

# **ICM-42607x and ICM-42670x DieID and WaferID Read Procedures**

## Table of Contents

1	Overview.....	3
2	ID read procedure .....	4
3	ID decode.....	5
4	ID read example software .....	6
5	Revision history .....	7

## **1 OVERVIEW**

The chip ID information is stored in an OTP bank at the time of manufacturing. It consists of a 15-bit DieID, six 6-bit lot ID Characters, 5-bit WaferID, and 6-bit Die Rev.

The ID read procedure will be presented in this document.

## 2 ID READ PROCEDURE

1. Write 0x10 to register bank0 0x02 (**SIGNAL\_PATH\_RESET**) to do signal path reset.
2. Write 0x10 to register bank0 0x1F (**PWR\_MGMT0**) to set IDEL bit to 1.
3. Enable OTP power (**OTP\_PWR\_DOWN=0**):
  - Write 0x28 to register bank0 0x7C to set **BLK\_SEL\_R**.
  - Write 0x06 to register bank0 0x7D to set **MADDR\_R**. Register **TRIGGER\_ST\_COPY**.
  - Wait 10  $\mu$ s
  - Read register bank0 0x7E and save the value as otp\_pwr for after ID read restore.
  - Write 0x28 to register bank0 0x79 to set **BLK\_SEL\_W**.
  - Write 0x06 to register bank0 0x7A to set **MADDR\_W**. Register **TRIGGER\_ST\_COPY**
  - Wait 10  $\mu$ s
  - Write (otp\_pwr & 0xFD) to register bank0 0x7B to set **OTP\_PWR\_DOWN** bit to 0
4. Read out IDs
  - Write 0x23 to register bank0 0x7C to set **BLK\_SEL\_R**.
  - Write 0xEE to register bank0 0x7D to set **MADDR\_R**.
  - Wait 10  $\mu$ s
  - Repeat read register bank0 0x7E for 11 times. Save the 11 read out data for ID decode.
5. Disable OTP power (**OTP\_PWR\_DOWN=1**):
  - Write 0x28 to register bank0 0x79 to set **BLK\_SEL\_W**.
  - Write 0x06 to register bank0 0x7A to set **MADDR\_W**.
  - Wait 10  $\mu$ s
  - Write (otp\_pwr) to register bank0 0x7B to restore production trim value
6. Write 0x00 to register bank0 0x79 to clear **BLK\_SEL\_W**.

## 3 ID DECODE

The table below lists the ID bit definitions.

IDS	FUNCTION	BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
0	MANU_ID	manu_id[7]	manu_id[6]	manu_id[5]	manu_id[4]	manu_id[3]	manu_id[2]	manu_id[1]	manu_id[0]
1	CHIP_ID	chip_id[7]	chip_id[6]	chip_id[5]	chip_id[4]	chip_id[3]	chip_id[2]	chip_id[1]	chip_id[0]
2	REV_ID	rev_id[7]	rev_id[6]	rev_id[5]	rev_id[4]	rev_id[3]	rev_id[2]	rev_id[1]	rev_id[0]
3	DIE_ID_L	die_id[7]	die_id[6]	die_id[5]	die_id[4]	die_id[3]	die_id[2]	die_id[1]	die_id[0]
4	DIE_ID_H	wp_id[1]	die_id[14]	die_id[13]	die_id[12]	die_id[11]	die_id[10]	die_id[9]	die_id[8]
5	CHAR1	char2[1]	char2[0]	char1[5]	char1[4]	char1[3]	char1[2]	char1[1]	char1[0]
6	CHAR2	char3[3]	char3[2]	char3[1]	char3[0]	char2[5]	char2[4]	char2[3]	char2[2]
7	CHAR3	char4[5]	char4[4]	char4[3]	char4[2]	char4[1]	char4[0]	char3[5]	char3[4]
8	CHAR4	char6[1]	char6[0]	char5[5]	char5[4]	char5[3]	char5[2]	char5[1]	char5[0]
9	CHAR5	waferid[3]	waferid[2]	waferid[1]	waferid[0]	char6[5]	char6[4]	char6[3]	char6[2]
10	DIE_REV	die_rev[5]	die_rev[4]	die_rev[3]	die_rev[2]	die_rev[1]	die_rev[0]	wp_id[0]	waferid[4]

CHAR1~6 have chip lot information. The Char are translated as follow

Char value	Char
0	0
1	1
2	2
.	.
9	9
10	A
11	B
12	C
.	.
15	F
16	G
17	H
.	.
33	X
34	Y
35	Z

## 4 ID READ EXAMPLE SOFTWARE

The below is an example software code for user reference.

```
void ReadIDs_I2C()
{
    uint8_t value[20], i, CHAR[7];
    int dieid, wid, otp_pwr;

    value[0] = 0x10; i2c_master_write_register(Address, 0x02, 1, &value); //signal path reset
    delay_ms(100);
    value[0] = 0x10; i2c_master_write_register(Address, 0x1F, 1, &value); //IDLE=1

    // otp_pwr_down = 0
    value[0] = 0x28; i2c_master_write_register(Address, 0x7C, 1, &value); //BLK_SEL_R = 0x28
    value[0] = 0x06; i2c_master_write_register(Address, 0x7D, 1, &value); //MADDR_R = 0x06
    delay_us(10);
    i2c_master_read_register(Address, 0x7E, 1, &value); otp_pwr = value[0]; //save org value

    value[0] = 0x28; i2c_master_write_register(Address, 0x79, 1, &value); //BLK_SEL_W
    value[0] = 0x06; i2c_master_write_register(Address, 0x7A, 1, &value); //MADDR_W
    delay_us(10);
    value[0] = otp_pwr & 0xFD; i2c_master_write_register(Address, 0x7B, 1, &value); //otp_pwr_down = 0 (bit1)

    //read IDs
    value[0] = 0x23; i2c_master_write_register(Address, 0x7C, 1, &value); //BLK_SEL_R
    value[0] = 0xEE; i2c_master_write_register(Address, 0x7D, 1, &value); //MADDR_R
    delay_us(10);
    for(i=0; i<11; i++){
        i2c_master_read_register(Address, 0x7E, 1, &value[i]);
    }
    value[0] = 0x28; i2c_master_write_register(Address, 0x79, 1, &value); //BLK_SEL_W
    value[0] = 0x06; i2c_master_write_register(Address, 0x7A, 1, &value); //MADDR_W
    delay_us(10);
    value[0] = otp_pwr; i2c_master_write_register(Address, 0x7B, 1, &value); //set back the original value

    value[0] = 0x00; i2c_master_write_register(Address, 0x79, 1, &value); //BLK_SEL_W = 0

    dieid = ((value[4] & 0x7F)<<8) | value[3];
    wid = (value[9]>>4) | ((value[10] & 0x01)<<4);
    CHAR[1] = value[5] & 0x3F;
    CHAR[2] = ((value[5] & 0xC0)>>6) | ((value[6] & 0x0F)<<2);
    CHAR[3] = ((value[6] & 0xF0)>>4) | ((value[7] & 0x03)<<4);
    CHAR[4] = (value[7] & 0xFC)>>2;
    CHAR[5] = value[8] & 0x3F;
    CHAR[6] = ((value[8] & 0xC0)>>6) | ((value[9] & 0x0F)<<2);
    printf("DIEID= %d, WID= %d, CHAR= %d,%d,%d,%d,%d,%d\r\n", dieid, wid, CHAR[1], CHAR[2], CHAR[3],
    CHAR[4], CHAR[5], CHAR[6]);
}
}
```

## **5 REVISION HISTORY**

REVISION DATE	REVISION	DESCRIPTION
03/23/2021	1.0	Initial Release



This information furnished by InvenSense or its affiliates ("TDK InvenSense") is believed to be accurate and reliable. However, no responsibility is assumed by TDK InvenSense for its use, or for any infringements of patents or other rights of third parties that may result from its use. Specifications are subject to change without notice. TDK InvenSense reserves the right to make changes to this product, including its circuits and software, in order to improve its design and/or performance, without prior notice. TDK InvenSense makes no warranties, neither expressed nor implied, regarding the information and specifications contained in this document. TDK InvenSense assumes no responsibility for any claims or damages arising from information contained in this document, or from the use of products and services detailed therein. This includes, but is not limited to, claims or damages based on the infringement of patents, copyrights, mask work and/or other intellectual property rights.

Certain intellectual property owned by InvenSense and described in this document is patent protected. No license is granted by implication or otherwise under any patent or patent rights of InvenSense. This publication supersedes and replaces all information previously supplied. Trademarks that are registered trademarks are the property of their respective companies. TDK InvenSense sensors should not be used or sold in the development, storage, production or utilization of any conventional or mass-destructive weapons or for any other weapons or life threatening applications, as well as in any other life critical applications such as medical equipment, transportation, aerospace and nuclear instruments, undersea equipment, power plant equipment, disaster prevention and crime prevention equipment.

©2021 InvenSense. All rights reserved. InvenSense, MotionTracking, MotionProcessing, MotionProcessor, MotionFusion, MotionApps, DMP, AAR, and the InvenSense logo are trademarks of InvenSense, Inc. The TDK logo is a trademark of TDK Corporation. Other company and product names may be trademarks of the respective companies with which they are associated.



©2021 InvenSense. All rights reserved.