VDCMD Debugging Development Guide

V1.0

Contents

This manual is intended to be used as a guide. Photographs, graphs, charts and illustrations provided in the manual are for explanation and illustration purposes only and may differ from the specific product, so please refer to the actual product. We strive to ensure the content of this manual is accurate. However, the company makes no express or implied representations or warranties regarding the contents of this manual.
Due to product version upgrades or other needs, Quanhom may update this manual. If you need the latest version of the manual, please contact our company. Quanhom recommends that you use this manual under the guidance of a professional.

Version History

Version	Time	Description
V1.0	2023-03-21	initial version

1. Manual Description

In general, when customers use Quanhom modules for their own development, SDK development is the most popular and highly recommended method, because SDK integrates relatively complete functions and encapsulates the module's self-defined protocols as well as communication protocols with the chip, so that customers do not need to pay attention to the low-level register addresses and configurations, which makes it quick and convenient to use.

However, some customers still consider Vendercmd development based on their own business processes, frameworks.

(1) To maintain compatibility with an existing software framework and avoid the significant workload of importing new libraries.

(2) Scenarios that do not use the module in depth or use some of the commands for early-stage debugging.

(3) Scenarios where platforms such as FPGA/DSP do not support the SDK.

Based on this, this manual has been developed to guide customers in the development of the Quanhom Module VDCMD commands.

2. Development and Debugging Basics

2.1. Toolkit Installation

Before debugging, users need to install the i2ctools toolkit, which is used to assist I2C communication debugging. There are two methods for installing i2ctools: software package management tools and source code. If the user have already installed a similar toolkit, this preparation step can be skipped.

2.1.1. Package Management Installation

(1) Ensure the target platform is connected to the internet and has the apt package management tool installed.

(2) Use the command sudo apt-get install i2c-tools to install.

2.1.2. Source Code Installation

(1) Download source code: https://mirrors.edge.kernel.org/pub/software/utils/i2c-tools/

(2) If the target platform has the gcc compiler installed, download the source code from the above link, unzip it and go to the i2c- tools-4.1 directory, then run make and make install directly to install it.

If the target platform does not have the gcc compiler installed, you need to install this tool through cross-compilation, which can be done with the following command: make CC=arm-linux-gnueabihf-gcc, USE_STATIC_LIB=1. The USE_STATIC_LIB means to use static compilation. After compilation, the following five executables will be generated in the tools directory: i2cdetect, i2cdump, i2cget, i2cset, and i2ctransfer. Copy these executables to the device. Without the USE_STATIC_LIB compile option, it will be compiled using dynamic linking. After compilation, you need to copy the dynamic library libi2c.so.0 from the i2c-tools-4.3/lib directory to the/usr/bin directory on the device.

2.2. Detailed Explanation of I2C Read/Write Commands

2.2.1. i2ctransfer Command Description

The i2ctransfer command is used to send user-defined I2C messages in a single transfer, allowing the creation of I2C messages and sending them as one merged transmission. The parameter options are as follows:

- -V: Output the current version number
- -f: Forces the use of this device address, even if the address is already in use; without this parameter, the address write may fail
- -y: The command is executed automatically yes, otherwise it will prompt to confirm the execution Continue? [Y/n] Y, without the parameter y, there will be a lot of execution prompts, which can help judgment
- -v: Enable detailed output
- -a: addresses between 0x00-0x02 and 0x78-0x7f are allowed

2.2.2. i2ctransfer Command Example

The following example demonstrates how to issue a complete command to read the device version number. Send the two commands below consecutively to get the device version number, the first command is used to send commands to the device, and the second command is used to get the data:

1) i2ctransfer -f -y 1 w10@0x3C 0x1d 0x00 0x05 0x84 0x04 0x00 0x00 0x04 0x00 0x04

The meanings of the parameters are as follows:

- 1 Identifies the i2c controller number
- W Indicates a write operation
- 10 Indicates the number of bytes written.
- @0x3C Identifies slave addr device address as 0x3C
- 0x1d00 is the address of the register to be written
- 0x05 0x84 0x04 0x00 0x00 0x04 0x00 0x04 for commands sent to the device

2) i2ctransfer -f -y 1 w2@0x3c 0x1d 0x08 r4

- 1 Identifies the i2c controller number
- w Indicates a write operation
- 2 Indicates the number of bytes written.
- @0x3C Identifies slave addr device address as 0x3C
- 0x1d08 is the address of the register to be written
- r4 Indicates that 4 bytes of data are to be read

2.3. I2C Device Detection

The following commands are commonly used for I2C device detection:

- Detect all i2c buses: i2cdetect -1
- Detect devices attached to the I2C bus: i2cdetect -r -y 1

The number 1 represents the device attached to the I2C-1 bus. The I2C slave device address for Quanhom's products is 0x3C, so users can detect whether Quanhom's devices are correctly connected to the I2C bus based on the actual hardware design.

- Chip communication status check: i2ctransfer -f -y 1 w2@0x3C RegisterAddress(2 bytes) r1

If the I2C slave device address is 0x3C and the register addresses are 0x0000 and 0x0001, the commands are as follows:

i2ctransfer -f -y 1 w2@0x3C 0x00 0x00 r1 //Detect 0x0000 register, which should return 0x53 if normal.

i2ctransfer -f -y 1 w2@0x3C 0x00 0x01 r1 //Detect 0x0001 register, which should return 0x52 if normal.

Note: Passing this step does not necessarily mean the device is fully functional. There are two modes of chips in the module: ROM mode and Cache mode. Only Cache mode is the normal state, if in ROM mode, it might indicate that the module's flash is not powered or there is a soldering issue with the pins.

- I2C Read Firmware Version Number

i2ctransfer -f -y 1 w10@0x3C 0x1d 0x00 0x05 0x84 0x04 0x00 0x00 0x04 0x00 0x04

i2ctransfer -f -y 1 w2@0x3C 0x1d 0x08 r4

Please refer to the example section for command explanation. This command can be used to test whether the chip enters the Cache mode. If the command returns correctly, the chip is in Cache mode. If it fails, the chip is in ROM mode, and the flash power supply should be checked.

2.3.1. Command Usage

(1) For commands whose results cannot be directly determined, this method can be used to obtain the execution result from the core.

(2) For read commands, this result can be read after the command is sent, and for write commands, this result can be obtained after both the command and the data are sent.

(3) Since it takes time for the core to execute the command, it is necessary to continuously get the result from the core by polling after sending the command. If bit[0] is 1, it means that the command has not been executed, then continue polling, if bit[0] is 0, the command execution is complete, and the result can be checked to determine whether the execution succeeded or failed.

(4) For most commands, the execution time is relatively short, the polling time can be set to 2s, for some special commands, the execution time will be longer, the polling time needs to be extended, which will be pointed out in the following section.

3. All Commands Analysis

I2C commands are divided into read and write commands based on whether there is return data in the read direction. Read commands retrieve data from the device and therefore always return data,such as version acquisition. Write commands are categorized into simple setup commands without data and commands that require sending additional data. For example, commands to switch data sources are simple settings, while commands for image output require additional data to be sent.

Different cores may support varying functionalities. Based on the generality of the commands, VDCMD commands are classified into three categories:

- Common commands: commands that apply to any of Quanhom's products, regardless of product.
- ASIC_2121W specific commands: Tiny series, P2 and Mini256 commands only.
- ASIC384/640/1280-specific commands: Only applies to commands specific to Mini384/640, G640/G1280 and Mini1280 products.

3.1. Register Address Description

Different types of commands read and write different register addresses.

- Read commands such as read firmware version read, get PN/SN, etc.

Function	Direction	Register Address	Command Analysis
Read Command	Write	0x1D00	i2ctransfer -f -y 1 wX @0x3C 0x1D 0x00 +Command X is the length of the command (in bytes), including the 2-byte register address
	Read	0x1D00 (Note 1#, base register address, adjust for offset in actual use)	i2ctransfer -f -y 1 wX @0x3C 0x1D 0x00 +rY. X is the command length (in bytes), including the 2-byte register address, and Y is the length of the data to be read.

- Write Commands Without Data (e.g., switching the data source for intermediate image output, manually triggering the shutter).

Function	Direction	Register Address	Command Analysis
Write Command (No Data)	Write	0x1D00	i2ctransfer -f -y 1 wX @0x3C 0x1D 0x00 + command. X is the length of the command (in bytes), including the 2-byte register address.

- Write Commands With Data (e.g., configuring resolution for image output, setting pseudo-color by writing pseudo-color code values):

Function	Direction	Register Address	Command Analysis
Write Command (WithData)	Write	0x9D00	i2ctransfer -f -y 1 wX @0x3C 0x9D 0x00 + command. X is the length of the command (in bytes), including the 2-byte register address.
	Write	0x1D00 (Note 1#, base register address, adjust for offset in actual use)	i2ctransfer -f -y 1 wX @0x3C 0x9D 0x00 +command. X is the length of the command (in bytes), including the 2-byte register address.

Note 1#: The actual address used is not 0x1D00, it is the offset address based on 0x1D00, the offset length depends on the length of the command in the previous write command, such as the length of 8 bytes in the previous write command, the actual address of the second write is 0x1D08.

3.2. Command Execution Result Query

3.2.1. Get Command Execution Result

(1) i2ctransfer -f -y 1 w2 @0x78 0x2C 0x00 r1

The acquired data is the current core state, and the acquired value corresponds to the actual state as follows:

bit[0] indicates whether the core is idle or not: when it is 0, it means the core is idle; when it is 1, it means the core is busy.

bit[1] indicates whether the command is finally executed successfully or not; when it is 0, it means the command is executed successfully; when it is 1, it means the command is failed.

bit[7:2] indicates the type of command execution failure: when all bits are 0, it means that there is no execution error; when any bit is 1, it means different failure types.

3.3. Common Commands

3.3.1. Basic System Functions

- Get firmware version number

(1) i2ctransfer -f -y 1 w10 @0x3C 0x1D 0x00 0x05 0x84 0x04 0x00 0x00 0x00 0x00 0x04

(2) i2ctransfer -f -y 1 w2@0x3C 0x1D 0x08 r4

This instruction has return data, the return value is the firmware version, the length is 4 bytes data[0]-data[3], where data[0] is the minor version number, data[1] is the major version number, and the remaining two bytes are reserved and not currently used.

- Get PN

(1) i2ctransfer -f -y 1 w10@0x3C 0x1D 0x00 0x05 0x84 0x06 0x00 0x00 0x00 0x00 0x30

(2) i2ctransfer -f -y 1 w2@0x3C 0x1D 0x08 r48

This command has return data; the return value is a PN string with a length of 48 bytes.

- Get SN

(1) i2ctransfer -f -y 1 w10@0x3C 0x1D 0x00 0x05 0x84 0x07 0x00 0x00 0x00 0x00 0x10

(2) i2ctransfer -f -y 1 w2@0x3C 0x1D 0x08 r16

This command has return data; the return value is a SN string with a length of 16 bytes.

- Save the set algorithm parameters

i2ctransfer -f -y 1 w10@0x3C 0x1D 0x00 0x01 0x0B 0x0C 0x00 0x00 0x00 0x00 0x00
This command saves algorithm parameters, including the version number, UART communication switch, shutter parameters, image parameters, overexposure parameters, TPD temperature measurement parameters, timestamp, bird mode, and shutter protection parameters.

- Reset Algorithm Parameters to Default

i2ctransfer -f -y 1 w10@0x3C 0x1D 0x00 0x01 0x0E 0x03 0x00 0x00 0x00 0x00 0x00
This command is used to restore the above attribute parameters (version number, UART communication switch, shutter parameter, image parameter, overexposure parameter, TPD temperature measurement parameter, timestamp, bird mode, shutter protection parameter, etc.) to their defaults, which takes a long time to execute, and therefore it is necessary to expand the polling time for the query result of the command to more than 5s.

- Image Output Command

(1) i2ctransfer -f -y 1 w10@0x3C 0x1D 0x00 0x0F 0xC1 0x00 0x00 0x00 0x00 0x00 0x08

(2) i2ctransfer -f -y 1 w10@0x3C 0x1D 0x08 data[0] ...... data[7]

The second command includes the image output configuration parameters, the length is 8 bytes, represented by data[0] to data[7]. These parameters are defined as follows (refer to commonly used command examples for more details):

data[0]: 0x00

data[1]: 0x00

data[2]:width_h

data[3]:width_1

data[4]:height_h

data[5]:height_1

data[6]:fps

data[7]:0x00（DVP）0x08（SPI

- Stop Image Output

i2ctransfer -f -y 1 w10@0x3C 0x1D 0x00 0x0F 0x02 0x00 0x00 0x00 0x00 0x00 0x00

- Switch Data Source to Intermediate Image Output

i2ctransfer -f -y 1 w10@0x3C 0x1D 0x00 0x0A 0x01 0x00 0x00 0x00 OutputFlag 0x00 0x00
OutputFlag indicates the intermediate output format, which is 1 byte long, with different values

corresponding to different data sources. Developers should enter values based on the intended purpose.

OutputFlag	Meaning
0x00	Switch data source to temperature data
0x01	Switch data source to IR data
0x02	Switch data source to KBC data
0x06	Switch data source to SNR data
0x08	Switch data source to DDE data

- Pause Intermediate Image Output Switch

i2ctransfer -f -y 1 w10@0x3C 0x1D 0x00 0x0A 0x02 0x00 0x00 0x00 0x00 0x00 0x00

After executing this command, the data source will switch to YUV format.

3.3.2. Shutter Related Functions

- Manual Shutter Release (B value correction)

i2ctransfer -f -y 1 w10@0x3C 0x1D 0x00 0x0D 0xC1 0x00 0x00 0x00 0x00 0x00 0x00

- Setting Automatic Shutter On/Off

(1) i2ctransfer -f -y 1 w10@0x3C 0x9D 0x00 0x14 0xC2 0x00 0x00 0x00 0x00 data[0] data[1]

(2) i2ctransfer -f -y 1 w10@0x3C 0x1D 0x08 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00

In Command 1, data[0] = value >> 8, data[1] = value & 0xFF, where value is the on/off state. A value of 1 indicates the shutter is open, with data[0] = 0x00 and data[1] = 0x01. A value of 0 indicates the shutter is closed, with data[0] = 0x00 and data[1] = 0x00.

- Getting The Automatic Shutter On/Off

(1) i2ctransfer -f -y 1 w10@0x3C 0x9D 0x00 0x14 0x82 0x00 0x00 0x00 0x00 0x00 0x00

(2) i2ctransfer -f -y 1 w10@0x3C 0x1D 0x08 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x02

(3) i2ctransfer -f -y 1 w2@0x3C 0x1D 0x10 r2

Command 3 return value is auto shutter switch, length is 2 bytes data[0] and data[1], according to value = data[0] << 8 + data[1], where value is 1 means open, 0 means closed.

- Setting the minimum time interval for the automatic shutter

(1) i2ctransfer -f -y 1 w10@0x3C 0x9D 0x00 0x14 0xC2 0x00 0x01 0x00 0x00 data[0] data[1]

(2) i2ctransfer -f -y 1 w10@0x3C 0x1D 0x08 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00

The value represents the time interval, with data[0] = value >> 8 and data[1] = value & 0xFF, ranging from 0 to 655.

- Getting the minimum time interval for the automatic shutter

(1) i2ctransfer -f -y 1 w10@0x3C 0x9D 0x00 0x14 0x82 0x00 0x01 0x00 0x00 0x00 0x00

(2) i2ctransfer -f -y 1 w10@0x3C 0x1D 0x08 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x02

(3) i2ctransfer -f -y 1 w2@0x3C 0x1D 0x10 r2

Command 3 returns the minimum time interval, with 2-byte result, data[0] and data[1], converted by value = data[0] << 8 + data[1].

- Setting the maximum time interval for the automatic shutter

(1) i2ctransfer -f -y 1 w10@0x3C 0x9D 0x00 0x14 0xC2 0x00 0x02 0x00 0x00 data[0] data[1]

(2) i2ctransfer -f -y 1 w10@0x3C 0x1D 0x08 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00

Command 1, data[0] = value >> 8, data[1] = value & 0xFF, value is the time interval, ranging from 0 to 655.

- Getting the maximum time interval for the automatic shutter

(1) i2ctransfer -f -y 1 w10@0x3C 0x9D 0x00 0x14 0x82 0x00 0x02 0x00 0x00 0x00 0x00

(2) i2ctransfer -f -y 1 w10@0x3C 0x1D 0x08 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x02

(3) i2ctransfer -f -y 1 w2@0x3C 0x1D 0x10 r2

Command 3, the return value is the maximum time interval and the length is 2 bytes data[0] and data[1], transferred by value = data[0] << 8 + data[1].

- Setting the automatic shutter correction trigger threshold (B-value correction)

(1) i2ctransfer -f -y 1 w10@0x3C 0x9D 0x00 0x14 0xC2 0x00 0x04 0x00 0x00 data[0] data[1]

(2) i2ctransfer -f -y 1 w10@0x3C 0x1D 0x08 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00

Command 1, data[0] = value >> 8,data[1] = value & 0xFF,value is the trigger threshold, the range is 0 to 65535, it means the amount of change of vtemp(the vtemp value reflects the temperature of the detector, and it has a linear relationship with it, 1℃ in the low temperature area corresponds to vtemp value of 39, and 1℃ in the high temperature area corresponds to vtemp value of 35, which varies slightly in different cores). By setting this value, the shutter action can be triggered when the core temperature changes to the corresponding range.

- Obtaining the automatic shutter correction trigger threshold (B-value correction)

(1) i2ctransfer -f -y 1 w10@0x3C 0x9D 0x00 0x14 0x82 0x00 0x04 0x00 0x00 0x00 0x00

(2) i2ctransfer -f -y 1 w10@0x3C 0x1D 0x08 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x02

(3) i2ctransfer -f -y 1 w2@0x3C 0x1D 0x10 r2

Command 3, The return value is the trigger threshold, which is 2 bytes long, data[0] and data[1], and is converted according to value = data[0] << 8 + data[1].

3.3.3. Image Related Commands

- Setting pseudo-colors

(1) i2ctransfer -f -y 1 w10@0x3C 0x9D 0x00 0x09 0xC4 0x00 0x00 0x00 0x00 0x00 0x01

(2) i2ctransfer -f -y 1 w3@0x3C 0x1D 0x08 data[0]

Command 2, data[0] is the pseudo-color value, the range is 1-12.

- Get pseudo-colors

(1) i2ctransfer -f -y 1 w10@0x3C 0x1D 0x00 0x09 0x84 0x00 0x00 0x00 0x00 0x00 0x01

(2) i2ctransfer -f -y 1 w2@0x3C 0x1D 0x08 r1

Command 2, The return value is a pseudo-color value ranging from 1 to 12.

- Setting the DDE Level

(1) i2ctransfer -f -y 1 w10@0x3C 0x9D 0x00 0x14 0xC3 0x00 0x02 0x00 0x00 data[0] data[1]

(2) i2ctransfer -f -y 1 w10@0x3C 0x1D 0x08 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00

Command 1, data[0] = value >> 8, data[1] = value & 0xFF, and value is the DDE level from 0 to 3.

- Obtaining the DDE Level

(1) i2ctransfer -f -y 1 w10@0x3C 0x9D 0x00 0x14 0x83 0x00 0x02 0x00 0x00 0x00 0x00

(2) i2ctransfer -f -y 1 w10@0x3C 0x1D 0x08 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x02

(3) i2ctransfer -f -y 1 w2@0x3C 0x1D 0x10 r2

Command 3, The return value is the DDE level, 2 bytes in length (data[0] and data[1], converted to value = data[0] << 8 + data[1].

- Setting the Brightness Level

(1) i2ctransfer -f -y 1 w10@0x3C 0x9D 0x00 0x14 0xC3 0x00 0x03 0x00 0x00 data[0] data[1]

(2) i2ctransfer -f -y 1 w10@0x3C 0x1D 0x08 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00

Command 1, data[0] = value >> 8, data[1] = value & 0xFF, value is the brightness level, the range is 0 to 255.

- Getting the Brightness Level

(1) i2ctransfer -f -y 1 w10@0x3C 0x9D 0x00 0x14 0x83 0x00 0x03 0x00 0x00 0x00 0x00

(2) i2ctransfer -f -y 1 w10@0x3C 0x1D 0x08 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x02

(3) i2ctransfer -f -y 1 w2@0x3C 0x1D 0x10 r2

Command 3, the return value is brightness level, length is 2 bytes data[0] and data[1], converted to value = data[0] << 8 + data[1].

- Setting the contrast level

(1) i2ctransfer -f -y 1 w10@0x3C 0x9D 0x00 0x14 0xC3 0x00 0x04 0x00 0x00 data[0] data[1]

(2) i2ctransfer -f -y 1 w10@0x3C 0x1D 0x08 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00

Command 1, data[0] = value >> 8, data[1] = value & 0xFF, value is the contrast level, the range is 0 to 255.

- Getting the contrast level

(1) i2ctransfer -f -y 1 w10@0x3C 0x9D 0x00 0x14 0x83 0x00 0x04 0x00 0x00 0x00 0x00

(2) i2ctransfer -f -y 1 w10@0x3C 0x1D 0x08 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x02

(3) i2ctransfer -f -y 1 w2@0x3C 0x1D 0x10 r2

Command 3, the return value is the contrast level, 2 bytes in length data[0] and data[1], converted to value = data[0] << 8 + data[1].

- Setting the AGC level

(1) i2ctransfer -f -y 1 w10@0x3C 0x9D 0x00 0x14 0xC3 0x00 0x05 0x00 0x00 data[0] data[1]

(2) i2ctransfer -f -y 1 w10@0x3C 0x1D 0x08 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00

Command 1, data[0] = value >> 8, data[1] = value & 0xFF, and value is the AGC level, ranging from 0 to 5.

- Getting the AGC level

(1) i2ctransfer -f -y 1 w10@0x3C 0x9D 0x00 0x14 0x83 0x00 0x04 0x00 0x00 0x00 0x00

(2) i2ctransfer -f -y 1 w10@0x3C 0x1D 0x08 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x02

(3) i2ctransfer -f -y 1 w2@0x3C 0x1D 0x10 r2

Command 3 The return value is AGC level, length is 2 bytes data[0] and data[1], transferred according to value = data[0] << 8 + data[1].

- Setting the Mirror/Flip

(1) i2ctransfer -f -y 1 w10@0x3C 0x9D 0x00 0x14 0xC3 0x00 0x09 0x00 0x00 data[0] data[1]

(2) i2ctransfer -f -y 1 w10@0x3C 0x1D 0x08 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00

Command 1, data[0] = value >> 8, data[1] = value & 0xFF, value is the mirroring/flip state, 0 means no mirroring and no flipping, 1 means mirroring and no flipping, 2 means no mirroring and flipping, 3 means mirroring and flipping.

- Getting the Mirror/Flip

(1) i2ctransfer -f -y 1 w10@0x3C 0x9D 0x00 0x14 0x83 0x00 0x09 0x00 0x00 0x00 0x00

(2) i2ctransfer -f -y 1 w10@0x3C 0x1D 0x08 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x02

(3) i2ctransfer -f -y 1 w2@0x3C 0x1D 0x10 r2

Command 3, return value is mirror/flip status, length is 2 bytes data[0] and data[1], transferred according to value = data[0] << 8 + data[1], 0 means no mirror and no flip, 1 means mirror and no flip, 2 means no mirror and flip, 3 means mirror and flip.

3.3.4. Temperature Measurement Related Commands

- Single-point secondary calibration

(1) i2ctransfer -f -y 1 w10@0x3C 0x9D 0x00 0x0E 0xC4 0x00 0x00 0x00 0x00 0x00 0x02

(2) i2ctransfer -f -y 1 w4@0x3C 0x1D 0x08 data[0] data[1]

Command 1 data[0] = temp >> 8, data[1] = temp & 0xFF, temp is the temperature in Kelvin. The execution time of this command is long, so it is necessary to expand the polling time of the command execution result query to more than 5s.

- Two-point secondary calibration

(1) i2ctransfer -f -y 1 w10@0x3C 0x9D 0x00 0x0E 0xC5 point_num 0x00 0x00 0x00 0x00 0x02

(2) i2ctransfer -f -y 1 w4@0x3C 0x1D 0x08 data[0] data[1]

In this command, point_num = 0x00 represents the low-temperature value, and point_num = 0x01 represents the high-temperature value. Low temperature should be calibrated first, followed by high temperature. data[0] = temp >> 8, data[1] = temp & 0xFF, where temp is the temperature in Kelvin. Since this command takes a long time to execute, the polling interval for querying the result should be extended to more than 20 seconds.

- Point temperature measurement

(1) i2ctransfer -f -y 1 w10@0x3C 0x9D 0x00 0x11 0x86 0x00 0x00 PointX[0] PointX[1] PointY[0] PointY[1]

(2) i2ctransfer -f -y 1 w10@0x3C 0x1D 0x08 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x02

(3) i2ctransfer -f -y 1 w2@0x3C 0x1D 0x10 r2

Where Point_X, Point_Y denote the horizontal and vertical coordinates of the point respectively, and the coordinate values start from (0,0).

- PointX[0] = (Point_X >> 8) & 0xFF
- PointX[1] = Point_X & 0xFF
- PointY[0] = (Point_Y >> 8) & 0xFF
- PointY[1] = Point_Y & 0xFF.

Command 3, the return value is the point temperature value, length 2 bytes data[0]
and data[1], transferred according to Point_Temp = data[0] << 8 + data[1].

- Line temperature measurement

(1) i2ctransfer -f -y 1 w10@0x3C 0x9D 0x00 0x11 0x89 0x00 0x00 LineXStart[0] LineXStart[1]
LineYStart[0] LineYStart[1]]

(2) i2ctransfer -f -y 1 w10@0x3C 0x1D 0x08 LineXEnd[0] LineXEnd[1] LineXEnd[0] LineXEnd[1] 0x00 0x00 0x00 0x0E

(3) i2ctransfer -f -y 1 w2@0x3C 0x1D 0x10 r14

PointX1 and PointY1 are the horizontal and vertical coordinates of the starting point of the line temperature measurement, and PointX2 and PointY2 are the horizontal and vertical coordinates of the end point of the line temperature measurement. All coordinate values start from (0,0).

- LineXStart[0] = (PointX1 >> 8) & 0xFF.
- LineXStart[1] = PointX1 & 0xFF
- LineYStart[0] = (PointY1 >> 8) & 0xFF
- LineYStart[1] = PointY1 & 0xFF
- LineXEnd[0] = (PointX2 >> 8) & 0xFF
- LineXEnd[1] = PointX2 & 0xFF
- LineYEnd[0] = (PointY2 >> 8) & 0xFF
- LineYEnd[1] = PointY2 & 0xFF

Command 3, the return value is the line temperature measurement result value, length 14 bytes, data[0] to data[13]. The return information is as follows.

- ave_temp = data[0] << 8 + data[1],average temperature.

- max_temp = data[2] << 8 + data[3],maximum temperature.
- min_temp = data[4] << 8 + data[5], the minimum temperature.
- max_temp_point.x = data[6] << 8 + data[7], the maximum temperature point horizontal coordinate.
- max_temp_point.y = data[8] << 8 + data[9], maximum temperature point vertical coordinate.
- min_temp_point.x = data[10] << 8 + data[11], horizontal coordinate of the lowest temperature point.
- min_temp_point.y = data[12] << 8 + data[13], the vertical coordinate of the lowest temperature point.

➢ Box temperature measurement

(1) i2ctransfer -f -y 1 w10@0x3C 0x9D 0x00 0x11 0x8C 0x00 0x00 RectXStart[0] RectXStart[1] RectYStart[0] RectYStart[1]

(2) i2ctransfer -f -y 1 w10@0x3C 0x1D 0x08 RectXEnd[0] RectXEnd[1] RectYEnd[0] RectYEnd[1] 0x00 0x00 0x00 0x0E

(3) i2ctransfer -f -y 1 w2@0x3C 0x1D 0x10 r14

PointX1 and PointY1 are the horizontal and vertical coordinates of the starting point of the box, and PointX2 and PointY2 are the horizontal and vertical coordinates of the end point of the box respectively. All coordinate values start from (0,0).

- RectXStart[0] = (PointX1 >> 8) & 0xFF.
- RectXStart[1] = PointX1 & 0xFF
- RectYStart[0] = (PointY1 >> 8) & 0xFF
- RectYStart[1] = PointY1 & 0xFF
- RectXEnd[0] = (PointX2 >> 8) & 0xFF
- RectXEnd[1] = PointX2 & 0xFF
- RectYEnd[0] = (PointY2 >> 8) & 0xFF
- RectYEnd[1] = PointY2 & 0xFF

Command 3, the return value is the line temperature measurement result value, length 14 bytes, data[0] to data[13]. The return information is as follows.

- ave_temp = data[0] << 8 + data[1], average temperature.
- max_temp = data[2] << 8 + data[3], maximum temperature.
- min_temp = data[4] << 8 + data[5], the minimum temperature.
- max_temp_point.x = data[6] << 8 + data[7], the maximum temperature point horizontal coordinate.
- max_temp_point.y = data[8] << 8 + data[9], maximum temperature point vertical coordinate.
- min_temp_point.x = data[10] << 8 + data[11], horizontal coordinate of the lowest temperature point.
- min_temp_point.y = data[12] << 8 + data[13], the vertical coordinate of the lowest

temperature point.

- ➢ Frame temperature measurement

(1) i2ctransfer -f -y 1 w10@0x3C 0x9D 0x00 0x11 0x83 0x00 0x00 0x00 0x00 0x00 0x00

(2) i2ctransfer -f -y 1 w10@0x3C 0x1D 0x08 0x00 0x00 0x00 0x00 0x00 0x00 0x0C

(3) i2ctransfer -f -y 1 w2@0x3C 0x1D 0x10 r12

Command 3, the return value is the line temperature measurement result value, length 12 bytes, data[0] to data[11]. The return information is as follows.

- max _temp = data[0] << 8 + data[1], maximum temperature.
- min _temp = data[2] << 8 + data[3], minimum temperature.
- max_temp_point.x = data[4] << 8 + data[5], maximum temperature point horizontal coordinate.
- max_temp_pointy = data[6] << 8 + data[7], maximum temperature point vertical coordinate.
- min_temp_point.x = data[8] << 8 + data[9], the horizontal coordinate of the lowest temperature point.
- min_temp_point.y= data[10] << 8 + data[11], vertical coordinate of the lowest temperature point.

- ➢ Setting High and Low Gain

(1) i2ctransfer -f -y 1 w10@0x3C 0x9D 0x00 0x14 0xC5 0x00 0x05 0x00 0x00 data[0] data[1]

(2) i2ctransfer -f -y 1 w10@0x3C 0x1D 0x08 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00

Command 1, data[0] = value >> 8, data[1] = value & 0xFF, value is the gain state, 0 is low gain, 1 is high gain.

The execution time of this command is long, so it is necessary to expand the polling time of the command execution result query to more than 5s.

This command can be used only for modules with high and low gain.

- ➢ Getting current gain

(1) i2ctransfer -f -y 1 w10@0x3C 0x9D 0x00 0x14 0x85 0x00 0x05 0x00 0x00 0x00 0x00

(2) i2ctransfer -f -y 1 w10@0x3C 0x1D 0x08 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x02

(3) i2ctransfer -f -y 1 w2@0x3C 0x1D 0x10 r2

Command 3, the return value is auto shutter switch, the length is 2 bytes data[0] and data[1], according to value = data[0] << 8 + data[1], 0 is low gain, 1 is high gain.

- ➢ Reset Temperature Measurement Data to Default

i2ctransfer -f -y 1 w10@0x3C 0x1D 0x00 0x01 0x0E 0x02 0x00 0x00 0x00 0x00 0x00

This command loads the default TPD temperature measurement data into the user area. The data mainly includes KT, BT, NUCT, and TPD temperature measurement parameters (including Ktemp, Btemp, Address_CA, NUC fitting parameters, etc.). Since the execution time of this command is long, the polling time for querying the result of the command should be extended to more than 5 seconds..

3.3.5. Blind Pixel Related Commands

➢ Save Blind Pixel Data

i2ctransfer -f -y 1 w10@0x3C 0x1D 0x00 0x01 0x0B 0x05 0x00 0x00 0x00 0x00 0x00

➢ Reset Blind Pixel Data to Default

i2ctransfer -f -y 1 w10@0x3C 0x1D 0x00 0x01 0x0E 0x05 0x00 0x00 0x00 0x00 0x00

The execution time of this command is long, so it is necessary to expand the polling time of the command execution result query to more than 5s.

➢ Manually Add Blind Pixel

(1) i2ctransfer -f -y 1 w10@0x3C 0x9D 0x00 0x13 0x01 0x00 0x00 0x00 0x00 X[0] X[1]

(2) i2ctransfer -f -y 1 w10@0x3C 0x1D 0x08 0x00 0x00 Y[0] Y[1] 0x00 0x00 0x00 0x00

Where PointX1, PointY1 represent the horizontal and vertical coordinates of the blind pixel points respectively.

- X[0] = (PointX1 >> 8) & 0xFF.
- X[1] = PointX1 & 0xFF
- Y[0] = (PointY1 >> 8) & 0xFF
- Y[1] = PointY1 & 0xFF

➢ Manually Delete Blind Pixel

(1) i2ctransfer -f -y 1 w10@0x3C 0x9D 0x00 0x13 0x02 0x00 0x00 0x00 0x00 X[0] X[1]

(2) i2ctransfer -f -y 1 w10@0x3C 0x1D 0x08 0x00 0x00 Y[0] Y[1] 0x00 0x00 0x00 0x00

where PointX1, PointY1 represent the horizontal and vertical coordinates of the blind pixel pixel points, respectively.

- X[0] = (PointX1 >> 8) & 0xFF.
- X[1] = PointX1 & 0xFF
- Y[0] = (PointY1 >> 8) & 0xFF
- Y[1] = PointY1 & 0xFF

3.4. ASIC_2121W Specific Commands

3.4.1. Basic System Functions

➢ Getting IR SENSOR information

(1) i2ctransfer -f -y 1 w10@0x3C 0x1D 0x00 0x05 0x84 0x03 0x00 0x00 0x00 0x1A

(2) i2ctransfer -f -y 1 w2@0x3C 0x1D 0x08 r26

Command 2 returns the IR SENSOR information as a 26-byte string.

➢ Reset all configurations to default data

i2ctransfer -f -y 1 w10@0x3C 0x1D 0x00 0x01 0x0E 0x00 0x00 0x00 0x00 0x00 0x00

➢ Reset K calibration to default data

i2ctransfer -f -y 1 w10@0x3C 0x1D 0x00 0x01 0x0E 0x01 0x00 0x00 0x00 0x00 0x00

➢ Reset user configuration to default data

i2ctransfer -f -y 1 w10@0x3C 0x1D 0x00 0x01 0x0E 0x04 0x00 0x00 0x00 0x00 0x00

This command needs to load the default area data into the user area to realize the role of restoring factory settings, and the data content mainly includes the ISP algorithm modules (CDC, HBC, SDPC, VBC, TNR, RMV, TPD, SNR, DDPC, FOCUS, AGC, DDE, GMMA, IR, KBC, OOC, etc.)

configuration parameters, the execution time is long, so it is necessary to expand the polling time of the command execution result query to more than 10s.

➢ Load default auto shutter parameters

(1) i2ctransfer -f -y 1 w10@0x3C 0x9D 0x00 0x14 0x01 0x00 0x00 0x00 0x00 0x00 0x00

(2) i2ctransfer -f -y 1 w10@0x3C 0x1D 0x08 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00

➢ Load Default Image Parameters

(1) i2ctransfer -f -y 1 w10@0x3C 0x9D 0x00 0x14 0x01 0x01 0x00 0x00 0x00 0x00 0x00

(2) i2ctransfer -f -y 1 w10@0x3C 0x1D 0x08 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00

Command 1 loads image parameters including level parameters (TNR, SNR, DDE, brightness, contrast), AGC module parameters, pseudo color mode, mirror flip mode, Zoom parameters, and video mode.

➢ Load Default Temperature Parameters

(1) i2ctransfer -f -y 1 w10@0x3C 0x9D 0x00 0x14 0x01 0x03 0x00 0x00 0x00 0x00 0x00

(2) i2ctransfer -f -y 1 w10@0x3C 0x1D 0x08 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00

Command 1 Load Temperature Default parameters mainly include correction distance, reflection temperature Tu, ambient temperature Ta, emissivity EMS, atmospheric transmittance Tau, gain and so on.

➢ Switching between different data sources

i2ctransfer -f -y 1 w10@0x3C 0x1D 0x00 0x0A 0x01 0x00 0x00 0x00 data[0] 0x00 0x00

data[0] represents the intermediate output format, length 1 byte, different values correspond to different data sources, developers need to fill in according to the purpose.

data[0]	Meaning
0x03	Switch data source to hbc_dpc data
0x04	Switch data source to VBC Data
0x05	Switch data source to TNR data
0x07	Switch data source to AGC data
0x09	Switch data source to gamma data
0x0a	Switch data source to mirror data

➢ Get current IR detector temperature

(1) i2ctransfer -f -y 1 w10@0x3C 0x1D 0x00 0x0D 0x8b 0x00 0x00 0x00 0x00 0x00 0x00

(2) i2ctransfer -f -y 1 w2@0x3C 0x1D 0x08 r2

Command 2, return value is current IR detector temperature, length is 2 bytes, data[0] to data[1]. Convert according to cur_vtemp = data[0] << 8 + data[1].

3.4.2. Shutter Related Commands

➢ Setting the manual shutter switch

i2ctransfer -f -y 1 w10@0x3C 0x1D 0x00 0x0C 0x42 value 0x00 0x00 0x00 0x00 0x00

Where value is the shutter switch, 0 is off and 1 is on.

➢ Getting the manual shutter switch and enable

(1) i2ctransfer -f -y 1 w10@0x3C 0x1D 0x00 0x0C 0x83 0x00 0x00 0x00 0x00 0x00 0x00

(2) i2ctransfer -f -y 1 w2@0x3C 0x1D 0x08 r2

Command 2 returns the shutter enable status and shutter switch, the length is 2 bytes data[0] and data[1], shutter enable status = data[0], 0 is uncontrollable, 1 is controllable. value = data[1], 0 is off, 1 is on.

➢ Setting the minimum interval between the automatic shutter and the manual shutter

(1) i2ctransfer -f -y 1 w10@0x3C 0x9D 0x00 0x14 0xC2 0x00 0x06 0x00 0x00 data[0] data[1]

(2) i2ctransfer -f -y 1 w10@0x3C 0x1D 0x08 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00

where value is the minimum interval between the automatic and manual shutter, the range is 0-655.data[0] = value >> 8, data[1]= value & 0xFF.

➢ Getting the minimum interval between the automatic and manual shutters

(1) i2ctransfer -f -y 1 w10@0x3C 0x9D 0x00 0x14 0x82 0x00 0x06 0x00 0x00 0x00 0x00

(2) i2ctransfer -f -y 1 w10@0x3C 0x1D 0x08 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x02

(3) i2ctransfer -f -y 1 w2@0x3C 0x1D 0x10 r2

Command 3 The return value is the minimum interval between auto shutter and manual shutter, the length is 2 bytes data[0] and data[1], transferred by value = data[0] << 8 + data[1].

3.4.3. Image Related Commands

➢ Setting the TNR cc

(1) i2ctransfer -f -y 1 w10@0x3C 0x9D 0x00 0x14 0xC3 0x00 0x00 0x00 0x00 data[0] data[1]

(2) i2ctransfer -f -y 1 w10@0x3C 0x1D 0x08 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00

Where value is the TNR level, range 0-3, data[0] = value >> 8, data[1] = value & 0xFF.

➢ Obtaining the TNR Level

(1) i2ctransfer -f -y 1 w10@0x3C 0x9D 0x00 0x14 0x83 0x00 0x00 0x00 0x00 0x00 0x00

(2) i2ctransfer -f -y 1 w10@0x3C 0x1D 0x08 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x02

(3) i2ctransfer -f -y 1 w2@0x3C 0x1D 0x10 r2

Command 3 The return value is the obtained TNR level, the length is 2 bytes data[0] and data[1], according to value = data[0]<< 8 + data[1] transferred, range 0-3.

➢ Setting the SNR Level

(1) i2ctransfer -f -y 1 w10@0x3C 0x9D 0x00 0x14 0xC3 0x00 0x010x00 0x00data[0] data[1]

(2) i2ctransfer -f -y 1 w10@0x3C 0x1D 0x08 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00

Where value is the SNR level, range 0-3, data[0] = value >> 8, data[1] = value & 0xFF.

➢ Getting SNR Level

(1) i2ctransfer -f -y 1 w10@0x3C 0x9D 0x00 0x14 0x83 0x00 0x010x00 0x00 0x00 0x00

(2) i2ctransfer -f -y 1 w10@0x3C 0x1D 0x08 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x02

(3) i2ctransfer -f -y 1 w2@0x3C 0x1D 0x10 r2

Command 3 The return value is the obtained SNR level, the length is 2 bytes data[0] and data[1], according to value = data[0].<< 8 + data[1] transferred,range0-3.

➢ Setting AGC Maximum Gain

i2ctransfer -f -y 1 w10@0x3C 0x9D 0x00 0x14 0xC3 0x00 0x060x00 0x00data[0] data[1]
i2ctransfer -f -y 1 w10@0x3C 0x1D 0x08 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00

Where value is the AGC maximum gain in the range 0-255, data[0] = value >> 8, and data[1] = value & 0xFF.

➢ Getting AGC Maximum Gain

(1) i2ctransfer -f -y 1 w10@0x3C 0x9D 0x00 0x14 0x83 0x00 0x060x00 0x00 0x00 0x00

(2) i2ctransfer -f -y 1 w10@0x3C 0x1D 0x08 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x02

(3) i2ctransfer -f -y 1 w2@0x3C 0x1D 0x10 r2

Command 3 The return value is the obtained AGC maximum gain, the length is 2 bytes data[0] and data[1], transferred according to value = data[0] << 8 + data[1], range 0-255.

➢ Setting the bird mode switch

(1) i2ctransfer -f -y 1 w10@0x3C 0x9D 0x00 0x14 0xC3 0x00 0x0a0x00 0x00data[0] data[1]

(2) i2ctransfer -f -y 1 w10@0x3C 0x1D 0x08 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00

Where value is the bird mode switch, 0 is off, 1 is on, data[0] = value >> 8, and data[1] = value & 0xFF.

➢ Getting the Bird Mode Switch

(1) i2ctransfer -f -y 1 w10@0x3C 0x9D 0x00 0x14 0x83 0x00 0x0a0x00 0x00 0x00 0x00

(2) i2ctransfer -f -y 1 w10@0x3C 0x1D 0x08 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x02

(3) i2ctransfer -f -y 1 w2@0x3C 0x1D 0x10 r2

Command 3 The return value is the acquired bird mode switch, the length is 2 bytes data[0] and data[1], according to value = data[0] << 8 + data[1] transferred, 0 is off, 1 is on.

➢ Scale by the Center Point

(1) i2ctransfer -f -y 1 w10@0x3C 0x9D 0x00 0x12 0x01 0x00 scale_step 0x00 0x00 0x00 0x00

(2) i2ctransfer -f -y 1 w10@0x3C 0x1D 0x08 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00

where scale_step is the scale step, the larger the step, the smaller the scale effect, range 1-4.

➢ Reduction by the Center Point

(1) i2ctransfer -f -y 1 w10@0x3C 0x9D 0x00 0x12 0x02 0x00 scale_step 0x00 0x00 0x00 0x00

(2) i2ctransfer -f -y 1 w10@0x3C 0x1D 0x08 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00

where scale_step is the reduction step, the larger the step, the smaller the reduction effect, the range is 1-4. note that the reduction is for the enlarged image, the origina image scale can not be reduced.

- Scale by the Specified Point

(1) i2ctransfer -f -y 1 w10@0x3C 0x9D 0x00 0x12 0x03 0x00 scale_step AddrX AddrY 0x00 0x00

(2) i2ctransfer -f -y 1 w10@0x3C 0x1D 0x08 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00

Where scale_step is the scale step, the larger the step, the smaller the scale effect, range 1-4. AddrX and AddrY are the horizontal and vertical coordinates of the specified point.

- Reduction by the Specified Points

(1) i2ctransfer -f -y 1 w10@0x3C 0x9D 0x00 0x12 0x04 0x00 scale_step AddrX AddrY 0x00 0x00

(2) i2ctransfer -f -y 1 w10@0x3C 0x1D 0x08 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00

Where scale_step is the reduction step, it means the reduction factor of each step is 1/(2^scale_step), so the larger the step is, the smaller the reduction effect is, the range is 1-4. Note that the reduction is for the enlarged image, and the original scale of the image can't be reduced.

- Setting the Temperature Measurement Parameters: Reflected Temperature

(1) i2ctransfer -f -y 1 w10@0x3C 0x9D 0x00 0x14 0xC5 0x00 0x01 0x00 0x00 data[1] data[0]

(2) i2ctransfer -f -y 1 w10@0x3C 0x1D 0x08 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00

where value is the reflected temperature in the range 230-500 (high gain), 230-900 (low gain) in K, data[0]= value >> 8, data[1] = value & 0xFF.

- Getting the Temperature Measurement Parameters: Reflected Temperature

(1) i2ctransfer -f -y 1 w10@0x3C 0x9D 0x00 0x14 0x85 0x00 0x01 0x00 0x00 0x00 0x00

(2) i2ctransfer -f -y 1 w10@0x3C 0x1D 0x08 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x02

(3) i2ctransfer -f -y 1 w2@0x3C 0x1D 0x10 r2

Command 3 The return value is to get the reflection temperature of the temperature measurement parameter, the length is 2 bytes data[0] and data[1], according to value = data[0].<< 8 + data[1] transferred in the range 230-500 (high gain), 230-900 (low gain).

- Setting Temperature Measurement Parameters: Atmospheric Temperature

(1) i2ctransfer -f -y 1 w10@0x3C 0x9D 0x00 0x14 0xC5 0x00 0x02 0x00 0x00 data[0] data[1]

(2) i2ctransfer -f -y 1 w10@0x3C 0x1D 0x08 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00

where value is the atmospheric temperature in the range 230-500 (high gain), 230-900 (low gain) in K, data[0]= value >> 8, data[1] = value & 0xFF.

- Getting the Temperature Measurement Parameters: Atmospheric Temperature

(1) i2ctransfer -f -y 1 w10@0x3C 0x9D 0x00 0x14 0x85 0x00 0x02 0x00 0x00 0x00 0x00

(2) i2ctransfer -f -y 1 w10@0x3C 0x1D 0x08 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x02

(3) i2ctransfer -f -y 1 w2@0x3C 0x1D 0x10 r2

Command 3 The return value is the atmospheric temperature of the temperature measurement parameter, the length is 2 bytes, data[0] and data[1], according to value = data[0].<< 8 + data[1] transferred in the range 230-500 (high gain), 230-900 (low gain).

➢ Setting Temperature Measurement Parameters: Emissivity

(1) i2ctransfer -f -y 1 w10@0x3C 0x9D 0x00 0x14 0xC5 0x00 0x03 0x00 0x00 data[0] data[1]

(2) i2ctransfer -f -y 1 w10@0x3C 0x1D 0x08 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00

Where value is the emissivity, in the range 1-128, data[0] = value >> 8, and data[1] = value & 0xFF.

➢ Getting the Thermometric Parameters: Emissivity

(1) i2ctransfer -f -y 1 w10@0x3C 0x9D 0x00 0x14 0x85 0x00 0x03 0x00 0x00 0x00 0x00

(2) i2ctransfer -f -y 1 w10@0x3C 0x1D 0x08 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x02

(3) i2ctransfer -f -y 1 w2@0x3C 0x1D 0x10 r2

Command 3 The return value is to get the emissivity of the temperature measurement parameter, the length is 2 bytes data[0] and data[1], according to value = data[0]<< 8 + data[1] transferred in the range 1-128.

➢ Setting Temperature Measurement Parameters: Atmospheric Transmittance

(1) i2ctransfer -f -y 1 w10@0x3C 0x9D 0x00 0x14 0xC5 0x00 0x04 0x00 0x00 data[0] data[1]

(2) i2ctransfer -f -y 1 w10@0x3C 0x1D 0x08 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00

where value is the atmospheric transmittance in the range 1-128, data[0] = value >> 8, and data[1] = value & 0xFF.

➢ Getting the Thermometric Parameters: Atmospheric Transmittance

(1) i2ctransfer -f -y 1 w10@0x3C 0x9D 0x00 0x14 0x85 0x00 0x04 0x00 0x00 0x00 0x00

(2) i2ctransfer -f -y 1 w10@0x3C 0x1D 0x08 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x02

(3) i2ctransfer -f -y 1 w2@0x3C 0x1D 0x10 r2

Command 3 the return value is atmospheric transmittance of the temperature measurement parameter, the length is 2 bytes data[0] and data[1], transferred according to value = data[0] << 8 + data[1], the range is 1-128.

➢ Getting the highest temperature of the whole frame

(1) i2ctransfer -f -y 1 w10@0x3C 0x9D 0x00 0x11 0x81 0x00 0x00 0x00 0x00 0x00 0x00

(2) i2ctransfer -f -y 1 w10@0x3C 0x1D 0x08 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x02

(3) i2ctransfer -f -y 1 w10@0x3C 0x1D 0x10 r2

Command 3 the return value is the highest temperature of the whole frame, the length is 2 bytes data[0] and data[1], according to max_temp = data[0].<< 8 + data[1] transferred in 1/16 K.

➢ Getting the lowest temperature of the whole frame

(1) i2ctransfer -f -y 1 w10@0x3C 0x9D 0x00 0x11 0x82 0x00 0x00 0x00 0x00 0x00 0x00

(2) i2ctransfer -f -y 1 w10@0x3C 0x1D 0x08 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x02

(3) i2ctransfer -f -y 1 w10@0x3C 0x1D 0x10 r2

Command 3 the return value is the lowest temperature of the whole frame, the length is 2 bytes, data[0] and data[1], according to min_temp = data[0] << 8 + data[1] transferred in 1/16 K.

3.4.4. .Vignetting Effect Related

- Close the de- vignetting effect enable

i2ctransfer -f -y 1 w10@0x3C 0x1D 0x00 0x0D 0x05 0x00 0x00 0x00 0x00 0x00 0x00

- Open the de- vignetting effect enable

i2ctransfer -f -y 1 w10@0x3C 0x1D 0x00 0x0D 0x0D 0x00 0x00 0x00 0x00 0x00 0x00 0x00

- Automatic vignetting effect calibration

i2ctransfer -f -y 1 w10@0x3C 0x1D 0x00 0x0D 0x0E zoom_scale 0x00 0x00 0x00 0x00 0x00
Where zoom_scale indicates the level of relevance, can take the value 1/2/4, the default value is 1.
The specific calibration procedure can be found in the document *Blind pixel calibration* (Dead pixel correction)

3.5. ASIC384/640/1280 Specific Commands

- Switching the UV Component Of YUYV

i2ctransfer -f -y 1 w10@0x3C 0x1D 0x00 0x0F 0x85 value 0x00 0x00 0x00 0x00 0x00
where Value is as follows:

- 0:UYVY
- 1:VYUY
- 2:YUYV
- 3:YVYU

- Image Enlarged by Center Point

(1) i2ctransfer -f -y 1 w10@0x3C 0x9D 0x00 0x12 0x06 0x00 scale_step 0x00 0x00 0x00 0x00

(2) i2ctransfer -f -y 1 w10@0x3C 0x1D 0x08 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00

Among them, scale_step represents the magnification, the range is 0~10, which corresponds to the magnification x1.0~x2.0, there are 11 steps in total.

- Clearing Blind Pixel Data

i2ctransfer -f -y 1 w10@0x3C 0x1D 0x00 0x01 0x0F 0x00 0x00 0x00 0x00 0x00 0x00

- Clearing Low Gain Vignetting Effect Data

i2ctransfer -f -y 1 w10@0x3C 0x1D 0x00 0x01 0x90 0x00 0x00 0x00 0x00 0x00 0x00

- Clearing High Gain Vignetting Effect Data

i2ctransfer -f -y 1 w10@0x3C 0x1D 0x00 0x01 0x91 0x00 0x00 0x00 0x00 0x00 0x00

- Automatic Vignetting Effect Calibration

(1) i2ctransfer -f -y 1 w10@0x3C 0x9D 0x00 0x0D 0xD1 0x01 0x00 0x00 0x00 0x00 0x01

(2) i2ctransfer -f -y 1 w3@0x3C 0x1D 0x08 0x00

4. Examples of common commands

4.1. I2C Image Output Example

4.1.1. DVP Interface I2C Image Output

Examples of commands to control the image output of different resolutions and frame rates during DVP output are as follows:

(1) 640*512 30 frames

i2ctransfer -f -y 1 w10@0x3C 0x9d 0x00 0x0f 0xc1 0x00 0x00 0x00 0x00 0x00 0x08

i2ctransfer -f -y 1 w10@0x3C 0x1d 0x08 0x00 0x00 0x02 0x80 0x02 0x00 0x1E 0x00

(2) 384*288 30 frames

i2ctransfer -f -y 1 w10@0x3C 0x9d 0x00 0x0f 0xc1 0x00 0x00 0x00 0x00 0x00 0x08

i2ctransfer -f -y 1 w10@0x3C 0x1d 0x08 0x00 0x00 0x01 0x80 0x01 0x20 0x1E 0x00

(3) 256*192 25 frames

i2ctransfer -f -y 1 w10@0x3C 0x9d 0x00 0x0f 0xc1 0x00 0x00 0x00 0x00 0x00 0x08

i2ctransfer -f -y 1 w10@0x3C 0x1d 0x08 0x00 0x00 0x01 0x00 0x00 0xc0 0x19 0x00

(4) Test Image Output (256*192 25 frames as an example)

i2ctransfer -f -y 1 w10@0x3C 0x9d 0x00 0x0f 0xc1 0x00 0x00 0x00 0x00 0x00 0x08

i2ctransfer -f -y 1 w10@0x3C 0x1d 0x08 0x00 0x80 0x01 0x00 0x00 0xc0 0x19 0x00

4.1.2. SPI Interface I2C Image Output

Examples of commands to control different resolutions and frame rates for SPI image output are as follows:

(1) 640*512 30 frames

i2ctransfer -f -y 1 w10@0x3C 0x9d 0x00 0x0f 0xc1 0x00 0x00 0x00 0x00 0x00 0x08

i2ctransfer -f -y 1 w10@0x3C 0x1d 0x08 0x00 0x00 0x02 0x80 0x02 0x00 0x1E 0x08

(2) 384*288 30 frames:

i2ctransfer -f -y 1 w10@0x3C 0x9d 0x00 0x0f 0xc1 0x00 0x00 0x00 0x00 0x00 0x08

i2ctransfer -f -y 1 w10@0x3C 0x1d 0x08 0x00 0x00 0x01 0x80 0x01 0x20 0x1E 0x08

(3) 256*192 25 frames

i2ctransfer -f -y 1 w10@0x3C 0x9d 0x00 0x0f 0xc1 0x00 0x00 0x00 0x00 0x00 0x08

i2ctransfer -f -y 1 w10@0x3C 0x1d 0x08 0x00 0x00 0x01 0x00 0x00 0xc0 0x19 0x08

(4) Test Image Output (256*192 25 frames)

i2ctransfer -f -y 1 w10@0x3C 0x9d 0x00 0x0f 0xc1 0x00 0x00 0x00 0x00 0x00 0x08

i2ctransfer -f -y 1 w10@0x3C 0x1d 0x08 0x00 0x80 0x01 0x00 0x00 0xc0 0x19 0x08

After the command is issued, you can see the print information of the module through the serial port log, the approximate information:

```
fixp info:256,192:26,24,8,10:26,24
fixp info:256,192:26,24,8,10:26,24
dvp clk sel:24000000,256,192,25
dvp clk sel:24000000,256,192,25
dvp clk sel:24000000,256,192,25
voc1 clk sel:1
vospi_init: 1
voc1 transfer start
RTC2121 IR sensor Turn on ,25 fps
dwIrTypeValCalClk:22287325
GenCfg:
0,0,0,0,0,0,0,0,0,0,0,0,0,0,a2,fe,51,a0,3,f,f,b0,82,67,64,64,90,65,cd,f3,71,40,5f,31,a2,6d,a4,1,e,c,0,c0,83,13,83,bb,ca,88,73,71,ee,95,2,10,e1,7f,a2,3b,87,bb,c3,fb,ff,f,
Config Access : 2, 0, Addr b5000, Len d9c8
OOC load user cfg
OOC base set to 12000, thd 500
wFsDlyTime: 4567
wLs2LsTime: 4729
dwLs2FsTime: 23080
dwIrCtlrClk: 24000000
ir_gen_cfg_amend
wLsLsTime : 4729
GenCfg:
0,0,0,0,0,0,0,0,0,0,0,0,0,0,a2,fe,51,a0,3,f,f,b0,82,67,64,64,90,65,cd,f3,71,40,5f,31,a2,6d,a4,1,e,c,0,c0,83,13,83,bb,ca,88,73,71,ee,95,2,10,e1,7f,a2,3b,87,bb,c3,fb,ff,f,
_isp1_clk_set: 24000000
rmvcover enabled
RMVCOVER ENABLE CFG DONE
------ Load Gain:1 tpd table
TPD Prop: Gain 1, Distance 32, Ems 128, Tau 128, Ta 300, Tu 300.
Gamma cacalculate done
Auto Correct Image Prop : 1 500 9000 360 5
OOC temp threshold = 360
B temp threshold = 5
PseudColor1 Mode : 1, PseudColor2 Mode 1
byApAnLevelSel:0x2, byMpMnLevelSel:0x2
set dde level: 2
set tnrlevel: 2
set Snrlevel: 2
Gamma cacalculate done
AGC MODE: 2
Over exposure Prop : 0 10000 0x7d00 10
TPD Prop: Gain 1, Distance 32, Ems 128, Tau 128, Ta 300, Tu 300.
current ddr addr before assignment:d6000
current ddr addr before assignment:d6000
dvp clk sel:24000000,256,192,25
dvp clk sel:24000000,256,192,25
dvp clk sel:24000000,256,192,25
voc1 clk sel:1
```

4.2. I2C Stop Image Output

Commands to control the stop image output are as follows:

i2ctransfer -f -y 1 w10@0x3C 0x1d 0x00 0x0f 0x02 0x00 0x00 0x00 0x00 0x00 0x00

After the command is issued, you can see the print information of the module through the serial port log, the approximate information:

```
isp2 frame idle now
IR Turn off.
```