Application Note

AN001033

AS7343 SMUX Configuration

Filter Channel Mapping by Multiplexing

v2-01 • 2021-Dec-14

Content Guide

1 General Description

The AS7343 integrates a so-called super multiplexer (SMUX). With the SMUX, it is possible to map all available photodiodes to one of the six available light-to-frequency converters (CH0 ADC to CH5 ADC). Every pixel has a multiplexer to map it to one of the engines – this multiplexer can be configured with three bits. (0 = pixel disabled or connected to GND; 1 to 6 = ADC 0 to ADC 5). The figures below show the SMUX pixel ID mapping to every individual diode or diodes pair. The “not mentioned” and the grey pixel ID are not used, and will be programmed with “0”. Reading and writing pixel configuration uses the first 10 bytes of the RAM starting at address 00h. For easier usage, the pixel configuration is stored in nibbles within the RAM (4 bits per pixel configuration, MSB not used). It is recommended to write the 20 bytes at once, and configure all the pixels together within one page write command.

Figure 1:
Sensor Array

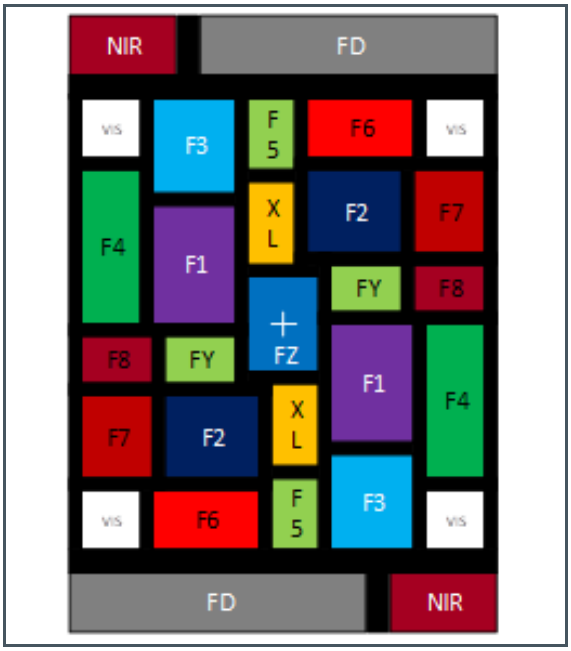

Figure 2:
SMUX Pixel ID Mapping to Diodes

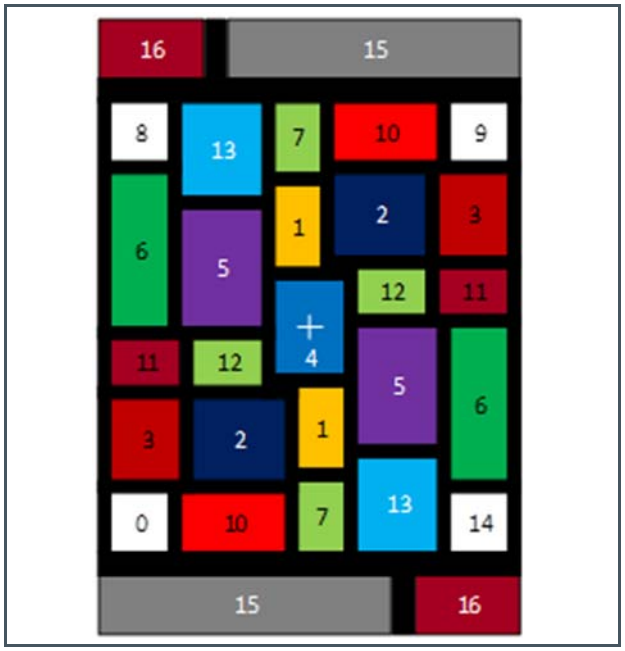

Figure 3
Chain Map Pixel ID (Line 1) vs. Filter (1) (Line 2)

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
VIS	XL	F2	F7	FZ	F1	F4	F5	VIS	VIS	F6	F8	FY	F3	VIS	FD	NIR	Ext	D	Not
LB								LT	RT					RB					used

(1) LT = Left Top VIS, LB = Left Bottom VIS, RT = Right Top VIS, Right Bottom VIS, NIR = Near Infrared, C = VIS , FD = Flicker Detection, D = Dark

2 SMUX Default Configuration Setting

Besides a default Configuration, there are several configuration methods: the RAM Method and Direct Chain Configuration. These methods are described in the following chapters.

On the AS7343, there is a Default Configuration Setting for data read-out, in which there is no need to adjust the SMUX in case the default fits the customer requirements.

The default setting is defined as n-integration chains, whereby n can be set in bit autorun_smux2 (bit 5 and 6 in register CFG20; default 0 – normal six channels) with the following settings:

Figure 4:
Settings for Bit autorun_smux2 to Set Number of Channels for Default SMUX

Value n	Mode	Integration Chain
0 (default)	Normal 6 channels	Integration Chain 1: Z, Y, XL, NIR , VIS(1) LT+RB, FD
1	Normal 6 channels	Integration Chain 1: Z, Y, XL, NIR , VIS(1) LT+RB, FD
2	Normal 12 channels	Integration Chain 1: Z, Y, XL, NIR , VIS(1) LT+RB, FD Integration Chain 2: F2, F3, F4, F6, VIS(1) LT+RB, FD
3	Normal 18 channels	Integration Chain 1: Z, Y, Xk, NIR , VIS(1) LT+RB, FD Integration Chain 2: F2, F3, F4, F6, VIS(1) LT+RB, FD Integration Chain 3: F1, F7, F8, F5, VIS(1) LT+RB, FD

(1) Only two of the four VIS [LT, LB, RT, RB] diodes in the default are read out, to verify a changed light situation within the three Chains

The data is stored in the Data registers accordingly. This means, only six channels are read by using default 0, or in the case of set n = 1. The results of 12 channels from Chains 1 and 2 are in the data registers. Eighteen (18) channels are read out after setting the bit autorun_smux2 n = 3.

3 Specific SMUX Multiplexer Mapping

In case the default settings do not meet customer requirements because other channels are needed, or faster measurement time is required, then channel read-out can be configured application-specifically via SMUX configuration.

In the "comfortable" method (see examples in chapters 3.1 and 3.2) of the RAM configuration, the SMUX data from I2C-Addr 0x0 are written into the RAM via the configuration of SMUXEN, and copied with the CMD SMUXEN=1 by the internal machine into the SMUX chain of the analog part. This method is limited to a configuration with a maximum of six channels (ALS 6) in Chain 1. If more channels/Chains are needed, the alternative method must be used.

The second method (see the examples in chapters 3.3 and 3.4) is Direct Configuration, where the chain for configuration is addressed directly via CHAINCMD. So one has direct access to the SMUX-chains 1, 2, 3 and can change the default configuration for all the modes ALS 6-12-18 (=maximum o channels) as desired.

Not depending on the method, the pixel IDs of the diodes must be mapped. The following table shows the principle mapping of the SMUX pixel IDs to address configuration bit positions in the I^2C and RAM address space (for “Chain Map Pixel ID vs. Filter” see Figure 3).

Figure 5:
Mapping of Pixel IDs to RAM Addresses

I^2C ADDR	RAM ADDR	<D7>	<D6>	<D5>	<D4>	<D3>	<D2>	<D1>	<D0>
0x00	0		Pixel ID 1 [6:4]				Pixel ID 0 [2:0]		
0x01			Pixel ID 3 [6:4]				Pixel ID 2 [2:0]		
0x02	1		Pixel ID 5 [6:4]				Pixel ID 4 [2:0]		
0x03			Pixel ID 7 [6:4]				Pixel ID 6 [2:0]		
0x04	2		Pixel ID 9 [6:4]				Pixel ID 8 [2:0]		
0x05			Pixel ID 11 [6:4]				Pixel ID 10 [2:0]		
0x06	3		Pixel ID 13 [6:4]				Pixel ID 12 [2:0]		
0x07			Pixel ID 15 [6:4]				Pixel ID 14 [2:0]		
0x08	4		Pixel ID 17 [6:4]				Pixel ID 16 [2:0]		
0x09			Unused				Pixel ID 18 [2:0]		

Figure 6:
SMUX Multiplexer Mapping

Addr: 0x00 – 0x09		SMUX Multiplexer Mapping		
Bit	**Bit Name**	**Default**	**Access**	**Bit Description**
7	Not used	0	RW	Reserved
6:4	Pixel IDx	0	RW	0: Connected to Ground/disabled 1: Pixel connected to ADC0 2: Pixel connected to ADC1 3: Pixel connected to ADC2 4: Pixel connected to ADC3 5: Pixel connected to ADC4 6: Pixel connected to ADC5 (Flicker) 7: Reserved
3	Not used	0	RW	Reserved
2:0	Pixel IDy	0	RW	0: Connected to Ground/disabled 1: Pixel connected to ADC0 2: Pixel connected to ADC1 3: Pixel connected to ADC2 4: Pixel connected to ADC3 5: Pixel connected to ADC4 6: Pixel connected to ADC5 (Flicker) 7: Reserved

3.1 RAM Configuration Readout F1-F4, VIS, NIR

The following example shows how to map individual PDs to dedicated ADCs using the SMUX. Each box in the chain map example in Figure 7 represents one nibble (4-bit per pixel ID). The number within the box is the value that needs to be programmed to map the pixel to the desired ADC. If no chain is defined via the CHAINCMD register, Chain 1 is always the default configuration target.

> F1 is mapped to ADC0, F2 is mapped to ADC1, F3 is mapped to ADC2, F4 is mapped to ADC3, VIS is mapped to ADC4, and NIR is mapped to ADC5.

Figure 7:
Chain Map Example of Readout F1 to F4, VIS, and NIR (Nibble Sequence Figure 3)

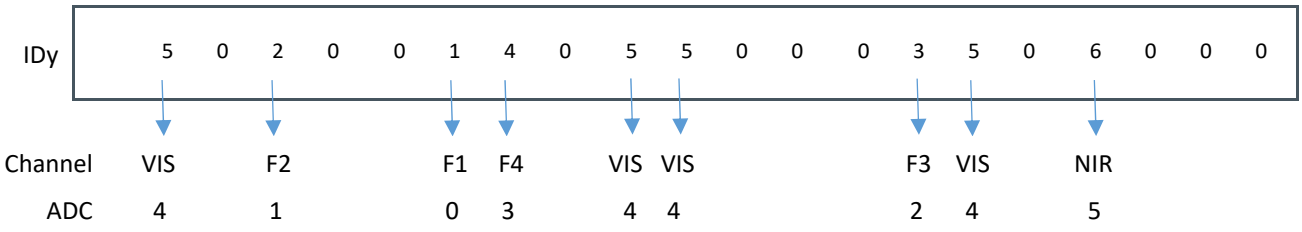

Figure 8:
Configuration Example of Readout F1-F4, VIS, NIR

Step	I²C command	Description
1	I2C_write(0x80, 0x01)	Enable power (set PON = “1”) and disable SP_EN (SP_EN=”0”). Register: ENABLE/0x80
2	I2C_write(0xCA, 0x10)	Enable special interrupt (SINT_SMUX). As soon as the SMUX command finishes, the interrupt is activated. Register: CFG9/0xCA
3	I2C_write(0xF9, 0x01)	Enable special interrupt SIEN. Register: INTENAB/0xF9
4	I2C_write(0xF5, 0x10)	Write SMUX configuration from the RAM to set the SMUX chain. Register: CFG6/0xF5
5	I2C_write(0x00, 0x05)	VIS LB connected to ADC4, XL not connected.
6	I2C_write(0x01, 0x02)	F2 connected to ADC1, F7 not connected.
7	I2C_write(0x02, 0x10)	F1 connected to ADC0, FZ not connected.
8	I2C_write(0x03, 0x04)	F4 connected to ADC3, F5 not connected.
9	I2C_write(0x04, 0x55)	VIS LT and VIS RT connected to ADC4.
10	I2C_write(0x05, 0x00)	F6 and F8 not connected.
11	I2C_write(0x06, 0x30)	F3 connected to ADC2, FY not connected.
12	I2C_write(0x07, 0x05)	VIS RB connected to ADC4, FD Flicker not connected.
13	I2C_write(0x08, 0x06)	NIR connected to ADC5, Ext not connected.
14	I2C_write(0x09, 0x00)	D (Dark) not connected.
25	I2C_write(0x80, 0x11)	Start the SMUX command while keeping the power on (SMUXEN = “1” and PON = “1”).
26		Wait for interrupt.
27	I2C_write(0x80, 0x00)	Power down (PON = “0”).

3.2 RAM Configuration Readout F5-F8, VIS, NIR

The following example shows how to map individual PDs to dedicated ADCs using the SMUX. If no Chain is defined via the CHAINCMD register, Chain 1 is always the default configuration target.

> F5 is mapped to ADC0, F6 is mapped to ADC1, F7 is mapped to ADC2, F8 is mapped to ADC3, VIS is mapped to ADC4, and NIR is mapped to ADC5.

Figure 9:
Chain Map Example of Readout F5 to F8, VIS, and NIR

5	0	0	2	0	0	0	4	5	5	1	3	0	0	5	0	6	0	0	0

Figure 10:
Configuration Example of Readout F5-F8, VIS, NIR

Step	I²C command	Description
1	I2C_write(0x80, 0x01)	Enable power (set PON = “1”). Register: ENABLE/0x80
2	I2C_write(0xCA, 0x10)	Enable special interrupt (SINT_SMUX). As soon as the SMUX command finishes, the interrupt is activated. Register: CFG9/0xCA
3	I2C_write(0xF9, 0x01)	Enable special interrupt SIEN. Register: INTENAB/0xF9
4	I2C_write(0xF5, 0x10)	Write SMUX configuration from the RAM to set the SMUX chain. Register: CFG6/0xF5
5	I2C_write(0x00, 0x05)	VIS LB connected to ADC4.
6	I2C_write(0x01, 0x20)	F6 connected to ADC1.
7	I2C_write(0x02, 0x00)	
8	I2C_write(0x03, 0x40)	F8 connected to ADC3.
9	I2C_write(0x04, 0x55)	VIS LT VIS RT connected to ADC4.
10	I2C_write(0x05, 0x31)	F5 connected to ADC0 F7 connected to ADC2.
11	I2C_write(0x06, 0x00)	
12	I2C_write(0x07, 0x05)	VIS RB connected to ADC4.
13	I2C_write(0x08, 0x06)	NIR connected to ADC5.
14	I2C_write(0x09, 0x00)	
25	I2C_write(0x80, 0x11)	Start the SMUX command while keeping the power on (SMUXEN = “1” and PON = “1”).
26		Wait for interrupt.
27	I2C_write(0x80, 0x00)	Power down (PON = “0”).

3.3 Direct Configuration Readout F5-F8, VIS, NIR

The following example shows how to map individual PDs, plus addressing the Chain n to dedicated ADCs using the SMUX. The (Chain) n is defined as a target for the configuration, by setting 0x46 to register CHAINCMD/0xE4 (alternatively, 0x56 for Chain2 and 0x66 for Chain 3).

- F5 is mapped to ADC0, F6 is mapped to ADC1, F7 is mapped to ADC2, F8 is mapped to ADC3, VIS is mapped to ADC4, and NIR is mapped to ADC5.

Figure 11:
Chain Map Example of Readout F5 to F8, VIS, and NIR

5	0	0	2	0	0	0	4	5	5	1	3	0	0	5	0	6	0	0	0

Figure 12:
Direct Configuration Chain1 Example F5-F8, VIS, NIR

Step	I²C command	Description
1	I2C_write(0x80, 0x01)	Enable power (set PON = “1”). Register: ENABLE/0x80
2	I2C_write(0xE4, 0x46)	SMUX Chain CMD. Register: CHAINCMD/0xE4
3	I2C_write(0xE7, 0x00)	Write pixel NA 18. Register: CHAIN_SMUX/0xE7
4	I2C_write(0xE4, 0x46)	SMUX Chain CMD. Register: CHAINCMD/0xE4
5	I2C_write(0xE7, 0x06)	Write pixel 17 16 NIR connected to ADC5. Register: CHAIN_SMUX/0xE7
6	I2C_write(0xE4, 0x46)	SMUX Chain CMD. Register: CHAINCMD/0xE4
7	I2C_write(0xE7, 0x05)	Write pixel 15 14 VIS RB connected to ADC4. Register: CHAIN_SMUX/0xE7
8	I2C_write(0xE4, 0x46)	SMUX Chain CMD. Register: CHAINCMD/0xE4
9	I2C_write(0xE7, 0x00)	Write pixel 13 12. Register: CHAIN_SMUX/0xE7
10	I2C_write(0xE4, 0x46)	SMUX Chain CMD. Register: CHAINCMD/0xE4
11	I2C_write(0xE7, 0x31)	Write pixel 11 10 F5 connected to ADC0 F7 connected to ADC2. Register: CHAIN_SMUX/0xE7
12	I2C_write(0xE4, 0x46)	SMUX Chain CMD. Register: CHAINCMD/0xE4 VIS RB connected to ADC4.
13	I2C_write(0xE7, 0x55)	Write pixel 9 8 VIS LT VIS RT connected to ADC4. Register: CHAIN_SMUX/0xE7
14	I2C_write(0xE4, 0x46)	SMUX Chain CMD. Register: CHAINCMD/0xE4
15	I2C_write(0xE7, 0x40)	Write pixel 7 6 F8 connected to ADC3. Register: CHAIN_SMUX/0xE7
16	I2C_write(0xE4, 0x46)	SMUX Chain CMD. Register: CHAINCMD/0xE4
17	I2C_write(0xE7, 0x00)	Write pixel 5 4. Register: CHAIN_SMUX/0xE7
18	I2C_write(0xE4, 0x46)	SMUX Chain CMD. Register: CHAINCMD/0xE4
19	I2C_write(0xE7, 0x20)	Write pixel 3 2 F6 connected to ADC1. Register: CHAIN_SMUX/0xE7
20	I2C_write(0xE4, 0x46)	SMUX Chain CMD. Register: CHAINCMD/0xE4

Step	I²C command	Description
21	I2C_write(0xE7, 0x05)	Write pixel 1 0 VISLB connected to ADC4. Register: CHAIN_SMUX/0xE7
22	I2C_write(0x80, 0x00)	Power down (PON = “0”).

3.4 Direct Configuration Code Examples for Chain 1 - 3

The following examples show the coded default configuration for Chain 1 – 3, which means for 18 channels (like the settings shown in Figure 4 - version Chain 1 - 3).

Figure 13:
Specification which Programs Chain 1 - 3

```
User        0x80            0x01        Set ALSEN=0 LTF ADC off

User        0xe7            0x00        // NC Flicker

User        0xe4            0x46        // smux command chain 1

User        0xe7            0x04        // EXT NIR

User        0xe4            0x46        // smux command chain 1

User        0xe7            0x65        // FD   RB

User        0xe4            0x46        // smux command chain 1

User        0xe7            0x02        // F3_473    Y

User        0xe4            0x46        // smux command chain 1

User        0xe7            0x00        // F7_685  F5_546

User        0xe4            0x46        // smux command chain 1

User        0xe7            0x05        // RT   LT

User        0xe4            0x46        // smux command chain 1

User        0xe7            0x00        // F8 _745  F4_514

User        0xe4            0x46        // smux command chain 1

User        0xe7            0x01        // F1_405   Z

User        0xe4            0x46        // smux command chain 1
```

```
User        0xe7            0x00        // F6_635   F2_424
User        0xe4            0x46        // smux command chain 1
User        0xe7            0x30        // X  LB
User        0xe4            0x46        // smux command chain 1
User        0xe7            0x00        // NC Flicker
User        0xe4            0x56        // smux command chain 2
User        0xe7            0x00        // EXT NIR
User        0xe4            0x56        // smux command chain 2
User        0xe7            0x60        // FLICKER   RB
User        0xe4            0x56        // smux command chain 2
User        0xe7            0x20        // F3_473    Y
User        0xe4            0x56        // smux command chain 2
User        0xe7            0x04        // F7_685  F5_546
User        0xe4            0x56        // smux command chain 2
User        0xe7            0x50        // RT   LT
User        0xe4            0x56        // smux command chain 2
User        0xe7            0x03        // F8 _745  F4_514
User        0xe4            0x56        // smux command chain 2
User        0xe7            0x00        // F1_405   Z
User        0xe4            0x56        // smux command chain 2
User        0xe7            0x01        // F6_635   F2_424
User        0xe4            0x56        // smux command chain 2
User        0xe7            0x05        // X  LB
User        0xe4            0x56        // smux command chain 2
User        0xe7            0x05        // NC  Flicker
```

```
User        0xe4            0x66        // smux command chain 3
User        0xe7            0x00        // EXT NIR
User        0xe4            0x66        // smux command chain 3
User        0xe7            0x60        // FLICKER  RB
User        0xe4            0x66        // smux command chain 3
User        0xe7            0x00        // F3_473  Y
User        0xe4            0x66        // smux command chain 3
User        0xe7            0x30        // F7_685  F5_546
User        0xe4            0x66        // smux command chain 3
User        0xe7            0x00        // RT  LT
User        0xe4            0x66        // smux command chain 3
User        0xe7            0x40        // F8_745  F4_514
User        0xe4            0x66        // smux command chain 3
User        0xe7            0x10        // F1_405  Z
User        0xe4            0x66        // smux command chain 3
User        0xe7            0x20        // F6_635  F2_424
User        0xe4            0x66        // smux command chain 3
User        0xe7            0x00        // X LB
User        0xe4            0x66        // smux command chain 3
```

Figure 14:
Example in Python for 3 Chains – 18 Channels

```
RA_CFG20 = 0xd6

RA_CHAINCMD = 0xe4

RA_CHAIN_SMUX = 0xe7
```

```
# Chain 1 Z,Y,Xk,NIR,LT+RB,FLICKER

# Chain 2 F424,F473,F514,F635,LT+RB,FLICKER

# Chain 3 F405,F670,F710,F546,LT+RB,FLICKER

# NA    D    EXT   NIR    FLICKER    RB  F473     Y  F710  F635    RT    LT  F546
F514  F405     Z  F670  F424    Xk    LB

default_chain1 = [ 0x00, 0x00, 0x00, 0x04, 0x06, 0x05, 0x00, 0x02, 0x00, 0x00,
0x00, 0x05, 0x00, 0x00, 0x00, 0x01, 0x00, 0x00, 0x03, 0x00 ]

default_chain2 = [ 0x00, 0x00, 0x00, 0x00, 0x06, 0x05, 0x02, 0x00, 0x00, 0x04,
0x00, 0x05, 0x00, 0x03, 0x00, 0x00, 0x00, 0x01, 0x00, 0x00 ]

default_chain3 = [ 0x00, 0x00, 0x00, 0x00, 0x06, 0x05, 0x00, 0x00, 0x03, 0x00,
0x00, 0x05, 0x04, 0x00, 0x01, 0x00, 0x02, 0x00, 0x00, 0x00 ]

# smux configuration without flicker diode based on default

no_fd_chain1   = [ 0x00, 0x00, 0x00, 0x04, 0x00, 0x05, 0x00, 0x02, 0x00, 0x00,
0x00, 0x05, 0x00, 0x00, 0x00, 0x01, 0x00, 0x00, 0x03, 0x00 ]

no_fd_chain2   = [ 0x00, 0x00, 0x00, 0x00, 0x00, 0x05, 0x02, 0x00, 0x00, 0x04,
0x00, 0x05, 0x00, 0x03, 0x00, 0x00, 0x00, 0x01, 0x00, 0x00 ]

no_fd_chain3   = [ 0x00, 0x00, 0x00, 0x00, 0x00, 0x05, 0x00, 0x00, 0x03, 0x00,
0x00, 0x05, 0x04, 0x00, 0x01, 0x00, 0x02, 0x00, 0x00, 0x00 ]

# smux configuration only with flicker diode

flicker_chain  = [ 0x00, 0x00, 0x00, 0x00, 0x06, 0x00, 0x00, 0x00, 0x00, 0x00,
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00 ]

class set_smux:

    def make_chains_default(self):

        for i in range(0,20,2):

            self.register_write(RA_CHAIN_SMUX, default_chain1[i] << 4 |
default_chain1[i+1])

            self.register_write(RA_CHAINCMD, 0x46)

        for i in range(0,20,2):

            self.register_write(RA_CHAIN_SMUX, default_chain2[i] << 4 |
default_chain2[i+1])
```

```
            self.register_write(RA_CHAINCMD, 0x56)

        for i in range(0,20,2):

            self.register_write(RA_CHAIN_SMUX, default_chain3[i] << 4 |
default_chain3[i+1])

            self.register_write(RA_CHAINCMD, 0x66)

    def set_smux_default(self):

        self.make_chains_default()

        self.register_write(RA_CFG20, 0x60)

    def register_write(self, address, value):

        pass

#end
```

3.5 Chip Library

ams provides a chip library for the AS7343 upon request. This library provides low-level access to the sensor as well as basic configuration possibilities including SMUX settings. However, the firmware for a sensor is always dependent on the system configuration used, i.e. the hardware and software of the sensor and the host computer. Therefore, the ChipLib is generic and considers exemplary hardware and software configurations.

4 Revision Information

Changes from previous version to current revision v2-01	Page
Changed filter names “Clear ==> VIS”	all
Correction number of bytes for mapping	3
Correction Mapping Addresses	5
ChipLib	14

- Page and figure numbers for the previous version may differ from page and figure numbers in the current revision.
- Correction of typographical errors is not explicitly mentioned.

5 Legal Information

Headquarters
ams AG
Tobelbader Strasse 30
8141 Premstaetten
Austria, Europe
Tel: +43 (0) 3136 500 0

Please visit our website at www.ams.com
Buy our products or get free samples online at www.ams.com/Products
Technical Support is available at www.ams.com/Technical-Support
Provide feedback about this document at www.ams.com/Document-Feedback
For sales offices, distributors and representatives go to www.ams.com/Contact
For further information and requests, e-mail us at ams_sales@ams.com