-bit) register of the AS5170/AS5171.			
Parameter name	**Data type**	**Type**	**Description**
com_port_in	char	in	Serial COM port of the programmer (e.g. COM2)
byte_count_in	uint8	in	Number of AGC and Magnitude reads
error_in	uint8	in	Error bus input
mag_data_out	uint16	out	Read Magnitude array
agc_data_out	uint16	out	Read AGC array
error_out	uint8	out	Error bus output

7.2.3.1.1.8 as5x7yPass2Func

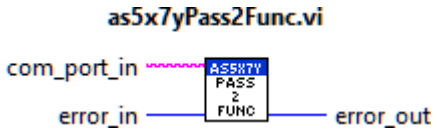


Table 7-17: Parameter list

Description			
This VI sets the AS5170/AS5171 into PASS2FUNC mode. This mode is needed for temporary testing before burning the OTP fuses.			
Parameter name	**Data type**	**Type**	**Description**
com_port_in	char	in	Serial COM port of the programmer (e.g. COM2)
error_in	uint8	in	Error bus input
error_out	uint8	out	Error bus output

7.2.3.1.1.9 as5x7yBurn

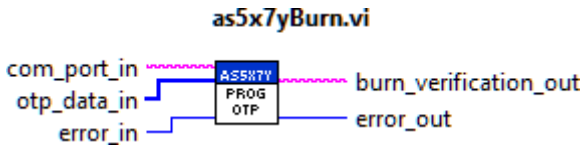


Table 7-18: Parameter list

Description			
This VI permanently burns the OTP fuses of the AS5170/AS5171 and does the verification according the AS5170/AS5171 datasheet programming flow.			
Parameter name	**Data type**	**Type**	**Description**
com_port_in	char	in	Serial COM port of the programmer (e.g. COM2)
otp_data_in	uint8	in	OTP register data for address 0x0A to 0x1E
error_in	uint8	in	Error bus input
burn_verification_out	char	out	OTP verification status
error_out	uint8	out	Error bus output

7.2.3.1.2 DLL

The DLL functions described hereafter can be found in the ***"UProg2Driver.dll"*** and ***"UProg2Driver.h"*** files delivered in the software package of the programmer.

7.2.3.1.2.1 uartInit

Table 7-19: Function list

Function	
This function initializes the UART interface for AS5170/AS5171.	
Function call	
`uint8_t uartInit(char com_port_in[], uint8_t error_in, double vio_in, double vdd_in, uint32_t baudrate_in)`	
Transfer values	
Parameter name	**Description**
`char com_port_in[]`	Serial COM port of the programmer (e.g. COM2)
`double vio_in`	Voltage for IO of the UART (typ. 5V)
`double vdd_in`	Voltage for sensor supply (typ. 5V)
`uint8_t error_in`	Error In
`uint32_t baudrate_in`	Baudrate for the UART (typ. 9600 baud)
Return value	
`uint8_t error_out`	Status: For detailed error information please refer to chapter 7.2.4 Error codes

7.2.3.1.2.2 as5x7yRead

Table 7-20: Function list

Function	
This function reads two consecutive registers starting at a specific address.	
Function call	
`uint8_t as5x7yRead(char com_port_in[], uint8_t error_in, uint8_t read_address_in, uint16_t *read_data_out)`	
Transfer values	
Parameter name	**Description**
`char com_port_in[]`	Serial COM port of the programmer (e.g. COM2)
`uint8_t read_address_in`	Read register address
`uint8_t error_in`	Error In
`uint16_t *read_data_out`	Register data of address and address+1
Return value	
`uint8_t error_out`	Status: For detailed error information please refer to chapter 7.2.4 Error codes

7.2.3.1.2.3 as5x7yWrite

Table 7-21: Function list

Function	
This function writes two consecutive registers starting at a specific address.	
Function call	
`uint8_t as5x7yWrite(char com_port_in[], uint8_t error_in, uint8_t write_address_in, uint16_t write_data_in)`	
Transfer values	
Parameter name	**Description**
`char com_port_in[]`	Serial COM port of the programmer (e.g. COM2)
`uint8_t write_address_in`	Write register address
`uint16_t write_data_in`	Register data for address and address+1
`uint8_t error_in`	Error In
Return value	
`uint8_t error_out`	Status: For detailed error information please refer to chapter 7.2.4 Error codes

7.2.3.1.2.4 as5x7yReadArray

Table 7-22: Function list

Function	
This function reads n number of registers starting at a specific address.	
Function call	
`uint8_t as5x7yReadArray(char com_port_in[], uint8_t error_in, uint8_t read_start_address_in, uint8_t byte_read_count, uint8_t read_data_out[], int32_t read_data_out_len)`	
Transfer values	
Parameter name	**Description**
`char com_port_in[]`	Serial COM port of the programmer (e.g. COM2)
`uint8_t read_start_address_in`	Read start address
`uint8_t byte_read_count`	Number of bytes to be read
`uint8_t read_data_out[]`	Read data array
`int32_t read_data_out_len`	Length of array
`uint8_t error_in`	Error In
Return value	
`uint8_t error_out`	Status: For detailed error information please refer to chapter 7.2.4 Error codes

7.2.3.1.2.5 as5x7yWriteArray

Table 7-23: Function list

Function	
This function writes a data array of n values starting at a specific register address.	
Function call	
`uint8_t as5x7yWriteArray(char com_port_in[], uint8_t error_in, uint8_t write_start_address_in, uint8_t write_data_in[], int32_t write_data_in_len)`	
Transfer values	
Parameter name	**Description**
`char com_port_in[]`	Serial COM port of the programmer (e.g. COM2)
`uint8_t write_start_address_in`	Write start address for the data array
`uint8_t write_data_in[]`	Data array (n bytes)
`int32_t write_data_in_len`	Length of array
`uint8_t error_in`	Error In
Return value	
`uint8_t error_out`	Status: For detailed error information please refer to chapter 7.2.4 Error codes

7.2.3.1.2.6 as5x7yReadCordic

Table 7-24: Function list

Function	
This function reads the CORDIC (14-bit angle) register of the AS5170/AS5171.	
Function call	
`uint8_t as5x7yReadCordic(char com_port_in[], uint8_t error_in, uint8_t read_count_in, uint16_t cordic_data_out[], int32_t cordic_data_out_len)`	
Transfer values	
Parameter name	**Description**
`char com_port_in[]`	Serial COM port of the programmer (e.g. COM2)
`uint8_t read_count_in`	Number of CORDIC reads
`uint16_t cordic_data_out[]`	Read CORDIC data array
`int32_t cordic_data_out_len`	Length of array
`uint8_t error_in`	Error In
Return value	
`uint8_t error_out`	Status: For detailed error information please refer to chapter 7.2.4 Error codes

7.2.3.1.2.7 as5270ReadAgcMag

Table 7-25: Function list

Function	
This function reads the AGC (8-bit) and Magnitude (8-bit) register of the AS5170/AS5171.	
Function call	
`uint8_t as5x7yReadAgcMag(char com_port_in[], uint8_t error_in, uint8_t read_count_in, uint8_t agc_data_out[], uint8_t mag_data_out[], int32_t agc_data_out_len, int32_t mag_data_out_len)`	
Transfer values	
Parameter name	**Description**
`char com_port_in[]`	Serial COM port of the programmer (e.g. COM2)
`uint8_t read_count_in`	Number of AGC and Magnitude reads
`uint16_t agc_data_out[]`	Read AGC array
`uint16_t mag_data_out[]`	Read Magnitude array
`int32_t agc_data_out_len`	Length of array
`int32_t mag_data_out_len`	Length of array
`uint8_t error_in`	Error In
Return value	
`uint8_t error_out`	Status: For detailed error information please refer to chapter 7.2.4 Error codes

7.2.3.1.2.8 as5x7yPass2Func

Table 7-26: Function list

Function	
This function sets the AS5170/AS5171 into PASS2FUNC mode. This mode is needed for temporary testing before burning the OTP fuses.	
Function call	
`uint8_t as5x7yPass2Func(char com_port_in[], uint8_t error_in)`	
Transfer values	
Parameter name	**Description**
`char com_port_in[]`	Serial COM port of the programmer (e.g. COM2)
`uint8_t error_in`	Error In
Return value	
`uint8_t error_out`	Status: For detailed error information please refer to chapter 7.2.4 Error codes

7.2.3.1.2.9 as5x7yBurn

Table 7-27: Function list

Function	
This function permanently burns the OTP fuses of the AS5170/AS5171 and does the verification according the AS5170/AS5171 datasheet programming flow.	
Function call	
`uint8_t as5x7yBurn(char com_port_in[], uint8_t error_in, uint8_t otp_data_in[], char burn_verification_out[], int32_t otp_data_in_len, int32_t burn_verification_out_len)`	
Transfer values	
Parameter name	**Description**
`char com_port_in[]`	Serial COM port of the programmer (e.g. COM2)
`uint8_t otp_data_in[]`	OTP register data for address 0x0A to 0x1E
`Char burn_verification_out[]`	OTP verification status
`int32_t otp_data_in_len`	Length of array
`int32_t burn_verification_out_len`	Length of array
`uint8_t error_in`	Error In
Return value	
`uint8_t error_out`	Status: For detailed error information please refer to chapter 7.2.4 Error codes

7.2.3.2 AS5172

7.2.3.2.1 LabVIEW

All LabVIEW drivers for AS5172 can be found in the ***"AS5172.llb"*** delivered in the software package of the programmer.

7.2.3.2.1.1 as5172InitPsi5

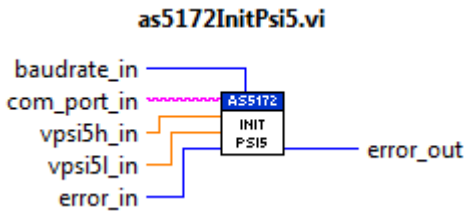
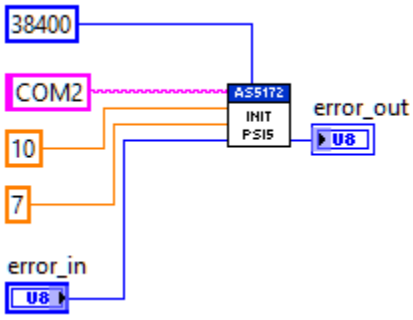


Table 7-28: Parameter list

Description			
This VI initializes the UART-over-PSI5 interface for AS5172.			
Parameter name	**Data type**	**Type**	**Description**
com_port_in	char	in	Serial COM port of the programmer (e.g. COM2)
baudrate_in	uint16	in	Baudrate for the UART-over-PSI5 communication (typ. 38400 baud)
vpsi5h_in	double	in	High level voltage of the UART-over-PSI5 signal (typ. 10V)
vpsi5l_in	double	in	Low level voltage of the UART-over-PSI5 signal (typ. 7V)
error_in	uint8	in	Error bus input
error_out	uint8	out	Error bus output

Example



This example shows the typical initialization of the UART-over-PSI5 for AS5172.

7.2.3.2.1.2 as5172Read

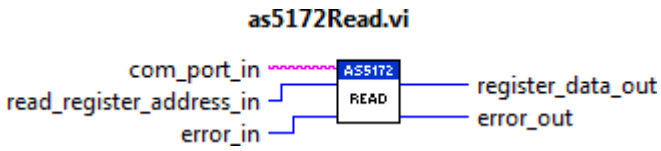


Table 7-29: Parameter list

Description			
This VI reads two consecutive registers starting at a specific address.			
Parameter name	**Data type**	**Type**	**Description**
com_port_in	char	in	Serial COM port of the programmer (e.g. COM2)
read_register_address_in	uint8	in	Read register address
error_in	uint8	in	Error bus input
register_data_out	uint16	out	Register data of address and address+1
error_out	uint8	out	Error bus output

7.2.3.2.1.3 as5172Write

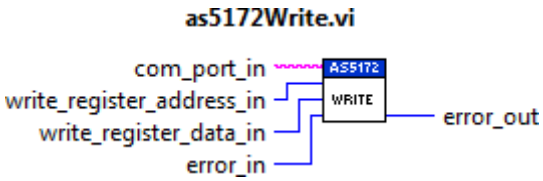


Table 7-30: Parameter list

Description			
This VI writes two consecutive registers starting at a specific address.			
Parameter name	**Data type**	**Type**	**Description**
com_port_in	char	in	Serial COM port of the programmer (e.g. COM2)
write_register_address_in	uint8	in	Write register address
write_register_data_in	uint16	in	Register data for address and address+1
error_in	uint8	in	Error bus input
error_out	uint8	out	Error bus output

7.2.3.2.1.4 as5172ReadArray

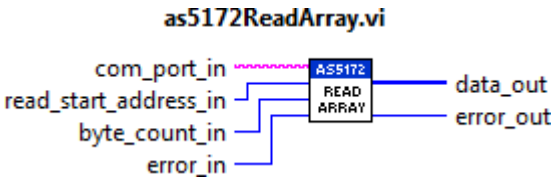


Table 7-31: Parameter list

Description			
This VI reads n number of registers starting at a specific address.			
Parameter name	**Data type**	**Type**	**Description**
com_port_in	char	in	Serial COM port of the programmer (e.g. COM2)
read_start_address_in	uint8	in	Read start address
byte_count_in	uint8	in	Number of bytes to be read
error_in	uint8	in	Error bus input
data_out	uint8	out	Read data array
error_out	uint8	out	Error bus output

7.2.3.2.1.5 as5172WriteArray

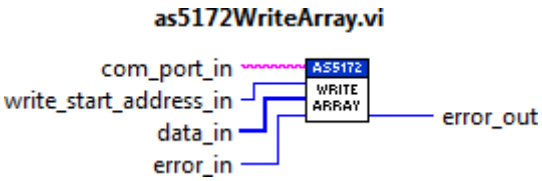


Table 7-32: Parameter list

Description			
This VI writes a data array of n values starting at a specific register address.			
Parameter name	**Data type**	**Type**	**Description**
com_port_in	char	in	Serial COM port of the programmer (e.g. COM2)
write_start_address_in	uint8	in	Write start address for the data array
data_in	uint8	in	Data array (n bytes)
error_in	uint8	in	Error bus input
error_out	uint8	out	Error bus output

7.2.3.2.1.6 as5172ReadCordic

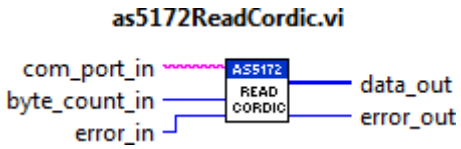


Table 7-33: Parameter list

Description			
This VI reads the CORDIC (14-bit angle) register of the AS5172.			
Parameter name	**Data type**	**Type**	**Description**
com_port_in	char	in	Serial COM port of the programmer (e.g. COM2)
byte_count_in	uint8	in	Number of CORDIC reads
error_in	uint8	in	Error bus input
data_out	uint16	out	Read CORDIC data array
error_out	uint8	out	Error bus output

7.2.3.2.1.7 as5172ReadAgcMag

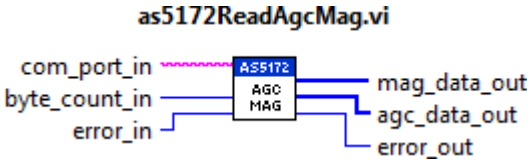


Table 7-34: Parameter list

Description			
This VI reads the AGC (8-bit) and Magnitude (8-bit) register of the AS5172.			
Parameter name	**Data type**	**Type**	**Description**
com_port_in	char	in	Serial COM port of the programmer (e.g. COM2)
byte_count_in	uint8	in	Number of AGC and Magnitude reads
error_in	uint8	in	Error bus input
mag_data_out	uint8	out	Read Magnitude array
agc_data_out	uint8	out	Read AGC array
error_out	uint8	out	Error bus output

7.2.3.2.1.8 as5172Pass2Func

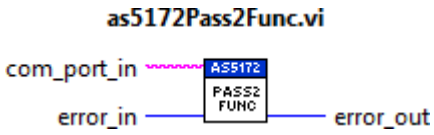


Table 7-35: Parameter list

Description			
This VI sets the AS5172 into PASS2FUNC mode. This mode is needed for temporary testing before burning the OTP fuses.			
Parameter name	**Data type**	**Type**	**Description**
com_port_in	char	in	Serial COM port of the programmer (e.g. COM2)
error_in	uint8	in	Error bus input
error_out	uint8	out	Error bus output

7.2.3.2.1.9 as5172BurnOtp

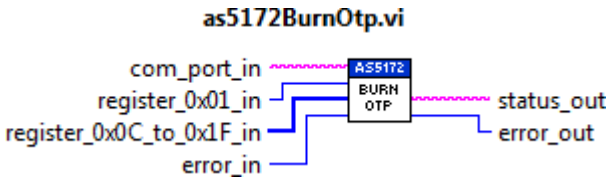


Table 7-36: Parameter list

Description			
This VI permanently burns the OTP fuses of the AS5172 and does the verification according the AS5172 datasheet programming flow.			
Parameter name	**Data type**	**Type**	**Description**
com_port_in	char	in	Serial COM port of the programmer (e.g. COM2)
register_0x01_in	uint8	in	Register data for address 0x01
register_0x0C_to_0x1F_in	uint8	in	Register data array for address 0x0C to 0x1F
error_in	uint8	in	Error bus input
status_out	char	out	OTP verification status
error_out	uint8	out	Error bus output

7.2.3.2.2 DLL

The DLL functions described hereafter can be found in the ***"UProg2Driver.dll"*** and ***"UProg2Driver.h"*** files delivered in the software package of the programmer.

7.2.3.2.2.1 as5172InitPsi5

Table 7-37: Function list

Function	
This function initializes the UART-over-PSI5 interface for AS5172.	
Function call	
`uint8_t as5172InitPsi5(char com_port_in[], uint8_t error_in, double vpsi5h_in, double vpsi5l_in, uint16_t baudrate_in)`	
Transfer values	
Parameter name	**Description**
`char com_port_in[]`	Serial COM port of the programmer (e.g. COM2)
`double vpsi5h_in`	High level voltage of the UART-over-PSI5 signal (typ. 10V)
`double vpsi5l_in`	Low level voltage of the UART-over-PSI5 signal (typ. 7V)
`uint8_t error_in`	Error In
`uint16_t baudrate_in`	Baudrate for the UART-over-PSI5 communication (typ. 38400 baud)
Return value	
`uint8_t error_out`	Status: For detailed error information please refer to chapter 7.2.4 Error codes

7.2.3.2.2.2 as5172Read

Table 7-38: Function list

Function	
This function reads two consecutive registers starting at a specific address.	
Function call	
`uint8_t as5172Read(char com_port_in[], uint8_t read_register_address_in, uint8_t error_in, uint16_t *register_data_out)`	
Transfer values	
Parameter name	**Description**
`char com_port_in[]`	Serial COM port of the programmer (e.g. COM2)
`uint8_t read_register_address_in`	Read register address
`uint8_t error_in`	Error In
`uint16_t *register_data_out`	Register data of address and address+1
Return value	
`uint8_t error_out`	Status: For detailed error information please refer to chapter 7.2.4 Error codes

7.2.3.2.2.3 as5172Write

Table 7-39: Function list

Function	
This function writes two consecutive registers starting at a specific address.	
Function call	
`uint8_t as5172Write(char com_port_in[], uint8_t error_in, uint8_t write_register_address_in, uint16_t write_register_data_in)`	
Transfer values	
Parameter name	**Description**
`char com_port_in[]`	Serial COM port of the programmer (e.g. COM2)
`uint8_t write_register_address_in`	Write register address
`uint16_t write_register_data_in`	Register data for address and address+1
`uint8_t error_in`	Error In
Return value	
`uint8_t error_out`	Status: For detailed error information please refer to chapter 7.2.4 Error codes

7.2.3.2.2.4 as5172ReadArray

Table 7-40: Function list

Function	
This function reads n number of registers starting at a specific address.	
Function call	
`uint8_t as5172ReadArray(char com_port_in[], uint8_t error_in, uint8_t read_start_address_in, uint8_t byte_count_in, uint8_t data_out[], int32_t data_out_len)`	
Transfer values	
Parameter name	**Description**
`char com_port_in[]`	Serial COM port of the programmer (e.g. COM2)
`uint8_t read_start_address_in`	Read start address
`uint8_t byte_count_in`	Number of bytes to be read
`uint8_t error_in`	Error In
`uint8_t data_out[]`	Read data array
`int32_t data_out_len`	Length of array
Return value	
`uint8_t error_out`	Status: For detailed error information please refer to chapter 7.2.4 Error codes

7.2.3.2.2.5 as5172WriteArray

Table 7-41: Function list

Function	
This function writes a data array of n values starting at a specific register address.	
Function call	
`uint8_t as5172WriteArray(char com_port_in[], uint8_t write_start_address_in, uint8_t data_in[], uint8_t error_in, int32_t data_in_len)`	
Transfer values	
Parameter name	**Description**
`char com_port_in[]`	Serial COM port of the programmer (e.g. COM2)
`uint8_t write_start_address_in`	Write start address for the data array
`uint8_t data_in[]`	Data array (n bytes)
`uint8_t error_in`	Error In
`int32_t data_in_len`	Length of array
Return value	
`uint8_t error_out`	Status: For detailed error information please refer to chapter 7.2.4 Error codes

7.2.3.2.2.6 as5172ReadCordic

Table 7-42: Function list

Function	
This function reads the CORDIC (14-bit angle) register of the AS5172.	
Function call	
`uint8_t as5172ReadCordic(char com_port_in[], uint8_t error_in, uint8_t byte_count_in, uint16_t data_out[], int32_t data_out_len)`	
Transfer values	
Parameter name	**Description**
`char com_port_in[]`	Serial COM port of the programmer (e.g. COM2)
`uint8_t byte_count_in`	Number of CORDIC reads
`uint8_t error_in`	Error In
`uint16_t data_out[]`	Read CORDIC data array
`int32_t data_out_len`	Length of array
Return value	
`uint8_t error_out`	Status: For detailed error information please refer to chapter 7.2.4 Error codes

7.2.3.2.2.7 as5172ReadAgcMag

Table 7-43: Function list

Function	
This function reads the AGC (8-bit) and Magnitude (8-bit) register of the AS5172.	
Function call	
`uint8_t as5172ReadAgcMag(char com_port_in[], uint8_t error_in, uint8_t byte_count_in, uint8_t mag_data_out[], uint8_t agc_data_out[], int32_t mag_data_out_len, int32_t agc_data_out_len)`	
Transfer values	
Parameter name	**Description**
`char com_port_in[]`	Serial COM port of the programmer (e.g. COM2)
`uint8_t byte_count_in`	Number of AGC and Magnitude reads
`uint8_t error_in`	Error In
`uint8_t mag_data_out[]`	Read Magnitude array
`uint8_t agc_data_out[]`	Read AGC array
`int32_t mag_data_out_len`	Length of Magnitude array
`int32_t agc_data_out_len`	Length of AGC array
Return value	
`uint8_t error_out`	Status: For detailed error information please refer to chapter 7.2.4 Error codes

7.2.3.2.2.8 as5172Pass2Func

Table 7-44: Function list

Function	
This function sets the AS5172 into PASS2FUNC mode. This mode is needed for temporary testing before burning the OTP fuses.	
Function call	
`uint8_t as5172Pass2Func(char com_port_in[], uint8_t error_in)`	
Transfer values	
Parameter name	**Description**
`char com_port_in[]`	Serial COM port of the programmer (e.g. COM2)
`uint8_t error_in`	Error In
Return value	
`uint8_t error_out`	Status: For detailed error information please refer to chapter 7.2.4 Error codes

7.2.3.2.2.9 as5172BurnOtp

Table 7-45: Function list

Function	
This function permanently burns the OTP fuses of the AS5172 and does the verification according the AS5172 datasheet programming flow.	
Function call	
`uint8_t as5172BurnOtp(char com_port_in[], uint8_t error_in, uint8_t register_0x01_in, uint8_t register_0x0C_to_0x1F_in[], char status_out[], int32_t register_0x0C_to_0x1F_len, int32_t status_out_len)`	
Transfer values	
Parameter name	**Description**
`char com_port_in[]`	Serial COM port of the programmer (e.g. COM2)
`uint8_t register_0x01_in`	Register data for address 0x01
`uint8_t register_0x0C_to_0x1F_in[]`	Register data array for address 0x0C to 0x1F
`uint8_t error_in`	Error In
`char status_out[]`	OTP verification status
`int32_t register_0x0C_to_0x1F_in_len`	Length of array
`int32_t status_out_len`	Length of array
Return value	
`uint8_t error_out`	Status: For detailed error information please refer to chapter 7.2.4 Error codes

7.2.3.3 AS5270

7.2.3.3.1 LabVIEW

All LabVIEW drivers for AS5270 can be found in the AS5270.llb delivered in the software package of the programmer. Before using the specific AS5270 VIs the UART interface needs to be initialized.
The necessary ***"uartDualInit.vi"*** can be found in the ***"Uprog_StandardSerial.llb".***

7.2.3.3.1.1 uartDualInit

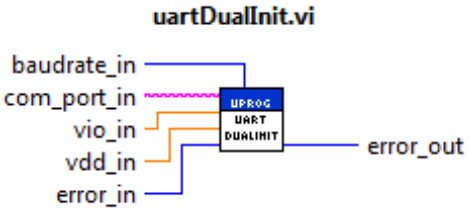


Table 7-46: Parameter list

Description			
This VI initializes the UART interface for AS5270.			
Parameter name	**Data type**	**Type**	**Description**
com_port_in	char	in	Serial COM port of the programmer (e.g. COM2)
baudrate_in	uint32	in	Baudrate for the UART (typ. 9600 baud)
vio_in	double	in	Voltage for IO of the UART (typ. 5V)
vdd_in	double	in	Voltage for sensor supply (typ. 5V)
error_in	uint8	in	Error bus input
error_out	uint8	out	Error bus output

7.2.3.3.1.2 as5270Read

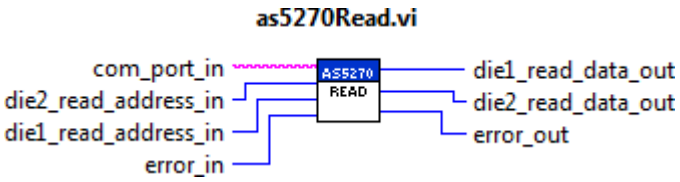


Table 7-47: Parameter list

Description			
This VI reads two consecutive registers starting at a specific address.			
Parameter name	**Data type**	**Type**	**Description**
com_port_in	char	in	Serial COM port of the programmer (e.g. COM2)
die1_read_address_in	uint8	in	Read register address (Sensor die1)
die2_read_address_in	uint8	in	Read register address (Sensor die2)
error_in	uint8	in	Error bus input
die1_read_data_out	uint16	out	Register data of address and address+1 (Sensor die1)
die2_read_data_out	uint16	out	Register data of address and address+1 (Sensor die2)
error_out	uint8	out	Error bus output

7.2.3.3.1.3 as5270Write

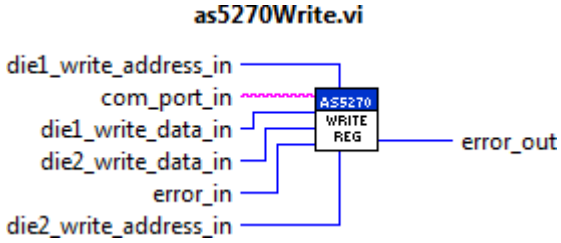


Table 7-48: Parameter list

Description			
This VI writes two consecutive registers starting at a specific address.			
Parameter name	**Data type**	**Type**	**Description**
com_port_in	char	in	Serial COM port of the programmer (e.g. COM2)
die1_write_address_in	uint8	in	Write register address (Sensor die1)
die2_write_address_in	uint8	in	Write register address (Sensor die2)
die1_write_data_in	uint16	in	Register data for address and address+1 (Sensor die1)
die2_write_data_in	uint16	in	Register data for address and address+1 (Sensor die2)
error_in	uint8	in	Error bus input
error_out	uint8	out	Error bus output

7.2.3.3.1.4 as5270ReadArray

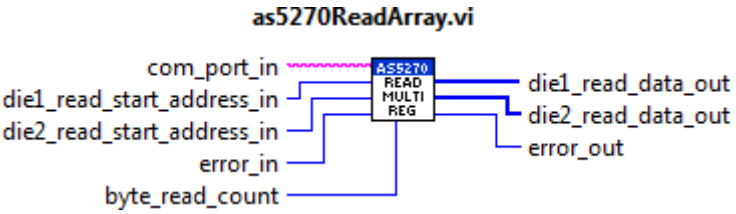


Table 7-49: Parameter list

Description			
This VI reads n number of registers starting at a specific address.			
Parameter name	**Data type**	**Type**	**Description**
com_port_in	char	in	Serial COM port of the programmer (e.g. COM2)
die1_read_start_address_in	uint8	in	Read start address (Sensor die1)
die2_read_start_address_in	uint8	in	Read start address (Sensor die2)
byte_read_count	uint8	in	Number of bytes to be read
error_in	uint8	in	Error bus input
die1_read_data_out	uint8	out	Read data array (Sensor die1)
die2_read_data_out	uint8	out	Read data array (Sensor die2)
error_out	uint8	out	Error bus output

7.2.3.3.1.5 as5270WriteArray

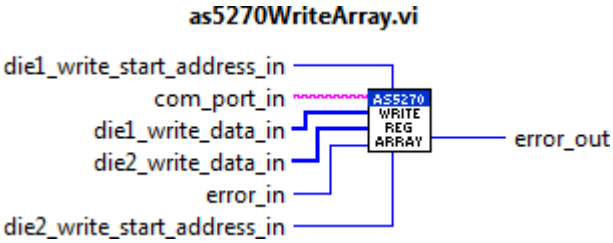


Table 7-50: Parameter list

Description			
This VI writes a data array of n values starting at a specific register address.			
Parameter name	**Data type**	**Type**	**Description**
com_port_in	char	in	Serial COM port of the programmer (e.g. COM2)
die1_write_start_address_in	uint8	in	Write start address for the data array (Sensor die1)
die2_write_start_address_in	uint8	in	Write start address for the data array (Sensor die2)
die1_write_data_in	uint8	in	Data array (n bytes) (Sensor die1)
die2_write_data_in	uint8	in	Data array (n bytes) (Sensor die2)
error_in	uint8	in	Error bus input
error_out	uint8	out	Error bus output

7.2.3.3.1.6 as5270ReadCordic

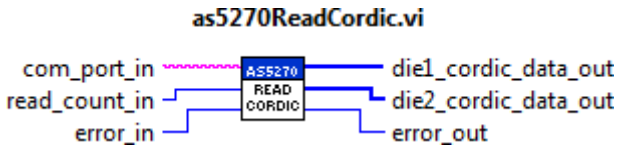


Table 7-51: Parameter list

Description			
This VI reads the CORDIC (14-bit angle) register of the AS5270.			
Parameter name	**Data type**	**Type**	**Description**
com_port_in	char	in	Serial COM port of the programmer (e.g. COM2)
read_count_in	uint8	in	Number of CORDIC reads
error_in	uint8	in	Error bus input
die1_cordic_data_out	uint16	out	Read CORDIC data array (Sensor die1)
die2_cordic_data_out	uint16	out	Read CORDIC data array (Sensor die2)
error_out	uint8	out	Error bus output

7.2.3.3.1.7 as5270ReadAgcMag

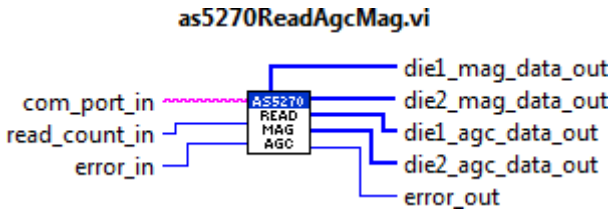


Table 7-52: Parameter list

Description			
This VI reads the AGC (8-bit) and Magnitude (8-bit) register of the AS5270.			
Parameter name	**Data type**	**Type**	**Description**
com_port_in	char	in	Serial COM port of the programmer (e.g. COM2)
read_count_in	uint8	in	Number of AGC and Magnitude reads
error_in	uint8	in	Error bus input
die1_mag_data_out	uint16	out	Read Magnitude array (Sensor die1)
die2_mag_data_out	uint16	out	Read Magnitude array (Sensor die2)
die1_agc_data_out	uint16	out	Read AGC array (Sensor die1)
die2_agc_data_out	uint16	out	Read AGC array (Sensor die2)
error_out	uint8	out	Error bus output

7.2.3.3.1.8 as5270Pass2Func

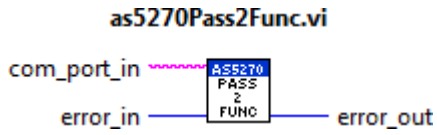


Table 7-53: Parameter list

Description			
This VI sets the AS5270 into PASS2FUNC mode. This mode is needed for temporary testing before burning the OTP fuses.			
Parameter name	**Data type**	**Type**	**Description**
com_port_in	char	in	Serial COM port of the programmer (e.g. COM2)
error_in	uint8	in	Error bus input
error_out	uint8	out	Error bus output

7.2.3.3.1.9 as5270Burn

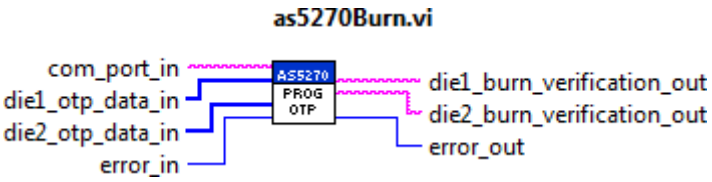


Table 7-54: Parameter list

Description			
This VI permanently burns the OTP fuses of the AS5270 and does the verification according the AS5270 datasheet programming flow.			
Parameter name	**Data type**	**Type**	**Description**
com_port_in	char	in	Serial COM port of the programmer (e.g. COM2)
die1_otp_data_in	uint8	in	OTP register data for address 0x0A to 0x1E (Sensor die1)
die2_otp_data_in	uint8	in	OTP register data for address 0x0A to 0x1E (Sensor die2)
error_in	uint8	in	Error bus input
die1_burn_verification_out	char	out	OTP verification status (Sensor die1)
die2_burn_verification_out	char	out	OTP verification status (Sensor die2)
error_out	uint8	out	Error bus output

7.2.3.3.2 DLL

The DLL functions described hereafter can be found in the ***"UProg2Driver.dll"*** and ***"UProg2Driver.h"*** files delivered in the software package of the programmer.

7.2.3.3.2.1 uartDualInit

Table 7-55: Function list

Function	
This function initializes the UART interface for AS5270.	
Function call	
`uint8_t uartDualInit(char com_port_in[], uint8_t error_in, double vio_in, double vdd_in, uint32_t baudrate_in)`	
Transfer values	
Parameter name	**Description**
`char com_port_in[]`	Serial COM port of the programmer (e.g. COM2)
`double vio_in`	Voltage for IO of the UART (typ. 3900 dec. = 5V)
`double vdd_in`	Voltage for sensor supply (typ. 3900 dec. = 5V)
`uint8_t error_in`	Error In
`double baudrate_in`	Baudrate for the UART (typ. 9600 baud)
Return value	
`uint8_t error_out`	Status: For detailed error information please refer to chapter 7.2.4 Error codes

7.2.3.3.2.2 as5270Read

Table 7-56: Function list

Function	
This function reads two consecutive registers starting at a specific address.	
Function call	
`uint8_t as5270Read(char com_port_in[], uint8_t error_in, uint8_t die1_read_address_in, uint8_t die2_read_address_in, uint16_t *die1_read_data_out, uint16_t *die2_read_data_out)`	
Transfer values	
Parameter name	**Description**
`char com_port_in[]`	Serial COM port of the programmer (e.g. COM2)
`uint8_t die2_read_address_in`	Read register address (Sensor die2)
`uint8_t die1_read_address_in`	Read register address (Sensor die1)
`uint8_t error_in`	Error In
`uint16_t *die1_read_data_out`	Register data of address and address+1 (Sensor die1)
`uint16_t *die2_read_data_out`	Register data of address and address+1 (Sensor die2)
Return value	
`uint8_t error_out`	Status: For detailed error information please refer to chapter 7.2.4 Error codes

7.2.3.3.2.3 as5270Write

Table 7-57: Function list

Function	
This function writes two consecutive registers starting at a specific address.	
Function call	
`uint8_t as5270Write(char com_port_in[], uint8_t error_in, uint8_t die1_write_address_in, uint8_t die2_write_address_in, uint16_t die2_write_data_in, uint16_t die1_write_data_in)`	
Transfer values	
Parameter name	**Description**
`char com_port_in[]`	Serial COM port of the programmer (e.g. COM2)
`uint8_t die2_write_address_in`	Write register address (Sensor die2)
`uint8_t die1_write_address_in`	Write register address (Sensor die1)
`uint16_t die2_write_data_in`	Register data for address and address+1 (Sensor die2)
`uint16_t die1_write_data_in`	Register data for address and address+1 (Sensor die1)
`uint8_t error_in`	Error In
Return value	
`uint8_t error_out`	Status: For detailed error information please refer to chapter 7.2.4 Error codes

7.2.3.3.2.4 as5270ReadArray

Table 7-58: Function list

Function	
This function reads n number of registers starting at a specific address.	
Function call	
`uint8_t as5270ReadArray(char com_port_in[], uint8_t error_in, uint8_t die1_read_start_address_in, uint8_t die2_read_start_address_in, uint8_t byte_read_count, uint8_t die1_read_data_out[], uint8_t die2_read_data_out[], int32_t die1_read_data_out_len, int32_t die2_read_data_out_len)`	
Transfer values	
Parameter name	**Description**
`char com_port_in[]`	Serial COM port of the programmer (e.g. COM2)
`uint8_t die1_read_start_address_in`	Read start address (Sensor die1)
`uint8_t die2_read_start_address_in`	Read start address (Sensor die2)
`uint8_t byte_read_count`	Number of bytes to be read
`uint8_t die1_read_data_out[]`	Read data array (Sensor die1)
`uint8_t die2_read_data_out[]`	Read data array (Sensor die2)
`int32_t die1_read_data_out_len`	Length of array
`int32_t die2_read_data_out_len`	Length of array
`uint8_t error_in`	Error In
Return value	
`uint8_t error_out`	Status: For detailed error information please refer to chapter 7.2.4 Error codes

7.2.3.3.2.5 as5270WriteArray

Table 7-59: Function list

Function	
This function writes a data array of n values starting at a specific register address.	
Function call	
`uint8_t as5270WriteArray(char com_port_in[], uint8_t error_in, uint8_t die1_write_data_in[], uint8_t die2_write_data_in[], uint8_t die1_write_start_address_in, uint8_t die2_write_start_address_in, int32_t die1_write_data_in_len, int32_t die2_write_data_in_len)`	
Transfer values	
Parameter name	**Description**
`char com_port_in[]`	Serial COM port of the programmer (e.g. COM2)
`uint8_t die1_write_start_address_in`	Write start address for the data array (Sensor die1)
`uint8_t die2_write_start_address_in`	Write start address for the data array (Sensor die2
`uint8_t die1_write_data_in[]`	Data array (n bytes) (Sensor die1)
`uint8_t die2_write_data_in[]`	Data array (n bytes) (Sensor die2)
`int32_t die1_write_data_in_len`	Length of array
`int32_t die2_write_data_in_len`	Length of array
`uint8_t error_in`	Error In
Return value	
`uint8_t error_out`	Status: For detailed error information please refer to chapter 7.2.4 Error codes

7.2.3.3.2.6 as5270ReadCordic

Table 7-60: Function list

Function	
This function reads the CORDIC (14-bit angle) register of the AS5270.	
Function call	
`uint8_t as5270ReadCordic(char com_port_in[], uint8_t error_in, uint8_t read_count_in, uint16_t die1_cordic_data_out[], uint16_t die2_cordic_data_out[], int32_t die1_cordic_data_out_len, int32_t die2_cordic_data_out_len)`	
Transfer values	
Parameter name	**Description**
`char com_port_in[]`	Serial COM port of the programmer (e.g. COM2)
`uint8_t read_count_in`	Number of CORDIC reads
`uint16_t die1_cordic_data_out[]`	Read CORDIC data array (Sensor die1)
`uint16_t die2_cordic_data_out[]`	Read CORDIC data array (Sensor die2)
`int32_t die1_cordic_data_out_len`	Length of array
`int32_t die2_cordic_data_out_len`	Length of array
`uint8_t error_in`	Error In
Return value	
`uint8_t error_out`	Status: For detailed error information please refer to chapter 7.2.4 Error codes

7.2.3.3.2.7 as5270ReadAgcMag

Table 7-61: Function list

Function	
This function reads the AGC (8-bit) and Magnitude (8-bit) register of the AS5270.	
Function call	
`uint8_t as5270ReadAgcMag(char com_port_in[], uint8_t error_in, uint8_t read_count_in, uint16_t die1_agc_data_out[], uint16_t die2_agc_data_out[], uint16_t die1_mag_data_out[], uint16_t die2_mag_data_out[], int32_t die1_agc_data_out_len, int32_t die2_agc_data_out_len, int32_t die1_mag_data_out_len, int32_t die2_mag_data_out_len)`	
Transfer values	
Parameter name	**Description**
`char com_port_in[]`	Serial COM port of the programmer (e.g. COM2)
`uint8_t read_count_in`	Number of AGC and Magnitude reads
`uint16_t die1_agc_data_out[]`	Read AGC array (Sensor die1)
`uint16_t die2_agc_data_out[]`	Read AGC array (Sensor die2)
`uint16_t die1_mag_data_out[]`	Read Magnitude array (Sensor die1)
`uint16_t die2_mag_data_out[]`	Read Magnitude array (Sensor die2)
`int32_t die1_agc_data_out_len`	Length of array
`int32_t die2_agc_data_out_len`	Length of array
`int32_t die1_mag_data_out_len`	Length of array
`int32_t die2_mag_data_out_len`	Length of array
`uint8_t error_in`	Error In
Return value	
`uint8_t error_out`	Status: For detailed error information please refer to chapter 7.2.4 Error codes

7.2.3.3.2.8 as5270Pass2Func

Table 7-62: Function list

Function	
This function sets the AS5270 into PASS2FUNC mode. This mode is needed for temporary testing before burning the OTP fuses.	
Function call	
`uint8_t as5270Pass2Func(char com_port_in[], uint8_t error_in)`	
Transfer values	
Parameter name	**Description**
`char com_port_in[]`	Serial COM port of the programmer (e.g. COM2)
`uint8_t error_in`	Error In
Return value	
`uint8_t error_out`	Status: For detailed error information please refer to chapter 7.2.4 Error codes

7.2.3.3.2.9 as5270Burn

Table 7-63: Function list

Function	
This function permanently burns the OTP fuses of the AS5270 and does the verification according the AS5270 datasheet programming flow.	
Function call	
`uint8_t as5270Burn(char com_port_in[], uint8_t error_in, uint8_t die1_otp_data_in[], uint8_t die2_otp_data_in[], char die1_burn_verification_out[], char die2_burn_verification_out[], int32_t die1_otp_data_in_len, int32_t die2_otp_data_in_len, int32_t die1_burn_verification_out_len, int32_t die2_burn_verification_out_len)`	
Transfer values	
Parameter name	**Description**
`char com_port_in[]`	Serial COM port of the programmer (e.g. COM2)
`uint8_t die1_otp_data_in[]`	OTP register data for address 0x0A to 0x1E (Sensor die1)
`uint8_t die2_otp_data_in[]`	OTP register data for address 0x0A to 0x1E (Sensor die2)
`char die1_burn_verification_out[]`	OTP verification status (Sensor die1)
`char die2_burn_verification_out[]`	OTP verification status (Sensor die2)
`int32_t die1_otp_data_in_len`	Length of array
`int32_t die2_otp_data_in_len`	Length of array
`int32_t die1_burn_verification_out_len`	Length of array
`int32_t die2_burn_verification_out_len`	Length of array
`uint8_t error_in`	Error In
Return value	
`uint8_t error_out`	Status: For detailed error information please refer to chapter 7.2.4 Error codes

7.2.4 Error codes

Following table shows all error codes which can occur when using the “UProg2” Programmer.

Table 7-64: Host interfaces

Error code	Description
0x01	No error
0x02	Serial communication error/UART error
0x03	Response timeout
0x04	Wrong response received

8 Legal information

8.1 Warranty Information

This product “UProg2_1v1” is still in the hardware and software status “prototype” and may be used for laboratory evaluation only. SmartDesign4you Ltd. guarantees the functions of the PSI5 interface as long as the programmer is used in the laboratory environment.

If the UProg2_1v1 fails due to faulty workmanship or materials it has to be shipped back to SmartDesign4you. In this case the customer has to take all shipping and customs clearance costs for returning the programmer to SmartDesign4you.

If the defect is due to improper damage or misuse as well as changes or repairs by unauthorized persons the guarantee is not valid any more. The warranty exceptions are listed in section 8.1.1 Warranty Exceptions.

8.1.1 Warranty Exceptions

Damage caused by other Devices
The warranty does not cover damages to the programmer due to wrong connections to different devices.

Short circuit damage
The warranty does not cover any damage to the programmer if there is a short circuit load somewhere on the connectors.

Over voltage damage
The warranty does not cover damage to the programmer if the specified voltage limits are not respected.

Over current damage
The warranty does not cover damage to the programmer when overcurrent is drawn from the power supply. The user is responsible for overcurrent protection.

WARNING!
Any damage caused by Electrostatic Discharge (ESD) to the programmer because of improper handling is not covered by the warranty.

8.2 Disclaimer

Every effort has been made to ensure that programming algorithms are correct at the time of their release. SmartDesign4you makes every effort and tries to endeavor to rectify any programming issues as quickly as possible after a validated fault report is received.

It is recommended that mass production always precedes quality control before production start in order to guarantee correct programming. SmartDesign4you cannot be held responsible for any third party claims which arise out of the use of this programmer including ‘consequential losses’ and ‘loss of profit’.

SmartDesign4You Ltd. cannot be held responsible for any programming problems which are ‘out of our control’.

9 Revision History

Revision history			
Date	Revi-sion	Comments	Author
12.06.2017	0.05	Initial Version	mzie
26.06.2017	0.06	Updated Key Features, Electrical Specification, Host Interfaces (USB/RS232), Device Connection and pinout Reorganized chapter 7	mzie
11.07.2017	0.10	Updated LabVIEW driver and DLL description	mzie
12.07.2017	0.11	Updated LabVIEW driver and DLL description	mzie
18.07.2017	0.12	Added Error code chapter	mzie
25.04.2018	0.20	Added support for AS5170/AS5171	mzie
10.02.2025	0.21	Update Device interfaces	aeld