Application Note

AN000699

AS7265x

Design Considerations

v1-00 • 2020-Jun-19

Content Guide

1 Introduction

This application note briefly describes system-level design considerations with **ams** AS7265x Multispectral Chipset solution.

1.1 Ordering Information

Ordering Code	Description
AS7265x Demo Kit	Demo Kit AS7265x Multispectral Chipset v3
AS7265x Demo Kit Housing	Black plastic housing from shell exterior retardant of ABS (UL94-V0) with inserted plastic diffuser

2 AS7265x Multispectral Chipset

AS7265x Multispectral Chipset consists of AS72651, AS72652, and AS72653 devices and each device has 6 optical filters so that AS7265x Multispectral Chipset has in total 18 channels for spectral identification from 400 nm to 1000 nm with FWHM of 20 nm.

AS72651 is the main device with a smart interface to a microcontroller. The microcontroller gets the sensor data including AS72652 and AS72653 through AS72651 AT commands or I^2C registers. AS72651 requires a flash memory to work with and the flash memory contains AS7265x Multispectral Chipset firmware[1].

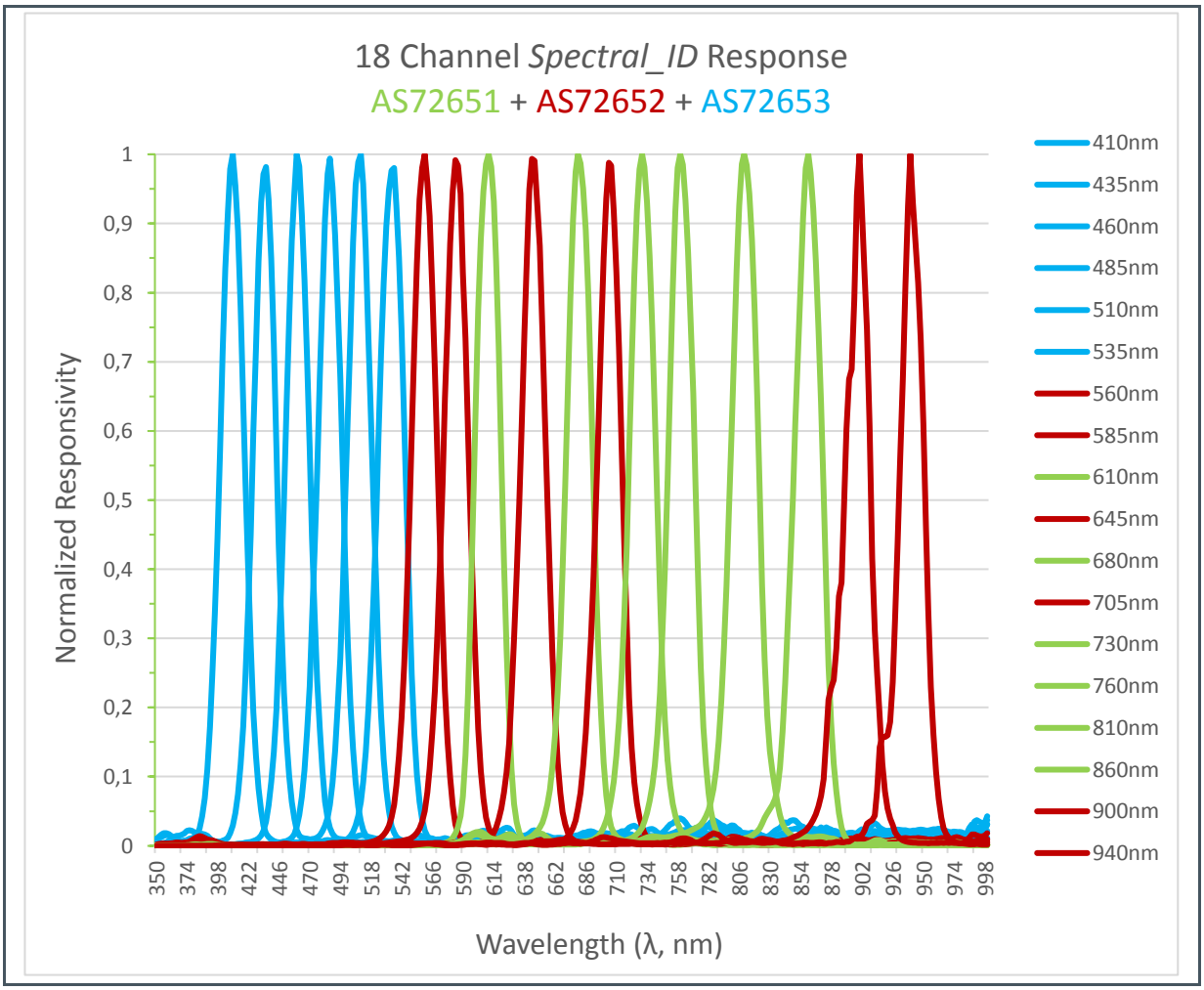
Each AS72651, AS72652, or AS72653 is packaged in a 20-pin LGA package.

Figure 1:
AS7265x Multispectral Chipset Optical Filters

Device	Channel	Filter Type	Center λ (nm)	FWHM (nm)
AS72653	A	Gaussian/BP	410	20
AS72653	B	Gaussian/BP	435	20
AS72653	C	Gaussian/BP	460	20
AS72653	D	Gaussian/BP	485	20
AS72653	E	Gaussian/BP	510	20
AS72653	F	Gaussian/BP	535	20
AS72652	G	Gaussian/BP	560	20
AS72652	H	Gaussian/BP	585	20
AS72651	R	Gaussian/BP	610	20
AS72652	I	Gaussian/BP	645	20
AS72651	S	Gaussian/BP	680	20
AS72652	J	Gaussian/BP	705	20
AS72651	T	Gaussian/BP	730	20
AS72651	U	Gaussian/BP	760	20
AS72651	V	Gaussian/BP	810	20
AS72651	W	Gaussian/BP	860	20
AS72652	K	Gaussian/BP	900	20
AS72652	L	Gaussian/BP	940	20

[1] See application note “AS72xx External Flash program and update”

Figure 2:
Typical Spectral Responsivity



3 Hardware Design Considerations

Depending on the used firmware releases, there exist for AS7265x schematic two alternative versions, named generation 1 and 2. Compared with the older generation 1, in AS7265x new generation 2, the INT and SLV_RST1 pins are swapped on default. Note, all components for generation 1 and 2 in firmware, software and hardware are not compatible.

The following Figure 3 and Figure 4 show the schematics for generation 1 and 2.

Figure 3:
Typical Schematic AS7265x (Version FW Generation 1)

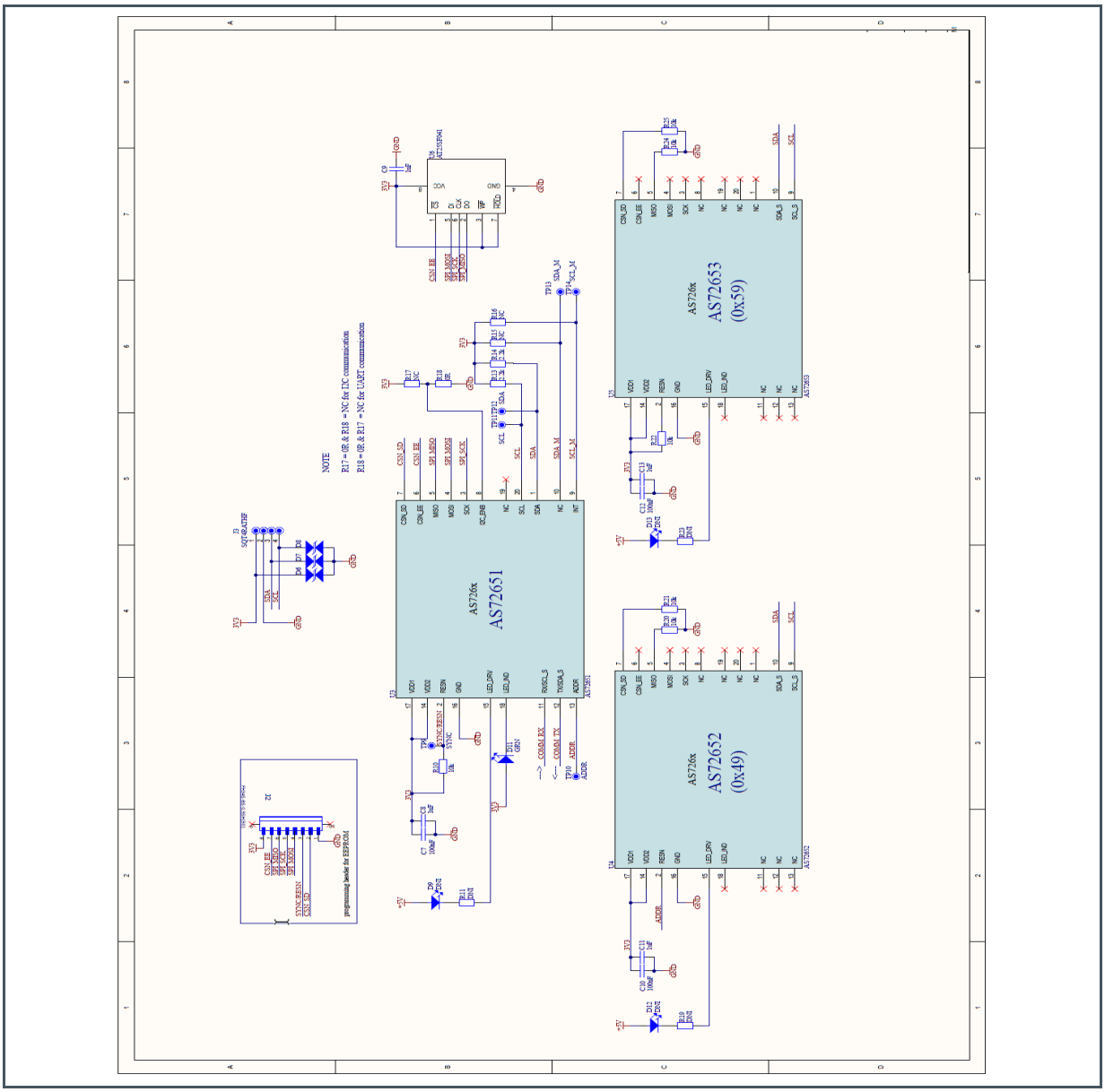
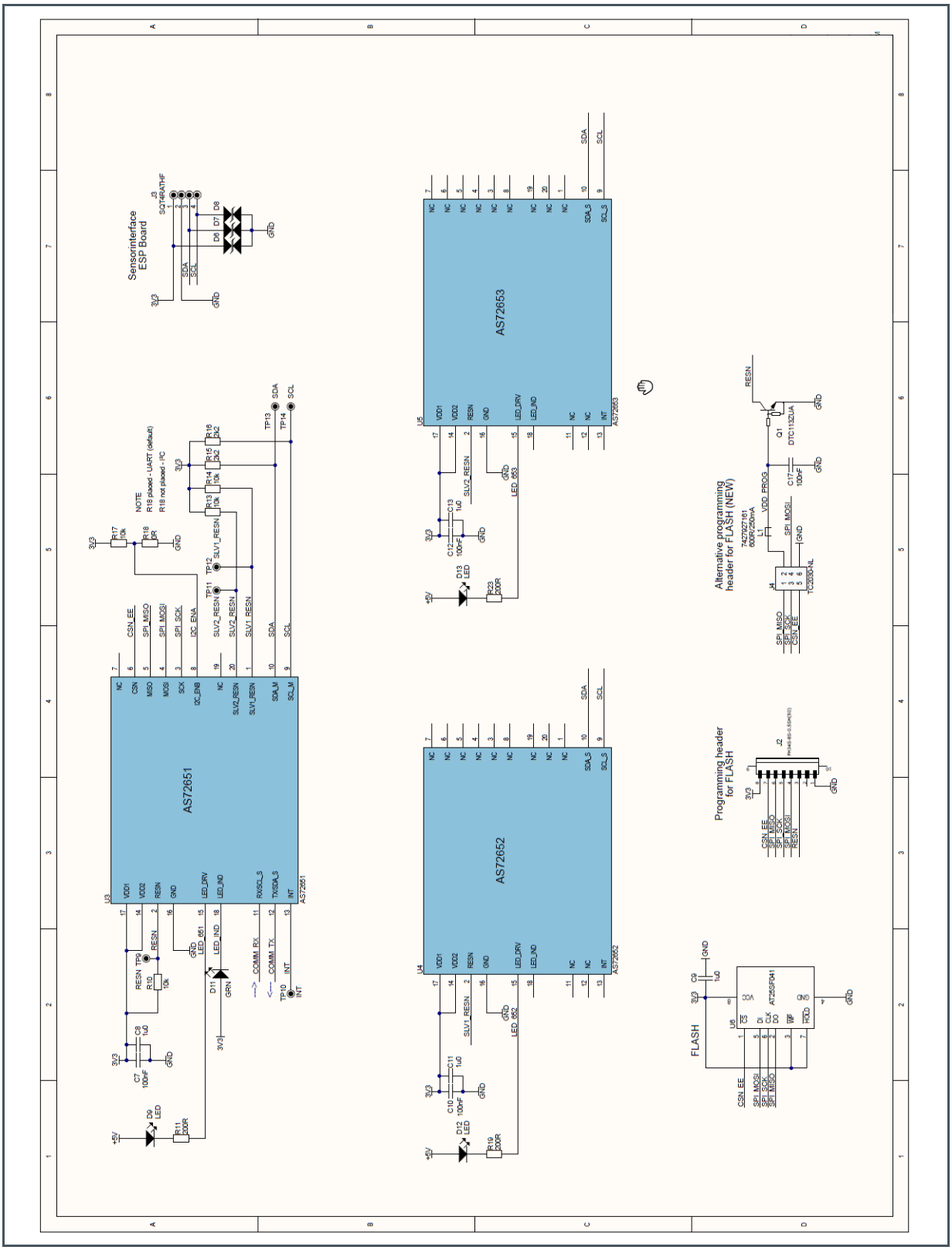


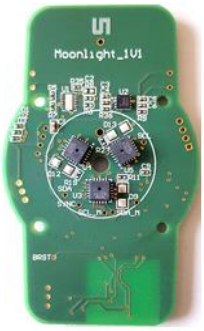
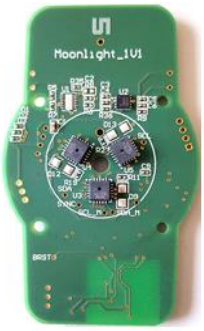
Figure 4:
Typical Schematic AS7265x (Version FW Generation 2)



3.1 HW/FW Generation 1 vs. HW/FW Generation 2

ams AS7265x enjoyed a production-level firmware release in 2018. As part of that release, v12 and higher firmware redefined several pin functions to enhance compatibility and system reliability. In addition, demo kits were updated to version 2V0 to support the revised “Generation 2” pinout. Column 1 of the following table describes the relevant Generation 1 pinout and pre-production firmware versions. Column 2 describes the Gen1 pin-compatible firmware option that is only recommended for layouts/designs that were in production prior to the v12 firmware release. Column 3 describes the Generation 2 pinout, which is the recommended implementation for all designs going forward.

Figure 5:
Variants of Hardware and Firmware for AS7265x Eval Kits

HW Generation 1 (Old Design)	HW Generation 1 (Old Design)	HW Generation 2 (New Design)
Hardware		
		
Board version “Moonlight_1V1”	Board version “Moonlight_1V1”	Board version “Moonlight_2V0”
I^2C bus pins 1 & 20 of AS72651 master are connected to the I^2C bus pins 9 & 10 of AS72652 and AS72653 slaves	I^2C bus pins 1 & 20 of AS72651 master are connected to the I^2C bus pins 9 & 10 of AS72652 and AS72653 slave devices	I^2C bus pins 9 & 10 of AS72651 master are connected to the I^2C bus pins 9 & 10 of AS72652 & AS72653 slave devices
Master AS72651 and slave AS72653 share same Reset pin, but to reset slave AS72652, master AS7651 has different reset pin (pin13)	Master AS72651 and slave AS72653 share same Reset pin, but to reset slave AS72652, master AS7651 has different reset pin (pin13)	Reset pin2 of AS72652 slave is connected to pin1 of AS72651 master and Reset pin2 of AS72653 slave is connected to pin20 of AS72651 master. Master AS72651’s reset pin2 is dedicated to master only.
For firmware update via Flashcat, it has a flat band cable micro-cable port connector.	For firmware update via Flashcat, it has a Flat band micro-cable port connector.	For firmware update via Flashcat, it has flat band micro-cable port and standard Tag band cable port connectors.
I^2C/UART connection via micro-USB connector port.	I^2C/UART connection via micro-USB connector port.	I^2C/UART connection via micro-USB connector port.

HW Generation 1 (Old Design)	HW Generation 1 (Old Design)	HW Generation 2 (New Design)
Firmware		
Firmware version lower than 9Vxx	Gen1 pin compatible v12 firmware	Gen2 pinout v12 firmware update via FlashCat requires 256 kb firmware bin file: “AS7265_complete.bin”
Only valid for tested and running systems, not compatible with any newer versions, not generally recommended for continued production use.	Update via FlashCat requires 256 kb firmware bin file: “AS7265_complete_moonlight.bin”	Firmware update via GUI with 56 kb firmware bin file: “AS7265_update.bin”
GUI		
GUI < 4Vx.x	GUI 4Vx.x and later	GUI 4Vx.x and later

3.2 UART Interface

AS72651 has a UART interface to communicate to the controller. AT commands can be used for data acquisition, sensors configuration, and LED drivers control. Please refer to the AS72651 datasheet for complete AT commands.

Pin11 of AS72651 is the RX of UART, which AS72651 receives the information from the controller. Pin12 of AS72651 is the TX of UART, which AS72651 transmits the information to the controller. Any windows terminal application with baud rate 115200, 8 data bit, 1 stop bit, and none parity can be used for AT commands.

Since pin11 and pin12 of AS72651 are also shared with the I^2C interface, the pin8, I^2C_ENB, has to be pulled down for UART interface configuration.

3.3 I^2C Interface

AS72651 has both I^2C master and I^2C slave interface. Both support I^2C fast mode (400 kHz) and standard mode (100 kHz).

AS72651 I^2C slave interface is used for communication with the controller. The pin11, SCL_S, is assigned to the I^2C bus clock and the pin12, SDA_S, is for the bus data. The pin8, I2C_ENB, has to be pulled HIGH.

AS72651 I^2C master interface is used for controlling AS72652 and AS72653. The pin20, SCL, is the I^2C bus clock and the pin1, SDA, is for the I^2C bus data. The communication between AS72651 and AS72652/AS72653 is managed by the firmware.

According to I²C specification, both SCL and SDA are open drain and need to be connected to a positive supply voltage via a pull-up resistor. The pull-up resistors, R13/R14 in the typical schematic, pull the line high when it is not driven low by the open drain interface. The maximum value of the pull-up resistor is limited by the bus capacitance, C_b, and the rise time, t_r, as below.

Equation 1:

$$R_{P(max)} = \frac{t_r}{(0.8473 * C_b)}$$

The bus capacitance is the total capacitance of wire, connections, and pins. I²C bus specifies the maximum rise time is 300 ns.

On the other hand, the minimum value of the pull-up resistor depends on the device's logical specifications and allows V_{OL} level to be read as a valid logical low.

Equation 2:

$$R_{P(min)} = \frac{V_{DD} - V_{OL(max)}}{I_{OL}}$$

For the AS7265x Multispectral Chipset application with 3.3 V supply voltage, 0.4 V maximum V_{OL}, and the specified minimum sink current of 3 mA for standard mode (100 kHz) or fast mode (400 kHz), the minimum pull-up resistor value is 966.7 Ω.

Then the decision of the pull-up resistor value would be based on the rise time, the total bus capacitance, and the power budget. A smaller resistor may get a short rise time but has higher power consumption.

Please note, the typical schematic here is configured as the UART interface by default so there are no pull-up resistors on the AS72651 I²C slave interface. If the I²C slave interface is needed, please add the pull-up resistors on either the controller side or on the AS72651 I²C interface.

3.4 Light Source Selection

Figure 6:
Generic Application

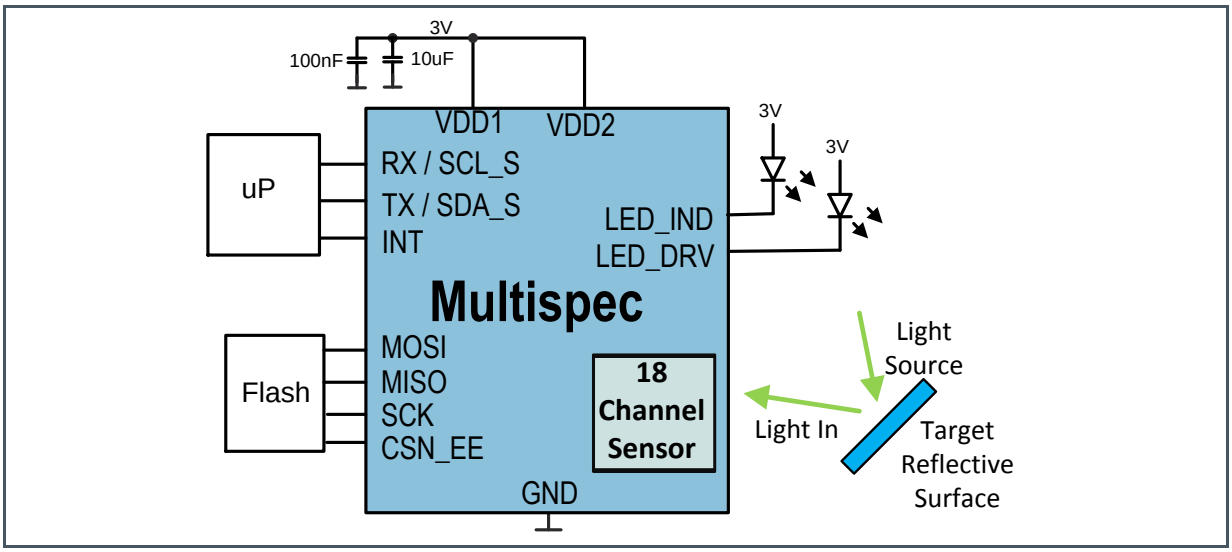


Figure 6 shows the AS7265x Multispectral Chipset generic application. AS7265x chipset produces the sensor output data based on the received reflection of light rays from the target. The light source selection would be dependent on the spectral responsivity of reflected light and characteristics of the target. For example, if the target is expected to absorb 610 nm light in the visible range and the application needs to distinguish the target from others, a broadband white LED might be used as the light source for AS7265x and AS7265x 610 nm channel should be checked. Various applications may require different light sources.

3.5 Other Connections

The AS72651 device needs a flash memory to store the firmware and the data. The flash memory should be at least 2 Mbits operating at SPI mode 0 with the byte write supported. With RESN pin, AS72651 supports various flash memory programming methodologies. Please refer to Application Note “AS72xx Flash program and update”.

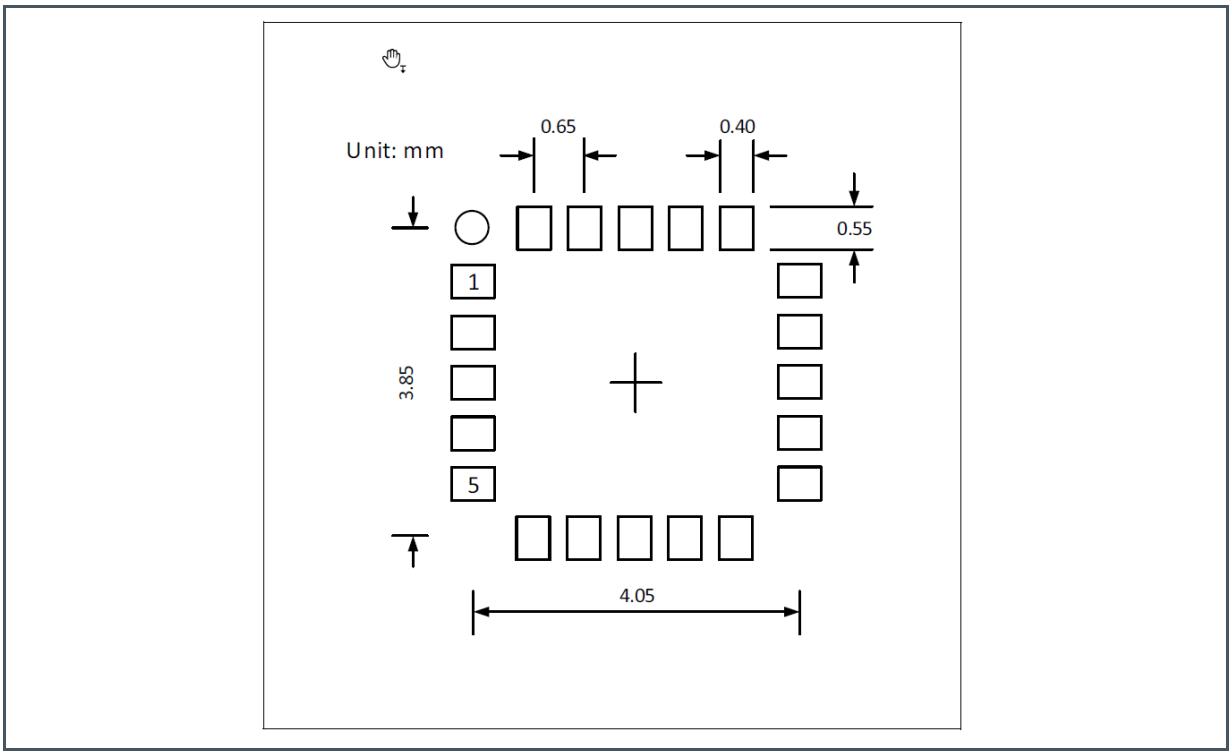
The LED, D11 in the typical schematic, is recommended for debugging purposes. During AS72651 power-up, D11 should be on for a short time and off. If D11 is blinking, it indicates there is an issue with accessing the flash memory content.

The AS72651/AS72652/AS72653 devices require a 3.3 V supply on both VDD1 and VDD2 pins associated with the decoupling capacitors, C7/C8, C10/C11, and C12/C13 in the schematic. Each LED_DRV pin on AS72651/AS72652/AS72653 can drive external LED sources with various amount of sink currents. The LED current can be configured as 12.5 mA, 25 mA, 50 mA and 100 mA.

3.6 PCB Layout Considerations

AS72651/AS72652/AS72653 have the same 20-pin LGA package sharing the same PCB footprint.

Figure 7:
AS72651/AS72652/AS72653 PCB Footprint Recommendation

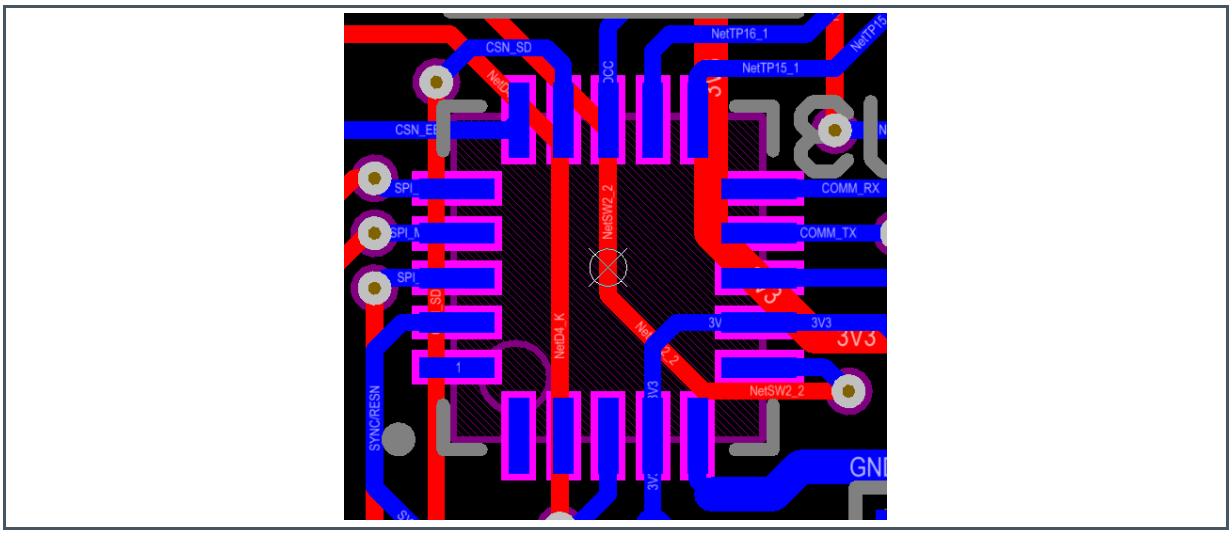


(1) Unless otherwise specified, all dimensions are in millimeters.
(2) Add 0.05 mm all around the nominal lead width and length for the PCB pad land pattern.

The schematic symbol and PCB layout footprint can also be provided in the Altium design format. Please contact **ams** support team to get the library file.

The PCB layout for AS72651/AS72652/AS72653 devices is simple. The first recommendation is to place the decoupling capacitors closed to VDD pins of AS72651/AS72652/AS72653 devices. The second recommendation is to avoid putting any via underneath the device. Please refer to the Figure below.

Figure 8:
Sample Layout of AS72651/AS72652/AS72653 Devices

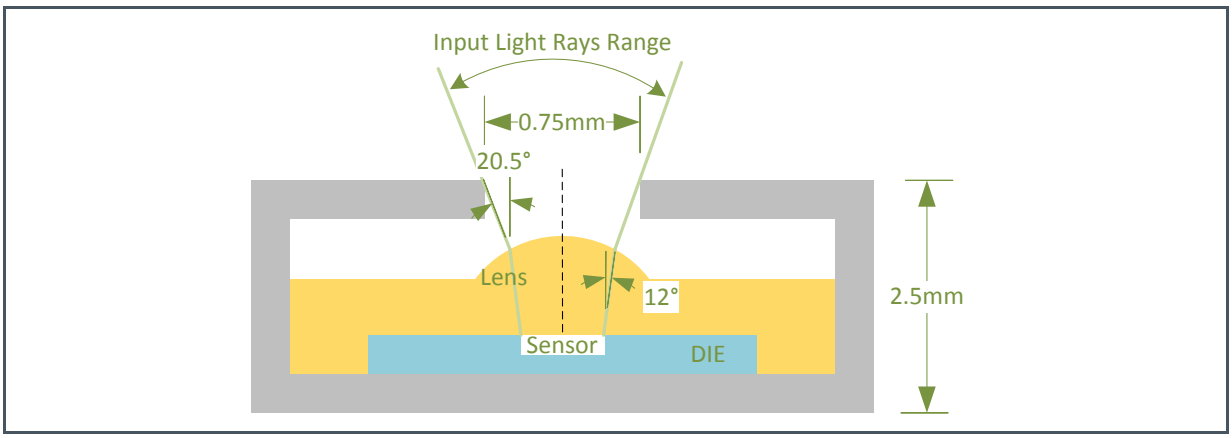


For the system level layout consideration, the generic PCB layout rules for digital designs should apply. In general, the wiring must be chosen so that crosstalk and interference to/from the bus lines is minimized. The I^2C bus specification also recommends that place VDD and/or GND between SDL and SDA if the traces are longer than 10 cm.

The length of the I^2C bus depends on the load of the bus and the speed you run at. The I^2C bus specification defines the maximum capacitance of the bus is 400 pF. This bus capacitance limit is specified to limit rise time reductions and allow operating at the rated frequency. In general, with a lower frequency and/or lower capacitance of the bus, you can have longer bus length.

3.7 Optic Considerations

Figure 9:
Aperture



Each AS7265x device has an open aperture on the surface. The diameter is 0.75 mm and the packaging field of view is ±20.5°. The light rays in the range as shown in the above figure would arrive at the sensor.

Since AS7265x Multispectral Chipset consists of three AS7265x devices, an external optical device might be needed so incident rays to each device is the same.

As an open-aperture device, precautions must be taken to avoid particulate or solvent contamination as a result of any manufacturing processes, including pick and place, reflow, cleaning, integration assembly and/or testing.

4 Software Design Considerations

In most system designs, AS72651 is controlled by a microcontroller. With the UART interface, the controller could configure the devices and get the sensors' data through some AT commands. The software of the microcontroller design would be simple.

The following sections would focus on the I^2C interface and the software of microcontroller design should satisfy both I^2C specification and AS7265x Multispectral Chipset register structure.

4.1 Features and Register Structure

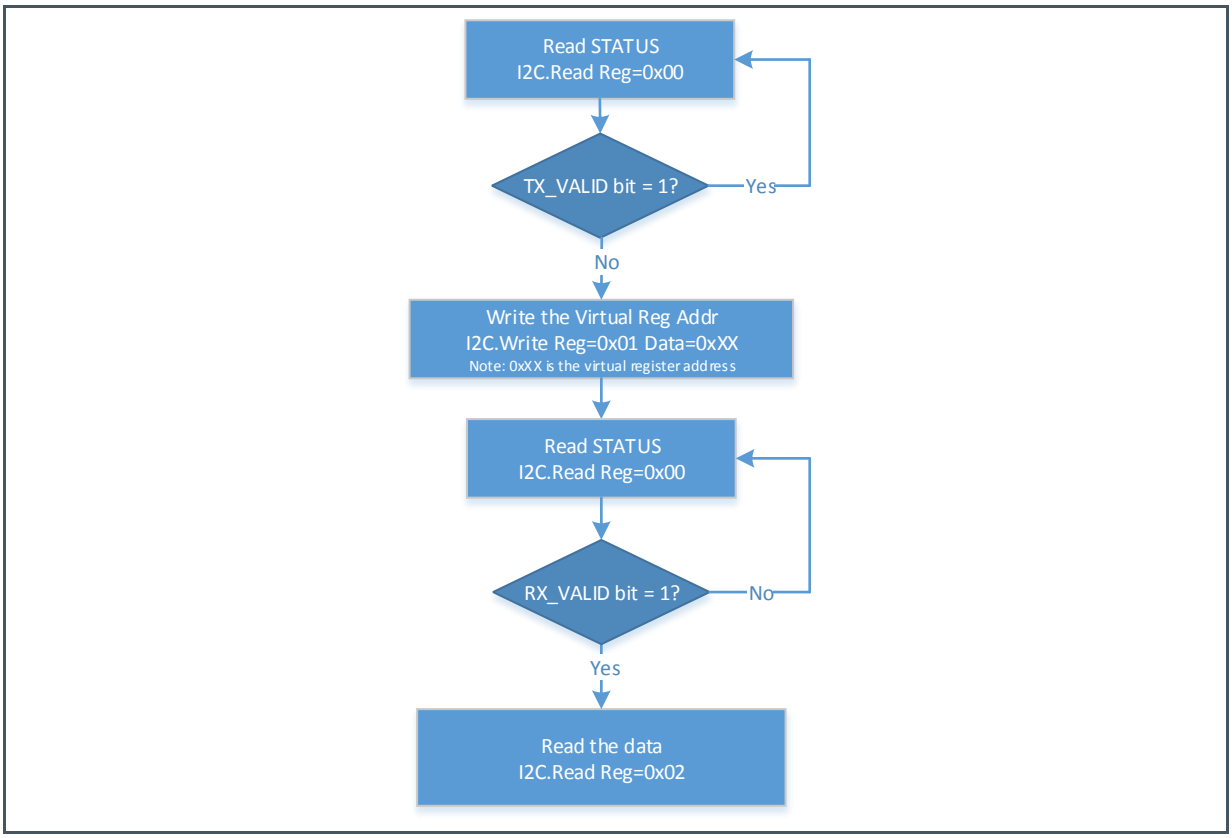
AS72651 supports I^2C both standard mode and fast mode. The addressing mode is 7+1-bit so when the controller sends a read command to AS72651, the slave address plus R/W bit should be 0x93 and when sending a write command, it should be 0x92. Both read and write are a single-byte process. AS72651 does not support the slave clock stretching mode.

AS72651 has only 3 hardware-based registers, STATUS (0x00), WRITE (0x01), and READ (0x02). The rest are implemented as virtual registers in the firmware. All virtual registers are accessed through WRITE and/or READ registers. Please refer to the datasheets for complete set of virtual registers.

4.2 I²C Virtual Register Read

To read an I²C virtual register, please follow the flow chart below.

Figure 10:
Flow Chart for Virtual Register Read



To poll the STATUS register, the controller should write the STATUS address then following a read command to get the value of the STATUS register. Figure 11 shows the format of the command for polling the STATUS register.

Figure 11:
Command for Polling the STATUS Register

Start	0x92	STATUS	Ack	Repeat Start	0x93	Data	Nack	Stop

To write the virtual register address, please program WRITE register with the virtual register address in the following format.

Figure 12:
Command for Writing the Virtual Register Address for Reading

Start	0x92	WRITE	Ack	Virtual Reg Addr	Ack	Stop

Finally below is the reading command to get the data.

Figure 13:
Command for Reading the READ Register

Start	0x92	READ	Ack	Repeat Start	0x93	Data	Nack	Stop

Figure 14:
Sample Code of Reading a Virtual Register

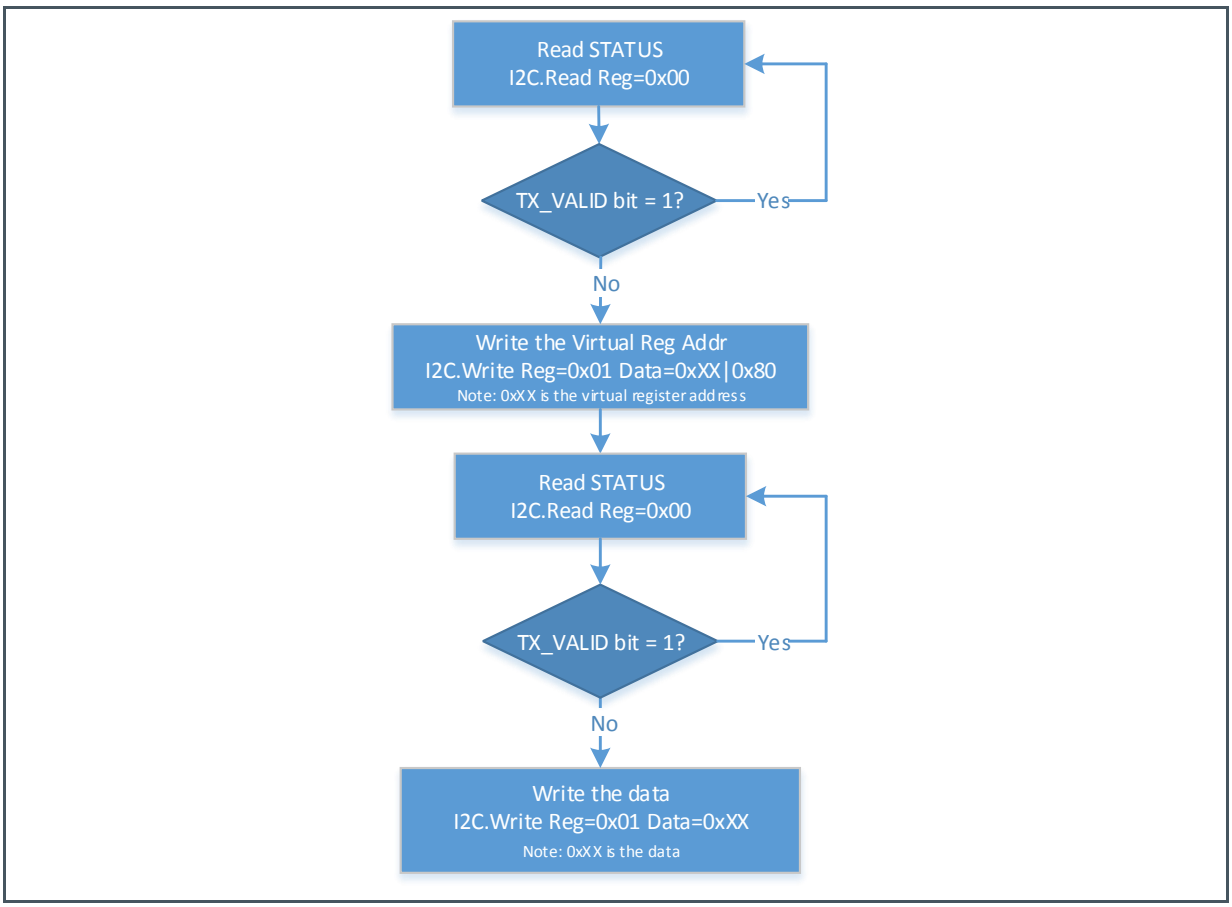
```
#define I2C_AS72XX_SLAVE_STATUS_REG   0x00
#define I2C_AS72XX_SLAVE_WRITE_REG     0x01
#define I2C_AS72XX_SLAVE_READ_REG     0x02
#define I2C_AS72XX_SLAVE_TX_VALID     0x02
#define I2C_AS72XX_SLAVE_RX_VALID     0x01

uint8_t i2cm_AS72xx_read(uint8_t virtualReg)
{
 volatile uint8_t status, d ;

 while (1)
{
        // Read slave I2C status to see if we can write the reg address.
        status = i2cm_read(I2C_AS72XX_SLAVE_STATUS_REG) ;

        if ((status & I2C_AS72XX_SLAVE_TX_VALID) == 0)
              // No inbound TX pending at slave.  Okay to write now.
              break ;
 }
 // Send the virtual register address
 i2cm_write(I2C_AS72XX_SLAVE_WRITE_REG, virtualReg) ;

 while (1)
{
        // Read the slave I2C status to see if our read data is available.
        status = i2cm_read(I2C_AS72XX_SLAVE_STATUS_REG) ;

        if ((status & I2C_AS72XX_SLAVE_RX_VALID) != 0)
              // Read data is ready for us.
```

```
30                  break ;
31     }
32     // Read the data to complete the operation.
33     d = i2cm_read(I2C_AS72XX_SLAVE_READ_REG) ;
34     return d ;
35   }
```

4.3 I²C Virtual Register Write

Writing to a virtual register is similar to the read.

Figure 15:
Flow Chart for Virtual Register Write



Please refer to the previous section for polling the STATUS register.

Writing the virtual register address for writing is not the same as the one for reading. The MSB of the virtual register address has to be set to 1 for writing.

Figure 16:
Command and for Writing the Virtual Register

Start	0x92	WRITE	Ack	Virtual Reg Addr | 0x80	Ack	Stop

Simple command for writing the data as below.

Figure 17:
Useful Command for Writing the Data

Start	0x92	WRITE	Ack	Data	Ack	Stop

Figure 18:
Sample Code of Writing a Virtual Register

```
1    void i2cm_AS72xx_write(uint8_t virtualReg, uint8_t d)
2    {
3     volatile uint8_t    status;
4
5     while (1)
6    {
7            // Read slave I2C status to see if we can write the reg address.
8            status = i2cm_read(I2C_AS72XX_SLAVE_STATUS_REG);
9
10           if ((status & I2C_AS72XX_SLAVE_TX_VALID) == 0)
11                  // No inbound TX pending at slave.  Okay to write now.
12                  break ;
13    }
14    // Send the virtual register address
15   // (setting bit 7 to indicate a pending write).
16    i2cm_write(I2C_AS72XX_SLAVE_WRITE_REG, (virtualReg | 0x80)) ;
17
18    while (1)
19   {
20           // Read the slave I2C status to see if we can write the data byte.
21           status = i2cm_read(I2C_AS72XX_SLAVE_STATUS_REG) ;
22
23           if ((status & I2C_AS72XX_SLAVE_TX_VALID) == 0)
24                  // No inbound TX pending at slave.  Okay to write data now.
25                  break ;
26    }
27    // Send the data to complete the operation.
28    i2cm_write(I2C_AS72XX_SLAVE_WRITE_REG, d) ;
29   }
```

5 Revision Information

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

- 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

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