

**AN514X-10 PPTRIM Programming
MAGNETIC ROTARY ENCODER
OTP Programming Guide**

APPLICATION NOTE

This application note describes the available options for PPTRIM programming of all related AS51xx devices:

- AS5140
- AS5145
- AS5245

Starting with the general permanent programming of the OTP memory register, it also describes how to do a “soft writing” for single or multiple non-permanent writing of the OTP. In addition the load function and the features for verification after programming are included.

For an overall description of the device, please refer to the relevant datasheet.

Note: The pin PROG at the AS5140 is identical to the PDIO pin of the AS5145.

1 Hardware Connections for OTP Memory Programming

For OTP memory access, 3 signals are required: PROG/PDIO, CSn and CLK. To read the angle position for zero position programming, signal DO is also required. The AS5xxx can be programmed in either 3.3V or 5V mode. The related programming voltage Vprog is always between 3.3V...3.8V.

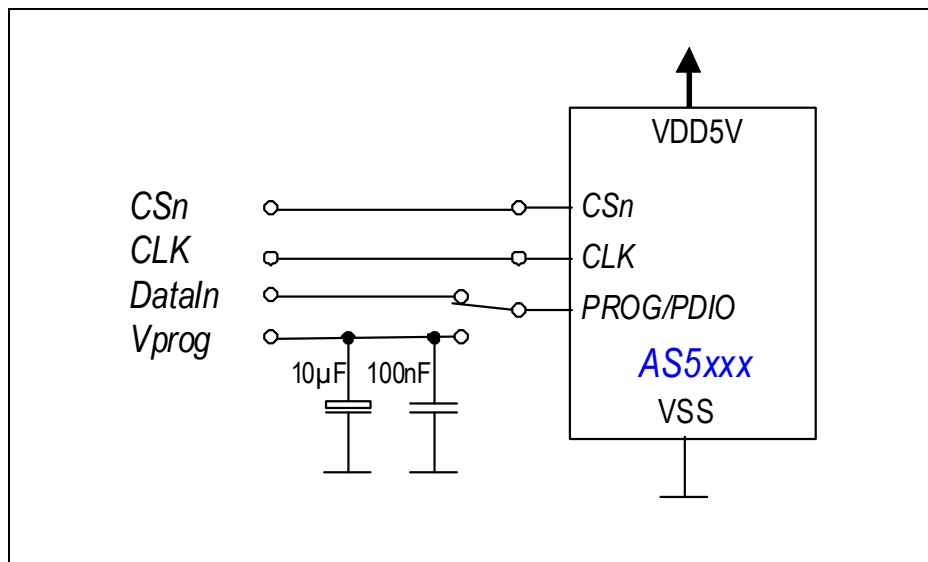


Figure 1 Programming circuit of AS5xxx

2 One Time Programmable Register (OTP)

The OTP block should add flexibility to the user. It allows to define a new zero position, and several other output modes. For internal test purpose a test mode can be activated. It is not recommended to change the factory settings of the device.

By default the peripheral pins CSn, CLK and PROG/PDIO are used for the SSI interface. To access the OTP block a special setup condition must be performed to access the block. Another exit condition must be performed to get out of the access!

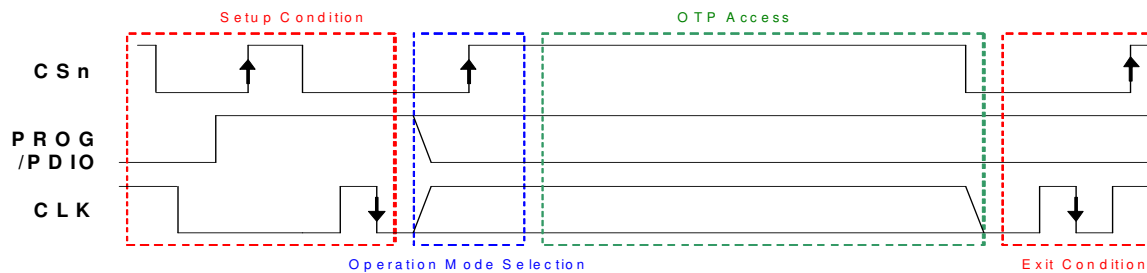


Figure 2 OTP register input and exit condition

2.1 AUTOLOAD Operation

This mode transfers the permanent stored data from each PolyFuse to the corresponding register. The outputs of the registers are internally available and buffered in parallel format. The autoload operation is performed at every power on sequence.

2.2 LOAD Operation

- Transfer DATA from PolyFuse to registers on demand.

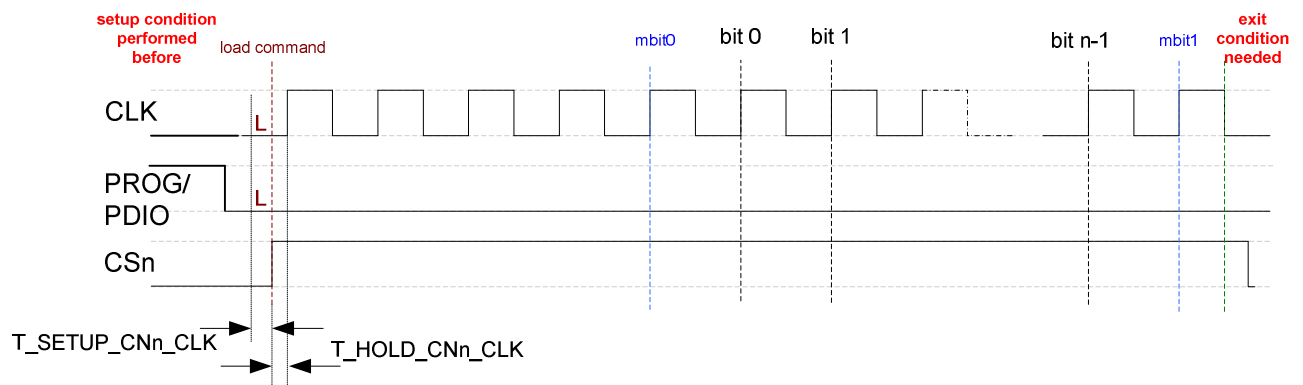


Figure 3 LOAD operation

At CLK=LOW and PROG/PDIO=LOW the rising edge on CSn signal latches the load command internally. Each rising edge at CLK gets another bit stored into the register. This mode is not recommended in application because the asynchronously change (bit after bit). A power down and AUTOLOAD can be used instead.

2.3 WRITE Operation

- Write DATA from External PROG/PDIO Pad to Register
- Operation stops automatically after DATA is latched
- DATA must be stable at rising edge of CLK

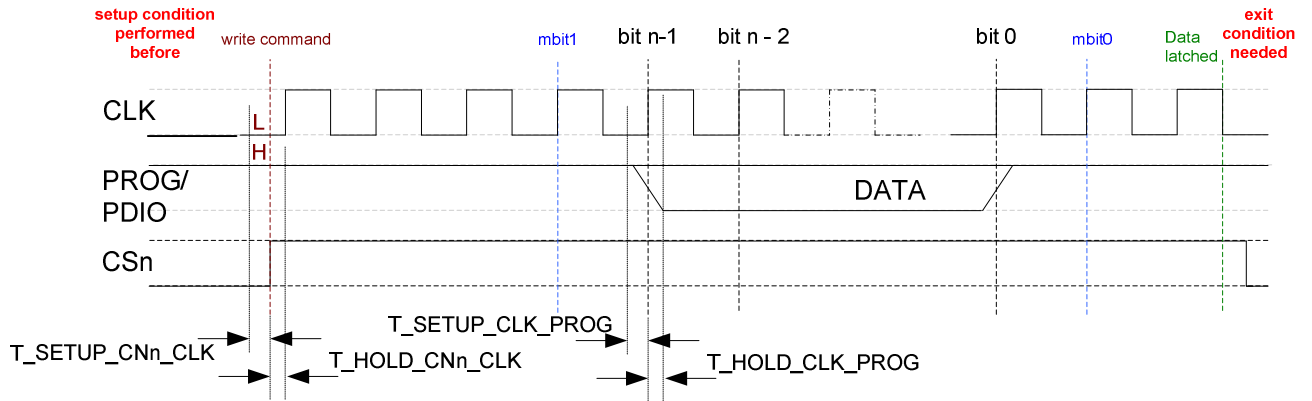


Figure 4 WRITE operation

This operation is defined by CLK=LOW and PROG/PDIO=HIGH during rising edge of CSn signal. The signal PROG/PDIO works as input and has to be stable at rising edges of CLK signal. All external data will be shifted into an internal shift register.

2.4 READ Operation

- Read DATA from Registers at PROG/PDIO
- DATA is looping until CSn stops operation

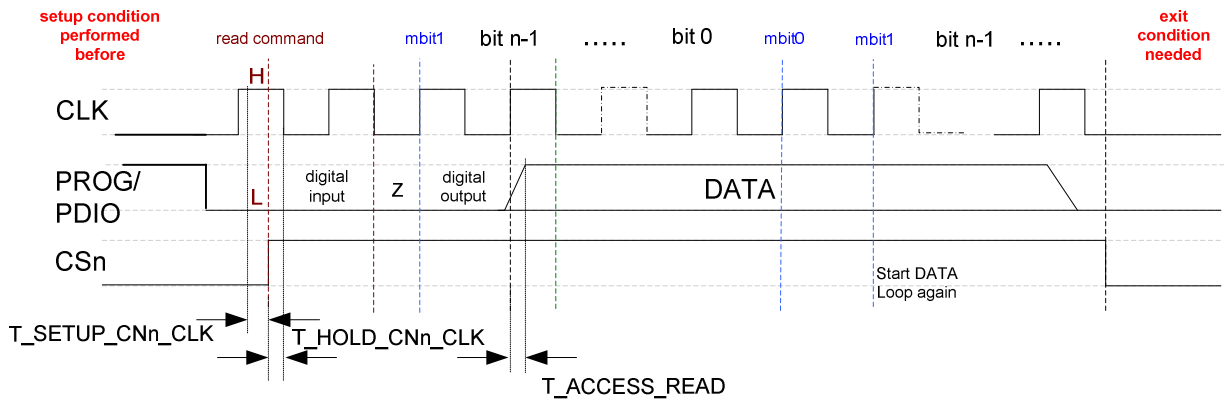


Figure 5 READ operation

This operation performs a serial read of the register data and is defined at CLK=HIGH and PROG/PDIO=LOW during rising edge of CSn signal. The signal PROG/PDIO works in data out direction after the second rising edge at CLK.

2.5 PROG Operation

- Permanent Storage of DATA from registers to PolyFuse

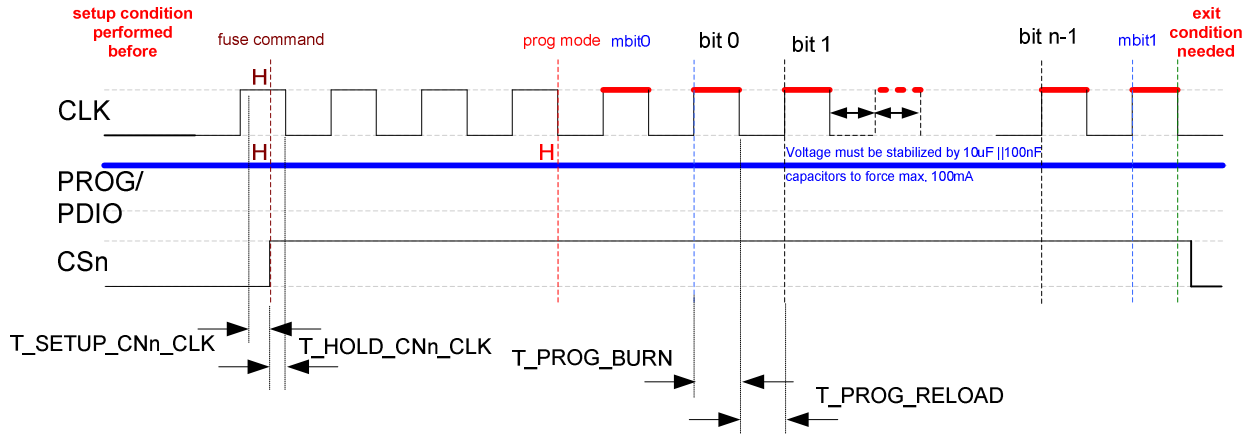


Figure 6 PROG operation

During this operation the data written into the Latch during write cycle operation are permanently stored in the PolyFuse cell. PROG/PDIO=HIGH and CLK=HIGH during rising edge of CSn signal starts this operation. The fusing is managed bit by bit due to the high current need for fusing. The programming time of PolyFuse cell is defined by the high pulse of CLK (PROG_BURN). As there is a maximum current of 100mA needed for permanently fusing, PROG/PDIO has to be stabilized by some capacitance during this operation. The low time of CLK (PROG_RELOAD) allows to recharge the capacitor. This is useful if the source has a limited current and can be compensated by increasing this time.

2.6 ANALOG Operation

Note: This operation is mandatory after programming. Every programmed fuse must be verified with this procedure!

- Reading Resistance of PolyFuse

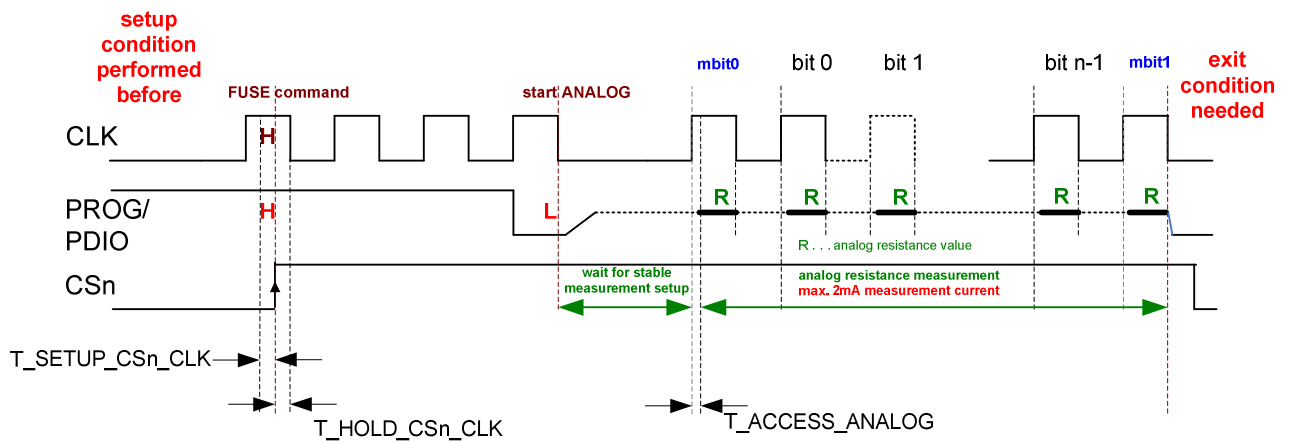


Figure 7 ANALOG operation

This operation is defined in a similar way than the PROG operation. At the 4th falling edge of CLK, PROG/PDIO must be LOW, further on internal switches enable the analogue resistance measurement between PROG/PDIO pad and GND. One rising edges of CLK later, the first PolyFuse resistance (mbit0) can be measured during CLK= HIGH. At each following CLK=HIGH, one PolyFuse element after the other is measured.

During resistance measurement, the maximum current must not exceed 2mA, to avoid unwanted programming to the PolyFuse elements. This analogue measurement is necessary because of:

- Guard band for the comparator level
- Guard band for a maximum lifetime drift
- Check of a low ohmic programming path

Resistance values and comparator level are within the following limits:

- Unprogrammed PolyFuse resistance: 50Ω-200Ω
- Comparator Level: 300Ω-2KΩ 1)
- Programmed PolyFuse resistance: >10KΩ

There are two possibilities to do resistance measurements:

Setting a fixed voltage of 100mV at PROG/PDIO and measuring of the current at CLK=HIGH.

This is the preferred measurement method to get exact resistance values.

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
I_{unprog}	Current for unprogrammed PolyFuse ²⁾	$V_{PDIO} = 100mV$	0.5	1.4	2	mA
I_{prog}	Current for programmed PolyFuse ²⁾	$V_{PDIO} = 100mV$	-	0.1	10	μA

Forcing a constant current of 200μA into PROG/PDIO pad and measuring of the voltage at CLK=HIGH.

This is the preferred measurement method for a fast implementation.

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
V_{unprog}	Voltage for unprogrammed PolyFuse ³⁾	$I_{PDIO} = 200\mu A$	10	14	40	mV
V_{prog}	Voltage for programmed PolyFuse ³⁾	$I_{PDIO} = 200\mu A$	$V_{DD}-40mV$	V_{DD}	-	V

1) Comparator level can not be measured from external 2) Current compliance must be set to 2mA 3) Voltage compliance must be set to V_{DD}

3 Operating Conditions

3.1 PROG Operation

PARAMETER	SYMBOL	MIN	MAX	UNIT	NOTE
Positive Supply Voltage (1) (=Programming Voltage)	VDD	3.3 3.5	3.6 3.6	V V	Positive supply for Temp < 70°C Positive supply for Temp < 150°C
Negative Supply Voltage	VSS	0V	0V	V	Ground =0V
Junction Temperature	Tjunction	0	150	°C	High temperature can cause a higher programming yield loss !
Programming Current into PROG/PDIO	I _{PROG,prog}		100	mA	required current to program a single PolyFuse element (2)
Programming refresh time	t _{PROG_REFRE SH}	1.0		us	minimal refresh time during fusing – pad CLK (3)
The PolyFuse cell can be programmed only once					

Notes:

- (1) The supply voltage must be fixed in the same range than the programming voltage
- (2) The PolyFuse cells are programmed in a bit by bit sequence due to the high current at fusing.
- (3) Refreshing time of the external capacitor depends on the compliance current of the fusing voltage source.

3.2 AC Characteristics: Timing Specifications

PARAMETER	PARAMETER	MIN	TYP	MAX	UNIT
T_SETUP_CS _n _CLK	Setup time for operation	5			ns
T_HOLD_CS _n _CLK	Hold time for operation	10			ns
T_SETUP_CLK_PROG	Setup time for data	5			ns
T_HOLD_CLK_PROG	Hold time for data	10			ns
T_ACCESS_READ	Access time for data			60	ns
T_PROG_BURN	Programming time	10	15	20	µs
T_PROG_RELOAD	Reload time of programming cap.	10			µs
T_ACCESS_ANALOG	Delay time of analog read of PolyFuse			10	µs

3.3 Operating Conditions for ANALOG Operation

PARAMETER	SYMBOL	MIN	MAX	UNIT	NOTE
Analog Voltage	V _{analog}	90	100	mV	Set on PROG(1)
Analog Current Prog. Fuse	I _{ana,1}	0	10	µA	10kOhms - infinite(1)
Analog Current Unprog. Fuse	I _{ana,0}	1	2	mA	50 - 100 Ohms (1)
Setup time for operation mode	T_Setup_MODE_CLK	5		ns	
Hold time for operation mode	T_HOLD_MODE_CLK	10		ns	
Delay time of analog read of Polyfuse	T_ACCESS_ANALOG		10	µs	

Notes: (1) The resistor or the PolyFuse can be calculated by setting a maximal voltage of 100mV on PROG/PDIO during PROG operation and measuring the current into the pin.

4 Test Sequence

4.1.1 Check AUTOLOAD

- 1) AUTOLOAD operation triggered with Power Up
- 2) READ operation: Data must be 0 for all bits (for non programmed devices)

4.1.2 Check Serial Interface

- 3) WRITE operation of checkerboard pattern (1010..)
- 4) READ operation data has to read checkerboard pattern
- 5) WRITE operation of anti checkerboard pattern (0101..)
- 6) READ operation data has to read anti checkerboard pattern

4.1.3 Check Comparator

- 7) WRITE operation: Every bit high (1111..)
- 8) LOAD operation performed to non programmed PolyFuses
- 9) READ operation: Data must be 0 for all bits (for non programmed devices)

4.1.4 Get Trimming Pattern

- 10) Test of user specific non trimmed parameter.
- 11) WRITE operation to trim parameter.

4.1.5 Program PolyFuses

- 12) PROG operation performed to program trimming pattern into PolyFuses.

4.1.6 Check programmed Pattern

- 13) AUTOLOAD operation (LOAD operation if AUTOLOAD will be performed on later tests)
- 14) READ operation: Compare digital pattern with trimming pattern
- 15) ANALOG operation: Read analog resistance levels for PolyFuses
- 16) Calculate maximum resistance value for non programmed PolyFuses
- 17) Calculate minimum resistance value for programmed PolyFuses
- 18) Retest of user specific trimmed parameter.

5 Example source code

The following source code is taken from the AS5140 and AS5145 DB Demoboards firmware. The microcontroller is a SiLabs C8051F320.

Three functions are represented in chapter 5.6:

```
void pptrimLoad(unsigned char num_bits)
void pptrimRead(unsigned char *buffer, unsigned char num_bits)
void pptrimWrite(unsigned char *buffer, unsigned char num_bits)
```

5.1 Zero position programming example

On the following example an AS5145 is used. The sequence is:

1. Read 18 bit SSI value from the AS5145 to a SSI buffer (3 bytes buffer for 18 bits)
2. Read 54 bit PPTrim OTP register to a buffer (7 bytes buffer for 54 bit)
3. Write the actual angle value from SSI buffer to the Zero position field in the PPTrim buffer
4. Write back the PPTrim buffer to the AS5145
5. Read back the new 18 SSI data. The angle should be 0.

This main (void) source code uses the three PPTRIM functions described above.

PPTrimBuffer structure for AS5140:

	Bit7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Byte 0	ID11 7	ID12 6	ID13 5	ID14 4	ID15 3	ID16 2	ID17 1	MBit 0 0
Byte 1	ID3 15	ID4 14	ID5 13	ID6 12	ID7 11	ID8 10	ID9 9	ID10 8
Byte 2	FS 8 23	FS 9 22	FS 10 21	FS 11 20	FS12 19	ID0 18	ID1 17	ID2 16
Byte 3	FS0 31	FS1 30	FS2 29	FS3 28	FS 4 27	FS 5 26	FS 6 25	FS 7 24
Byte 4	Z8 39	Z9 38	CCW 37	RA0 36	RA1 35	RA2 34	RA3 33	RA4 32
Byte 5	Z0 47	Z1 46	Z2 45	Z3 44	Z4 43	Z5 42	Z6 41	Z7 40
Byte 6	-	-	MBit 1 53	Md0 52	Md1 51	Div0 50	Div1 49	Index 48

SSIBuffer structure for AS5140:

	Bit7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Byte 0	D1 7	D0 6	OCF 5	COF 4	LIN 3	MagINC 2	MagDEC 1	Parity 0
Byte 1	D9 15	D8 14	D7 13	D6 12	D5 11	D4 10	D3 9	D2 8

PPTrimBuffer structure for AS5145:

	Bit7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Byte 0	ID11 7	ID12 6	ID13 5	ID14 4	ID15 3	ID16 2	ID17 1	MBit 0 0
Byte 1	ID3 15	ID4 14	ID5 13	ID6 12	ID7 11	ID8 10	ID9 9	ID10 8
Byte 2	FS 6 23	FS 7 22	FS 8 21	FS 9 20	FS10 19	ID0 18	ID1 17	ID2 16
Byte 3	RA3 31	RA4 30	FS0 29	FS1 28	FS 2 27	FS 3 26	FS 4 25	FS 5 24
Byte 4	Z8 39	Z9 38	Z10 37	Z11 36	CCW 35	RA0 34	RA1 33	RA2 32
Byte 5	Z0 47	Z1 46	Z2 45	Z3 44	Z4 43	Z5 42	Z6 41	Z7 40
Byte 6	-	-	MBit 1 53	PWM Half 52	MagCompEN 51	PWM Dis 50	MD0 49	MD1 48

SSIBuffer structure for AS5145:

	Bit7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Byte 0	D1 7	D0 6	OCF 5	COF 4	LIN 3	MagINC 2	MagDEC 1	Parity 0
Byte 1	D9 15	D8 14	D7 13	D6 12	D5 11	D4 10	D3 9	D2 8
Byte 2	- 23	- 22	- 21	- 20	- 19	- 18	D11 17	D10 16

```

void main(void)
{
    xdata unsigned char SSIBuffer[3], PPTrimBuffer[7];
    xdata unsigned char ZeroTemp;
    xdata unsigned short angle;

    // Step 1: Read the 18 bit SSI value from the AS5145 (AS5140 uses 16 bit)
    ssiRead(SSIBuffer, 18);
    angle = extractAngleValueFromSsiBuffer(SSIBuffer);

    // Step 2: Read the 54 PPTRIM OTP bits from the AS5145
    pptrimRead(PPTrimBuffer, 54);

    // Step 3: Write the actual angle to the zero position field of the OTP bits
    ZeroTemp = (SSIBuffer[2] << 2) + (SSIBuffer[1] >> 6);
    ZeroTemp = reversebits(ZeroTemp);
    PPTrimBuffer[4] = ZeroTemp;

    ZeroTemp = (SSIBuffer[1] << 2) + (SSIBuffer[0] >> 6);
    ZeroTemp = reversebits(ZeroTemp);

```

```
PPTrimBuffer[5] = ZeroTemp;

// Step 4: Write back the 54 PPTRIM OTP bits containing the Zero position (which is
the actual angle) to the AS5145
pptrimWrite(PPTrimBuffer, 54);

// Step 5: Read the new angle. Must be 0 (+-1, if the mechanics is not very stable)
ssiRead(SSIBuffer, 18);
angle = extractAngleValueFromSsiBuffer(SSIBuffer);
}
```

5.2 Definitions: Macro

```
#define CLEAR_CSN()      do { CSN = 0; } while(0)
#define SET_CSN()       do { CSN = 1; } while(0)
#define CLEAR_CLK()     do { CLK = 0; } while(0)
#define SET_CLK()       do { CLK = 1; } while(0)
#define CLEAR_PROG()    do { PROG = 0; } while(0)
#define SET_PROG()      do { PROG = 1; } while(0)

#define PROG_HIGH_IMPED() do { POMDOUT = 0x5F; PROG = 1;} while(0) // Pushpull disabled, Hi-Z
#define PROG_LOW_IMPED() do { POMDOUT = 0xDF; } while(0) // Pushpull enabled
```

5.3 Utility functions

```
#define PPTRIMDelay 50
static void pptrimDelay(volatile unsigned int value)
{
    for(value; value>0; value--);
    {
        unsigned char foo = 30;
        while(foo--);
    }
}

void initPPTRIM()
{
    PROG_LOW_IMPED();
    CLEAR_PROG();
}

static void clkPulses(unsigned char num)
{
    unsigned char i;
    for(i = 0; i < num; i++)
    {
        SET_CLK();    pptrimDelay(PPTRIMDelay);
        CLEAR_CLK();  pptrimDelay(PPTRIMDelay);
    }
}

unsigned char reversebits(unsigned char value) // Endian switch
{
    unsigned char i=0, result=0;
    while (i<8)
    {
        result += (value<<i)&0x80;
        if (i<7) result = result >> 1;
        i++;
    }

    return result;
}
```

5.4 Setup and exit conditions

```
static void setupCondition()
```

```
{  
    CLEAR_CSN();  
    pptrimDelay(PPTRIMDelay);  
    CLEAR_CLK();  
    pptrimDelay(PPTRIMDelay);  
    SET_PROG();  
    pptrimDelay(PPTRIMDelay);  
    SET_CSN();  
    pptrimDelay(PPTRIMDelay);  
    CLEAR_CSN();  
    pptrimDelay(PPTRIMDelay);  
    SET_CLK();  
    pptrimDelay(PPTRIMDelay);  
    CLEAR_CLK();  
    pptrimDelay(PPTRIMDelay);  
}
```

```
static void exitCondition()
```

```
{  
    PROG_LOW_IMPED();  
    pptrimDelay(PPTRIMDelay);  
    CLEAR_CSN();  
    pptrimDelay(PPTRIMDelay);  
    SET_CLK();  
    pptrimDelay(PPTRIMDelay);  
    CLEAR_CLK();  
    pptrimDelay(PPTRIMDelay);  
    SET_CLK();  
    pptrimDelay(PPTRIMDelay);  
    SET_CSN();  
    pptrimDelay(PPTRIMDelay);  
    CLEAR_PROG();  
    pptrimDelay(PPTRIMDelay);  
}
```

5.5 Operation modes

```
static void operationModeLoad()
```

```
{  
    CLEAR_PROG();  
    pptrimDelay(PPTRIMDelay);  
    SET_CSN();  
    pptrimDelay(PPTRIMDelay);  
    clkPulses(4);  
}
```

```
static void operationModeRead()
```

```
{  
    CLEAR_PROG();  
    pptrimDelay(PPTRIMDelay);  
    SET_CLK();  
    pptrimDelay(PPTRIMDelay);  
    SET_CSN();  
    pptrimDelay(PPTRIMDelay);  
    CLEAR_CLK();  
    pptrimDelay(PPTRIMDelay);  
    clkPulses(1);  
    PROG_HIGH_IMPED();  
}
```

```
static void operationModeWrite()
```

```
{  
    SET_CSN();  
    pptrimDelay(PPTRIMDelay);  
    clkPulses(3);  
}
```

```
static void operationModeProg()
```

```
{  
    SET_CLK();  
    pptrimDelay(PPTRIMDelay);  
    SET_CSN();  
    pptrimDelay(PPTRIMDelay);  
    CLEAR_CLK();  
    pptrimDelay(PPTRIMDelay);  
    clkPulses(4);  
}
```

5.6 PPTrim functions

```

void pptrimLoad(unsigned char num_bits)
{
    setupCondition();
    operationModeLoad();
    clkPulses(num_bits);
    exitCondition();
}

void pptrimRead(unsigned char *buffer, unsigned char num_bits)
{
    xdata unsigned char current_byte = 0;
    xdata unsigned char current_bit = 0;
    xdata unsigned char temp = 0;

    if(!num_bits) return;

    current_byte = num_bits >> 3;
    current_bit = num_bits & ~0x07;

    setupCondition();
    operationModeRead();
    clkPulses(1); // position the first bit to read

    //-- read OTP Data --
    temp = 0;

    temp += (SSI_PROG) ? 1 : 0;

    for(current_bit = num_bits; current_bit; current_bit--)
    {
        if(((current_bit - 1) & 0x07) == 0)
        {
            buffer[current_bit >> 3] = temp;

            temp = 0;
        }

        if (current_bit)
        {
            temp <<= 1;

            SET_CLK();
            pptrimDelay(200);
            temp += (SSI_PROG) ? 1 : 0;
        }
    }
}

```

```

        CLEAR_CLK();
        pptrimDelay(200);
    }
}
exitCondition();
}

void pptrimWrite(unsigned char *buffer, unsigned char num_bits)
{
    xdata unsigned char *current_byte;
    xdata unsigned char current_bit = 0;
    xdata unsigned char temp = 0;

    current_byte = buffer + ((num_bits-1)>>3);
    temp = *current_byte;
    if(num_bits % 8)
        temp <<= 8 - (num_bits % 8);

    setupCondition();
    operationModeWrite();

    //-- send OTP Data
    for(current_bit = num_bits; current_bit; current_bit--)
    {
        if(temp & 0x80)
            SET_PROG();
        else
            CLEAR_PROG();

        pptrimDelay(100);
        SET_CLK();
        pptrimDelay(300); // delay, tzapp=2us(typ.)
        CLEAR_CLK();
        pptrimDelay(PPTRIMDelay);

        temp <<= 1;
        if(((current_bit-1) & 0x07) == 0)
        {
            temp = *(--current_byte);
        }
    }
    SET_PROG();
    pptrimDelay(100);

    clkPulses(1); // data latched

    // END OTP-Write
    exitCondition();
}

```

Revision History

Revision	Date	Description
1.1	September, 2009	initial revision
1.2	December, 2009	Rename document from AN514X-10 to AN5000-30; Insert device list
1.3	July, 2010	Rename document and note for analog readback function (page 18)

Copyrights

Copyright © 1997-2009, austriamicrosystems AG, Schloss Premstaetten, 8141 Unterpremstaetten, Austria-Europe.

Trademarks Registered ®. All rights reserved. The material herein may not be reproduced, adapted, merged, translated, stored, or used without the prior written consent of the copyright owner.

All products and companies mentioned are trademarks or registered trademarks of their respective companies.

Disclaimer

Devices sold by austriamicrosystems AG are covered by the warranty and patent indemnification provisions appearing in its Term of Sale. austriamicrosystems AG makes no warranty, express, statutory, implied, or by description regarding the information set forth herein or regarding the freedom of the described devices from patent infringement. austriamicrosystems AG reserves the right to change specifications and prices at any time and without notice. Therefore, prior to designing this product into a system, it is necessary to check with austriamicrosystems AG for current information. This product is intended for use in normal commercial applications. Applications requiring extended temperature range, unusual environmental requirements, or high reliability applications, such as military, medical life-support or lifesustaining equipment are specifically not recommended without additional processing by austriamicrosystems AG for each application.

The information furnished here by austriamicrosystems AG is believed to be correct and accurate. However, austriamicrosystems AG shall not be liable to recipient or any third party for any damages, including but not limited to personal injury, property damage, loss of profits, loss of use, interruption of business or indirect, special, incidental or consequential damages, of any kind, in connection with or arising out of the furnishing, performance or use of the technical data herein. No obligation or liability to recipient or any third party shall arise or flow out of austriamicrosystems AG rendering of technical or other services.

Contact Information

Headquarters

austriamicrosystems AG

A-8141 Schloss Premstaetten, Austria

Tel: +43 (0) 3136 500 0

Fax: +43 (0) 3136 525 01

For Sales Offices, Distributors and Representatives, please visit:

<http://www.austriamicrosystems.com>