APPLICATION NOTE

Using a numeric camera OV7670

RENESAS

POLYTECH®
CLERMONT-FERRAND

Geoffrey Raynal

Summary

Figure Table

Table Table

Introduction

This Application Note shows how to use a numeric camera OV7670 on a RX64M microcontroller. Explanation and diagrams are specific to the hardware and come from manufacturers. Results and numbers which are given here could not be use with your hardware, in this case refer on your Hardware manual.

Numeric cameras have an important place in electronics today, image processing algorithm become more and more efficiently. Many devices use video to work, as drones, car, robot. So it will be necessary to integrate video processing. The aim of this last year study project was to realize a MJPEG encoder on a RX64M, and coding a video coming from an OV7670 camera device. Method used to get pictures on the chip is describe in this AN.

Establishing

The device used was an OV7670 made by Omni vision.

Frequently, the device was sold welded on a board, and contains only 18 pins instead of 24.

The device should be connected on the J6 port on the RX64M's starter kit board. All pins of the Camera are explained below Figure 1

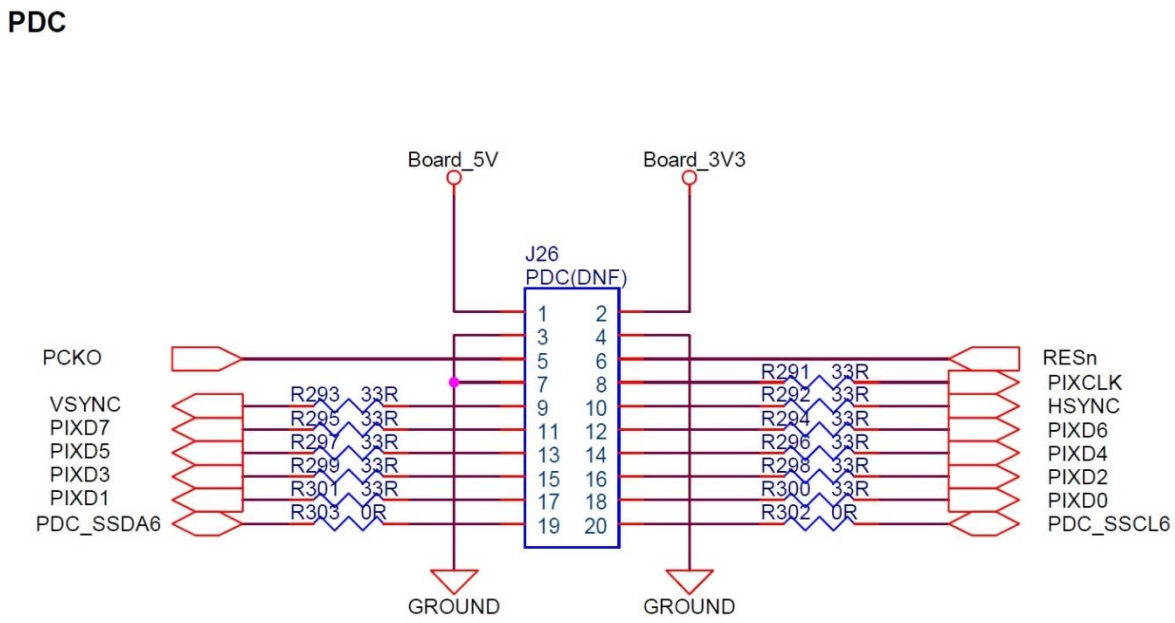


Figure 1 J6 port on RX64M Board

Tableau 1 pin description

N° broche	Nom	I/O	Description	Connexion port J6
1	VDD	I		Pin 2 : Board_3V3
2	GND	I		Pin 3, 4, 7 : GROUND

3	SIOC	I	I2C Clock	Pin 20 : PDC_SSCL6
4	SIOD	I	I2C Data	Pin 19 : PDC_SSDA6
5	VSYNC	O	Vetical syncrhonization signal	Pin 9 : VSYNC
6	HREF	O	Horizontal synchronization signal	Pin 10 : HSYNC
7	PCLK	O	Pixel clock from the camera	Pin 5 : PCKO
8	XCLK	O	Pixel clock from the microcontroler	Pin 8 : PIXCLK
9	D7	O		Pin 11 : PIXD7
10	D6	O		Pin 12 : PIXD6
11	D5	O		Pin 13 : PIXD5
12	D4	O		Pin 14 : PIXD4
13	D3	O		Pin 15 : PIXD3
14	D2	O		Pin 16 : PIXD2
15	D1	O		Pin 17 : PIXD1
16	D0	O		Pin 18 : PIXD0
17	Reset	I	Reset pin	Pin 6 : RESn
18	Pwdn	I	Power down pin	NC*

Communications

Generally cameras have two type of communications. A slow one, as RS232 or like here, I2C at 400 kHz, this communication serves to setup the device to send pictures in a particular format, GVB, VGA, and at the whished speed. The second communication, faster (from 15 to 30 MHz) is an 8 bit parallel communication which allow to keep image pixels.

The setup communication of the OV7670 is a SCCB liaison, an Omni vision's communication, this communication work with simple I2C. The difference whit the I2C standard is that there is no acknowledgment after a byte reception. So do not take into account this acknowledge. Chronogram of this communication is given in figure 2.

Figure 4 SCCB Timing Diagram

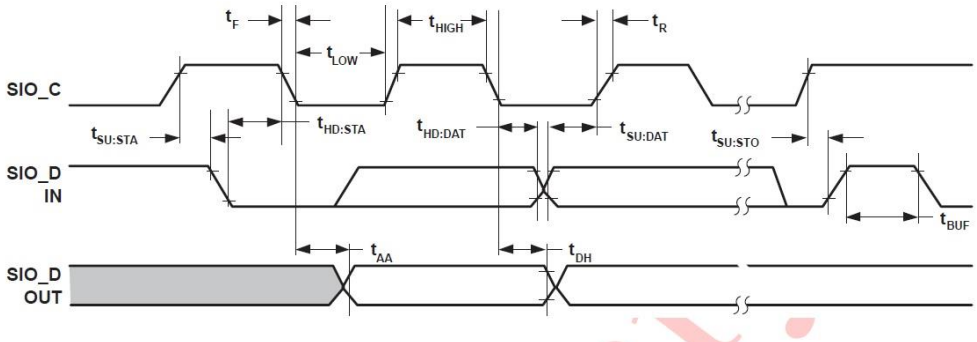


Figure 2 timing reference for SCCB liaison

Communication are made in 3 steps figure 3

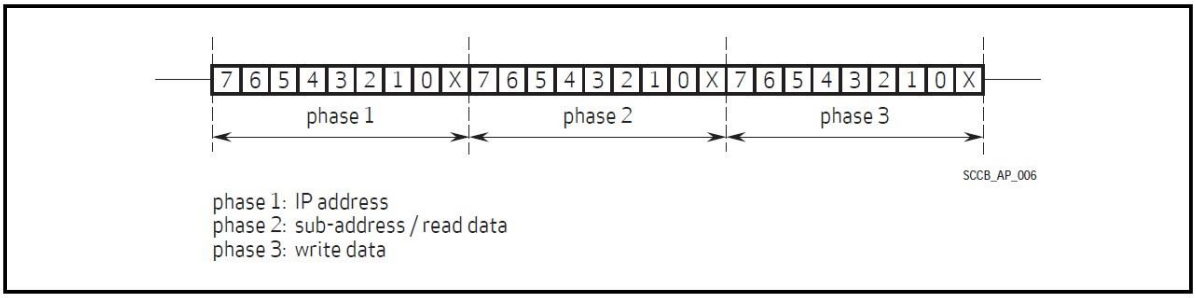


Figure 3 3 phases communication

Writing a register use the three phases, as shown in the figure 4

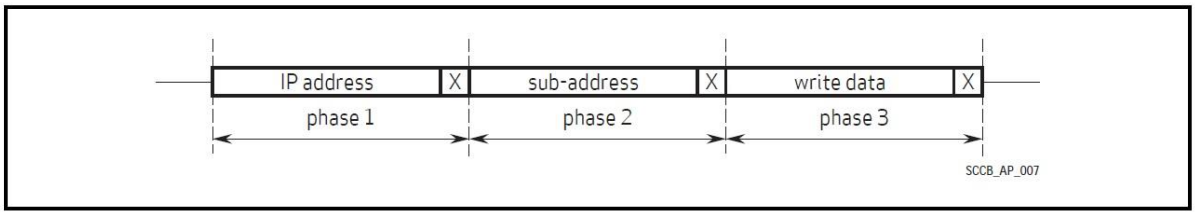


Figure 4 3 phases writing communication

Reading is made in a 2 phases as shown in figure 6, after make a 2 or 3 phases writing. Figure 4-5

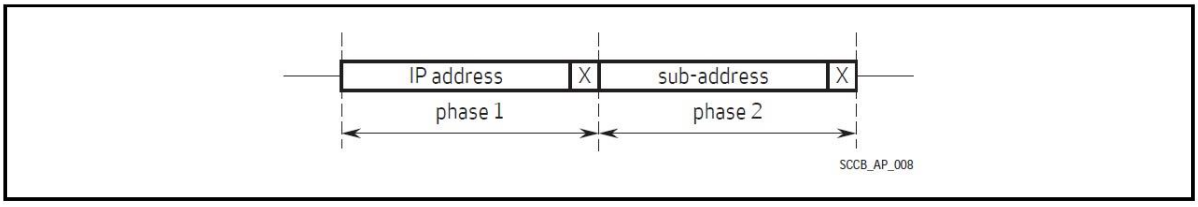


Figure 5 2 phases writing communication

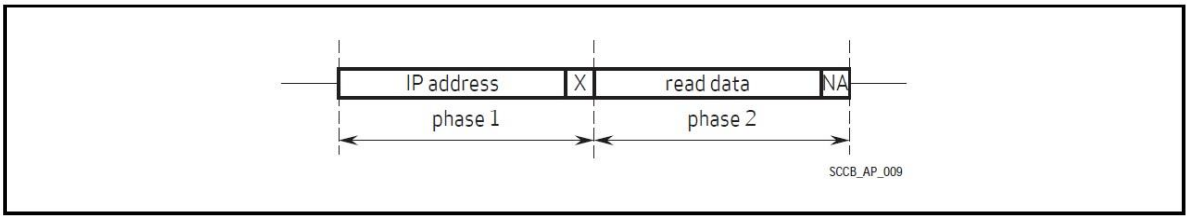
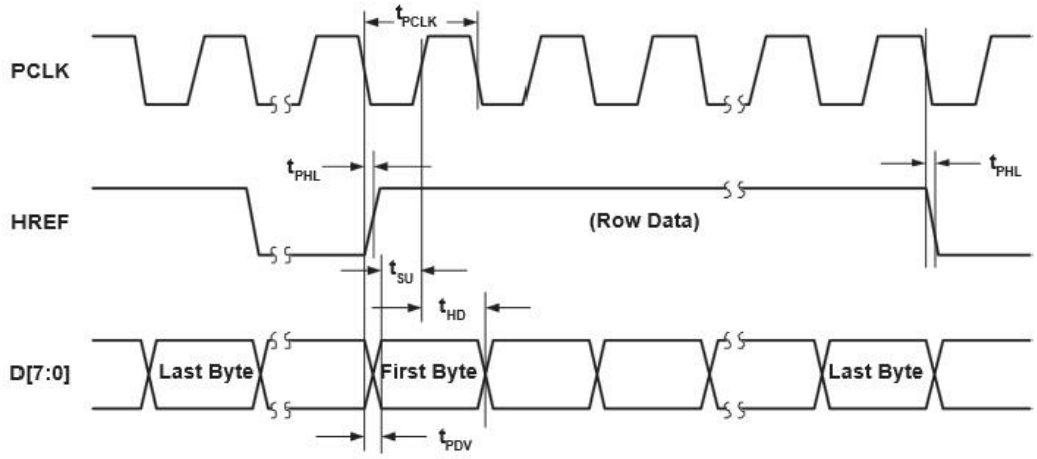


Figure 6 2 phases read communication

The default I2C address of the device is in hexadecimal 0x42

The communication which allow to keep the picture, is a 8 bit parallel communication which is synchronized with 3 signals : VSYNC, HREF, PICLK Figure 7. The state machine to implement the algorithm is given in annex 1.

Horizontal Timing



VGA Frame Timing

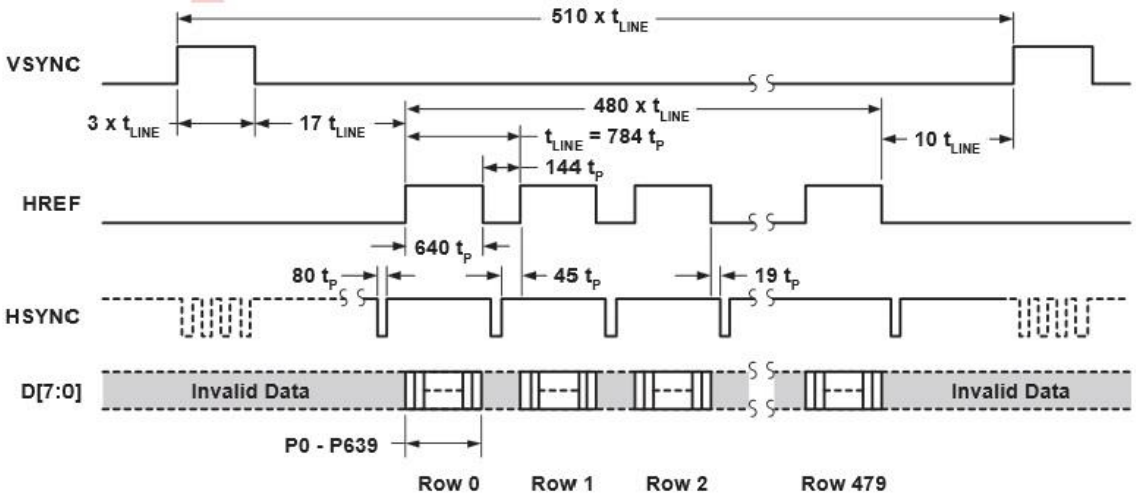


Figure 7 Timing reference parallel communication

Register configuration

Setting up the camera begin by writing 0x80 in the register COM7, followed by waiting 1 ms. This allow to reinitialize all the device registers, and put default values inside.

Register to initialize after is describe in table 2

Address (HEX)	Register Name	Default value (HEX)	Description	Note
11	CLKRC	80	Internal Clock Allow to set Pll and pre-scale for the output clock	
3A	TSLB	0C	Line Buffer Test Option	

			Allow to set the output sequence with COM13	
3D	COM13	99	Common Control 13 Allow to set the output sequence with TSLB	Enabling Gamma and UV saturation, and set the output sequence
15	COM10	00	Common Control 10 Synchonization signal option	

Tableau 2 Initialization register

Table 3 describes configuration of the device to set it in QVGA mode without zoom

Address (HEX)	Register Name	Setting value (HEX)	Description
12	COM7	10	QVGA selection
0C	COM3	04	Down sampling selected
3E	COM14	19	Enable DCW and scaling PCLK Scaling parametters can be adjusted manually PCLK divided by 2

Tableau 3 Initialization register for QVGA

Timing constraint are given in the page 6 of the camera's hardware manual. It's necessary to apply them during the initialization flow, and the setting of communication peripherals on RX64M.

Communications Peripherals

The SCCB liaison works with 400 kHz I2C standard communication. As it said below this communication serves to setup the camera's registers. This communication is very simple to instantiate on a microcontroller and some code are available on the internet to make it by yourself. In order to use the hardware peripheral it is necessary to know where the SCCB pins of the camera are connected. On RX64M starter kit board these pins are connected on Simple Communication Interface number 6. The flow charts to initialize this communication is given in figure 8.

The most complex communication protocol, allow to keep pixel of an image, this communication is an 8 bit parallel communication in 30MHz. One microcontroller's peripherals allow to keep data's automatically. On the RX64M this peripheral, Parallel Data Capture, is setting up as shown on figure 9.

Figure 10 shows data transfer between camera and the microcontroller. Data's are saved in a FIFO memory which is needed to read it at each flag generation. In the interrupt routine it's necessary to read the FIFO 8 time when data are available. Details of implementation of this peripheral is given in annex.

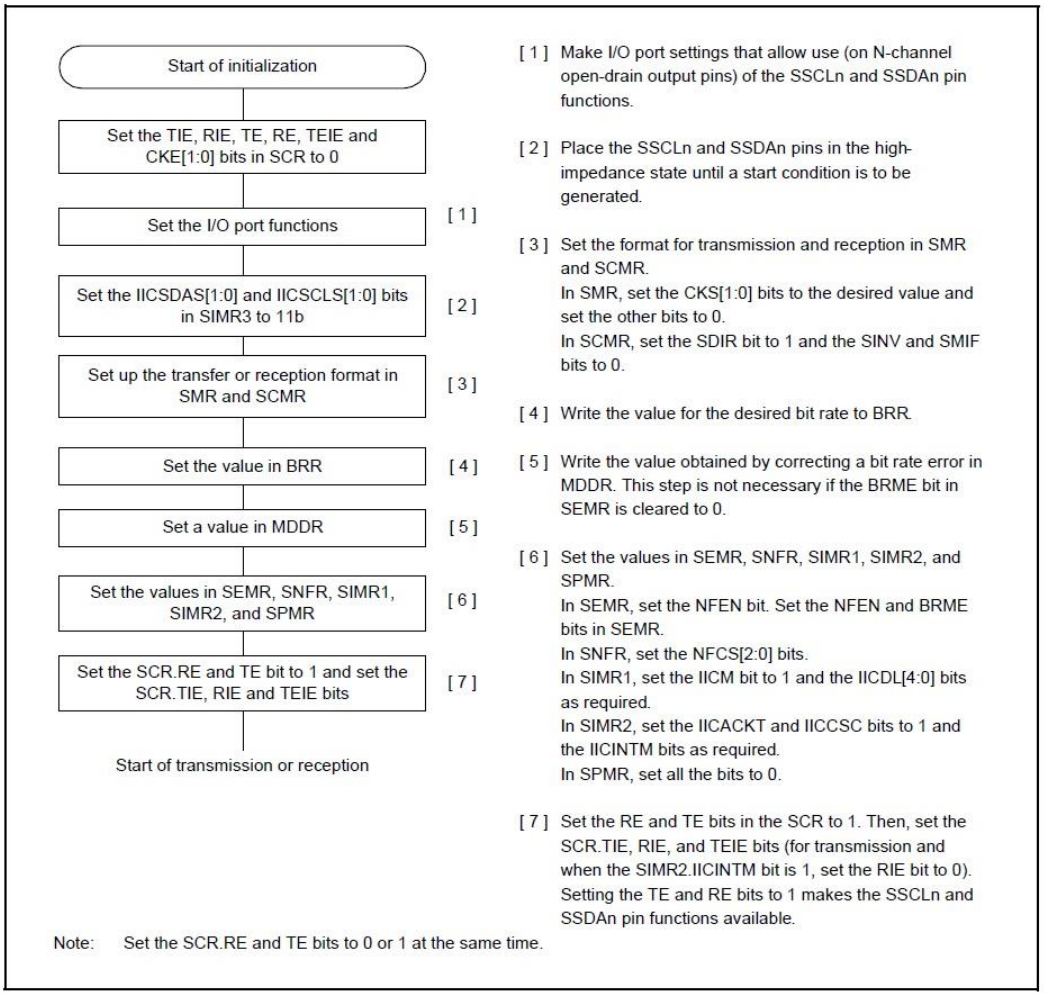


Figure 8 Flow I2C communication initialization

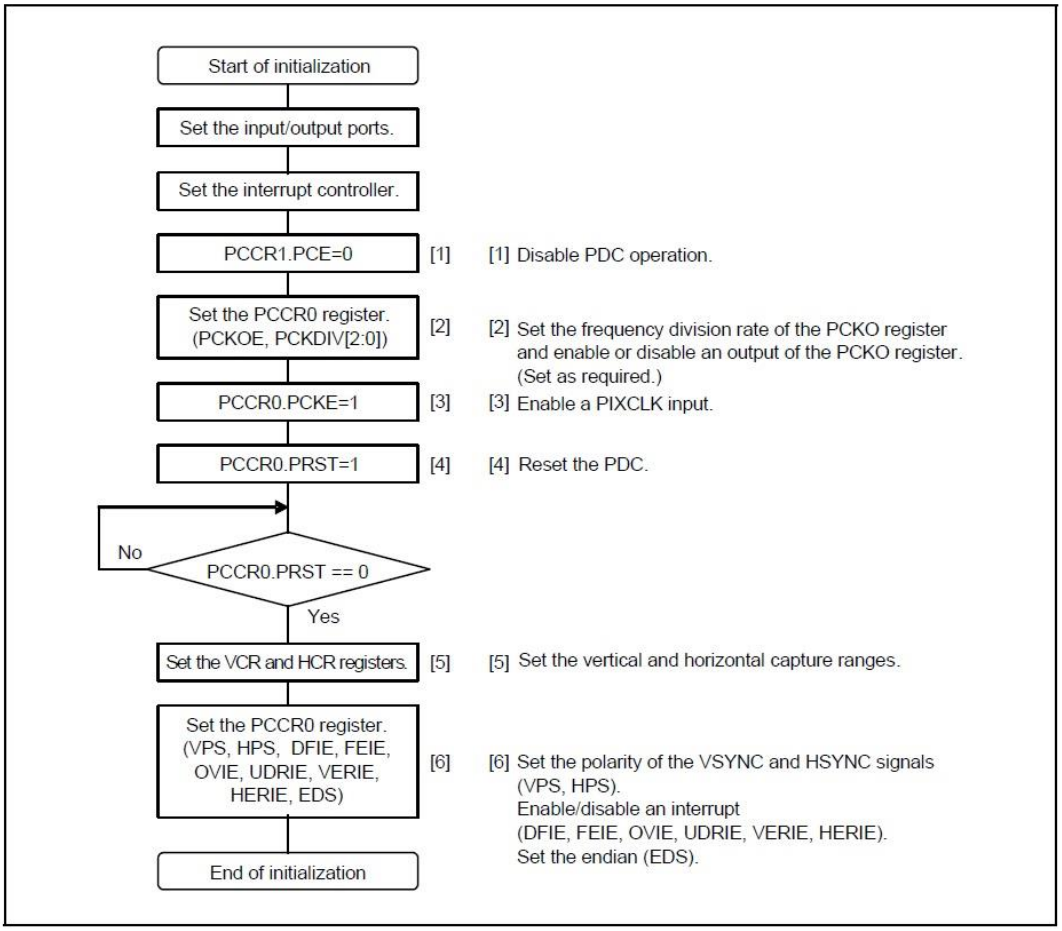


Figure 9 PDC flow chart initialization

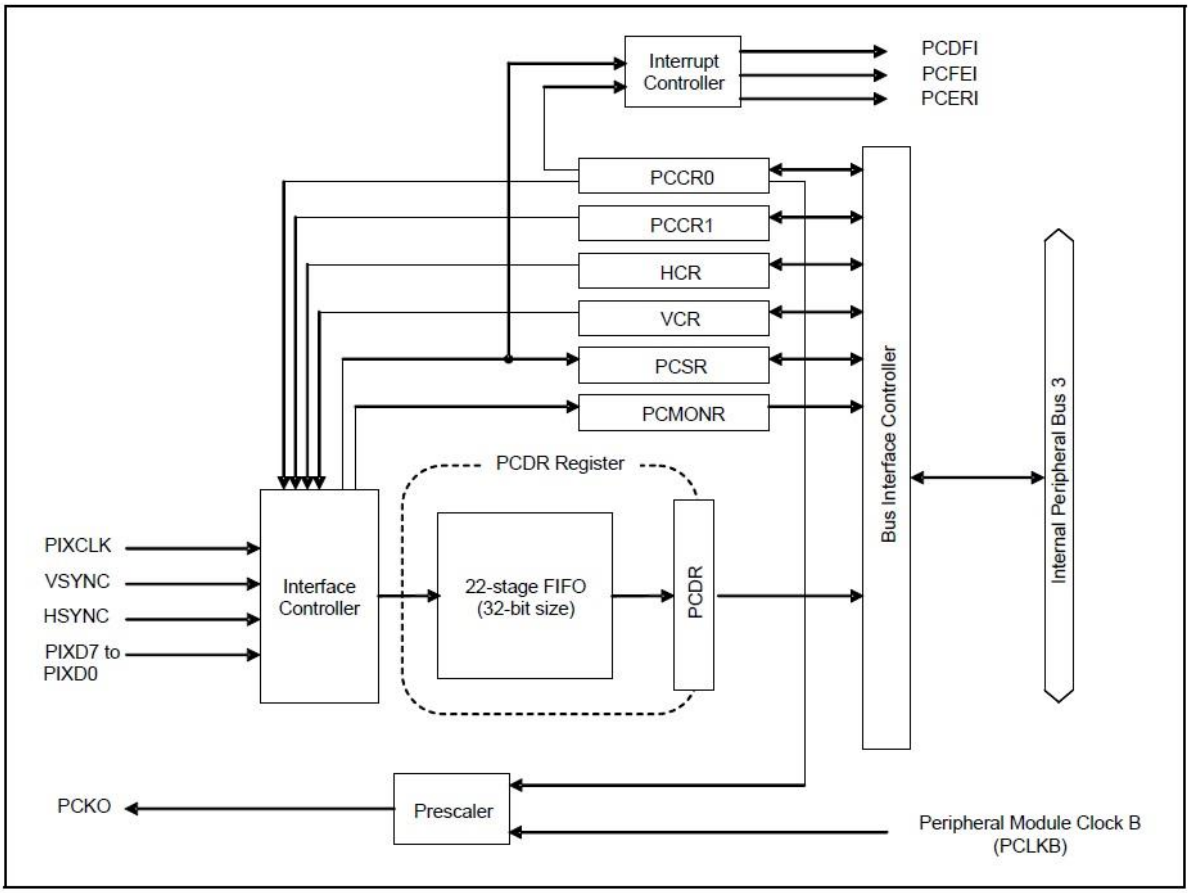


Figure 10 Data flow of PDC

Bibliography

- OV7670_CMOS.pdf 42p.
  OV7670/OV7171 CMOS VGA (640x480) CAMERACHIP with OmniPixel ® Technology

- OV7670 Implementation Guide (V1.0).pdf 64p.
  OV7670/OV7171 CMOS VGA (640x480) CameraChip Implementation Guide

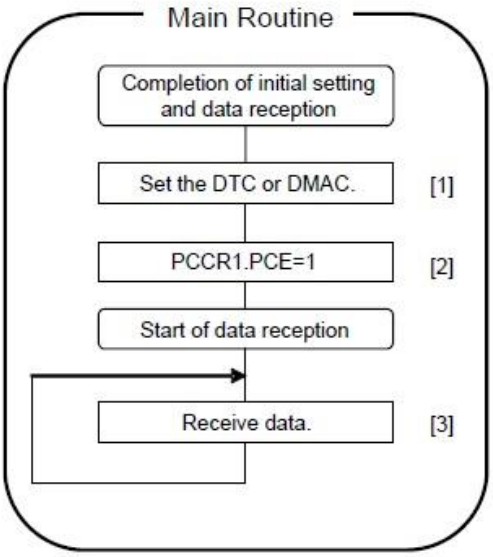
- SCCBSpec_AN (2.2).pdf 24p
  OmniVision Serial Camera Control Bus (SCCB) Functional Specification

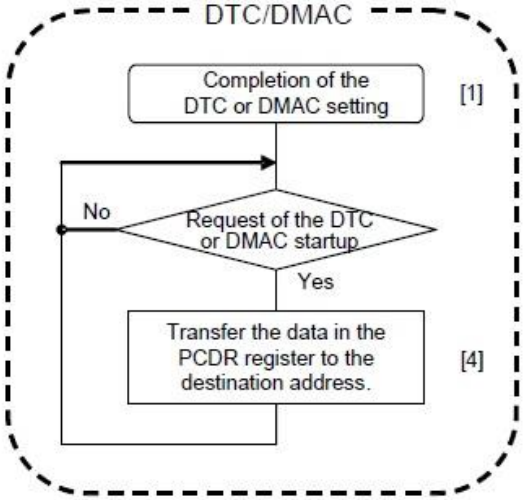
- r01uh0377ej0100_rx64m.pdf 2903p.
  User's manual : Hardware

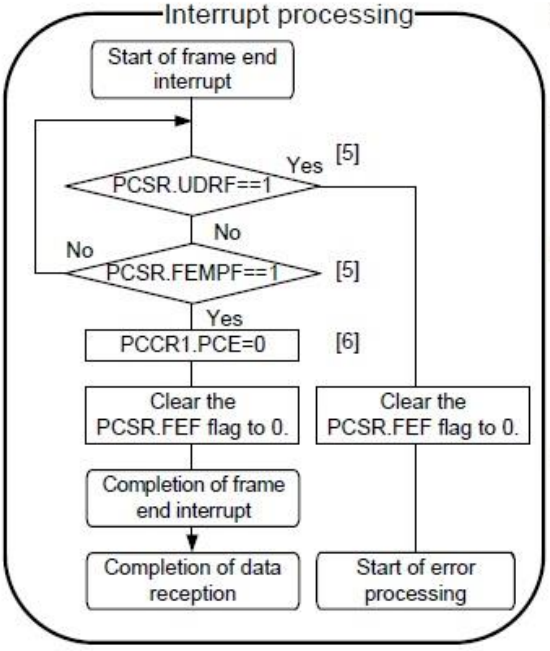
- r20ut2589eg0100_rsk+rx64m_schema 17p.

ANNEX TABLE

PDC implementation flow chart







[1] Setting of the DTC or DMAC:
Set the DTC or DMAC according to the size of the captured image.

[2] Enabling of PDC operation:
Set the PCE bit in the PCCR1 register to 1 to enable PDC operation.

[3] Data reception:
Receive the data from the external I/O and wait for generation of a receive data ready interrupt (DTC/DMAC startup) or a frame end interrupt.

[4] Transfer by the DTC or DMAC:
Transfer the data in the PCDR register to the destination address in response to a receive data ready interrupt (DTC/DMAC startup).
Read the PCDR register in longword access and transfer 32 bytes of data per receive data ready interrupt.

[5] Frame end interrupt:
Read the FEMPF flag in the PCSR register and verify completion of the DTC or DMAC transfer.
If an underrun occurs during DTC or DMAC transfer, setting of the FEMPF flag may not proceed. When an underrun occurs, clear the FEF flag to 0 and branch to the appropriate error processing routine.

[6] Completion of data reception:
After completion of the transfer, clear the PCE bit in the PCCR1 register to 0, and then clear the FEF flag in the PCSR register to 0. This completes handling of the frame end interrupt and data reception.

PDC implementation initialization code

```
void R_PDC_Create()
{
	/*Cancel PDC module stop state*/
	SYSTEM.PRCR.WORD = 0xA50BU;

	/* Enable writing to MPC pin function control registers */
	MPC.PWPR.BIT.B0WI = 0U;
	MPC.PWPR.BIT.PFSWE = 1U;

	int i;
	MSTP( PDC ) = 0U;

	IR(PDC, PCDFI) = 0U;
	IPR(PDC, PCDFI) = 15;
	IEN(PDC, PCDFI) = 1U;
	/*Set PIXD0 & PIXD3 pins*/
	PORT1.PMR.BYTE |= 0xACU;
	MPC.P15PFS.BYTE = 0x1CU;
	MPC.P17PFS.BYTE = 0x1CU;

	/*Set PIXD4, PIXD5, PIXD6, PIXD7, PIXCLK, HSYNC pins*/
	PORT2.PMR.BYTE |= 0x3FU;
	MPC.P20PFS.BYTE = 0x1CU;
	MPC.P21PFS.BYTE = 0x1CU;
	MPC.P22PFS.BYTE = 0x1CU;
	MPC.P23PFS.BYTE = 0x1CU;
	MPC.P24PFS.BYTE = 0x1CU;
	MPC.P25PFS.BYTE = 0x1CU;

	/*Set VSYNC & PCKO pins*/
	PORT3.PMR.BYTE |= 0x0CU;
	MPC.P32PFS.BYTE = 0x1CU;
	MPC.P33PFS.BYTE = 0x1CU;

	/*Set PIXD1, PIXD2 pins */
	PORT8.PMR.BYTE |= 0xC0U;
	MPC.P86PFS.BYTE = 0x1CU;
	MPC.P87PFS.BYTE = 0x1CU;

	/*Disable Operation on PDC*/
	PDC.PCCR1.LONG = 0U;

	/*Setting PCLKO*/
	PDC.PCCR0.BIT.PCKOE = 1U;	//enable PCKO
	PDC.PCCR0.BIT.PCKDIV = 1U;		//Divide PCLKB by 4, PCKO = 15MHz

	/*Enable PIXCLK input*/
	PDC.PCCR0.BIT.PCKE = 1;

	/*Reset the PDC*/
	PDC.PCCR0.BIT.PRST = 1;
	while(PDC.PCCR0.BIT.PRST);
```

```
/*Set vertical and horizontal capture range*/
       PDC.VCR.BIT.VST = 0U;
       PDC.VCR.BIT.VSZ = 480;          // 1E0h = 480
       PDC.HCR.BIT.HST = 0U;
       PDC.HCR.BIT.HSZ = 640;          // 0280h = 640

       /*Set the polarity of the VSYNC and HSYNC signals*/
       PDC.PCCR0.BIT.VPS = 1U;               //active low
       PDC.PCCR0.BIT.HPS = 0U;               //active high

       /*Enable end frame interrupt and data ready interrupt*/
       PDC.PCCR0.BIT.DFIE = 1U;
       PDC.PCCR0.BIT.FEIE = 1U;
       PDC.PCCR0.BIT.OVIE = 1U;
       PDC.PCCR0.BIT.UDRIE = 1U;
       PDC.PCCR0.BIT.VERIE = 1U;
       PDC.PCCR0.BIT.HERIE = 1U;

       /*Set Little Endian*/
       PDC.PCCR0.BIT.EDS = 0U;

    MPC.PWPR.BIT.PFSWE = 0U;
    MPC.PWPR.BIT.B0WI = 1U;

    /* Enable protection */
    SYSTEM.PRCR.WORD = 0xA500U;

       for(i = 0; i < 240; i++)
       {
              image_cam[i] = &Pixel_cam[i*320];
       }

}
```

PDC implementation interrupt routine reading FIFO

```
void R_PDC_Data_Ready_Interrupt(void)
{
       static uint32_t value = 0;


       Pixel_cam[value++].data = PDC.PCDR.LONG;
       Pixel_cam[value++].data = PDC.PCDR.LONG;
       Pixel_cam[value++].data = PDC.PCDR.LONG;
       Pixel_cam[value++].data = PDC.PCDR.LONG;
       Pixel_cam[value++].data = PDC.PCDR.LONG;
       Pixel_cam[value++].data = PDC.PCDR.LONG;
       Pixel_cam[value++].data = PDC.PCDR.LONG;
       Pixel_cam[value++].data = PDC.PCDR.LONG;

       value %= 76800;
}
```