Application Note

AN000666

SMUX Configuration

How to Configure SMUX for Reading Out AS7341 Sensor Results

v1-00 • 2021-Jan-21

Content Guide

1 Introduction

The AS7341 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 3 bits. (0 = pixel disabled / connected to GND; 1 to 6 = ADC 0 to ADC 5).

The figures below show the SMUX pixel ID mapping to every individual diode. In addition to the 4x4 pixel array, flicker detection, NIR, and CLEAR diodes, three IDs are available for external photodiodes connected to pin GPIO and INT, as well as an on-chip DARK photodiode (PD covered with black filter). Unmentioned and grey pixel IDs are not used, and shall be programmed with “0”.

Reading and writing pixel configuration uses the first 20 bytes of the RAM starting at address 00h. For easier usage, the pixel configuration is stored in nibbles within the RAM (4bits per pixel configuration, MSB not used). It is recommended to write the 20 bytes at once and configure all pixels together within one page write command.

Figure 1:
Sensor Array (left) and SMUX Pixel ID Mapping to Diodes (right)

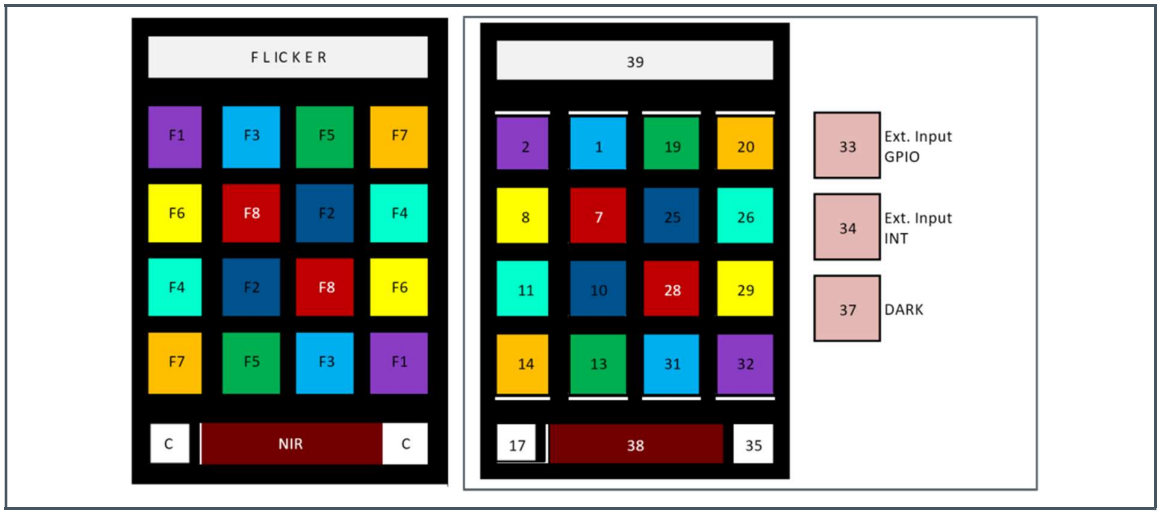
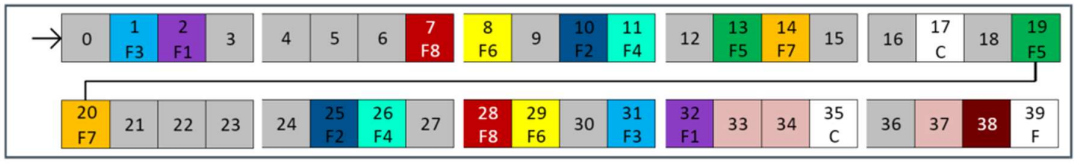


Figure 2:
Chain Map Pixel ID vs Filter



2 SMUX Multiplexer Mapping

The following table shows the mapping of the SMUX pixel IDs to address configuration bit positions, in I²C and RAM address space.

Figure 3:
SMUX Multiplexer Mapping

I²C ADDR	RAM ADDR	<D7>	<D6>	<D5>	<D4>	<D3>	<D2>	<D1>	<D0>
0x00	0	Pixel ID 1 [6:4]					Unused		
0x01		Unused					Pixel ID 2 [2:0]		
0x02	1	Unused					Unused		
0x03		Pixel ID 7 [6:4]					Unused		
0x04	2	Unused					Pixel ID 8 [2:0]		
0x05		Pixel ID 11 [6:4]					Pixel ID 10 [2:0]		
0x06	3	Pixel ID 13 [6:4]					unused		
0x07		Unused					Pixel ID 14 [2:0]		
0x08	4	Pixel ID 17 [6:4]					Unused		
0x09		Pixel ID 19 [6:4]					Unused		
0x0A	5	Unused					Pixel ID 20 [2:0]		
0x0B		Unused					Unused		
0x0C	6	Pixel ID 25 [6:4]					Unused		
0x0D		Unused					Pixel ID 26 [2:0]		
0x0E	7	Pixel ID 29 [6:4]					Pixel ID 28 [2:0]		
0x0F		Pixel ID 31 [6:4]					Unused		
0x10	8	Pixel ID 33 [6:4]					Pixel ID 32 [2:0]		
0x11		Pixel ID 35 [6:4]					Pixel ID 34 [2:0]		
0x12	9	Pixel ID 37 [6:4]					Unused		
0x13		Pixel ID 39 [6:4]					Pixel ID 38 [2:0]		

Figure 4:
Addr: 0x00 – 0x13 SMUX Multiplexer Mapping

Bit	Bit Name	Default	Access	Bit Description
7	Not used	0	RW	Reserved

Bit	Bit Name	Default	Access	Bit Description
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 Configuration Example 1

The following example shows how to map individual PDs to dedicated ADCs using the SMUX. In the example below, each box in the chain map represents one nibble (4-bit per pixel ID). The number within the box is the value, which needs to be programmed to map the pixel to the desired ADC.

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

Figure 5:
Chain Map Example Read Out F1 to F4, CLEAR and NIR

Figure 6:
I²C Commands Example 1

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(0xB2, 0x10)	Enable special interrupt (SINT_SMUX). As soon as SMUX command has finished interrupt is activated. Register: CFG9 / 0xB2
3	I2C_write(0xF9, 0x01)	Enable special interrupt SIEN Register: INTENAB / 0xF9
4	I2C_write(0xAF, 0x10)	Write SMUX configuration from RAM to set SMUX chain Register: CFG6 / 0xAF
5	I2C_write(0x00, 0x30)	F3 left set to ADC2
6	I2C_write(0x01, 0x01)	F1 left set to ADC0
7	I2C_write(0x02, 0x00)	
8	I2C_write(0x03, 0x00)	F8 left disabled
9	I2C_write(0x04, 0x00)	F6 left disabled
10	I2C_write(0x05, 0x42)	F4 left connected to ADC3 / F2 left connected to ADC1
11	I2C_write(0x06, 0x00)	F5 left disabled
12	I2C_write(0x07, 0x00)	F7 left disabled
13	I2C_write(0x08, 0x50)	CLEAR connected to ADC4
14	I2C_write(0x09, 0x00)	F5 right disabled
15	I2C_write(0x0A, 0x00)	F7 right disabled
16	I2C_write(0x0B, 0x00)	
17	I2C_write(0x0C, 0x20)	F2 right connected to ADC1

Step	I²C Command	Description
18	I2C_write(0x0D, 0x04)	F4 right connected to ADC3
19	I2C_write(0x0E, 0x00)	F6/F8 right disabled
20	I2C_write(0x0F, 0x30)	F3 right connected to ADC2
21	I2C_write(0x10, 0x01)	F1 right connected to ADC0
22	I2C_write(0x11, 0x50)	CLEAR right connected to ADC4
23	I2C_write(0x12, 0x00)	
24	I2C_write(0x13, 0x06)	NIR connected to ADC5
25	I2C_write(0x80, 0x11)	Start SMUX command while keeping power on (SMUXEN = “1” and PON = “1”)
26		Wait for interrupt
27	I2C_write(0x80, 0x00)	Power down (PON = “0”)

4 Configuration Example 2

The following example shows how to map individual PDs to dedicated ADCs using the SMUX.

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

Figure 7:
Chain Map Example Read Out F5 to F8, CLEAR and NIR

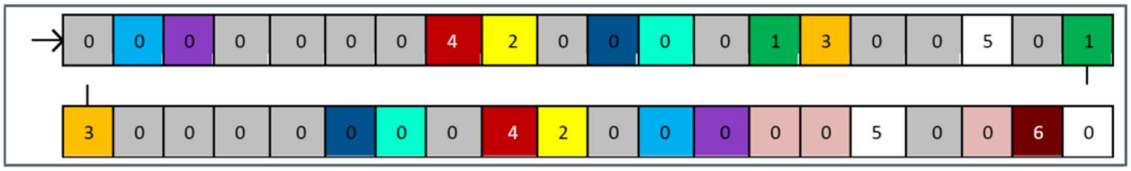


Figure 8:
I²C Commands Example 2

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(0xB2, 0x10)	Enable special interrupt (SINT_SMUX). As soon as SMUX command has finished interrupt is activated. Register: CFG9 / 0xB2
3	I2C_write(0xF9, 0x01)	Enable special interrupt SIEN Register: INTENAB / 0xF9
4	I2C_write(0xAF, 0x10)	Write SMUX configuration from RAM to set SMUX chain Register: CFG6 / 0xAF
5	I2C_write(0x00, 0x00)	F3 left disabled
6	I2C_write(0x01, 0x00)	F1 left disabled
7	I2C_write(0x02, 0x00)	
8	I2C_write(0x03, 0x40)	F8 left connected to ADC3
9	I2C_write(0x04, 0x02)	F6 left connected to ADC1
10	I2C_write(0x05, 0x00)	F4/F2 disabled
11	I2C_write(0x06, 0x10)	F5 left connected to ADC0
12	I2C_write(0x07, 0x03)	F7 left connected to ADC2
13	I2C_write(0x08, 0x50)	CLEAR connected to ADC4
14	I2C_write(0x09, 0x10)	F5 right connected to ADC0
15	I2C_write(0x0A, 0x03)	F7 right connected to ADC2
16	I2C_write(0x0B, 0x00)	
17	I2C_write(0x0C, 0x00)	F2 right disabled
18	I2C_write(0x0D, 0x00)	F4 right disabled
19	I2C_write(0x0E, 0x24)	F8 right connected to ADC3 / F6 right connected to ADC1

Step	I²C Command	Description
20	I2C_write(0x0F, 0x00)	F3 right disabled
21	I2C_write(0x10, 0x00)	F1 right disabled
22	I2C_write(0x11, 0x50)	CLEAR right connected to ADC4
23	I2C_write(0x12, 0x00)	
24	I2C_write(0x13, 0x06)	NIR connected to ADC5
25	I2C_write(0x80, 0x11)	Start SMUX command while keeping power on (SMUXEN = “1” and PON = “1”)
26		Wait for interrupt
27	I2C_write(0x80, 0x00)	Power down (PON = “0”)

5 Revision Information

Changes from previous version to current revision v1-00	Page
Initial version	all

- 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.

6 Legal Information

Error! Not a valid filename.

Error! Not a valid filename.

Error! Not a valid filename.