基于 NXP S32V234 的 ISP 工程建立介绍

WPI

文件标识	基于 NXP S32V234 的 ISP 工程建立介绍		
当前版本	V1.0	联系方式	Alva.Hong@wpi-group.com
作者	Alva Hong	撰写日期	2020.02.14
审核者		审核日期	

版本历史

版本	日期	描述	作者
V1.0	2020.02.14	建立文档	Alva Hong

WPI

目录

WPI

1. 芯片性能概述

S32V234 是 NXP 推出的一款汽车视觉微处理器，具有 4 个主频达 1GHz 的 A53 核以及 1 个主频达 133MHz 的 M4 内核，具有强大的运算处理能力。 S32V 中包含的 APEX 和 ISP 核，可以轻松实现图像和视频的检测，识别，分类等应用。 S32V234 的框架图如下：

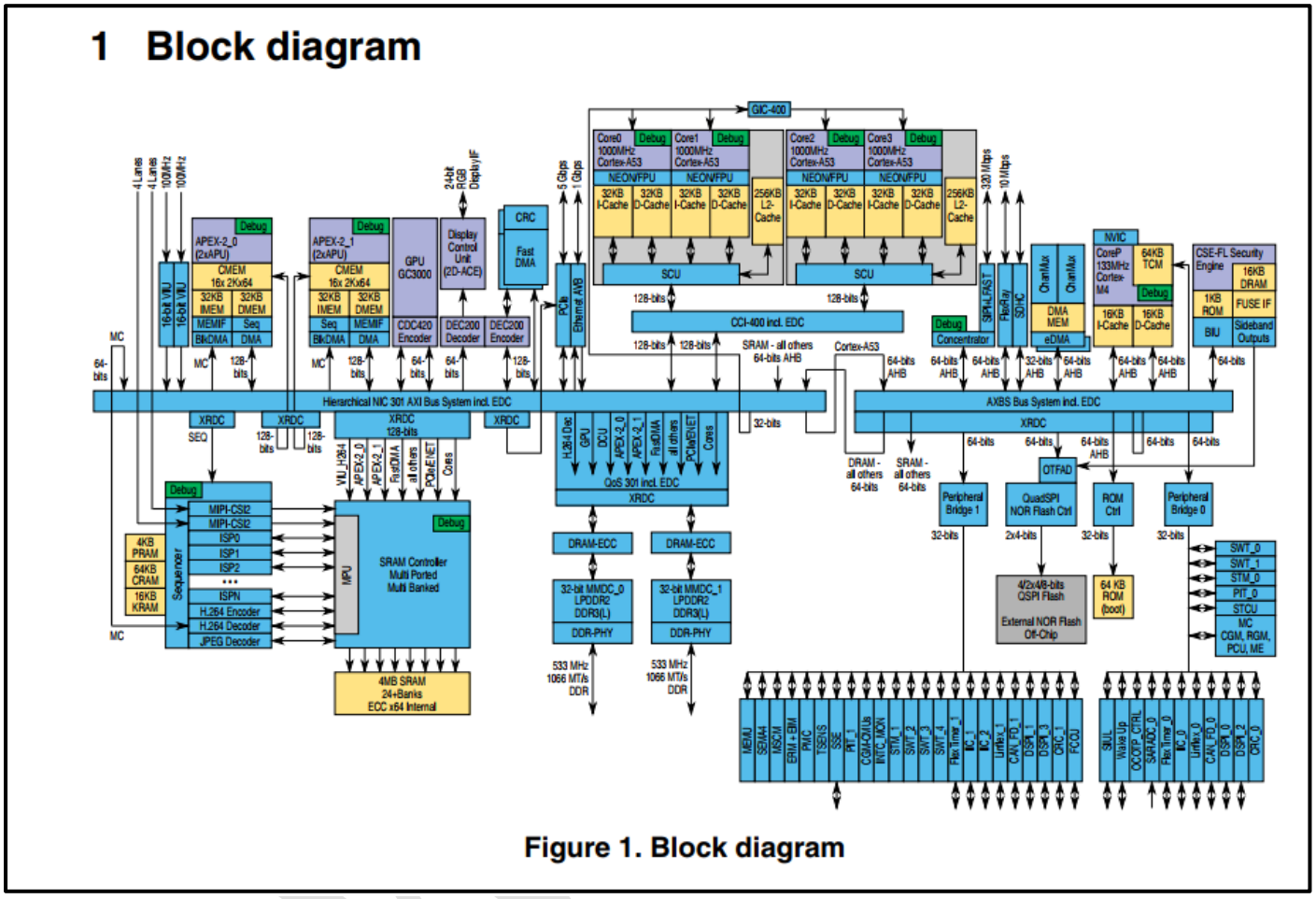


Figure 1. Block diagram

2. 功能介绍

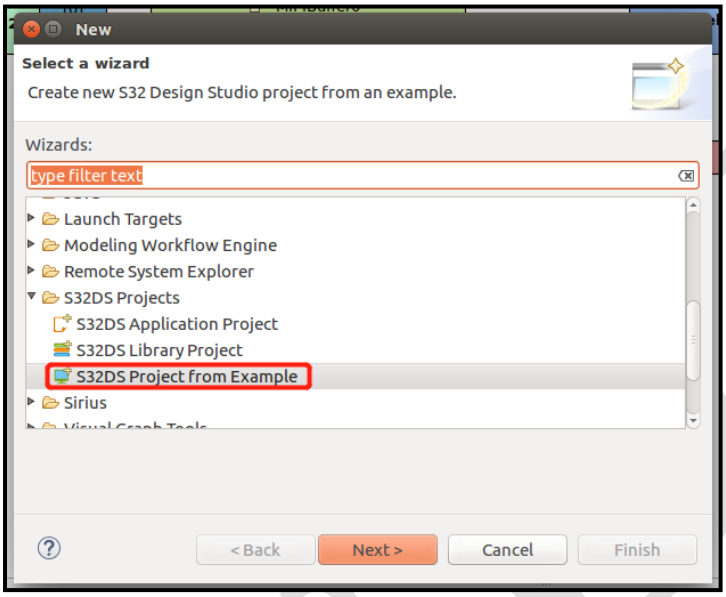
ISP 核是一个 S32V234 处理图像数据主要核心，通过其强大的图像数据转化能力，可以针对性地完成图像的格式转化、效果优化、数据处理等功能。

NXP S32V234 简化了 ISP 处理繁琐的开发过程，通过提供 DS 工具来实现 ISP 工程的建立以及代码生成，方便开发者更好地使用工具来进行 ISP 开发。只需要通过绘制和配置数据处理的 Graph 图形，便可以生成 ISP 处理的核心代码。下面会分别介绍三种通过 NXP 提供的 DS 工具建立 ISP 工程的方式，分别是利用 DS 工具自带 Example 的 ISP 工程、利用已有的 VSDK_Graph 建立 ISP 工程、自主建立 ISP 工程等三种建立 ISP 工程的方式。

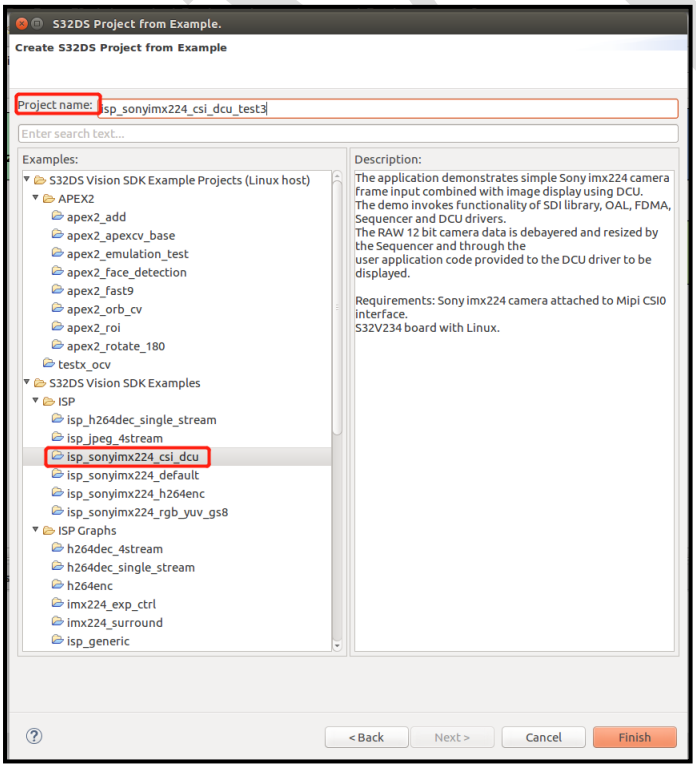
3. 功能实现

3.1 建立一个使用 Example 的 ISP 工程

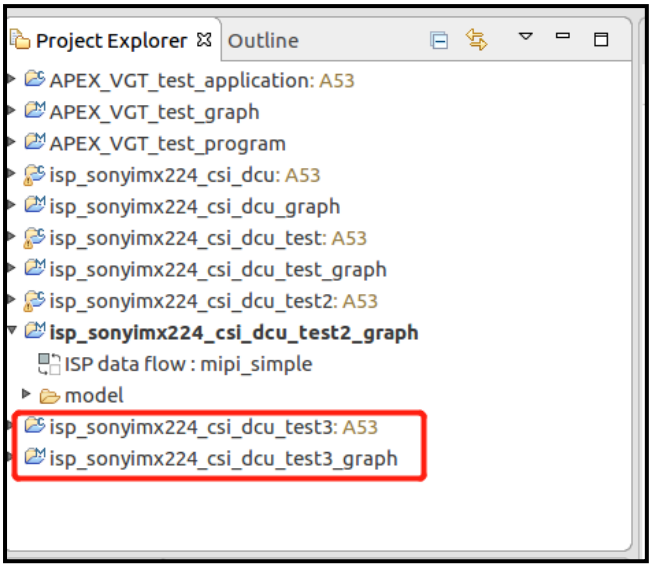
① 点击 Create ISP Project ，出现以下选项，选择 S32DS Project from Example



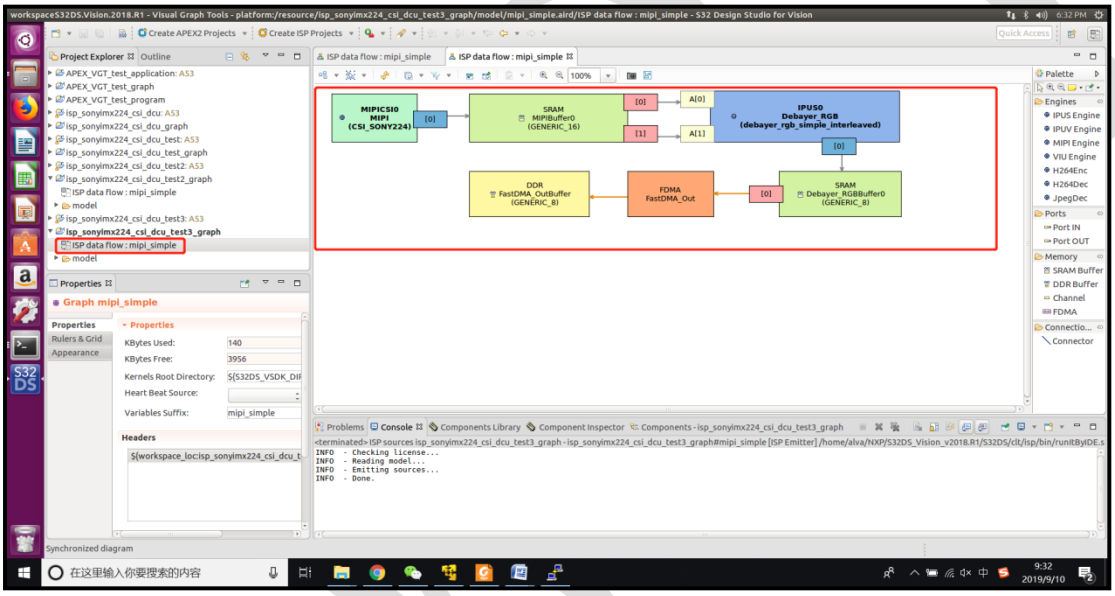
② 选择 isp_sonyimx224_csi_dcu , Project name 可以顺便起名



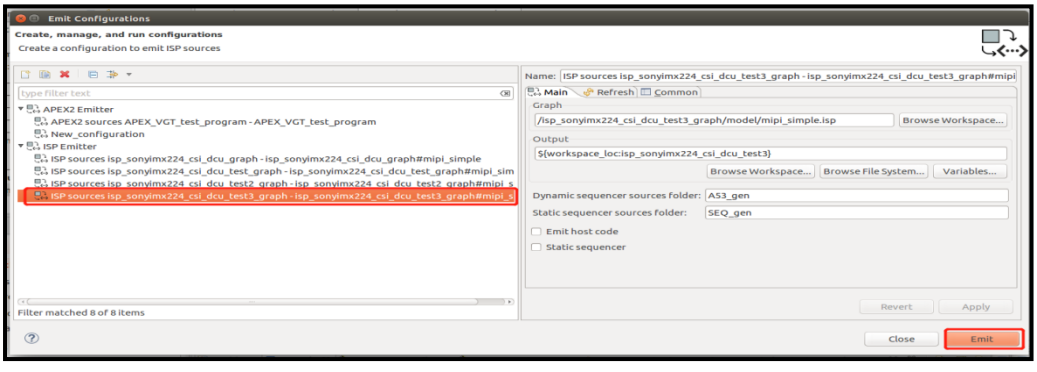
③ 点击 Finish 之后，会在 Project Explorer 出现以下两个项目



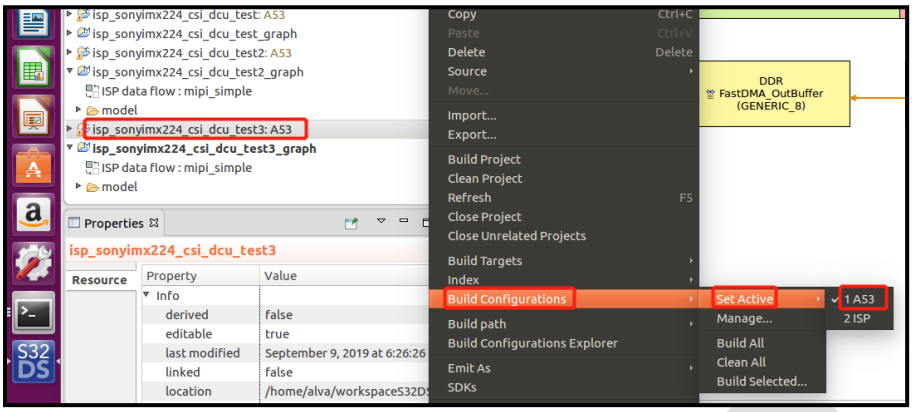
④ 选择 isp_sonyimx224_csi_dcu_test3_graph，并双击 mipi_simple



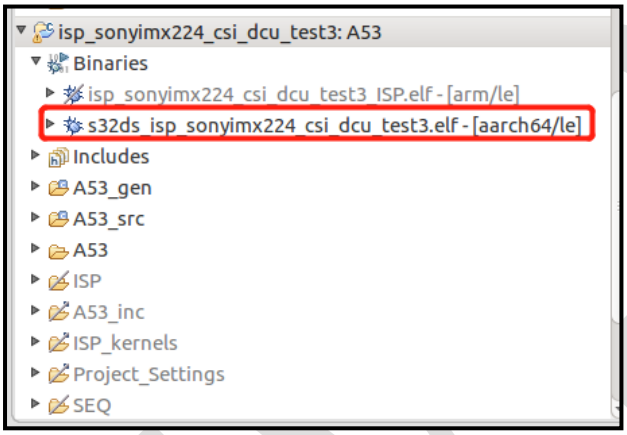
⑤ 在右边图像框点击右键 ，选中 Emit As – > Emit Configurations，并在出现的界面下选择对应的工程，然后 Emit



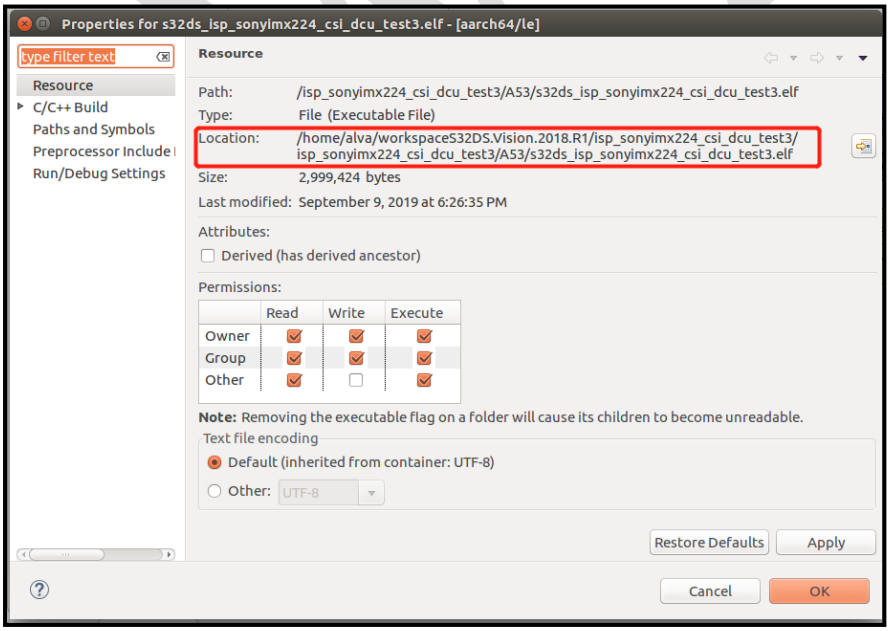
⑥ 接下来选择上面的工程 isp_sonyimx224_csi_dcu_test3 ， 对着这个工程点击右键 ， 然后点击 Build Configurations -> Set Active -> 选择 A53



⑦ isp_sonyimx224_csi_dcu_test3 点击右键 ， 选择 Build Project ， 编译完成之后，会生成以下 s32ds_isp_sonyimx224_csi_dcu_test3.elf 可执行文件

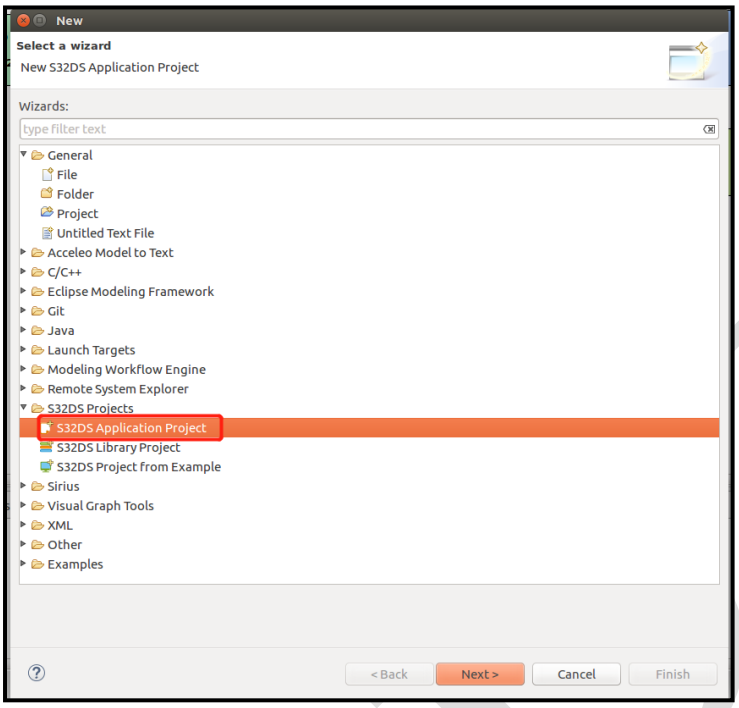


⑧ 对着这个文件点击右键 -> Properties 可以看到 .elf 应用程序所在的路径

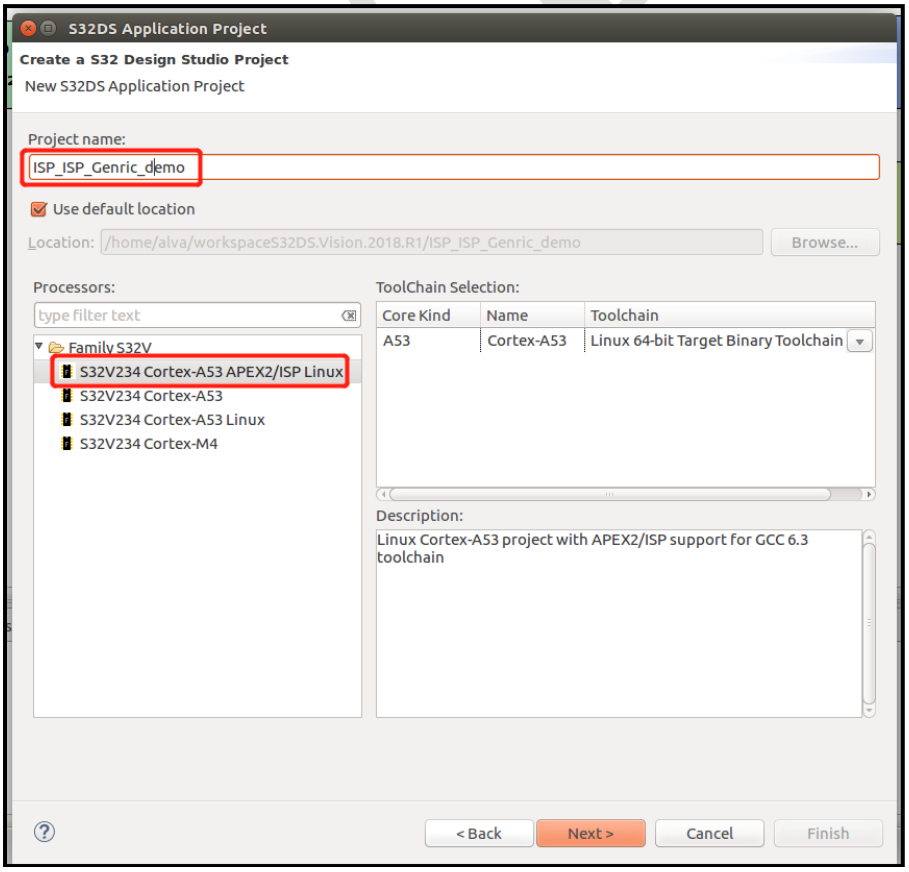


3.2 根据已存在的 VSDK_Graph 建立一个 ISP 工程

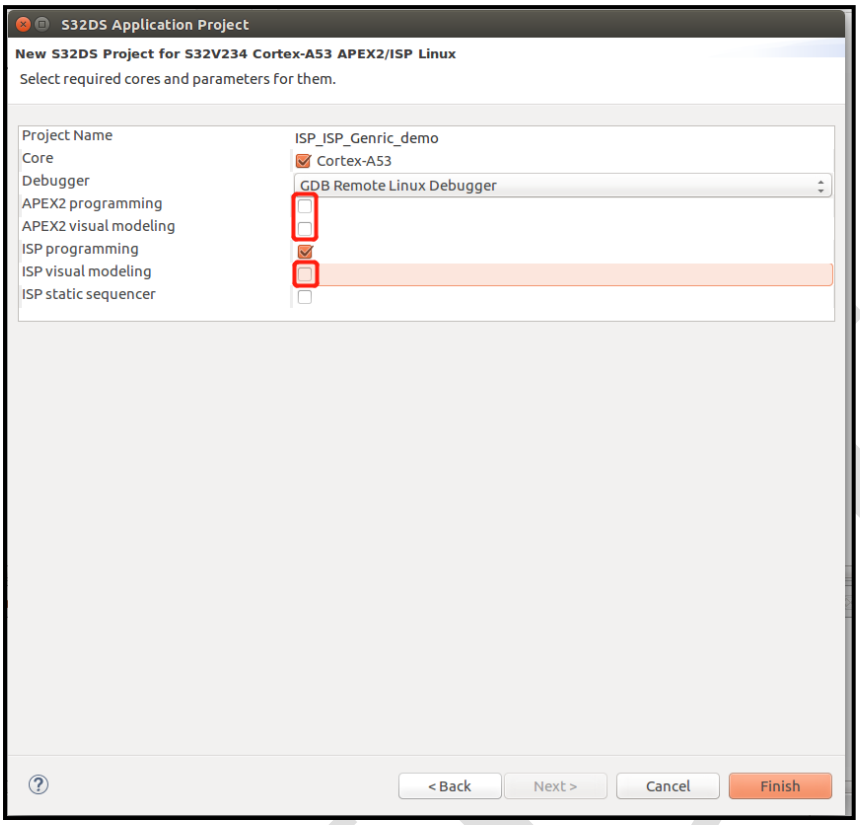
① 点击 Create ISP Project ，出现以下选项，选择 S32DS Application Project



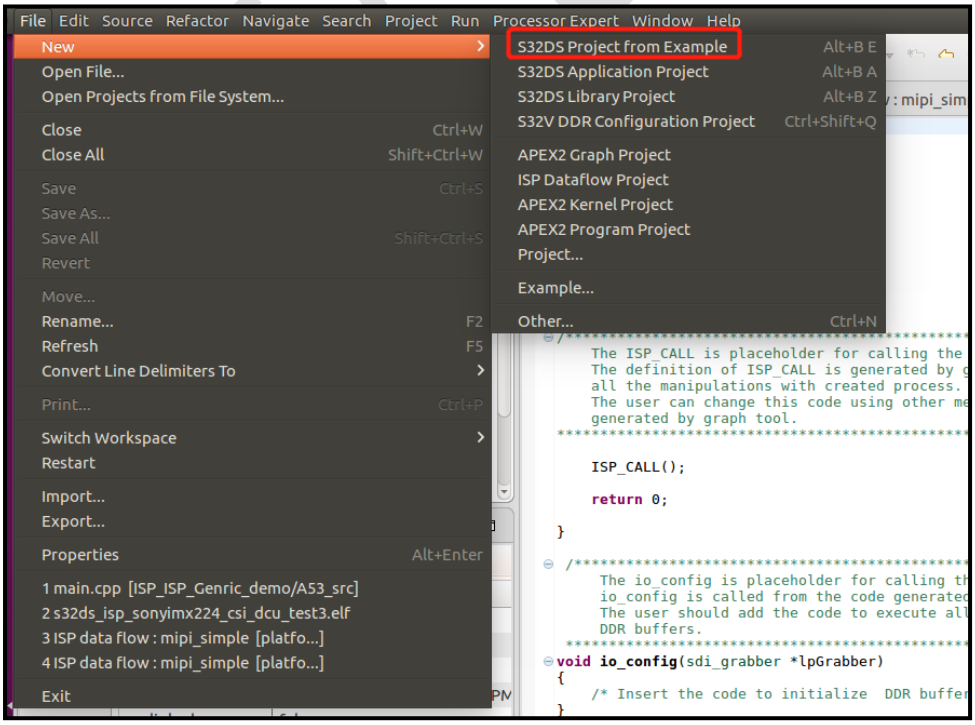
② 选择 S32V234 Cortex-A53 APEX2/ISP Linux， Project name 可以随便起



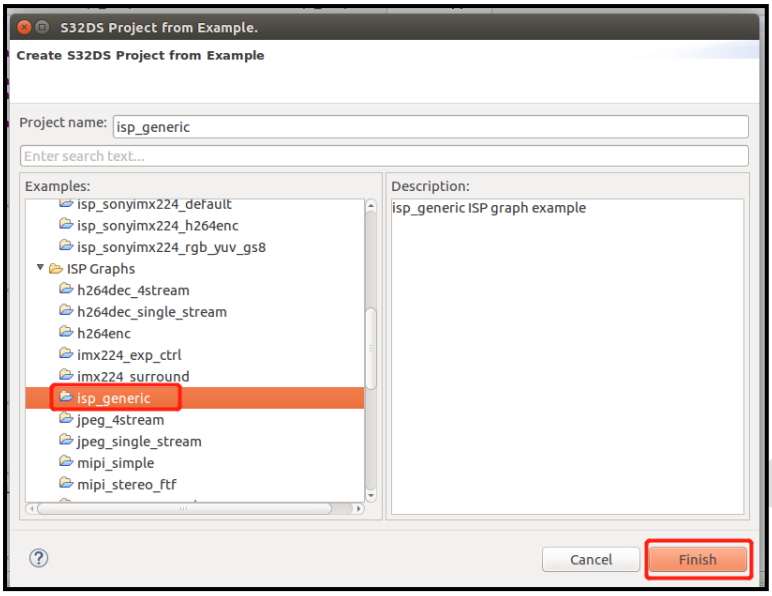
③ 在下面选项中取消以下画出的红框中的 √



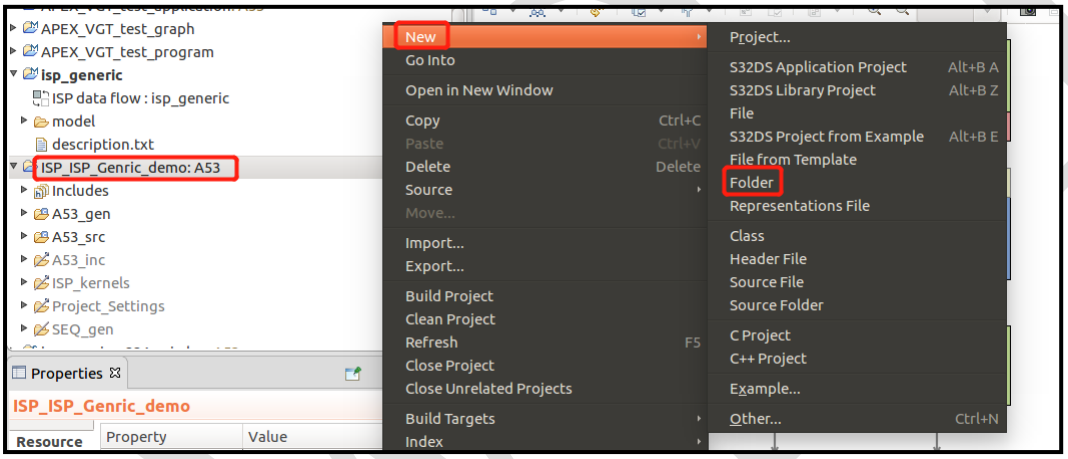
④ 新建一个 S32DS Project from Example



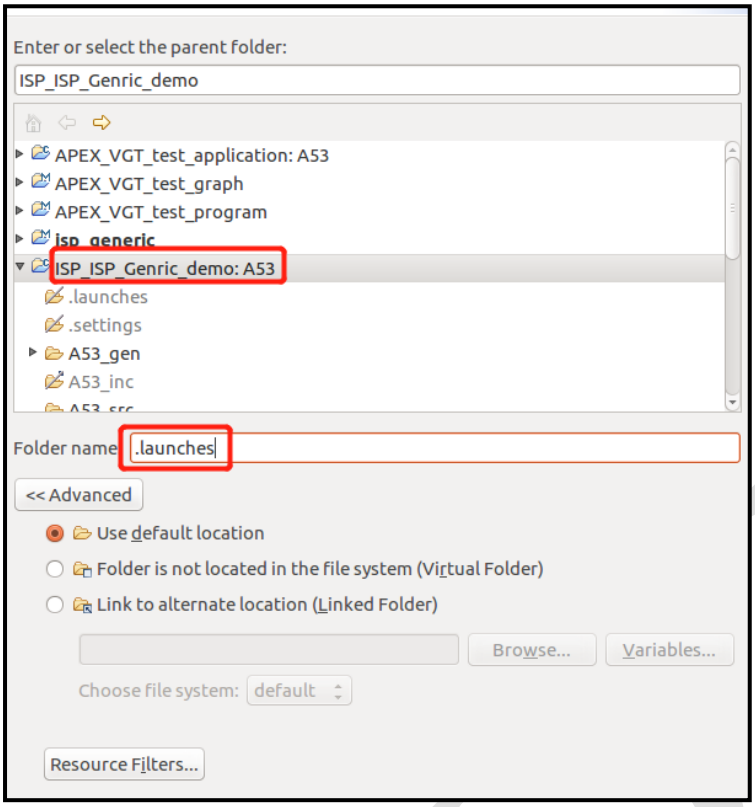
⑤ 选中 isp_generic ， 然后点击 Finish



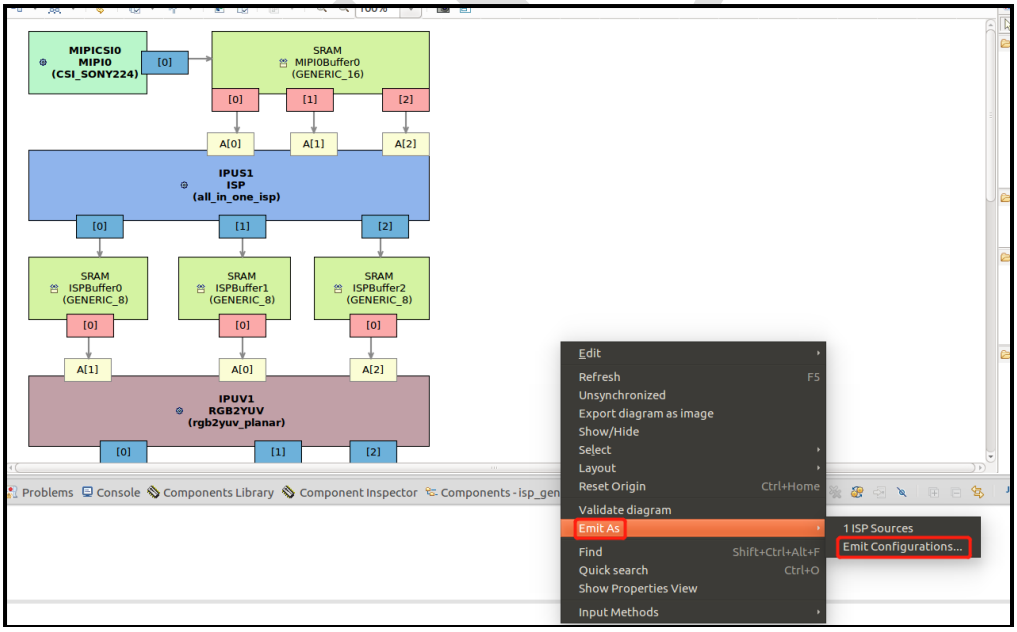
⑥ 选中 isp_generic ，然后点击 Finish



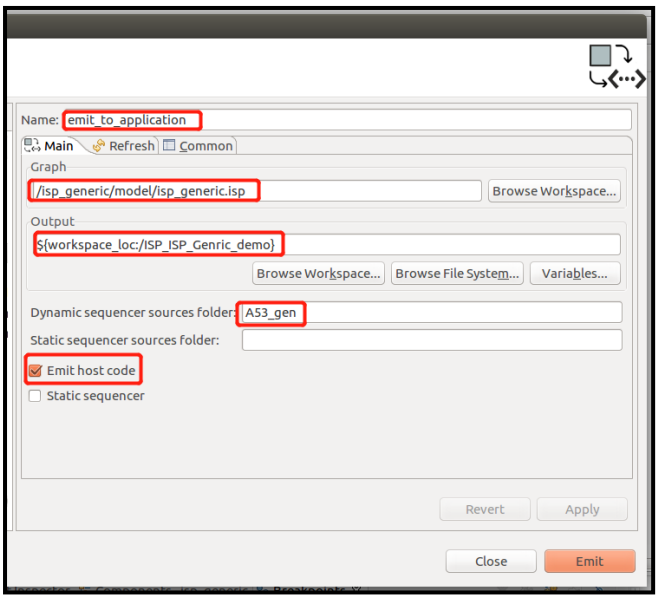
⑦ 选中 ISP_ISP_Genric_demo:A53 ，并在 Folder name 中为 Folder 起名

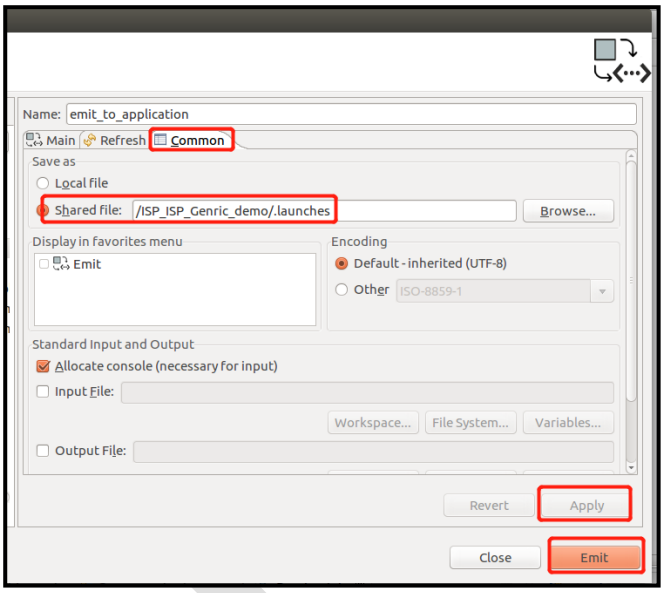


⑧ 在右边图像框点击右键 ，选中 Emit As – > Emit Configurations



⑨ 选项设置如下





⑩ 复制 isp_sonyimx224_csi_dcu/A53_src/main.cpp 去覆盖 ISP_ISP_ Generic_demo/ A53_src / main.cpp ，并且修改文件如下：

```
#ifdef __STANDALONE__
#include "frame_output_dcu.h"
#define CHNL_CNT io::IO_DATA_CH3
#else // #ifdef __STANDALONE__
#include "frame_output_v234fb.h"
#define CHNL_CNT io::IO_DATA_CH3
#endif // else from #ifdef __STANDALONE__

#include "sdi.hpp"
#include <isp_generic_c.h>

#include "vdb_log.h"
#include <common_helpers.h>
```

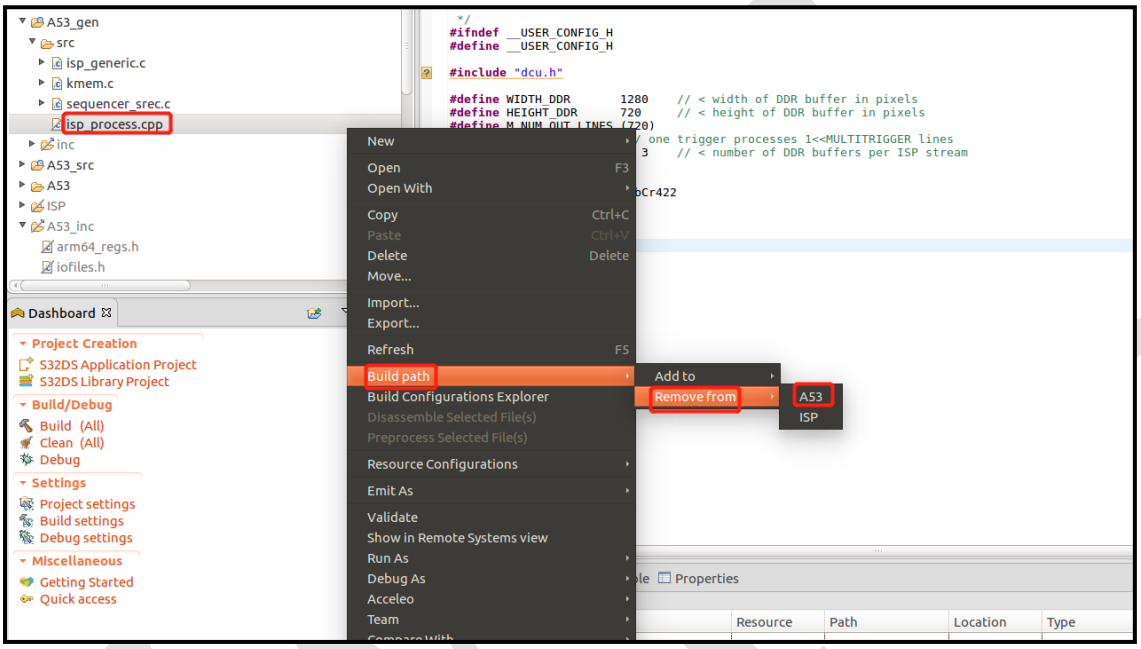
```
static int32_t DdrBuffersPrepare(AppContext& arContext)
{
  // *** RGB full buffer array ***
  // modify DDR frame geometry to fit display output
  SDI_ImageDescriptor lFrmDesc;
  lFrmDesc = SDI_ImageDescriptor(WIDTH, HEIGHT, RGB888);

  if(arContext.mpFdma->DdrBufferDescSet(FDMA_IX_ISP_OUTPUT,
                                        lFrmDesc) != LIB_SUCCESS)
  {
    printf("Failed to set image descriptor.\n");
    return -1;
  } // if frame descriptor setup failed

  // allocate DDR buffers
  if(arContext.mpFdma->DdrBuffersAlloc(DDR_BUFFER_CNT) != LIB_SUCCESS)
  {
    printf("Failed to allocate DDR buffers.\n");
    return -1;
  } // if ddr buffers not allocated

  return 0;
} // DdrBuffersPrepare(AppContext &arContext)
```

```
if(arContext.mpGrabber->ProcessSet(
        gpGraph, &gGraphMetadata,
        kmem_srec, sequencer_srec) != LIB_SUCCESS)
{
  printf("Failed to set ISP graph to grabber.\n");
  return -1;
} // if ISP graph not set
```

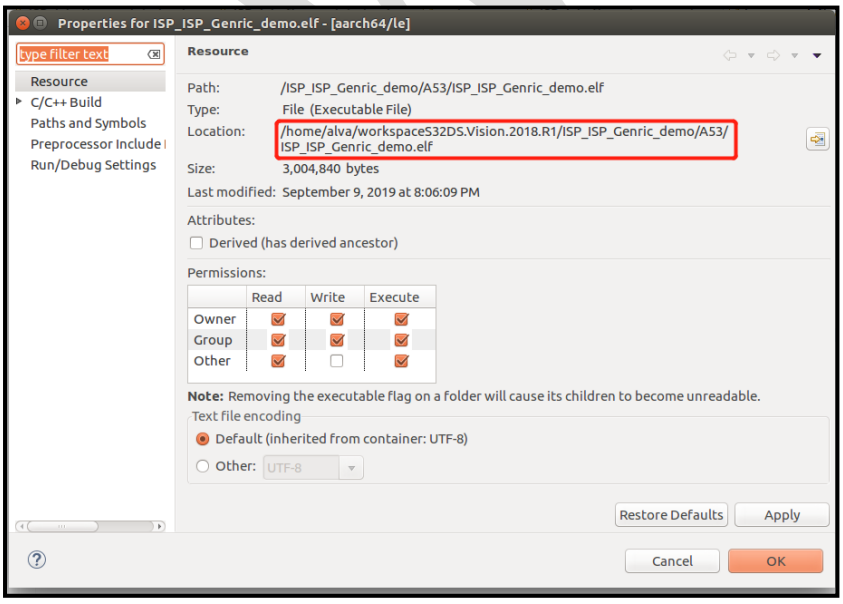
⑪ 修改 ISP_ISP_Generic_demo/A53_inc/isp_user_define.h

```
/*
 * User's configurations
 *
 */
#ifndef __USER_CONFIG_H
#define __USER_CONFIG_H

#include "dcu.h"

#define WIDTH_DDR       1280    // < width of DDR buffer in pixels
#define HEIGHT_DDR      720     // < height of DDR buffer in pixels
#define M_NUM_OUT_LINES (720)
#define  MULTITRIGGER 0  // one trigger processes 1<<MULTITRIGGER lines
#define DDR_OUT_BUFFER_CNT 3    // < number of DDR buffers per ISP stream
#define LOOP_NUM        30

#define DCU_BPP_DCU_BPP_YCbCr422
#undef __DCU_BPP

#endif // __USER_CONFIG_H
```

⑫ 将 isp_process.cpp 从 A53 的路径中移除



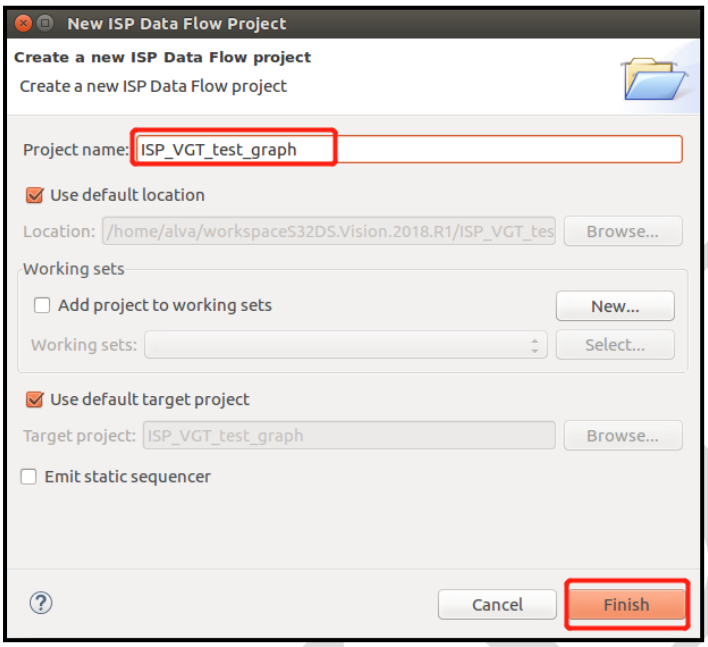
⑬ ISP_ISP_Genric_demo:A53 点击右键 ， 选择 Build Project ， 编译完成之后，会生成以下 ISP_ISP_Genric_demo.elf 可执行文件



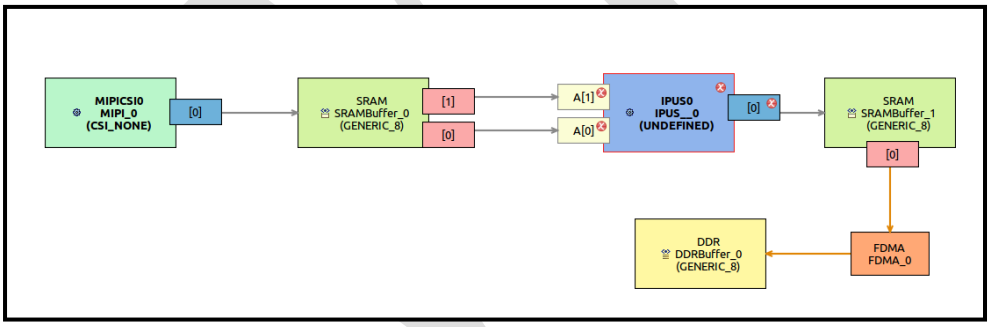
3.3 自主建立 ISP 工程

3.3.1 Make a Graph

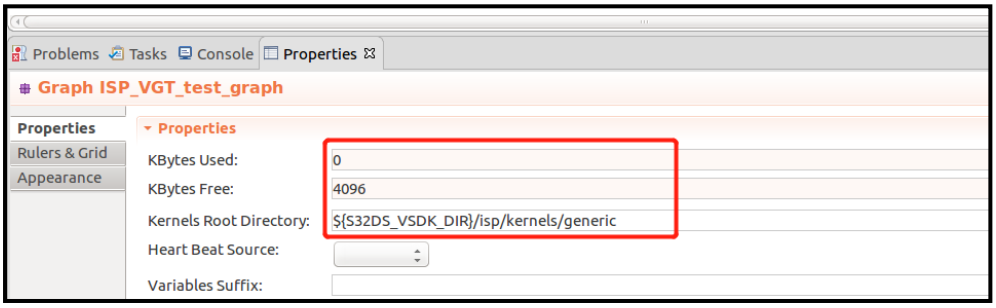
① 建立 ISP DataFlow Project



② 建立 ISP DataFlow Project ，并且画出一下的 graph

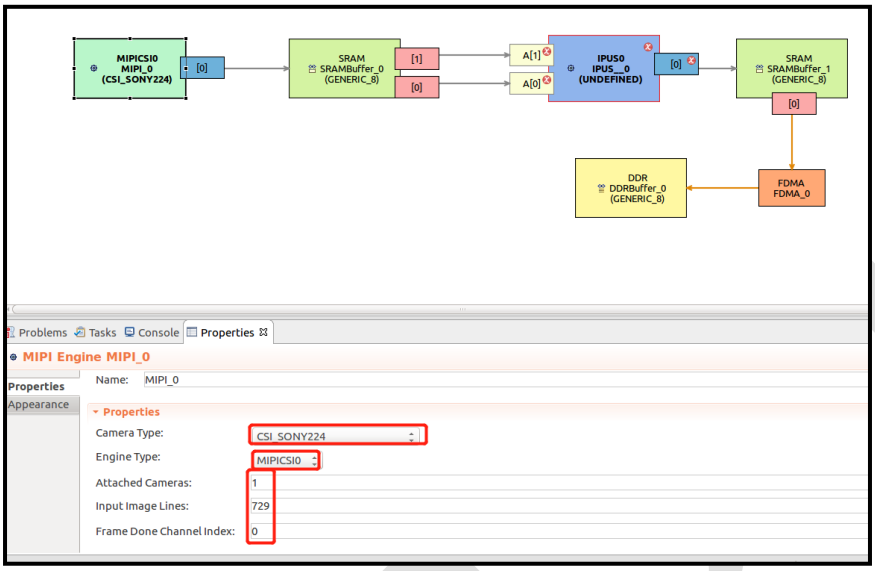


③ 点击 graph 空白处，然后配置下面参数

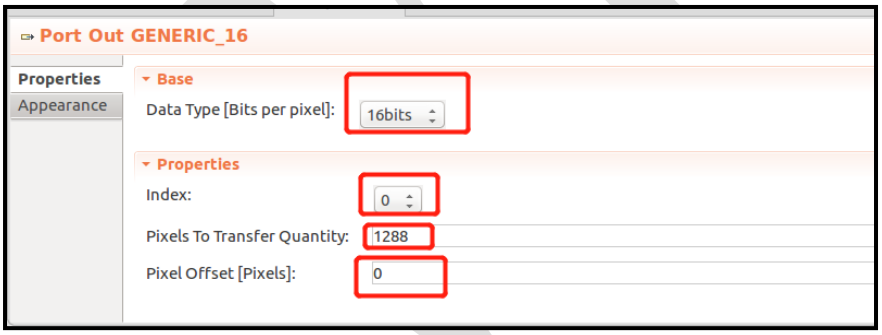


(注： KBytes Used: 一般情况下不超过 1M
Kbytes Free: 表示图中未使用的 SRAM 的空间 ， 这里写了 4M ， 但是只有 1M 是为 ISP 优化的
Kernel Root Directory: 包含 ISP 内核的文件夹)

④ 配置 MIPICSI0 以及输出口

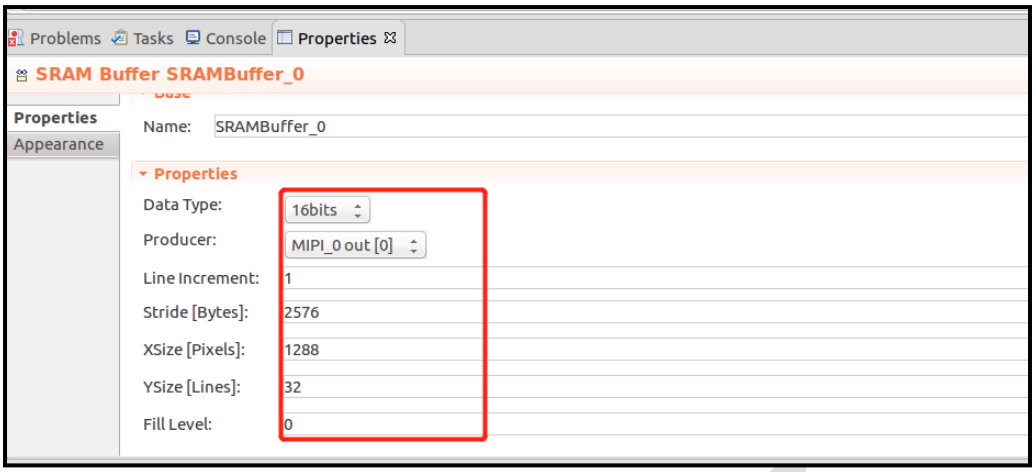


(注： Camera type: Camera 类型
Engine Type: 连接的 MIPI_CSI 接口
Attached Camera: 与此 MIPI 口连接的 Camera 数
Input Image Lines: MIPI 捕获到的行数)



(注： Data Type：像素数据的数据类型，由于 Sony Camera 的是 12 bits
Index：对应 MIPI_CSI 的虚拟通道
Pixel To Transfer Quantity：图像每行的像素数， Sony Camera 是 1288
Pixel Offset：每一行中跳过的字节数)

⑤ 配置 SRAMBuffer_0

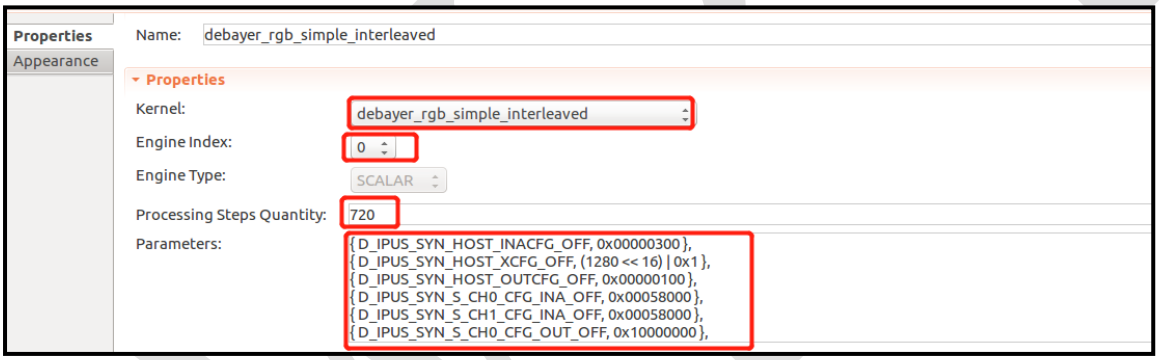


(注： Stride：步幅必须等于或大于 XSize 。它是每一行的字节数。它也可以增加，以扩展线与一些黑色像素（0x0）像素数

XSize：每行像素数

YSize: 缓冲区的大小（ 以行数为单位 ）；缓冲区的大小不需要非常大，特别是在这种情况下，只有一个 FDMA 通道将运行 ）

⑥ 配置 IPU



(注：

Processing Steps Quantity： IPU 要处理的行数

{ D_IPUS_SYN_HOST_INACFG_OFF, 0x00000300 },

=> Enable InA[0] and InA[1] inputs

{ D_IPUS_SYN_HOST_XCFG_OFF, (1280 << 16) | 0x1 },

=> 1280 pixels per lines, pixel processed one at a time (XPOS incremented by 1 with “pixel done” kernel instruction)

{ D_IPUS_SYN_HOST_OUTCFG_OFF, 0x00000100 },

=> Enable OUT[0] output

{ D_IPUS_SYN_S_CH0_CFG_INA_OFF, 0x00058000 },

=> InA[0] configuration: 16 bits, streamed pixel not repeated, every pixels of a lines is used, no added padding on the image

border

{ D_IPUS_SYN_S_CH1_CFG_INA_OFF, 0x00058000 },

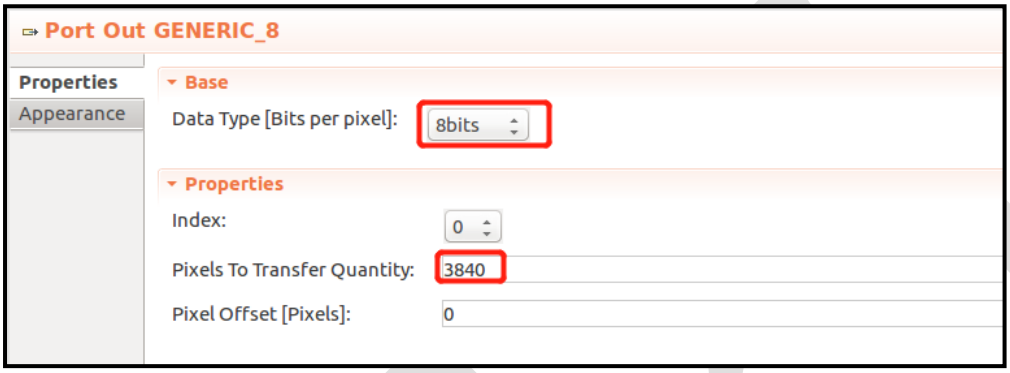
=> InA[1] configuration: 16 bits, streamed pixel not repeated, every pixels of a lines is used, no added padding on the image

border

{ D_IPUS_SYN_S_CH0_CFG_OUT_OFF, 0x10000000 },

=> OUT[0] configuration: 8bits (the frame will be in RGB888: R, G and B will be outputted one per one)

)



(注： RGB888 传输的数据位为 8 bits

Pixels to transfer quantity 是 3*1280)

⑦ 配置 SRAMBuffer_1

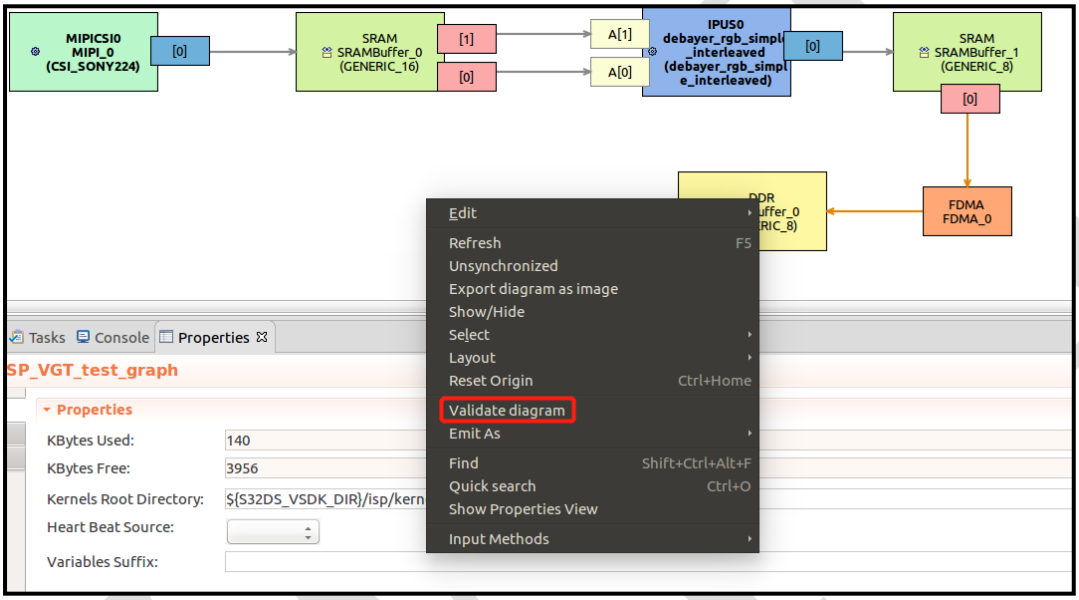
(Stride：步幅必须等于或大于 XSize 。它是每一行的字节数。它也可以增加，以扩展线与一些黑色像素（0x0）像素数

XSize：每行像素数

YSize: 缓冲区的大小（以行数为单位）；缓冲区的大小不需要非常大，特别是在这种情况下，只有一个 FDMA 通道将运行)

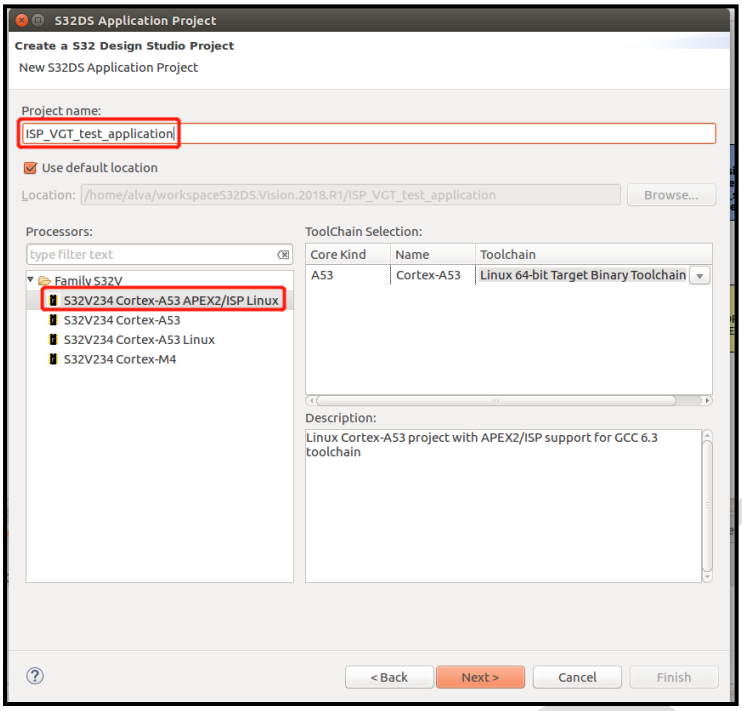
⑧ 配置 DDR

⑨ 检查错误

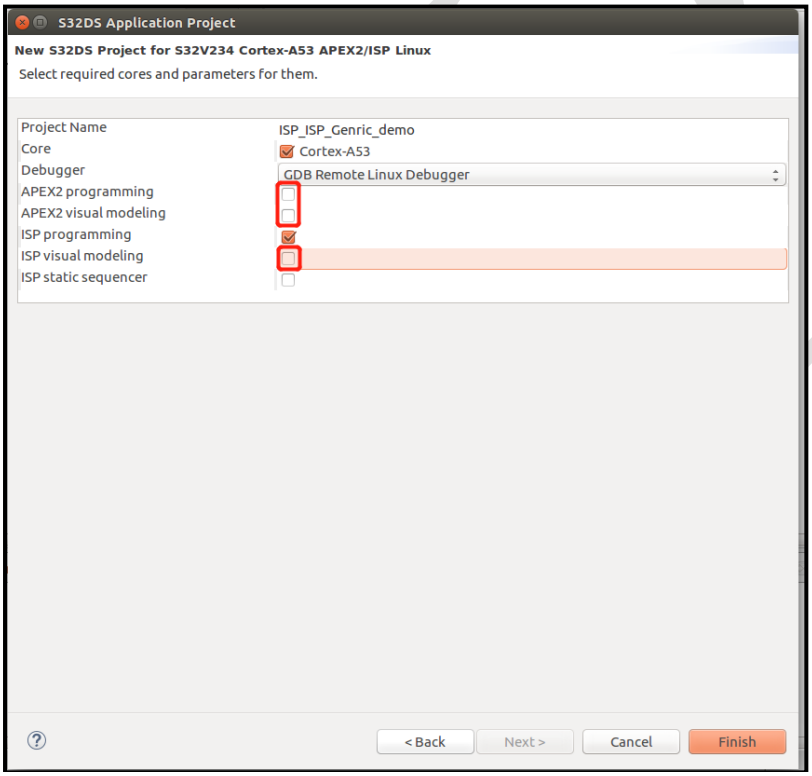


3.3.2 Make a Linux application project without ISP graph

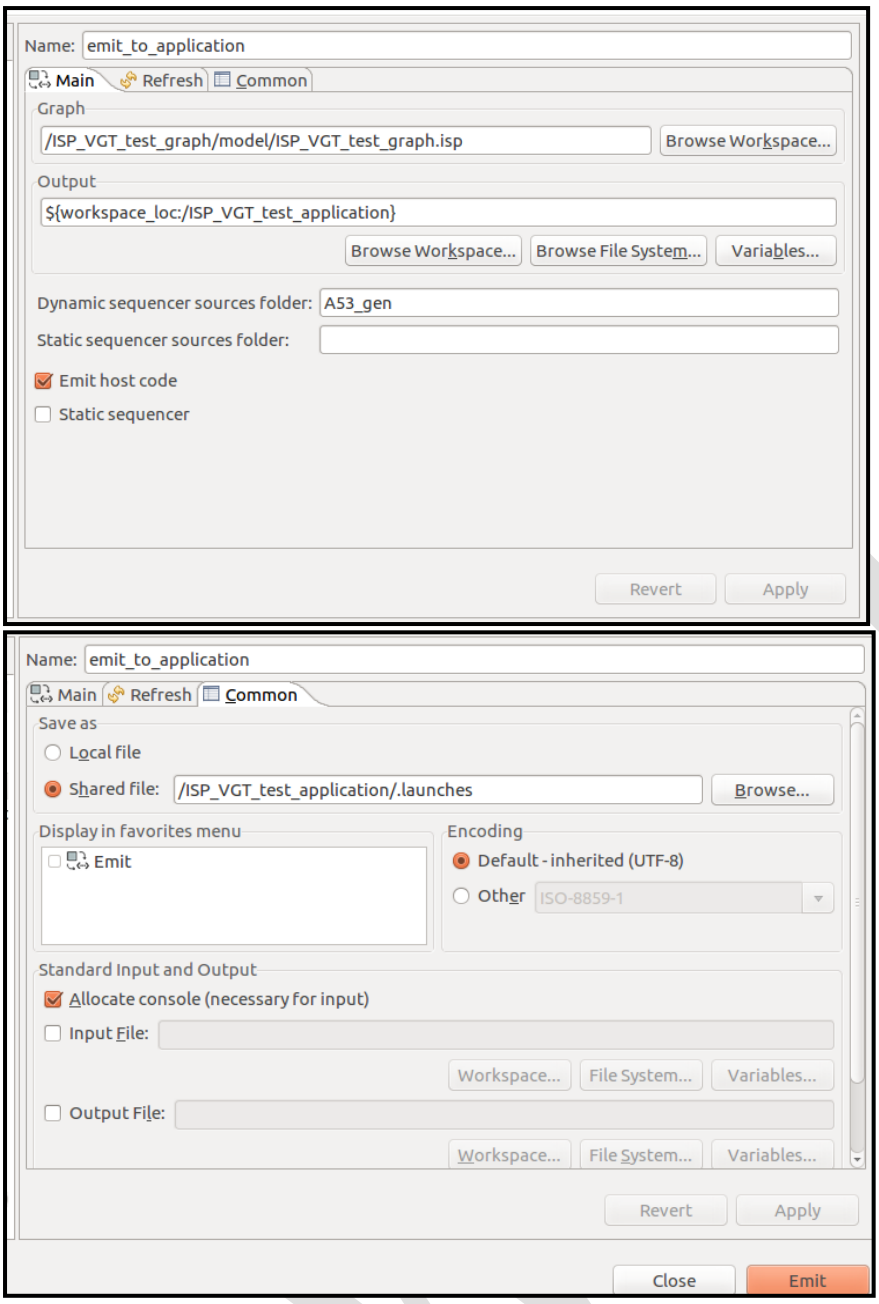
① 选择 S32V234 Cortex-A53 APEX2/ISP Linux ， Project name 可以随便起



② 在下面选项中取消以下画出的红框中的 √



③ 在 Graph 空白处右击进入 Emit Configure ， 选项配置如下：



④ 点击 Emit ， 完成工程搭建

3.3.3 Application Code for ISP

① 修改 A53_inc/isp_user_define.h

#define DCU_BPP DCU_BPP_24

~~#define DCU_BPP DCU_BPP_YCbCr422~~

② 增加 #include "isp_vgt_test_graph_c.h"

③ 对下面函数进行修改

```
void io_config(sdi_grabber *lpGrabber)
{
    /* Insert the code to initialize DDR buffers */
    // *** prepare IOs ***
    sdi_FdmaIO *lpFdma = (sdi_FdmaIO*)lpGrabber->IoGet(SEQ_OTHRIX_FDMA);
    // modify DDR frame geometry to fit display output
    SDI_ImageDescriptor lFrmDesc = SDI_ImageDescriptor(WIDTH_DDR, HEIGHT_DDR, RGB888);
    lpFdma->DdrBufferDescSet(FDMA_IX_FDMA_0, lFrmDesc);
    //*** allocate DDR buffers ***
    lpFdma->DdrBuffersAlloc(FDMA_IX_FDMA_0, DDR_OUT_BUFFER_CNT);
}
```

3.3.4 Build Project

正常执行后会生成 .elf 可执行文件

如果报错如下图

Description	Resource	Path
Errors (9 items)		
'GraphMetadata_t' does not name a type	isp_vgt_test_graph_c.h	/ISP_VGT_test_application/A53_gen/inc
'SEQ_Buf_t' does not name a type	isp_vgt_test_graph_c.h	/ISP_VGT_test_application/A53_gen/inc
'SEQ_Buf_t' does not name a type	isp_vgt_test_graph_c.h	/ISP_VGT_test_application/A53_gen/inc
'SEQ_Buf_t' does not name a type	isp_vgt_test_graph_c.h	/ISP_VGT_test_application/A53_gen/inc
'SEQ_Head_Ptr_t' does not name a type	isp_vgt_test_graph_c.h	/ISP_VGT_test_application/A53_gen/inc
make: *** [all] Error 2	ISP_VGT_test_application	
make[1]: *** [A53_src/main.o] Error 1	ISP_VGT_test_application	
recipe for target 'A53_src/main.o' failed	subdir.mk	/ISP_VGT_test_application/A53/A53_src
recipe for target 'all' failed	makefile	/ISP_VGT_test_application/A53
Warnings (1 item)		
string length '80524' is greater than the length '	isp_sonyimx224_csi_dcu_test3	

可以在 isp_vgt_test_graph_c.h 中添加下面两个头文件：

#include "seq_graph.h"

#include "seq_graph_meta.h"

3.4 总结

上面列出了三种利用 NXP 提供的 DS 工具建立 ISP 工程的方式，其中比较主要的是第三种，自主建立 ISP 工程的方式，其他两种主要是做了解为主，以及更好地熟悉 DS 工具。后面，会开始介绍在设备节点的应用开发上，主要是对 UART1 的使用以及配置，敬请期待。

4. 参考资料

博文操作都是参考 NXP 的 DS 软件操作手册。

[1] 《 HOWTO_Create_An_ISP_Project_From_Example_in_S32DS_for_Vision 》操作手册

[2] 《 HOWTO_Create_An_ISP_Project_From_Existing_VSDK_Graph_in_S32DS_for_Vision 》操作手册

[3] 《 ISP_graph_tool_in_depth_tutorial 》操作手册

[4] 《 S32DS_Vision_User_Guide 》操作手册