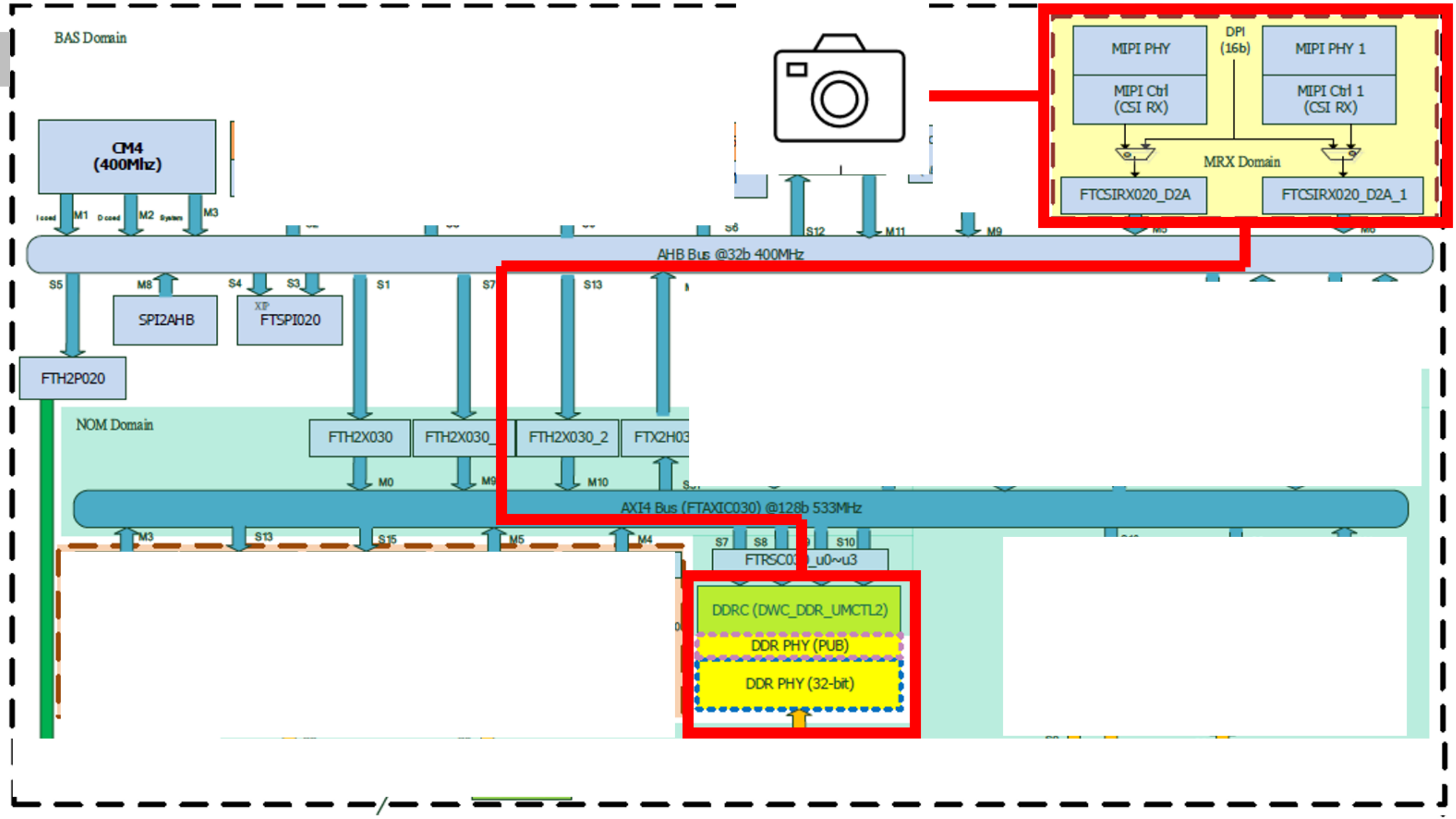
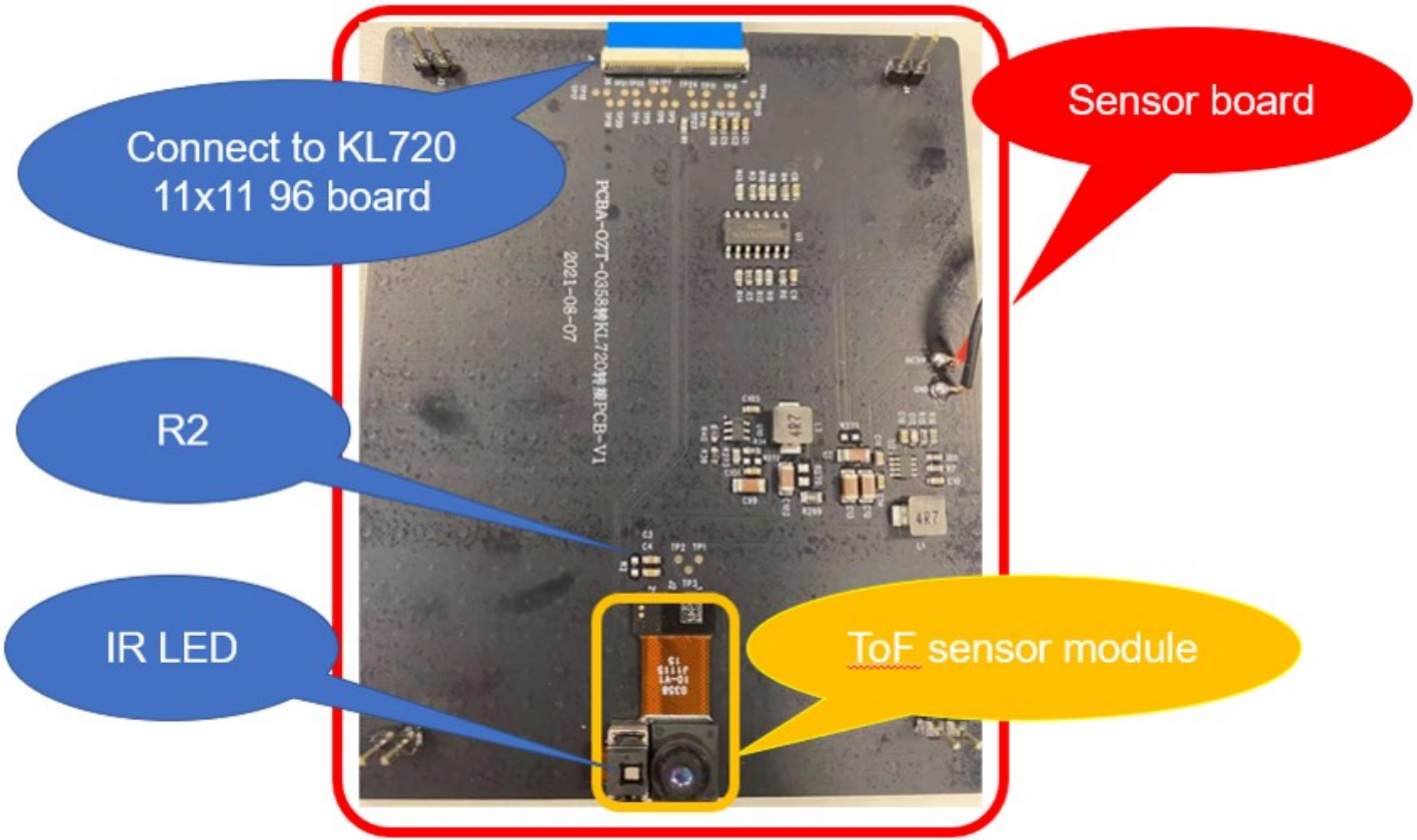
MIPI driver introduction on KL720 with ToF sensor

Outline

1. ToF document introduction.
2. HW schematic introduction
3. Sensor device init sequence
4. MIPI clk setting
5. CSIRX PHY and CSIRX Controller setting
6. D2A setting
7. Analysis tool – Octave
8. Summary

KL720


BAS Domain
CM4
(400Mhz)
I code
M1
D code
M2
System
M3
MIPI PHY
DPI
(16b)
MIPI PHY 1
MIPI Ctrl
(CSI RX)
MIPI Ctrl 1
(CSI RX)
MRX Domain
FTCSIRX020_D2A
FTCSIRX020_D2A_1
S6
S12
M11
M9
AHB Bus @32b 400MHz
S5
M8
S4
S3
S1
S7
S13
SPI2AHB
XIP
FTSPI020
FTH2P020
NOM Domain
FTH2X030
FTH2X030_
FTH2X030_2
FTX2H03
M0
M9
M10
AXI4 Bus (FTAXIC030) @128b 533MHz
M3
S13
S15
M5
M4
S7
S8
S10
FTRSC03
u0~u3
DDRC (DWC_DDR_UMCTL2)
DDR PHY (PUB)
DDR PHY (32-bit)


Connect to KL720
11x11 96 board
Sensor board
R2
IR LED
ToF sensor module

ToF document introduction.

IRS2877C Features.pdf

IRS2877C Features

• ToF-Pixel-Array

- Dual frequency : 640 x 2169 (640 x 241 x 9) raw pixel.
- Single frequency : 640 x 1205 (640 x 241 x 5) raw pixel.
- MIPI compliant CSI-2 interface: 2-data lanes, maximum 1Gbit/s..
- Depth resolution:640 x 240 x 2byte (uint16 Depth16 format).
- IR resolution:640 x 240(uint8).

E. Sensor Rawdata 无法正常撷取数据.

1. 请先确保 Sensor init 成功.
2. 确认 MIPI Data type:RAW12.
3. Data lane:2-lanes.
4. 确认 MIPI CLK 速度~~(384Mbit/s).~~

500Mhz

ToF document introduction.

IRS2877C Features.pdf

F. OMS 自研开发 Raw data 无法正常译码:

1. Sensor Raw 12-bit data 收集大小不正确, Dual frequency:2082240 bytes, Single Frequency:1156800 bytes.

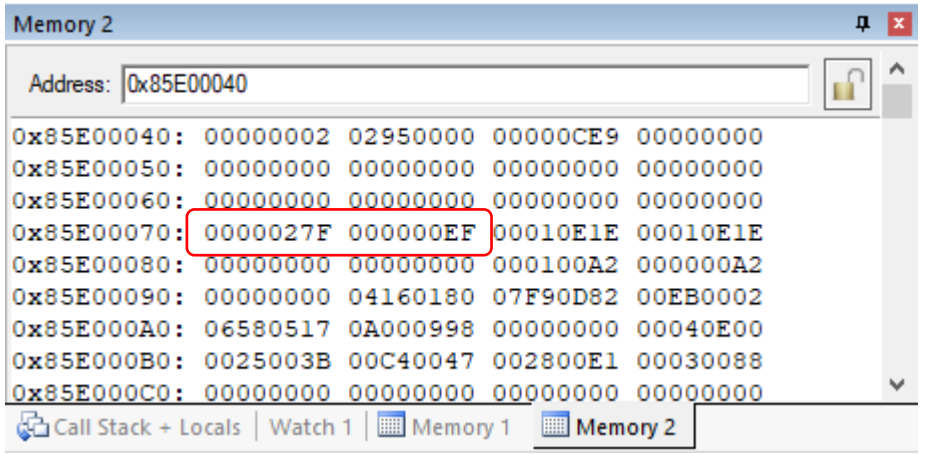
2. 若使用的 SoC 平台 ISP 是接收 16-bit 格式,Dual frequency:2776320 bytes, Single Frequency:1542400 bytes.

3. 因应各 SoC 平台可能需透过 ISP 收集 MIPI rawdata,若有被 ISP 排序过 Raw data 确认格式正确,可看收集的目前是观测抓取下来的 37~45 byte 数据分别代表 25~30 pixel 的资料其中 pixel 25 / 27 为 ROI 值 分别是 639 和 239.

   0x27F 0xEF

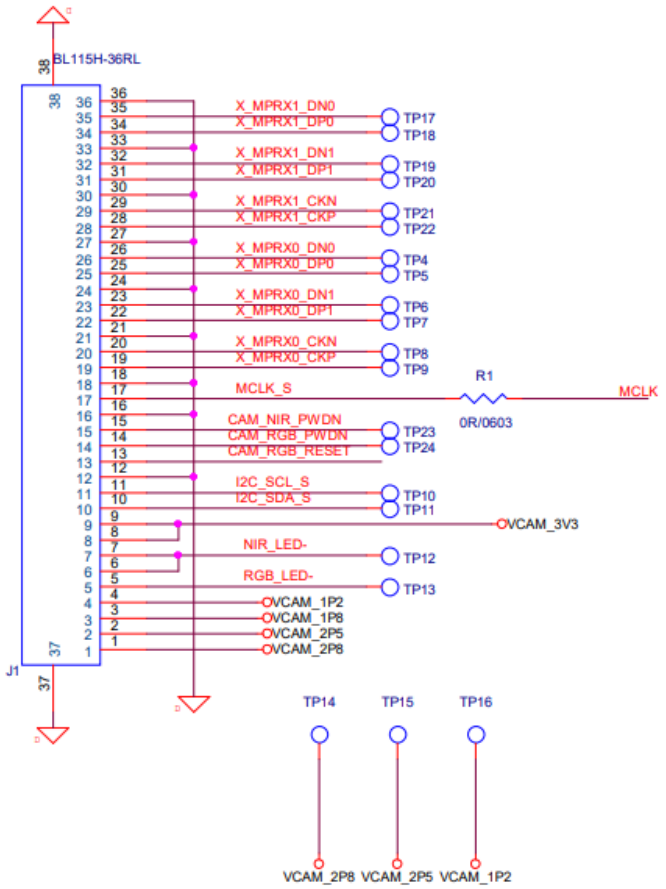
   (如下图 12-bit mode),前提是收集 bytes 要正确.

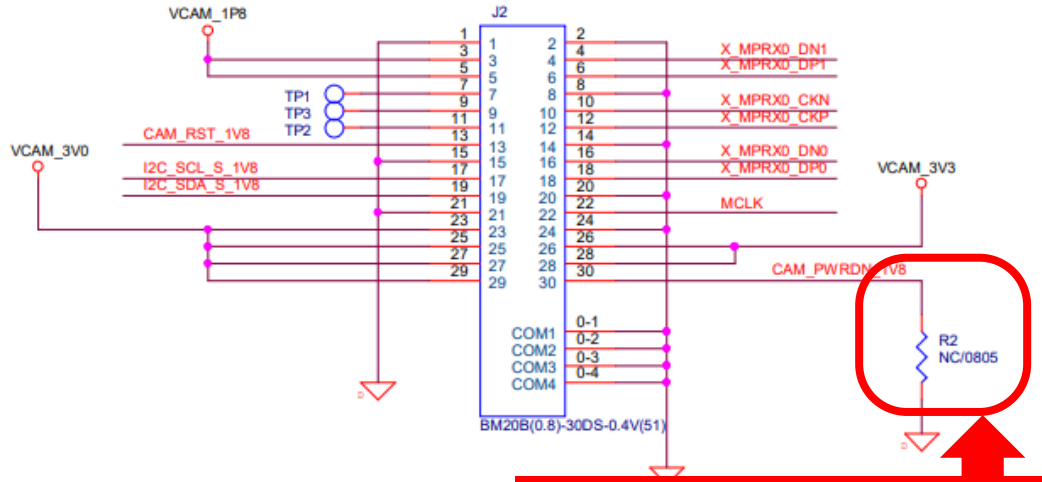
00	00	02	00	AF	30	28	00	0B	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00	00	00	00	27	00	0F	0E	00	0F	E1	00	1E	E1	00	1E
00	00	00	00	00	00	0A	00	10	0A	00	00	00	01	00	00
08	60	D6	7F	92	05	12	C0	52	5B	F6	8F	80	04	00	80
00	80	0A	00	0F	09	BD	0D	01	6C	06	03	E9	02	00	30



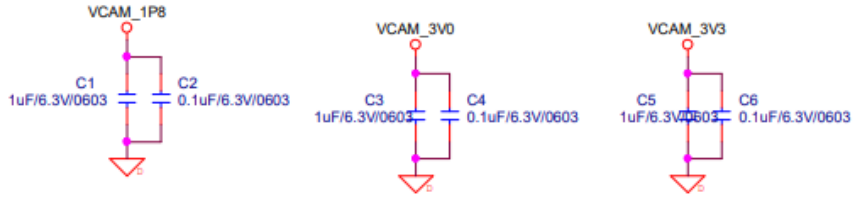
HW schematic introduction

OZT-0358转KL720-DSN-20210809.pdf





R2 need to short for single mode

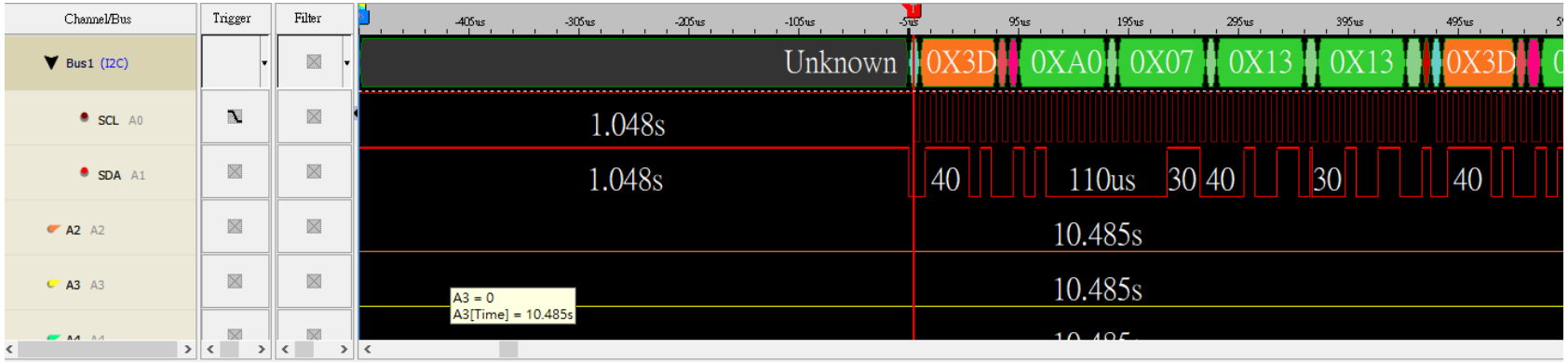


Recommend to inject 1p8 and 3p3 from power supply

Sensor device init sequence

0378_80_30FPS_990.txt -> kdev_sensor_irs2877c.c

```
struct sensor_init_seq irs2877c_init_regs[] = {
    //Original init sequence
  {0xA007, 0x1313},
  {0xA008, 0x1513},
  {0xA001, 0x0003},
  {0xA039, 0x16A1},
  {0xA03A, 0x5555},
  {0xA03B, 0x0005},
  {0xA03C, 0x0000},
  {0xA03D, 0x04D0},
  {0xA03E, 0x0000},
  {0xA03F, 0x000F},
  {0xA058, 0x0A08},
  {0xA05B, 0x7422},
  {0x9000, 0x1E1E},
  {0x9002, 0x66D3},
  {0x9004, 0x66D3},
  {0x9006, 0x66D3},
  {0x9008, 0x66D3},
  {0x900A, 0x5D1E},
  {0x900C, 0x5D1E},
  {0x900E, 0x5D1E},
  {0x9010, 0x5D1E},
  {0x9080, 0x1E1E},
  {0x9082, 0x10A2},
  {0x9083, 0x00A2},
  {0x9084, 0x0000},
  {0x9085, 0x66D3},
  {0x9087, 0x0000},
  {0x9088, 0x0000},
  {0x9089, 0x0000},
  {0x908A, 0x66D3},
  {0x908C, 0x0000},
  {0x908D, 0x0000},
```



Settings ... | Refresh | Export... | Synch. Navigation ...

Packet #	Name	TimeStamp	Start	Address	Write	A-ACK	Data	D-ACK	Data	D-ACK	Data	D-ACK	Data	D-ACK	Stop
1	Bus1(I2C)	-5us	Start	3D	Write	A-ACK	A0	D-ACK	07	D-ACK	13	D-ACK	13	D-ACK	Stop
2	Bus1(I2C)	470us	Start	3D	Write	A-ACK	A0	D-ACK	08	D-ACK	15	D-ACK	13	D-ACK	Stop
3	Bus1(I2C)	942.5us	Start	3D	Write	A-ACK	A0	D-ACK	01	D-ACK	00	D-ACK	03	D-ACK	Stop
4	Bus1(I2C)	1.4175ms	Start	3D	Write	A-ACK	A0	D-ACK	39	D-ACK	16	D-ACK	A1	D-ACK	Stop
5	Bus1(I2C)	1.8925ms	Start	3D	Write	A-ACK	A0	D-ACK	3A	D-ACK	55	D-ACK	55	D-ACK	Stop
6	Bus1(I2C)	2.3675ms	Start	3D	Write	A-ACK	A0	D-ACK	3B	D-ACK	00	D-ACK	05	D-ACK	Stop
7	Bus1(I2C)	2.84ms	Start	3D	Write	A-ACK	A0	D-ACK	3C	D-ACK	00	D-ACK	00	D-ACK	Stop

Sensor device init sequence

0378_80_30FPS_990.txt

Settings ... | Refresh | Export... | Synch. Navigation ...

Packet #	Name	TimeStamp	Start	Address	Write	A-ACK	Data	D-ACK	Data	D-ACK	Data	D-ACK	Data	D-ACK	Stop
204	Bus1(I2C)	96.2925ms	Start	3D	Write	A-ACK	92	D-ACK	FD	D-ACK	7E	D-ACK	5C	D-ACK	Stop
205	Bus1(I2C)	96.765ms	Start	3D	Write	A-ACK	94	D-ACK	01	D-ACK	00	D-ACK	02	D-ACK	Stop
206	Bus1(I2C)	97.24ms	Start	3D	Write	A-ACK	98	D-ACK	0A	D-ACK	FE	D-ACK	FF	D-ACK	Stop
207	Bus1(I2C)	97.715ms	Start	3D	Write	A-ACK	98	D-ACK	0C	D-ACK	3F	D-ACK	00	D-ACK	Stop
208	Bus1(I2C)	98.19ms	Start	3D	Write	A-ACK	98	D-ACK	0D	D-ACK	3F	D-ACK	00	D-ACK	Stop
209	Bus1(I2C)	98.6625ms	Start	3D	Write	A-ACK	A0	D-ACK	A4	D-ACK					

Packet #	Name	TimeStamp	Start	Address	Read	A-ACK	Data	D-ACK	Data	D-NACK	Stop	Description
210	Bus1(I2C)	98.9525ms	Start	3D	Read	A-ACK	28	D-ACK	77	D-NACK	Stop	Master NACK

Packet #	Name	TimeStamp	Start	Address	Write	A-ACK	Data	D-ACK	Data	D-ACK	Data	D-ACK	Data	D-ACK	Stop
211	Bus1(I2C)	144.0325ms	Start	3D	Write	A-ACK	94	D-ACK	00	D-ACK	00	D-ACK	01	D-ACK	Stop

Sensor ChipID

ChipID Register Address: **0xA0A4.**

CHIP ID: **0x2877.**

When developing the sensor driver, you can check the chip id.

D. ToF Sensor Start/Stop configurations :

Operation	Configurations	Comments
Camera START	0x9400 ← 0x0001	0x9400 is the command register address
Camera STOP	0x9400 ← 0x0000	0x9400 is the command register address

Kneron

MIPI clk setting

MIPI_PLL.xlsx

		ToF sensor	
	Sensor	MIPI0	MIPI1
	MIPI Lanes	2	
	bpp	12	
	MIPI data rates	1000	
	$F_{RxBwcClkHS}$	125	
	F_{txtin}	166.6666667	
PLL target	Pixel clk	166.6666667	0
	csi_clk$_{min}$	62.5	0
	ES clk	16	16
	Sensor	MIPI0	MIPI1
Result (MHz)	Pixel clk	165	720
	csi_clk$_{min}$	73.33333333	720
	ES clk	20.625	720

PLL3	FREF	MS	NS	PS	pll3_out	before deividex
	12	1	220	1	2640	1320
		1	DC	1		
			divider	neg		
PLL3	csirx0_EscClk	64	63	3F		
RGB	csirx0_vc0	8	7	7		
(0X5C)	csirx0_csi	18	17	11		

Table 4-50. Clock Divider Register 0 (Offset: 0x0058)

Bit	Symbol	Access	Default	Description
21:16	pll2_esc1_div_factor	RW	0x23	csirx1_EscClk divider.
12:8	pll2_csi1_vc_div_factor	RW	0x1	csirx1_vc_pclk divider.
4:0	pll2_csi1_div_factor	RW	0x3	csirx1_csi_clk divider.

Table 4-51. Clock Divider Register 1 (Offset: 0x005C)

Bit	Symbol	Access	Default	Description
21:16	pll3_esc0_div_factor	RW	0x23	csirx0_EscClk divider.
12:8	pll3_csi0_vc_div_factor	RW	0x1	csirx0_vc_pclk divider.
4:0	pll3_csi0_div_factor	RW	0x3	csirx0_csi_clk divider.

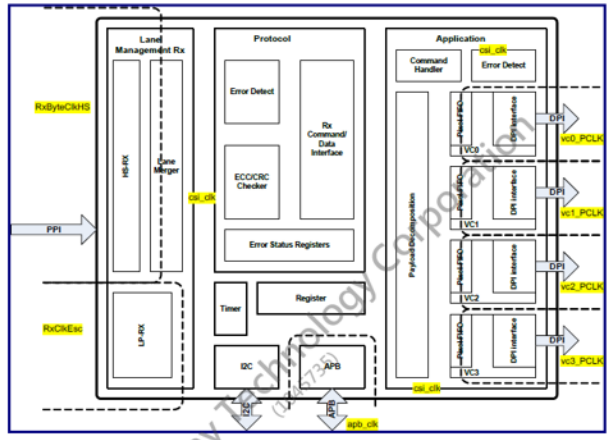
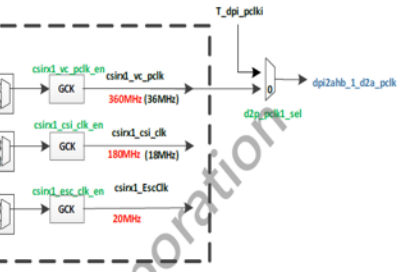


Figure 1-1. Functional Block Diagram



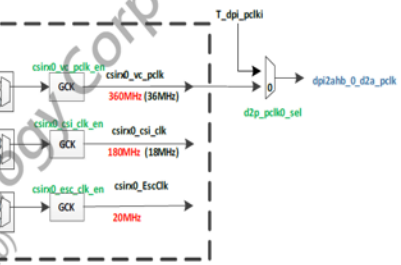


Figure 2-3. Clock Distribution Diagram – clk_gen_MRX

MIPI clk setting

project.h, kdrv_pll.h and kdrv_clock.c

```
/*===========================================================================
clock setting
===========================================================================*/
#define SCPU_MHZ                                SCPU_400
#define AXI_DDR_MHZ                             AXI_DDR_533
#define MRX1_MHZ                                MRX1_720
#define MRX0_MHZ                                MRX0_1320
#define NPU_MHZ                                 NPU_500
#define DSP_MHZ                                 DSP_500
#define AUDIO_MHZ                               ADO_12p288

//PLL3  1440MHZ
#define PLL3_1440_MS            1
#define PLL3_1440_NS            120     //0x78
#define PLL3_1440_PS            0
#define PLL3_1440_IS            0
#define PLL3_1440_MS_MASK       ((PLL3_1440_MS&0x07 )<<16)
#define PLL3_1440_NS_MASK       ((PLL3_1440_NS&0x1FF)<<20)
#define PLL3_1440_PS_MASK       ((PLL3_1440_PS&0x0F )<<12)
#define PLL3_1440_IS_MASK       ((PLL3_1440_IS&0x03 )<< 8)
#define PLL3_TIMER              0x4C0

//PLL3  2640MHZ
#define PLL3_2640_MS            1
#define PLL3_2640_NS            220//220     //0xdc
#define PLL3_2640_PS            0
#define PLL3_2640_IS            0
#define PLL3_2640_MS_MASK       ((PLL3_2640_MS&0x07 )<<16)
#define PLL3_2640_NS_MASK       ((PLL3_2640_NS&0x1FF)<<20)
#define PLL3_2640_PS_MASK       ((PLL3_2640_PS&0x0F )<<12)
#define PLL3_2640_IS_MASK       ((PLL3_2640_IS&0x03 )<< 8)
#define PLL3_TIMER              0x4C0

//CSIRX0_DIV  //EscClk=36(20MHz), csirx_vc_pclk=12(60MHz), csirx_clk=24(30MHz)
#define PLL3_CSI0_TXESC         0x23
#define PLL3_CSI0_CSI           0x17
#define PLL3_CSI0_VC0           0x0B
#define PLL3_DIV_MASK           ((PLL3_CSI0_CSI)|(PLL3_CSI0_VC0<<8)|(PLL3_CSI0_TXESC<<16))

//CSIRX0_DIV //For ToF IRS2877c sensor
#define PLL3_CSI0_TXESC_2       0x3F
#define PLL3_CSI0_CSI_2         0x11
#define PLL3_CSI0_VC0_2         0x07
#define PLL3_DIV_MASK_2         ((PLL3_CSI0_CSI_2)|(PLL3_CSI0_VC0_2<<8)|(PLL3_CSI0_TXESC_2<<16))
```

```
const T_PLLnConfig mrx0_clk_tbl[MRX0_CLK_TOTAL_SUPPORTED] =
{
    //720MHz CFG1
    {
        (720000000),
        (PLL3_1440_MS_MASK|PLL3_1440_NS_MASK|PLL3_1440_PS_MASK|PLL3_1440_IS_MASK|PLLn_DISABLE_MASK),
        (PLL3_TIMER),
        (NULL),
        (PLL3_DIV_MASK)
    },

    //1320MHz CFG1
    {
        (1320000000),
        (PLL3_2640_MS_MASK|PLL3_2640_NS_MASK|PLL3_2640_PS_MASK|PLL3_2640_IS_MASK|PLLn_DISABLE_MASK),
        (PLL3_TIMER),
        (NULL),
        (PLL3_DIV_MASK_2)
    }
}
```

Kneron

CSIRX PHY and CSIRX Controller setting

board.h and project.h

- board.h

```
//CSIRX
#define CSIRX_VSTU_LINE                 0
#define CSIRX_VSTU_PIXEL                1

#define CSIRX_MCR_LSB                   0
#define CSIRX_MCR_MSB                   1


#define CSIRX_VSTR_IRS2877C             0x50    //0x1
#define CSIRX_VSTER_IRS2877C            0x0     //0x8
#define CSIRX_HSTR_IRS2877C             0x8     //0x8
#define CSIRX_PFTR_IRS2877C             0x15    //0x30
#define CSIRX_SETTLE_CNT_IRS2877C       0xA     //0xB//0x7
```

- project.h

```
#define CAM_ID_MAX                      1

#define IMGSRC_NUM                      CAM_ID_MAX
#define MIPI_CAM_RGB                    0
#define MIPI_CAM_NIR                    1
#define MIPI_LANE_RGB                   2
#define MIPI_LANE_NIR                   2

#define SENSOR_RES_RGB                  SENSOR_RES_640_480
#define SENSOR_RES_NIR                  SENSOR_RES_480_640
#define IMGSRC_FORMAT_RGB               IMG_FORMAT_RGB565
#define IMGSRC_FORMAT_NIR               IMG_FORMAT_RAW8

#define CSIRX_0_TCN                     50
#define CSIRX_0_HRTV                    100
#define CSIRX_0_VSTU                    CSIRX_VSTU_PIXEL
#define CSIRX_0_MCR                     CSIRX_MCR_LSB
#define CSIRX_0_VSTR                    CSIRX_VSTR_IRS2877C
#define CSIRX_0_VSTER                   CSIRX_VSTER_IRS2877C
#define CSIRX_0_HSTR                    CSIRX_HSTR_IRS2877C
#define CSIRX_0_PFTR                    CSIRX_PFTR_IRS2877C
#define CSIRX_0_SETTLE_CNT              CSIRX_SETTLE_CNT_IRS2877C
```

dpi2ahb setting

board.h and project.h

- board.h

```
/* D2A Packet type register */
#define D2A_PT_YUV422                0x1E
#define D2A_PT_RGB565                0x22
#define D2A_PT_RGB888                0x24
#define D2A_PT_RAW8                  0x2A
#define D2A_PT_RAW10                 0x2B
#define D2A_PT_RAW12                 0x2C
#define D2A_PT_RAW14                 0x2D
#define D2A_PT_RAW16                 0x2E

#define D2A_SRC_CSIRX                0
#define D2A_SRC_EXT_DPI              1

#define D2A_DA_LSB                   0
#define D2A_DA_MSB                   1

#define D2A_POLARITY_L               0
#define D2A_POLARITY_H               1

#define D2A_FT_IRS2877C              0x180
#define D2A_FT_GC2145                0x180
#define D2A_FT_SC132GS               0x180
#define D2A_FT_EXTERN                0x380

#define D2A_TILE_AVG_SIZE128         0
#define D2A_TILE_AVG_SIZE64          1
#define D2A_TILE_AVG_SIZE32          2

#define D2A_VSYNC_PL_IRS2877C        D2A_POLARITY_L
#define D2A_VSYNC_PL_GC2145          D2A_POLARITY_L
#define D2A_VSYNC_PL_SC132GS         D2A_POLARITY_L
#define D2A_VSYNC_PL_EXT             D2A_POLARITY_H
```

- project.h

```
#define D2A_0_INPUT_SRC              D2A_SRC_CSIRX
#define D2A_0_FIFO_THRE              D2A_FT_IRS2877C
#define D2A_0_DROP_FRAME_NUM         0
#define D2A_0_PACKET_TYPE            D2A_PT_RAW12
#define D2A_0_DATA_ALIGN             D2A_DA_LSB
#define D2A_0_VSYNC_PL               D2A_VSYNC_PL_IRS2877C
#define D2A_0_HSYNC_PL               D2A_HSYNC_PL_IRS2877C
#define D2A_0_TILE_AVG_EN            IMGSRC_0_TILE_AVG
#define D2A_0_TILE_AVG_SIZE          D2A_TILE_AVG_SIZE128
```

Analysis tool - Octave

Octave_show_img.m

- https://www.gnu.org/software/octave/download

Microsoft Windows

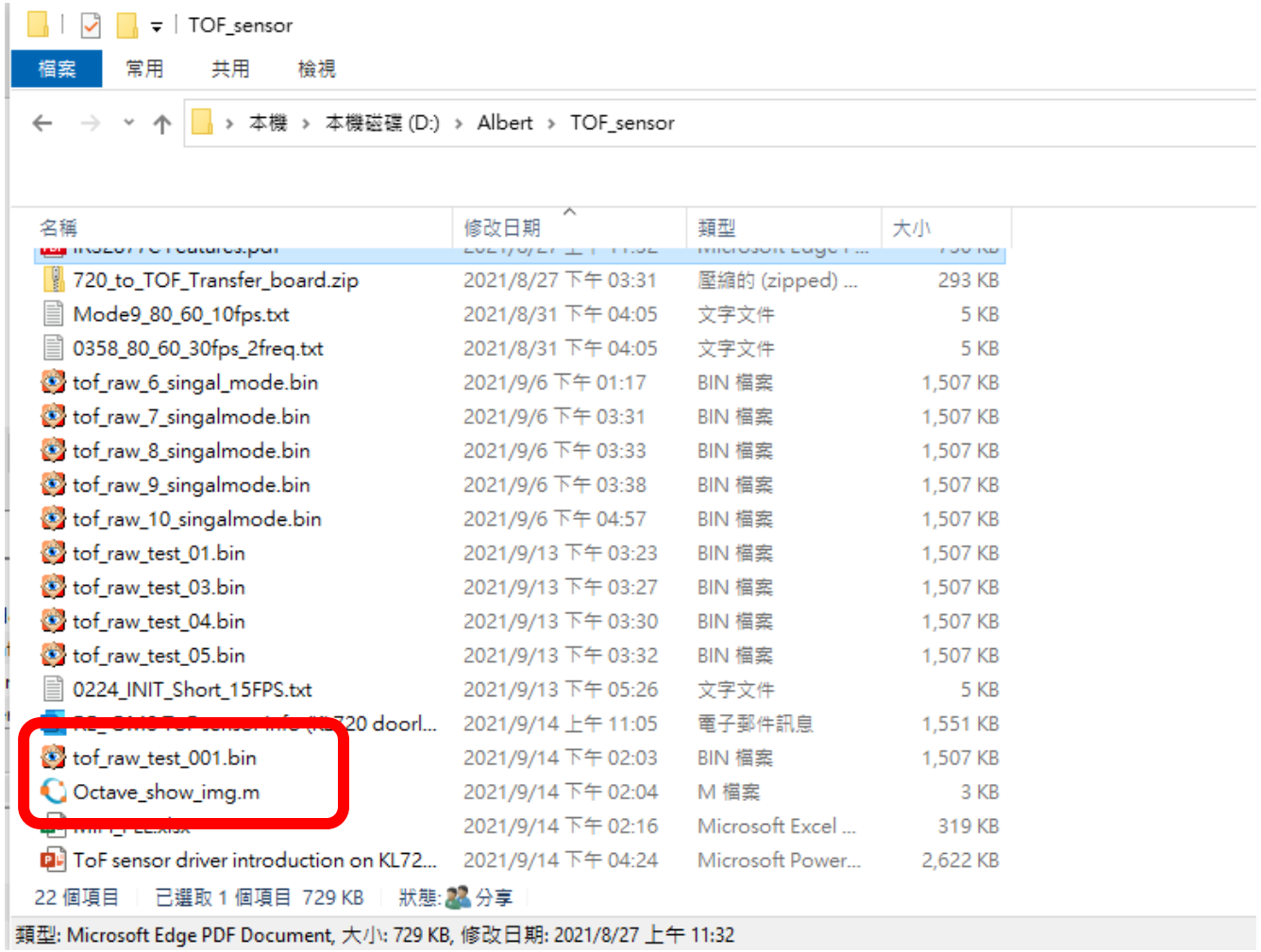
Note: All installers below bundle several **Octave packages** so they don't have to be installed separately. After installation type `pkg list` to list them. Read more.

- Windows-64 (recommended)
  - octave-6.3.0-w64-installer.exe (~ 325 MB) [signature]
  - octave-6.3.0-w64.7z (~ 319 MB) [signature]
  - octave-6.3.0-w64.zip (~ 568 MB) [signature]
- Windows-32 (old computers)

Kneron

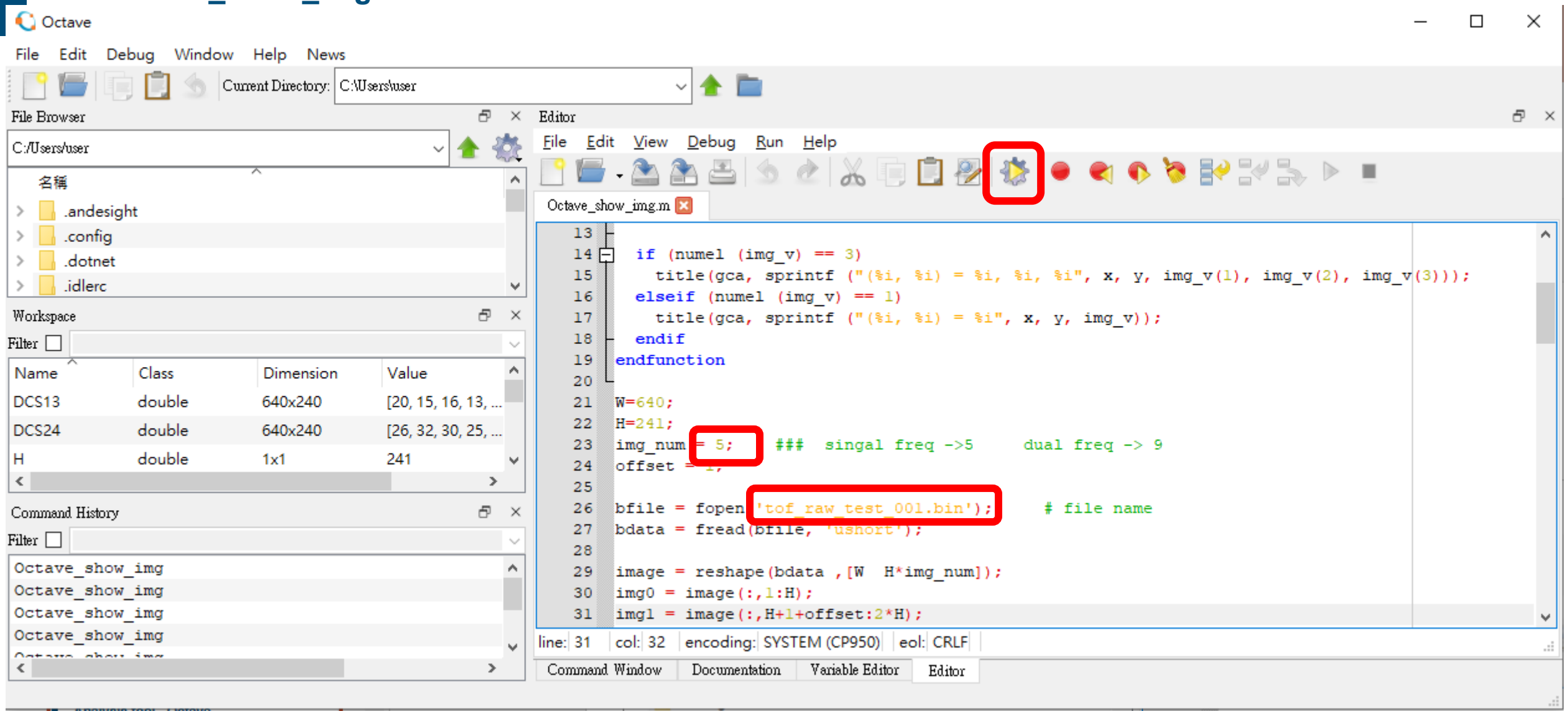
Analysis tool - Octave

Octave_show_img.m



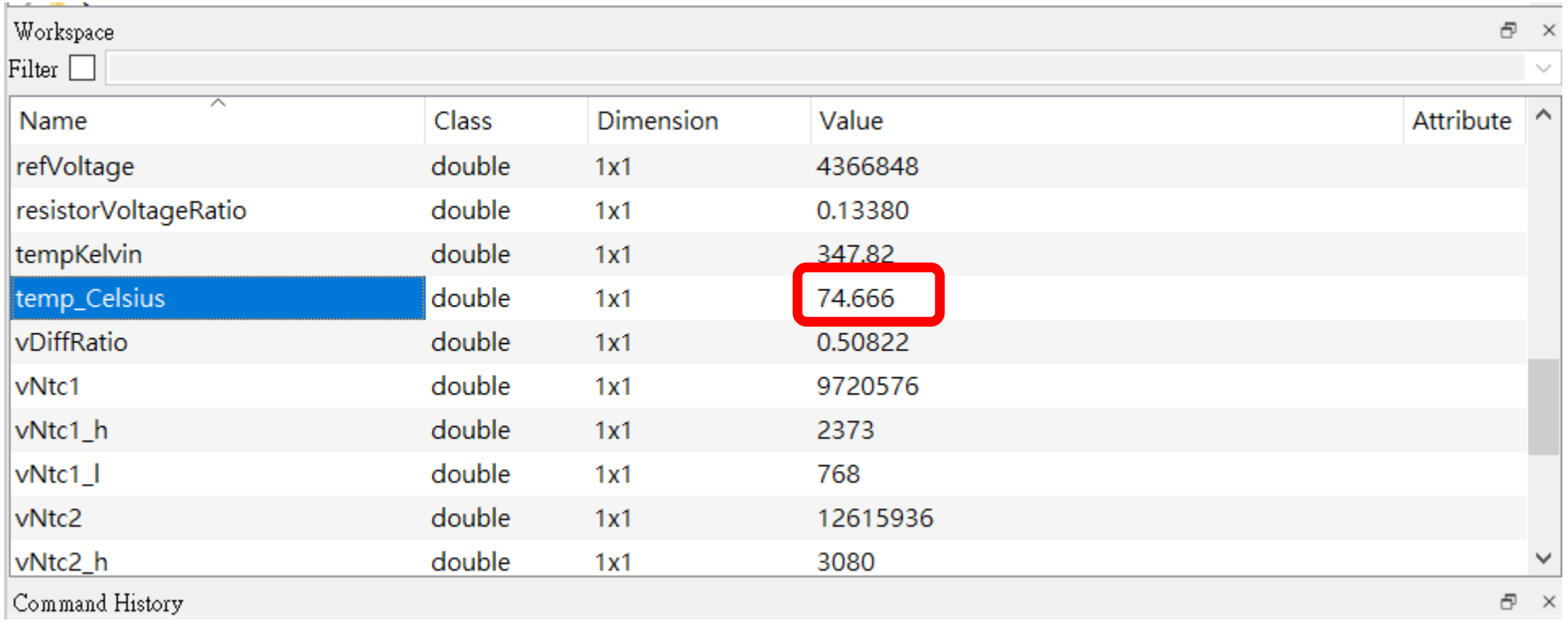
Analysis tool - Octave

Octave_show_img.m



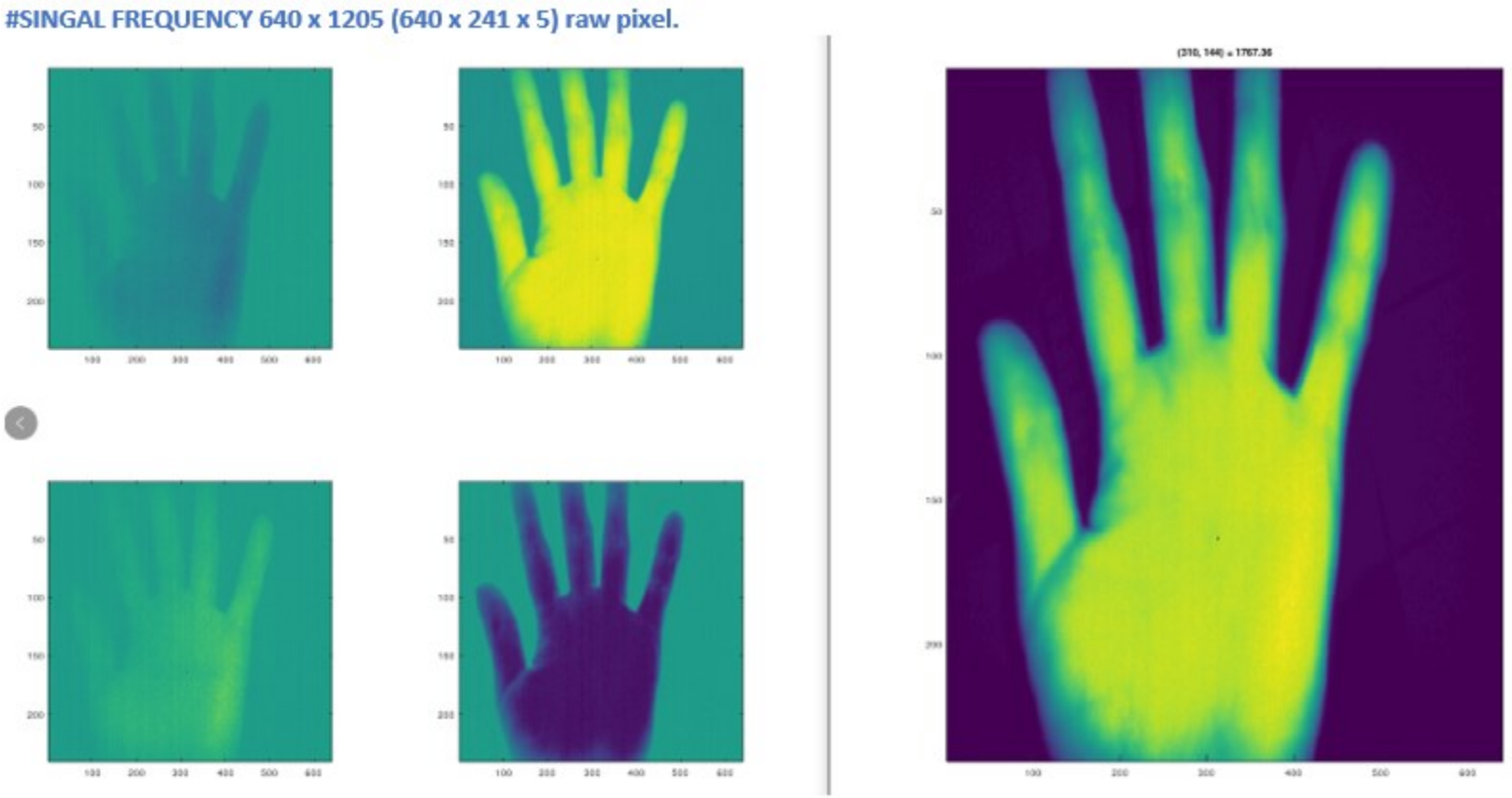
Analysis tool - Octave

Octave_show_img.m



Analysis tool - Octave

Octave_show_img.m



Analysis tool - Octave

Octave_show_img.m