RDACM20 Automotive Camera Platform
Technical Datasheet

Subject to Change Without Prior Notice

Preliminary
Rev 0.2

Revision History

Version	Description	Author	Date
0.1	Draft release	FDC	23 July 2015
0.2	Updated Block diagram, key features, configuration and use. Updated naming convention to include lens variant	FDC	29 December 2015

This document is a guide to those wishing to develop applications using the RDACM20 Automotive Camera Platform. No license is granted by implication or otherwise of any intellectual property rights of IMI.

The information contained in this document is subject to change without prior notice.

Disclaimer: The specifications and test results presented in this document are from a limited number of samples characterized.

TABLE OF CONTENTS

1. General Description 3
1.1 Overview 3
1.2 Key Features 4
1.3 Applications 4
2. Hardware Description 4
2.1 Block Diagram 4
2.2 Typical Operating Conditions 5
2.3 Technical Data 5
2.4 Key Optical Specifications 5
2.5 Mechanical Specifications 6
2.6 Suggested Camera Mounting 7
2.7 Connection Specifications 8
2.8 Mating Cable Assembly Specifications 8
3. Configuration and Use 9
3.1 Camera Control Interface 9
3.2 Camera MCU Commands 10
3.3 Profile Structure 12
3.4 Other Control Inputs 13

1. General Description

1.1 Overview

RDACM20 is an automotive camera module suitable for advanced driver assistance (ADAS) applications. It provides a digital video stream to an electronic control unit (ECU) which processes the data into information that aids in the operation of a motor vehicle. The module is powered directly by the ECU.

The RDACM20 is a version of Minicube camera platform. Its specifications are representative of what is needed by an ADAS camera. Customization may be required however for it to conform to the exact requirements of a customer.

The camera is connected to the ECU through a coaxial cable. Power is delivered on the same conductor that is used to transmit video and control data between the ECU and the camera (power over coax or PoC).

The lens and electronic components are housed in an enclosure with an integrated FAKRA connector for connection to the ECU wire harness.

The camera module has a default configuration that is loaded on power up. The camera module can be reconfigured by the ECU through a bidirectional control channel after the default configuration has been loaded. The default configuration can be altered and stored in non-volatile memory provided for the purpose.

The RDACM20 has superior thermal characteristics and was designed to be able to withstand harsh environmental conditions.

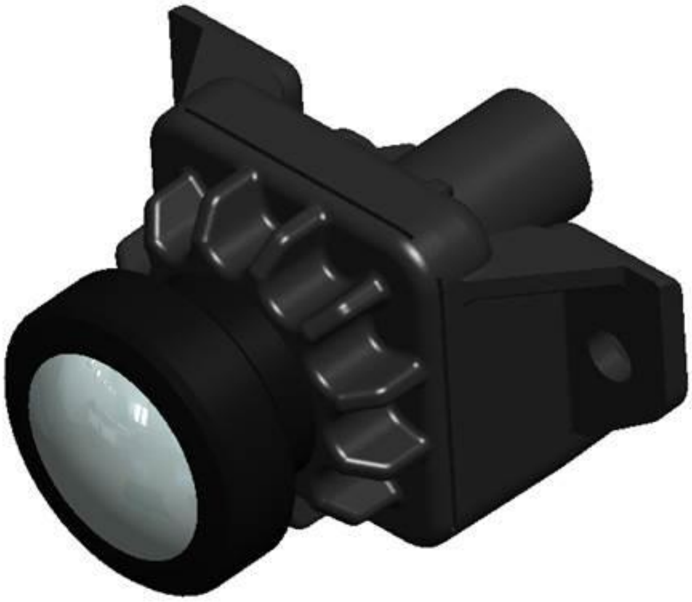


Figure 1.1: RDACM20 with DSL267 Lens

1.2 Key Features

- Suitable for ADAS and mirror replacement application
- Very small size (20.5mm x 20.5mm x 21.5mm)
- Ultra lightweight (< 10 g excluding lens)
- Adaptable to various types of lenses
- Excellent thermal management performance
- Perfect camera axis alignment.
- Power over coax interface with multiplexed video stream and control channel
- Internal MCU for camera initialization
- Patent Pending

1.3 Applications

- Automotive driver assistance
- Mirror replacement
- Unattended surveillance
- Process Automation
- Machine Vision

2. Hardware Description

2.1 Block Diagram

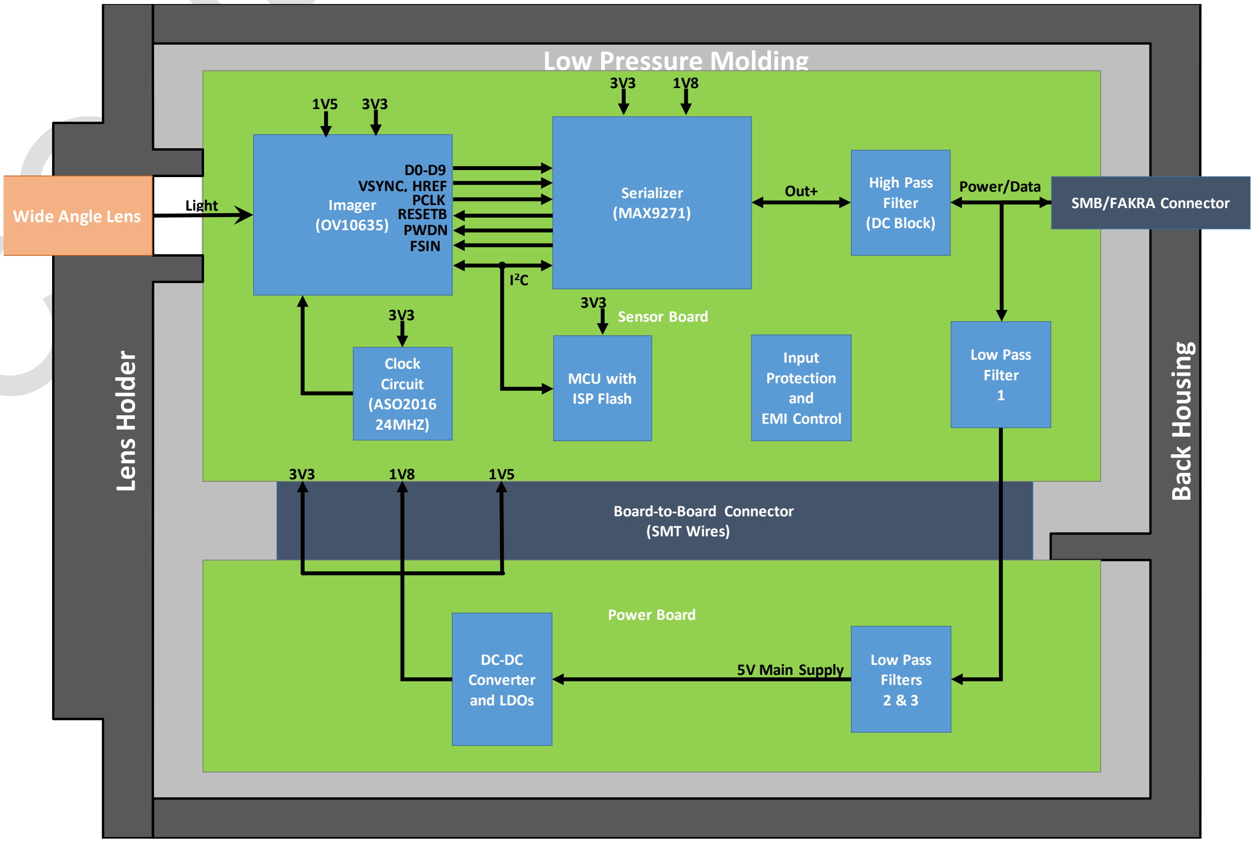


Figure 2.1: RDACM20 Block Diagram

2.2 Typical Operating Conditions

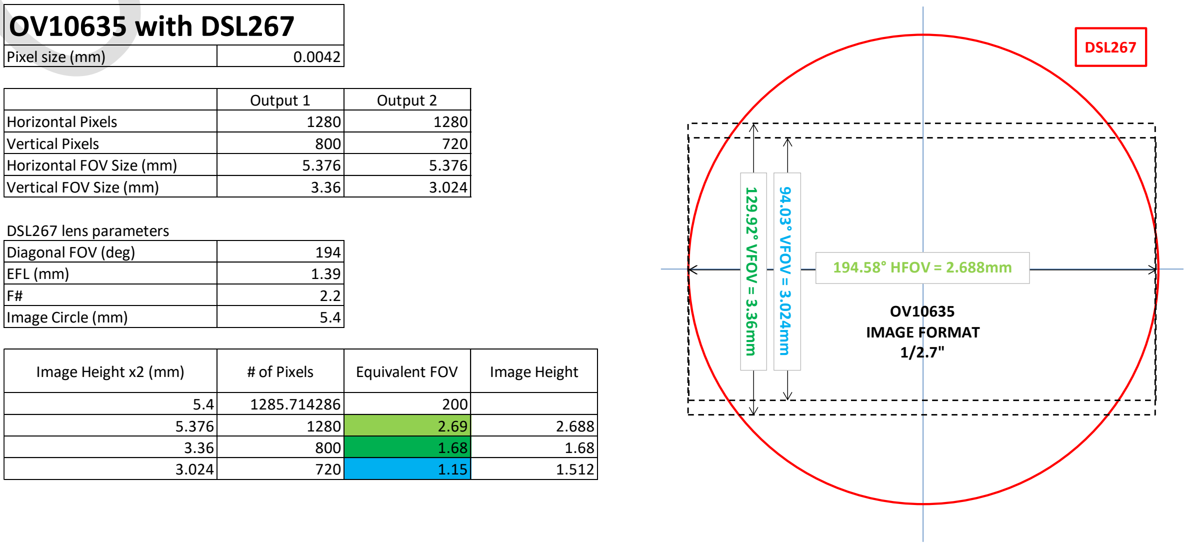
- Operating voltage: +5V to +12 V (+5V typ.)
- Current draw: TBD
- Power consumption: <1.2 W at maximum data rate
- Operating temperature = -40°C to 85°C
- Storage temperature range = -40°C to 105°C

2.3 Technical Data

Parameter	
Sensor Type	OV10635
Optical Format	1/2.7 inch
Resolution	1280 x 800 pixels
Sensor Pixel Size	4.2 µm x 4.2 µm
Sensor Sensitivity	3650 mV/lux-sec
Sensor Dynamic Range	115dB
Data Output Format	10-bit raw, 8-/10-bit YUV422
Maximum Readout	10b YUV 1280x800 30fps 88MHz 10b YUV 1280x800 30fps 96MHz BT656 1280x800 30fps 88MHz

2.4 Key Optical Specifications

- OV10635 with DSL267
- Horizontal field of view: 194.6°
- Vertical field of view: 129.92°
- Lens aperture: F# 2.2



OV10635 with DSL267	
Pixel size (mm)	0.0042

	Output 1	Output 2
Horizontal Pixels	1280	1280
Vertical Pixels	800	720
Horizontal FOV Size (mm)	5.376	5.376
Vertical FOV Size (mm)	3.36	3.024

DSL267 lens parameters

Diagonal FOV (deg)	194
EFL (mm)	1.39
F#	2.2
Image Circle (mm)	5.4

Image Height x2 (mm)	# of Pixels	Equivalent FOV	Image Height
5.4	1285.714286	200	
5.376	1280	2.69	2.688
3.36	800	1.68	1.68
3.024	720	1.15	1.512

Figure 2.4: Image Circle of DSL267 Lens vs. OV10635 Sensor Active Area

2.5 Mechanical Specifications

- Camera dimensions:
  - With DSL267 lens:
    - DxWxH: 31.48mm x 20.25mm x 20.37mm
  - Without lens:
    - DxWxH: 21.09mm x 20.25mm x 20.37mm

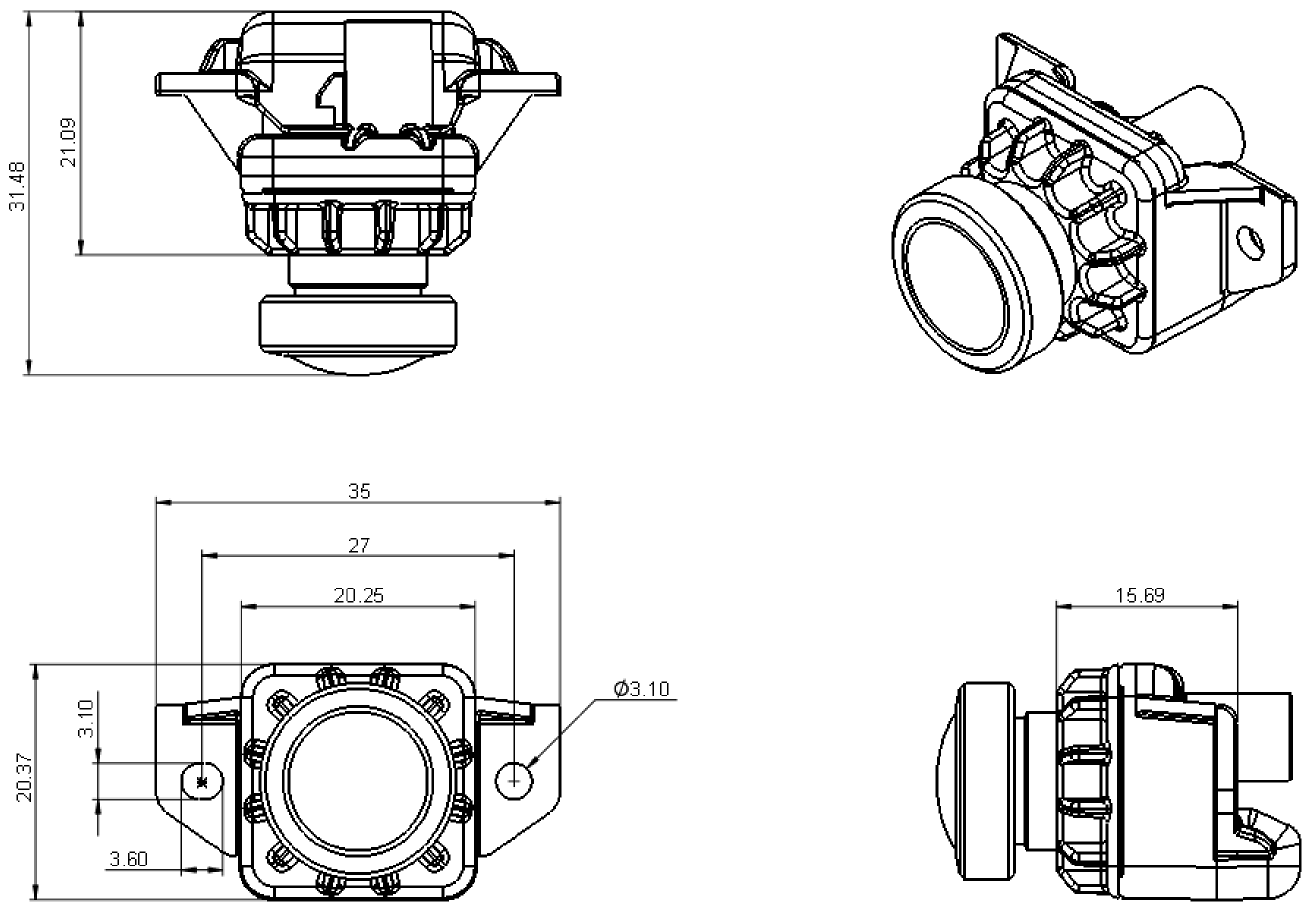


Figure 2.5: Mechanical Dimensions with DSL267 Lens

2.6 Suggested Camera Mounting

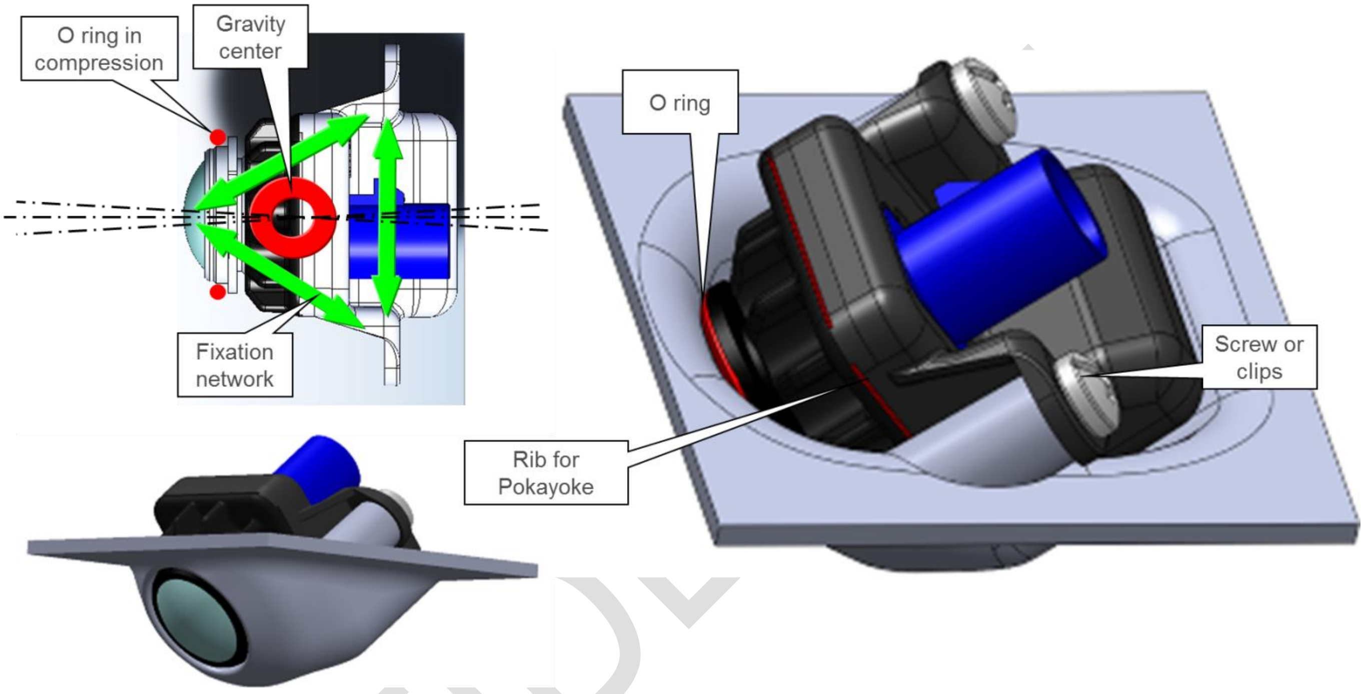


Figure 2.6 – Mounting Detail

Figure 2.6 shows a possible camera mounting scheme. Mounting holes are provided on the camera housing that accommodate M3 machine screws. It is preferred that the lens holder be supported, as illustrated, for robustness.

2.7 Connection Specifications

Interface to the camera is through a 2-coaxial cable which carries the video stream, control channel and camera voltage supply.

Connector specification: FAKRA Straight Plug Connector Interface Code – Z

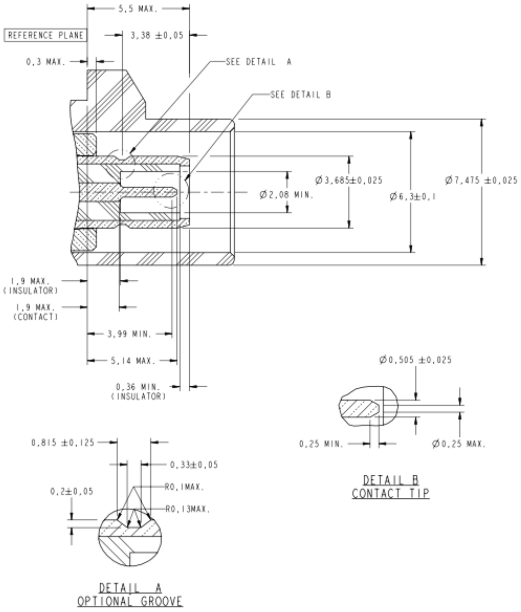


Figure 2.7 Camera Connector Details

2.8 Mating Cable Assembly Specifications

Cable assembly Connector: FAKRA Straight Jack Connector Interface Code – Z
Cable Type: RG174

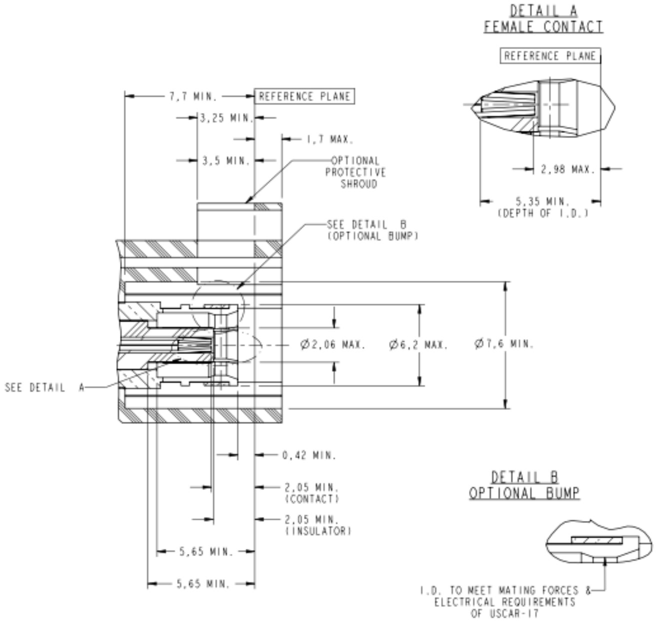


Figure 2.8 Mating Cable Connector Details

3. Configuration and Use

3.1 Camera Control Interface

The camera module has a control interface using the bidirectional control channel of the serializer-deserializer link. For compatibility, the deserializer should be implemented using the MAX9272 IC.

Figure 3.1 is a recommended filtering scheme for implementation on the deserializer side.

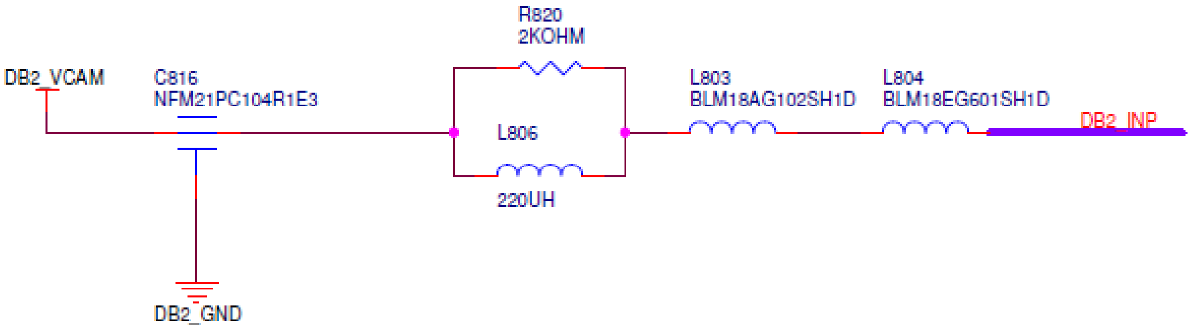


Figure 3.1 Recommended PoC Filtering Scheme on Deserializer Side

Control commands are sent via I2C. An I2C master attached to the deserializer can access the image sensor, the MCU and the serializer through the following addresses:

- Serializer (MAX9270): 0x80
- image sensor (OV10635): 0x60
- microcontroller (MCU): 0xA0

The I2C master should support clock stretching.

Access to the image sensor should follow the procedures as defined in the OV10635 datasheet. Access to the serializer should follow the procedures as defined in the MAX9271 IC data sheet.

3.2 Camera MCU Commands

The camera module MCU supports the following commands:

GET_STATUS (0x00)
This command allows the I2C master to retrieve the following information from the camera module:

- firmware version (1 byte)
- number of initialization bytes sent to the serializer prior imager initialization (1 byte)
- number of initialization bytes sent to the serializer after imager initialization (1 byte)
- number of bytes sent to the imager for initialization (1 word)
- number of bytes received and stored to the flash memory in the last STORE_PROFILE command (1 word)

GET_PROFILE (0x01)
Allows the ECU or any I2C master connected to the deserializer to read back the profile stored in the camera module. Can be used after a STORE_PROFILE command to ensure that the whole profile was received and saved without errors.

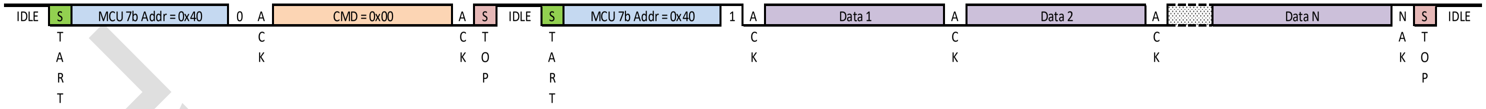
STORE_PROFILE (0x02)
Allows the ECU or any I2C master connected to the deserializer to save a new profile in the camera module.

RELOAD_PROFILEx (0x03) – Forces the camera module to re-initialize and load the profile currently stored in the flash memory of the camera.

Figure 3.2 illustrates the correct sequence for the various commands supported by the camera module.

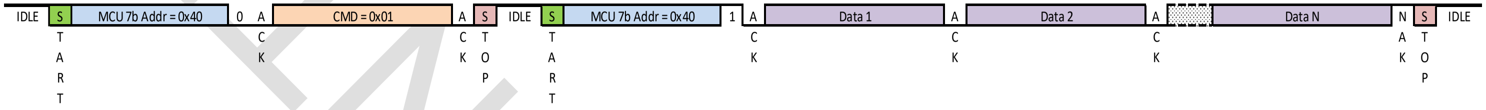
Communications format:

Command = GET_STATUS = 0x00



Data 1: Firmware Version (0x01)
Data 2: Number of bytes sent to serializer on power-up/profile reload (before imager set-up)
Data 3: Number of bytes sent to serializer on power-up/profile reload (after imager set-up)
Data 4: Number of bytes sent to imager on power-up/profile reload (MSB)
Data 5: Number of bytes sent to imager on power-up/profile reload (LSB)
Data 6: Number of bytes received/written to flash memory in last STORE_PROFILE command (MSB); resets to 0 on power-up
Data 7: Number of bytes received/written to flash memory in last STORE_PROFILE command (LSB); resets to 0 on power-up

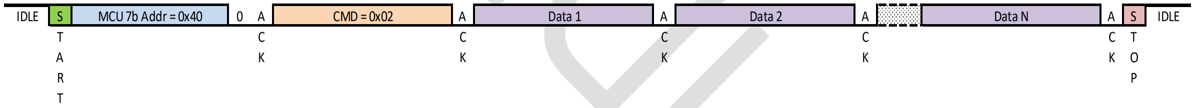
Command = GET_PROFILE = 0x01



Data 1-N: Profile (variable length, multiples of 3)
* The end of the profile has been reached when 3 successive 0xFF values are returned
* The total number of bytes to be read can be calculated from the 14th(MSB) and 15th (LSB) byte of the profile, using the following formula:

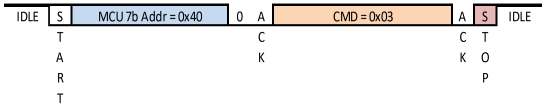
Total(Bytes) = ((Profile[13] << 8) | Profile[14]) * 3

Command = STORE_PROFILE = 0x02



Data 1-N: Profile (variable length, multiples of 3)
* The whole profile should be sent after a single STORE_PROFILE command
* Although the whole profile is to be sent in bulk, data is committed to flash memory in pages of 64 bytes; If the last page is less than 64 bytes, it will not be committed to flash memory until another command is sent
* The new profile / settings will not take effect until a RELOAD_PROFILE command is executed or a power cycle is completed

Command = RELOAD_PROFILE = 0x03



* This will load the profile currently saved in flash memory. Can be issued after a STORE_PROFILE command so the new profile will take effect.

Figure 3.2: I2C Read and Write Sequences for MCU Commands

3.3 Profile Structure

The profile stored in the flash memory of the MCU is a sequence of bytes used by the MCU to initializer the serializer and the imager on power up. It is organized in rows with each row consisting of 3 bytes. It can contain up 1365 rows for a total length of 4096 bytes.

The profile shall consist of 4 parts: Header, Serializer Sequence 1, Serializer Sequence 2, and the Imager Sequence.

HEADER
The header contains information on the I2C slave IDs and indices pointing to where the other 3 parts end.

SLAVE ID
The slave ID of the MCU can be changed using the profile. The slave IDs of the imager and the serializer should not be changed as this should match the hardware configuration. Otherwise, the initialization sequence will fail.

SERIALIZER SEQUENCE 1
The serializer sequence 1 section contain the register addresses and settings that will be sent to the serializer by the MCU before the imager is initialized. These bytes are sent over I2C to the slave ID of the serializer indicated in the header.

SERIALIZER SEQUENCE 2
The Serializer Sequence 2 section contain the register addresses and settings that will be sent to the serializer by the MCU after the imager is initialized. These bytes are sent over I2C to the slave ID of the serializer indicated in the header.

IMAGER SEQUENCE
The Imager Sequence contain the register addresses and corresponding values that will be loaded on the imager on power up. These bytes are sent over I2C to the slave ID of the imager indicated in the header.

SAMPLE PROFILE STRUCTURE:

```
0xAA 0xAA 0xAA          <-- Index 0: Start, always 0xAA 0xAA 0xAA
0x40 0x48 0xBA          <-- Index 1: Slave IDs - MCU, Imager, Serializer
0xAA 0x00 0x07          <-- Index 2: Index of end of Serializer Seq. 1
0xAA 0x00 0x0A          <-- Index 3: Index of end of Serializer Seq. 2
0xAA 0x00 0xAC          <-- Index 4: Index of end of Imager Sequence
0x03 0xC1 0x00          <-- Index 5: Start of Serializer Sequence 1
0x0D 0x51 0x00          <-- Index 6:
0x0D 0x59 0x00          <-- Index 7: End of Serializer Sequence 1
0x11 0x82 0x00          <-- Index 8: Start of Serializer Sequence 2
0x12 0x82 0x00          <-- Index 9:
0x03 0xC5 0x00          <-- Index 10: End of Serializer Sequence 1
0x00 0x22 0x00          <-- Index 11: Start of Imager Sequence
0x00 0x0C 0x04          <-- Index 12:
0x00 0x06 0x00          <-- Index 13:
…
…
0x00 0x2C 0x51          <-- Index 226:
0x00 0x05 0x01          <-- Index 227: End of Imager Sequence
```

3.4 Other Control Inputs

GPO is connected to FSIN and can be used to synchronize frames from multiple cameras. GPIO1/BWS is connected to RESETB and must be set high or in high-impedance for normal operation.