



# **ZCU104(MPSoC) System Monitor Usage with Vitis**

**Field Application Engineer**

Adaptive and Embedded Computing Group (AECG)

# Revision History

Date	Version	Description
02/06/24	1.0	Initial version for flow introduction.

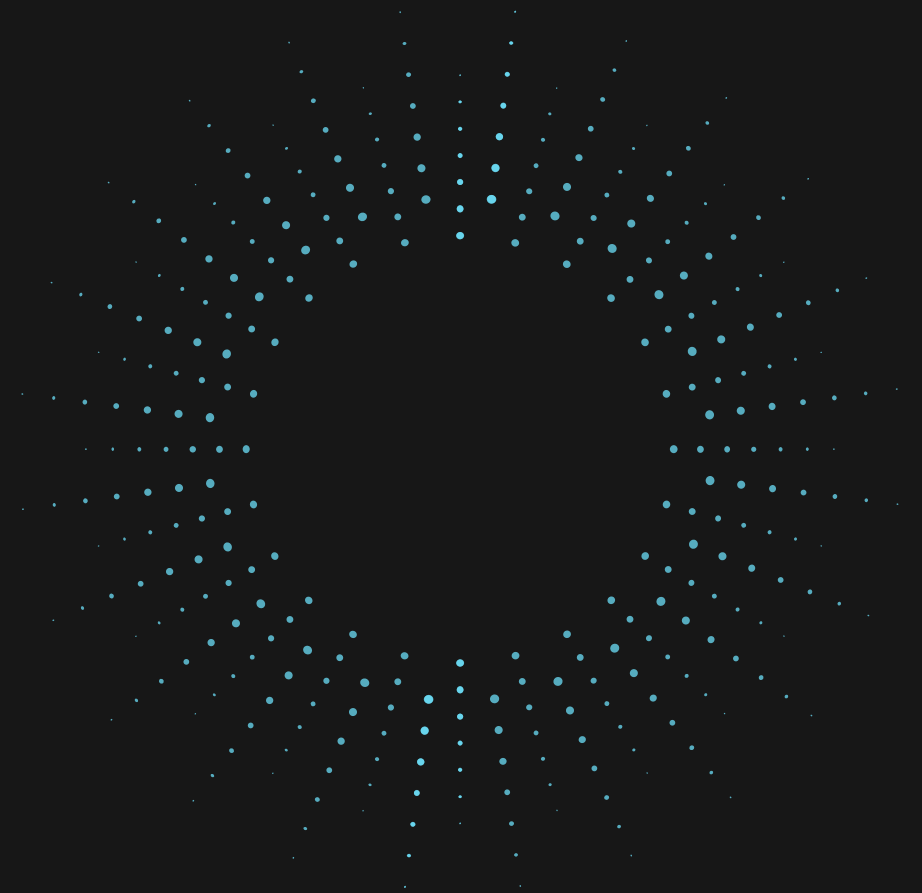
© Copyright 2021 Xilinx, Inc. Xilinx, the Xilinx logo, Artix, ISE, Kintex, Spartan, Virtex, Vivado, Zynq, and other designated brands included herein are trademarks of Xilinx in the United States and other countries. All other trademarks are the property of their respective owners.

NOTICE OF DISCLAIMER: The information disclosed to you hereunder (the "Information") is provided "AS-IS" with no warranty of any kind, express or implied. Xilinx does not assume any liability arising from your use of the Information. You are responsible for obtaining any rights you may require for your use of this Information. Xilinx reserves the right to make changes, at any time, to the Information without notice and at its sole discretion. Xilinx assumes no obligation to correct any errors contained in the Information or to advise you of any corrections or updates. Xilinx expressly disclaims any liability in connection with technical support or assistance that may be provided to you in connection with the Information. XILINX MAKES NO OTHER WARRANTIES, WHETHER EXPRESS, IMPLIED, OR STATUTORY, REGARDING THE INFORMATION, INCLUDING ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NONINFRINGEMENT OF THIRD-PARTY RIGHTS.

---

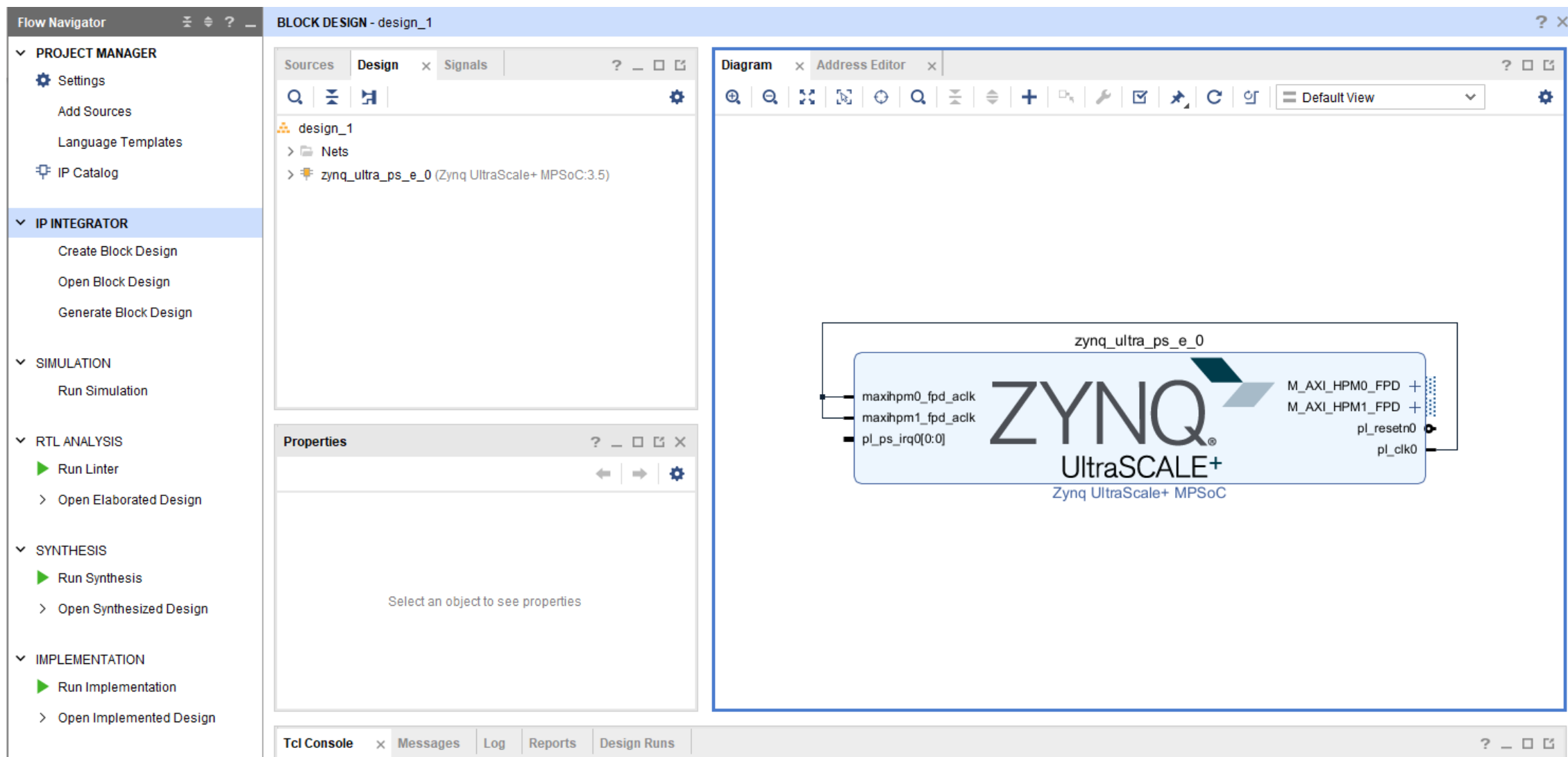
# Vivado 2023.2 Part

---



# ZCU104(MPSoC) System Monitor Usage with Vitis

叫出 ZYNQ MPSoC IP 就好，我們要調用 PS 端而已



# ZCU104(MPSoC) System Monitor Usage with Vitis

## Block Design Steps

The screenshot displays the Vitis IDE interface. On the left, the 'Sources' panel shows a project hierarchy with 'design\_1 (design\_1.bd)' selected. A context menu is open over this file, listing actions such as 'Source Node Properties...', 'Open File', 'Create HDL Wrapper...', 'View Instantiation Template', 'Generate Output Products...', 'Reset Output Products...', 'Replace File...', 'Copy File Into Project', and 'Copy All Files Into Project'. The 'Create HDL Wrapper...' option is highlighted. Below the 'Sources' panel, the 'Hierarchy' tab is active, showing 'IP Sources' and 'Libraries'. On the right, the 'Diagram' panel is visible. At the bottom, a console window shows the command: `add_files -fileset constrs_1 -norecurse C:/BIST/XVES_0019/src/constr/ZC702.xdc`. A dialog box titled 'No Implementation Results Available' is overlaid on the right side of the IDE. The dialog contains a question mark icon and the text: 'There are no implementation results available. OK to launch synthesis and implementation? 'Generate Bitstream' will automatically start when synthesis and implementation completes.' Below this text is a checkbox labeled 'Don't show this dialog again'. At the bottom right of the dialog are 'Yes' and 'No' buttons. The 'Yes' button is highlighted in green.

**IMPLEMENTATION**

- Run Implementation
- Open Implemented Design

**PROGRAM AND DEBUG**

- Generate Bitstream**
- Open Hardware Manager

**No Implementation Results Available**

There are no implementation results available. OK to launch synthesis and implementation? 'Generate Bitstream' will automatically start when synthesis and implementation completes.

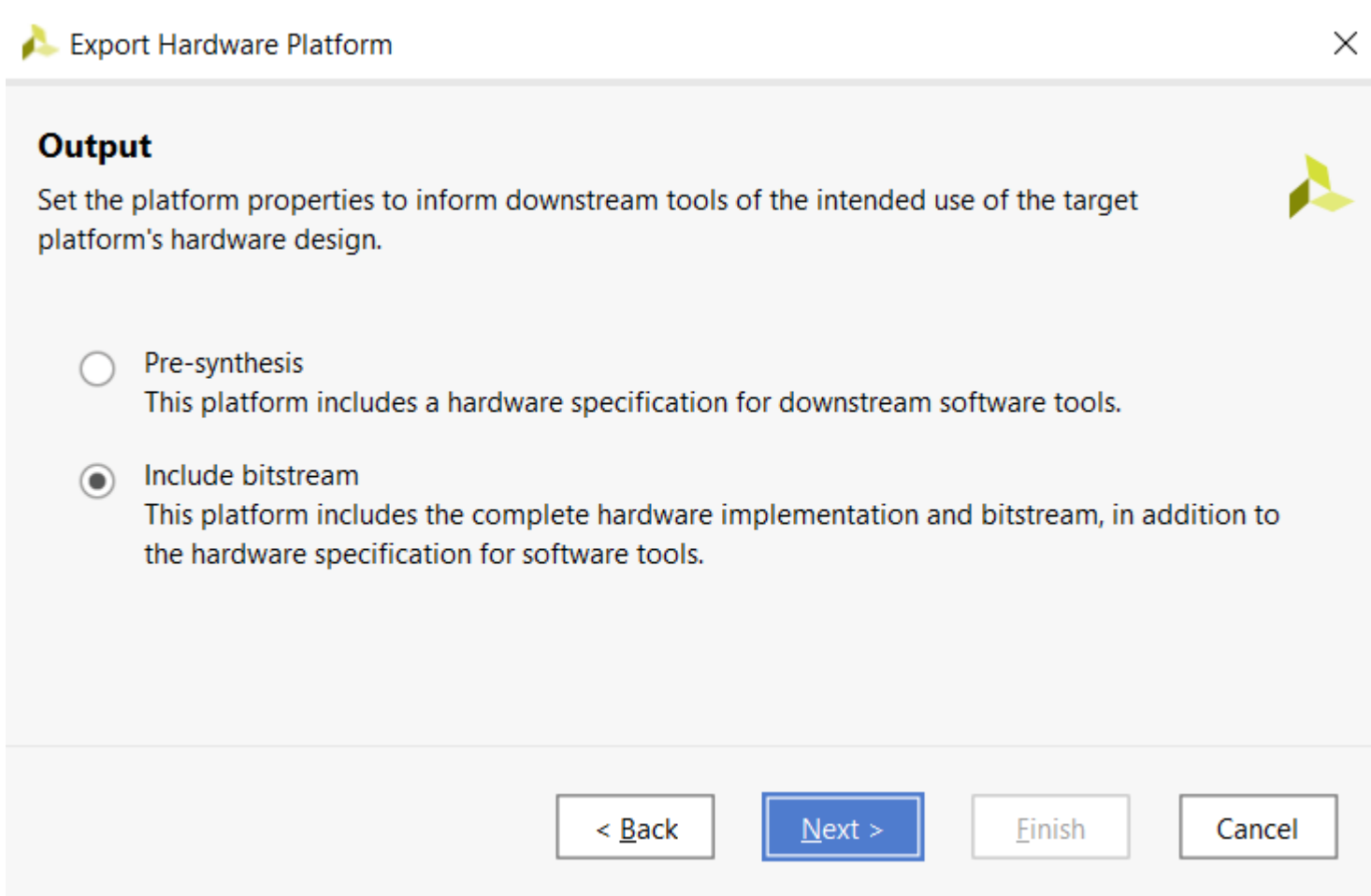
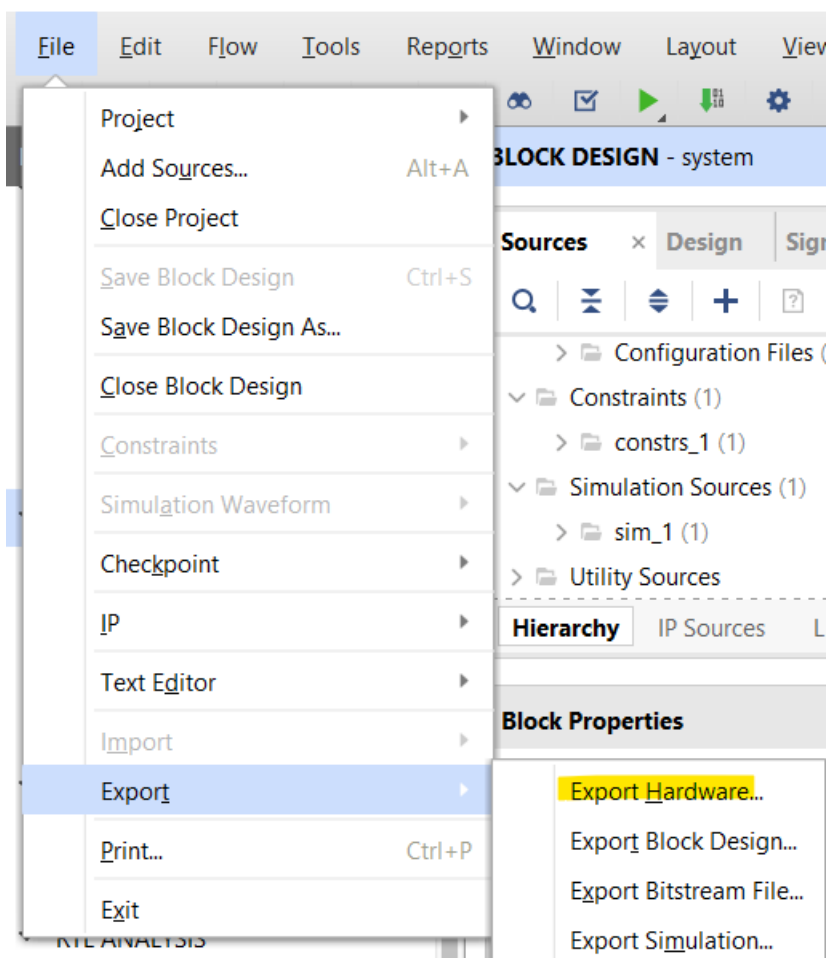
☐ Don't show this dialog again

Yes No



**THIRTY-SEVEN  
MINUTES  
LATER...**

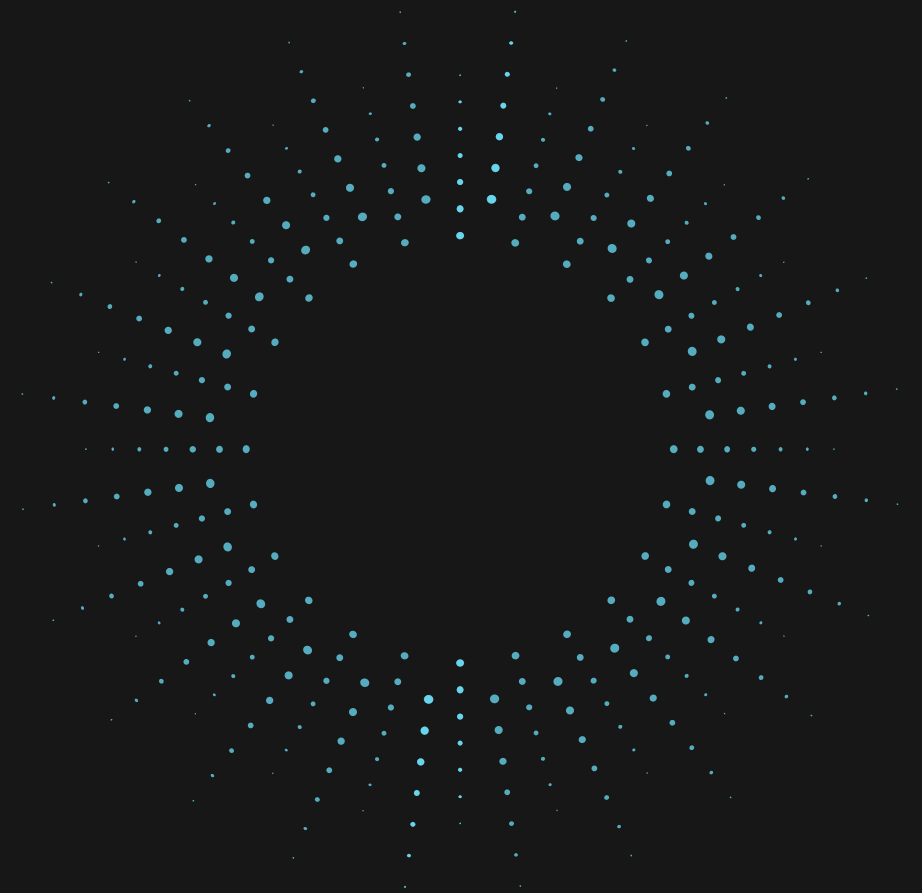
# ZCU104(MPSoC) System Monitor Usage with Vitis



---

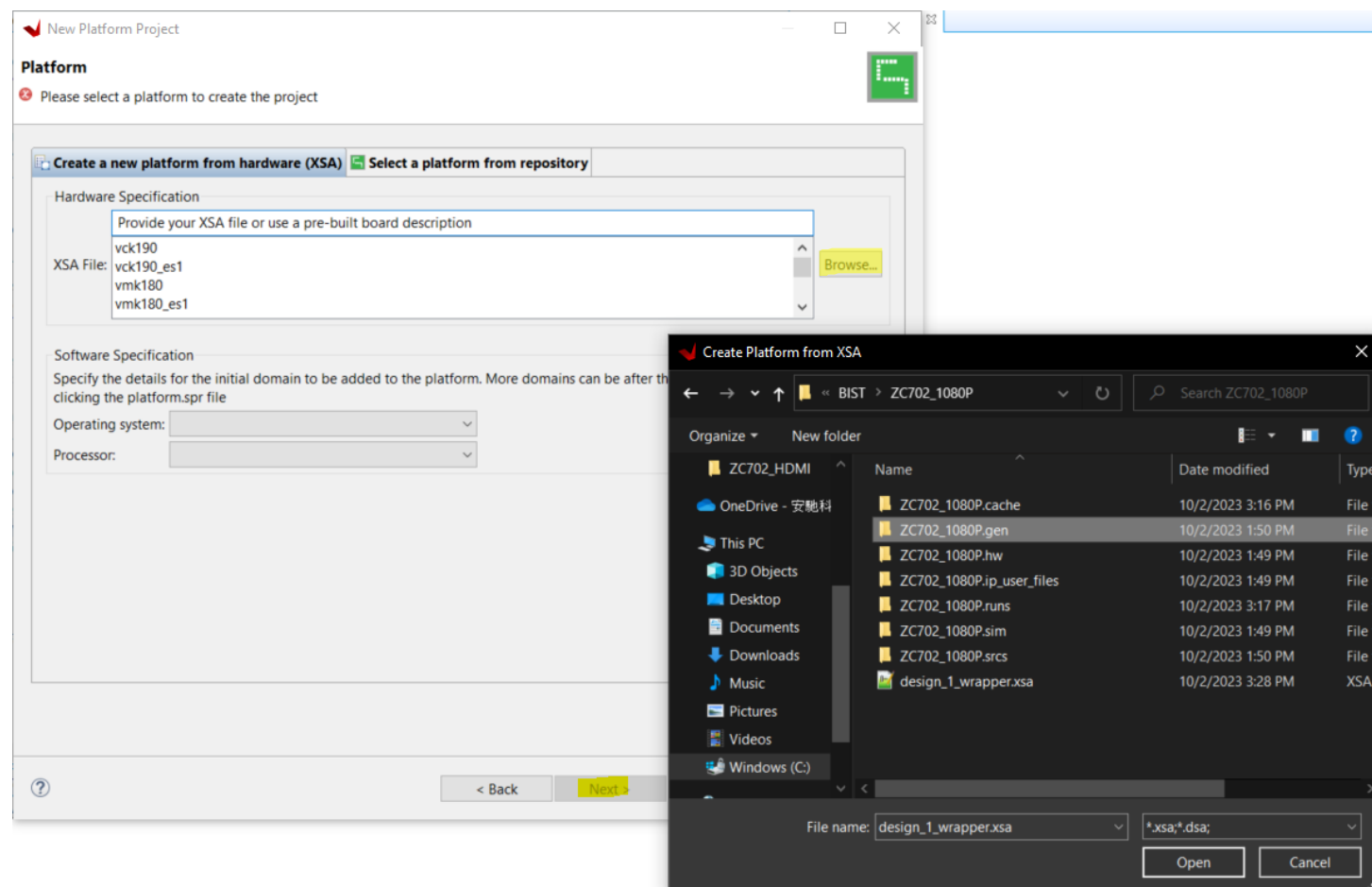
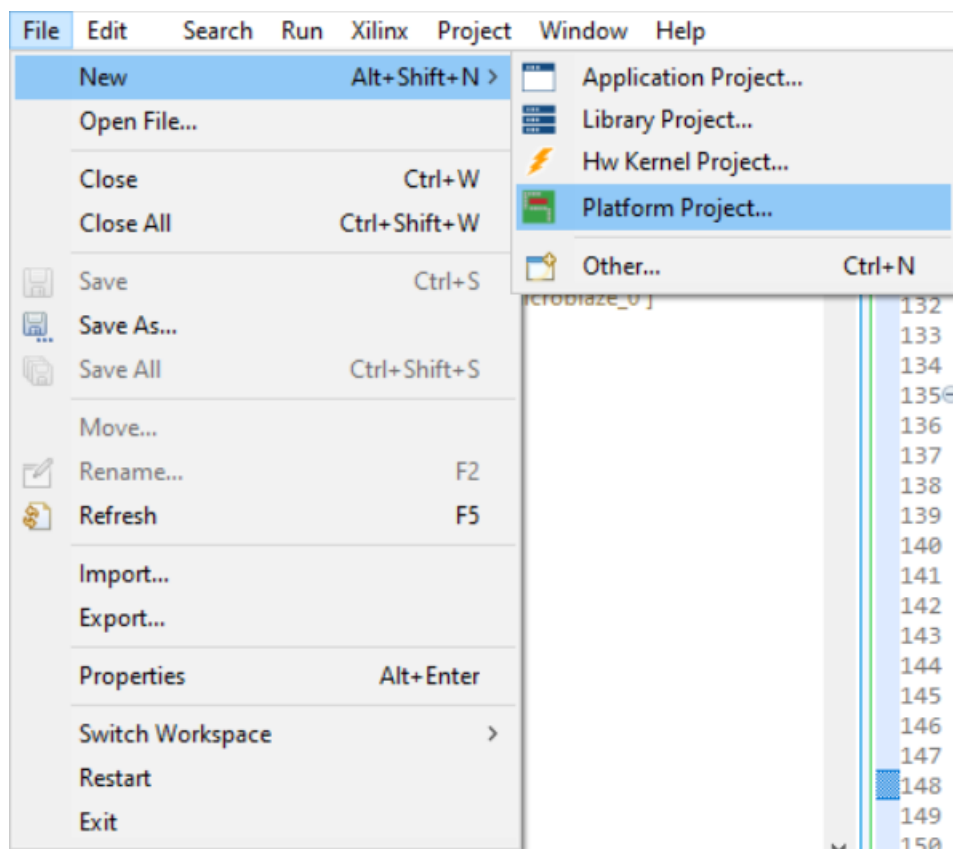
# Vitis 2023.2 Part

---





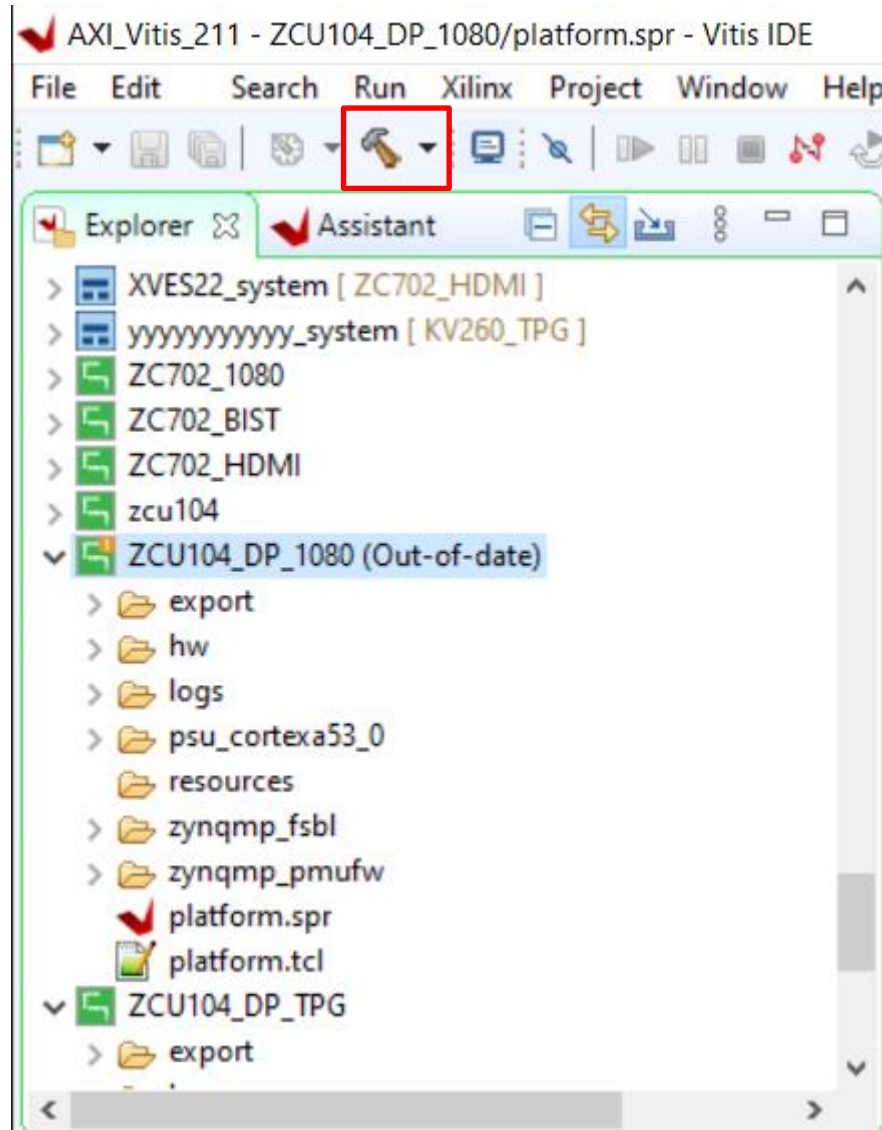
# ZCU104(MPSoC) System Monitor Usage with Vitis



\*\*\* 實際檔案請按照自己設定的位置與名稱去開啟

# ZCU104(MPSoC) System Monitor Usage with Vitis

記得 **Build**



# ZCU104(MPSoC) System Monitor Usage with Vitis

抓 PS SYSMON 的 example code 下來改

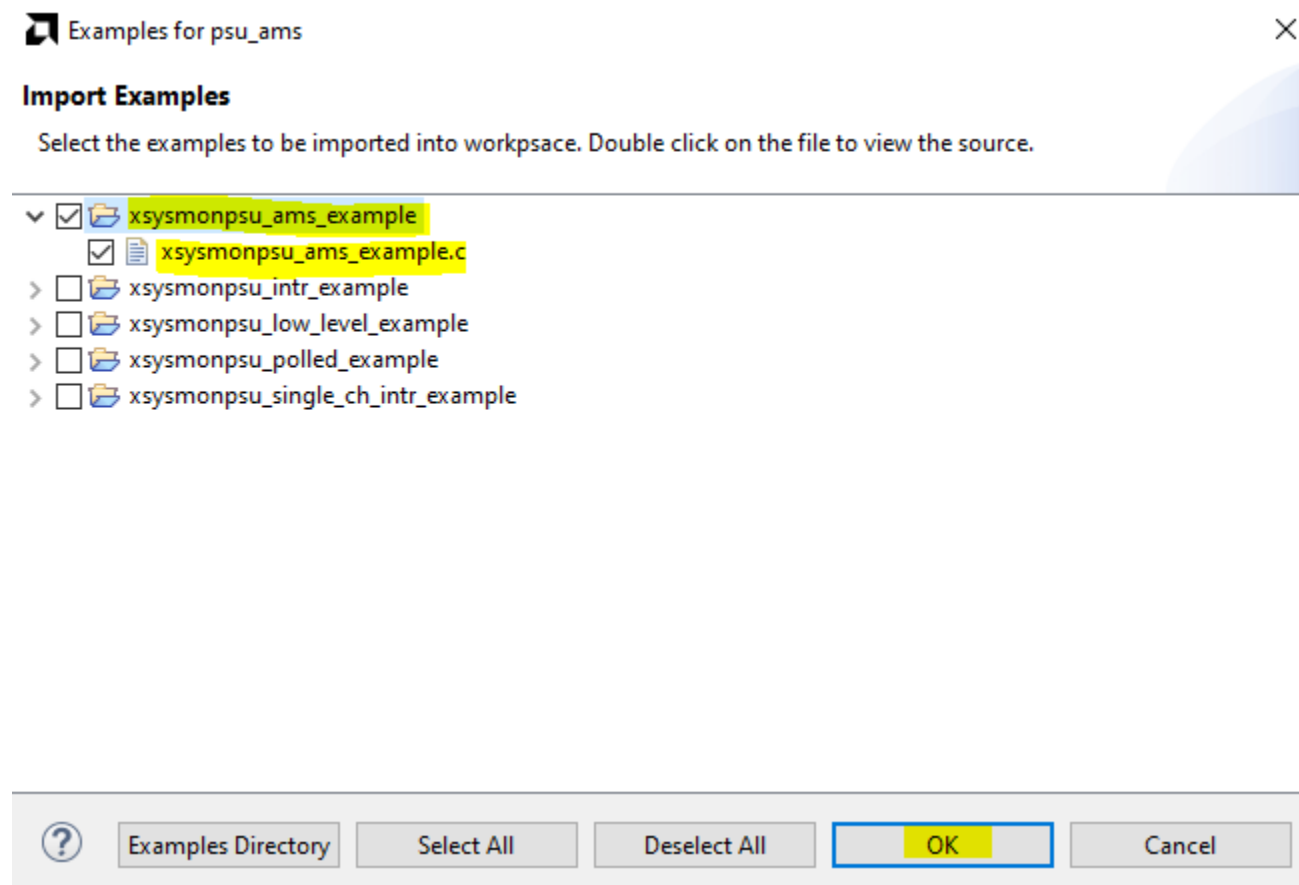
The screenshot displays the Vitis IDE interface with the following components:

- Explorer:** Shows the project hierarchy. The **ZCU104\_SD\_SYSMON\_Platform** project is selected, showing its subdirectories: **export**, **hw**, **logs**, **psu\_cortexa53\_0**, **resources**, **zynqmp\_fsbl**, **zynqmp\_pmufw**, **platform.spr**, and **platform.tcl**.
- Assistant:** Shows the build configuration for the **KD240\_PWM\_system** application, with the **Debug** configuration selected.
- Board Support Package:** The **Board Support Package** is selected in the **psu\_cortexa53\_0** directory. The configuration page shows the **Name** as **standalone** and the **Version** as **9.0**. The description states: "Standalone is a simple, low-level software layer. It provides access to basic processor features such as caches, interrupts and exceptions as well as the basic features of a hosted environment, such as standard input and output, profiling, abort and exit." The **Documentation** link is provided.
- Drivers Table:** A table listing available drivers and their configurations.

Name	Driver	Documentation	Examples
psu_afi_3	generic	-	-
psu_afi_4	generic	-	-
psu_afi_5	generic	-	-
psu_afi_6	generic	-	-
psu_ams	sysmonpsu	<a href="#">Documentation Link</a>	<a href="#">Import Examples</a>
psu_apm_0	axipmon	<a href="#">Documentation Link</a>	<a href="#">Import Examples</a>
psu_apm_1	axipmon	<a href="#">Documentation Link</a>	<a href="#">Import Examples</a>
psu_apm_2	axipmon	<a href="#">Documentation Link</a>	<a href="#">Import Examples</a>
psu_apm_5	axipmon	<a href="#">Documentation Link</a>	<a href="#">Import Examples</a>
psu_apu	generic	-	-
psu_can_1	canps	<a href="#">Documentation Link</a>	<a href="#">Import Examples</a>
psu_cci_gpv	generic	-	-

# ZCU104(MPSoC) System Monitor Usage with Vitis

抓 PS SYSMON 的 example code 下來改



# ZCU104(MPSoC) System Monitor Usage with Vitis

主要修改 SysMonPsuAMSEExample 這個 Function

```
int SysMonPsuAMSEExample(XScuGic* XScuGicInstancePtr,  
                        XSysMonPsu* SysMonInstPtr,  
                        u32 SysMonDeviceId,  
                        u16 SysMonIntrId)  
{  
    int Status;  
    XSysMonPsu_Config *ConfigPtr;  
  
    s32 TempRawData;  
    u32 VccPSIO1RawData;  
    u32 VccAuxRawData;  
    u32 VccPSDDRRawData;  
    u32 VccPSINTLPRawData;  
    u32 PLVPVNRawData;  
  
    int pass_time;  
    int days;  
    int hours;  
    int minutes;  
    int seconds;  
  
    float TempData;  
    float VccPSIO1Data;  
    float VccAuxData;  
    float VccPSDDRData;  
    float VccPSINTLPData;  
    float PLVPVNData;  
  
    XTime tEnd, tStart;  
    u64 IntrStatus;  
  
    /* Initialize the SysMon driver. */  
    ConfigPtr = XSysMonPsu_LookupConfig(SysMonDeviceId);  
    if (ConfigPtr == NULL) {  
        return XST_FAILURE;  
    }  
}
```

# ZCU104(MPSoC) System Monitor Usage with Vitis

當然跟使用一般 IP 一樣，要先 instance SysMon，Configure SysMon，Get SysMon DeviceID 等等

```
#include "xsysmonpsu.h"
```

```
#ifndef SDT
#define SYSMON_DEVICE_ID    XPAR_XSYSMONPSU_0_DEVICE_ID
#else
#define SYSMON_DEVICE_ID    0xffa50000

```

➡ 設定 Device ID

```
...
static XSysMonPsu SysMonInst;          /* System Monitor driver instance */
...
int SysMonPsuAMSExample(XScuGic* XScuGicInstancePtr,
                        XSysMonPsu* SysMonInstPtr,
                        u32 SysMonDeviceId,
                        u16 SysMonIntrId)
{

```

```
    XSysMonPsu_Config *ConfigPtr;

    /* Initialize the SysMon driver. */
    ConfigPtr = XSysMonPsu_LookupConfig(SysMonDeviceId);
    if (ConfigPtr == NULL) {
        return XST_FAILURE;
    }
    XSysMonPsu_CfgInitialize(SysMonInstPtr, ConfigPtr, ConfigPtr->BaseAddress);

```

➡ 抓 Device ID，然後 Config

```
    /* Self Test the System Monitor device. */
    Status = XSysMonPsu_SelfTest(SysMonInstPtr);
    if (Status != XST_SUCCESS) {
        return XST_FAILURE;
    }

```

➡ 先測看看 SYSMON 會不會通

```
    /* Clear any bits set in the Interrupt Status Register. */
    IntrStatus = XSysMonPsu_IntrGetStatus(SysMonInstPtr);
    XSysMonPsu_IntrClear(SysMonInstPtr, IntrStatus);

    /* Set the sequencer in Single channel mode. */
    XSysMonPsu_SetSequencerMode(SysMonInstPtr, XSM_SEQ_MODE_SINGCHAN, XSYSMON_PS);
}

```

➡ 清掉 interrupt register 以進行下一個選項的觀測，每次只觀測一個選項

# ZCU104(MPSoC) System Monitor Usage with Vitis

以 VCC\_PSBATT 為例

```
/*  
 * Set the configuration registers for single channel continuous mode  
 * of operation for the VCC_PSBATT channel.  
 */
```

```
Status= XSysMonPsu_SetSingleChParams(SysMonInstPtr, XSM_CH_RESERVE1,  
                                     FALSE, FALSE, FALSE, XSYSMON_PS);  
if(Status != XST_SUCCESS) {  
    return XST_FAILURE;  
}
```

➡ 設定要觀測的項目，可以從兩個參數去做參考

```
XSysmonPsu_Poll_timeout(SysMonInstPtr->Config.BaseAddress +  
                        XSYSMONPSU_ISR_1_OFFSET, &IntrStatus,  
                        (IntrStatus & XSYSMONPSU_ISR_1_EOC_MASK) == XSYSMONPSU_ISR_1_EOC_MASK,  
                        EOC_POLLING_TIMEOUT);  
  
xil_printf("\r\n8. EOC: %s , VCC_PSINTFP_DDR: ", (IntrStatus & 0x8) ? "Done" : "Timeout");
```

➡ 測試有沒有 Timeout，可以省略

```
VccPsIntFpRawData = XSysMonPsu_GetAdcData(SysMonInstPtr, XSM_CH_RESERVE1, XSYSMON_AMS);  
if (VccPsIntFpRawData == 0U)  
    return XST_FAILURE;
```

➡ 抓 VCC\_PSBATT 在 AMS 上的 RawData

```
VccPsIntFpData = XSysMonPsu_RawToVoltage(VccPsIntFpRawData);  
xil_printf("%0d.%03d Volts\r\n", (int)(VccPsIntFpData), SysMonPsuFractionToInt(VccPsIntFpData));
```

➡ 把 RawData 轉成電壓值

```
/* Set the sequencer in Single channel mode. */  
XSysMonPsu_SetSequencerMode(SysMonInstPtr, XSM_SEQ_MODE_SINGCHAN, XSYSMON_PS);  
  
/* Clear any bits set in the Interrupt Status Register. */  
IntrStatus = XSysMonPsu_IntrGetStatus(SysMonInstPtr);  
XSysMonPsu_IntrClear(SysMonInstPtr, IntrStatus);
```



# ZCU104(MPSoC) System Monitor Usage with Vitis

以 VCC\_PSBATT 為例

```
/*
 * Set the configuration registers for single channel continuous mode
 * of operation for the VCC_PSBATT channel.
 */
Status= XSysMonPsu_SetSingleChParams(SysMonInstPtr, XSM_CH_RESERVE1,
                                     FALSE, FALSE, FALSE, XSYSMON_PS);
if(Status != XST_SUCCESS) {
    return XST_FAILURE;
}

XSysMonPsu_Poll_timeout(SysMonInstPtr->Config.BaseAddress +
                       XSYSMONPSU_ISR_1_OFFSET, &IntrStatus,
                       (IntrStatus & XSYSMONPSU_ISR_1_EOC_MASK) == XSYSMONPSU_ISR_1_EOC_MASK);

xil_printf("\r\n8. EOC: %s , VCC_PSINTFP_DDR: ", (IntrStatus & 0x8) ? "Done" : "Timeout");

VccPsIntFpRawData = XSysMonPsu_GetAdcData(SysMonInstPtr, XSM_CH_RESERVE1, XSYSMON_AMS);
if (VccPsIntFpRawData == 0U)
    return XST_FAILURE;

VccPsIntFpData = XSysMonPsu_RawToVoltage(VccPsIntFpRawData);
xil_printf("%0d.%03d Volts\r\n", (int)(VccPsIntFpData), SysMonPsuFractionToInt(VccPsIntFpData));

/* Set the sequencer in Single channel mode. */
XSysMonPsu_SetSequencerMode(SysMonInstPtr, XSM_SEQ_MODE_SINGCHAN, XSYSMON_PS);

/* Clear any bits set in the Interrupt Status Register. */
IntrStatus = XSysMonPsu_IntrGetStatus(SysMonInstPtr);
XSysMonPsu_IntrClear(SysMonInstPtr, IntrStatus);
```

```
/* BaseAddress Offsets */
#define XSYSMON_PS 1U
#define XSYSMON_PL 2U
#define XSYSMON_AMS 3U
#define XPS_BA_OFFSET 0x00000800U
#define XPL_BA_OFFSET 0x00000C00U
#define XSM_ADC_CH_OFFSET 0x00000200U
#define XSM_AMS_CH_OFFSET 0x00000600U
#define XSM_MIN_MAX_CH_OFFSET 0x00000800U
```

```
/**
 * @name Indexes for the different channels.
 * @{
 */
#define XSM_CH_TEMP 0x0U /**< On Chip Temperature */
#define XSM_CH_SUPPLY1 0x1U /**< SUPPLY1 VCC_PSINTLP */
#define XSM_CH_SUPPLY2 0x2U /**< SUPPLY2 VCC_PSINTFP */
#define XSM_CH_VPVN 0x3U /**< VP/VN Dedicated analog inputs */
#define XSM_CH_VREFP 0x4U /**< VREFP */
#define XSM_CH_VREFN 0x5U /**< VREFN */
#define XSM_CH_SUPPLY3 0x6U /**< SUPPLY3 VCC_PSAUX */
#define XSM_CH_SUPPLY_CALIB 0x08U /**< Supply Calib Data Reg */
#define XSM_CH_ADC_CALIB 0x09U /**< ADC Offset Channel Reg */
#define XSM_CH_GAINERR_CALIB 0x0AU /**< Gain Error Channel Reg */
#define XSM_CH_SUPPLY4 0x0DU /**< SUPPLY4 VCC_PSDDR_504 */
#define XSM_CH_SUPPLY5 0x0EU /**< SUPPLY5 VCC_PSI03_503 */
#define XSM_CH_SUPPLY6 0x0FU /**< SUPPLY6 VCC_PSI00_500 */
#define XSM_CH_AUX_MIN 16U /**< Channel number for 1st Aux Channel */
#define XSM_CH_AUX_MAX 31U /**< Channel number for Last Aux channel */
#define XSM_CH_SUPPLY7 32U /**< SUPPLY7 VCC_PSI01_501 */
#define XSM_CH_SUPPLY8 33U /**< SUPPLY8 VCC_PSI02_502 */
#define XSM_CH_SUPPLY9 34U /**< SUPPLY9 PS_MGTRAVCC */
#define XSM_CH_SUPPLY10 35U /**< SUPPLY10 PS_MGTRAVTT */
#define XSM_CH_VCCAMS 36U /**< VCCAMS */
#define XSM_CH_TEMP_REMTE 37U /**< Temperature Remote */
#define XSM_CH_VCC_PSLL0 48U /**< VCC_PSLL0 */
#define XSM_CH_VCC_PSLL3 51U /**< VCC_PSLL3 */
#define XSM_CH_VCCINT 54U /**< VCCINT */
#define XSM_CH_VCCBRAM 55U /**< VCCBRAM */
#define XSM_CH_VCCAUX 56U /**< VCCAUX */
#define XSM_CH_VCC_PSDDRPLL 57U /**< VCC_PSDDRPLL */
#define XSM_CH_DDRPHY_VREF 58U /**< DDRPHY_VREF */
#define XSM_CH_RESERVE1 63U /**< PSGT_AT0 */
```



# ZCU104(MPSoC) System Monitor Usage with Vitis

以溫度為例

```
//Temperature
Status= XSysMonPsu_SetSingleChParams(SysMonInstPtr, XSM_CH_TEMP, FALSE, FALSE, FALSE, XSYSMON_PS);

TempRawData = XSysMonPsu_GetAdcData(SysMonInstPtr, XSM_CH_TEMP, XSYSMON_PS);
if (TempRawData == 0U)
    return XST_FAILURE;

TempData = XSysMonPsu_RawToTemperature_OnChip(TempRawData);

/* Set the sequencer in Single channel mode. */
XSysMonPsu_SetSequencerMode(SysMonInstPtr, XSM_SEQ_MODE_SINGCHAN, XSYSMON_PS);

/* Clear any bits set in the Interrupt Status Register. */
IntrStatus = XSysMonPsu_IntrGetStatus(SysMonInstPtr);
XSysMonPsu_IntrClear(SysMonInstPtr, IntrStatus);
```

➡ 把 RawData 轉成溫度值

# ZCU104(MPSoC) System Monitor Usage with Vitis

因此可以透過以上流程和 **example code**，將自己想要觀測的照參數填寫進去，最後再 **xil\_printf** 出來

```
//PSI01_Data  
Status= XSysMonPsu_SetSingleChParams(SysMonInstPtr, XSM_CH_SUPPLY7, FALSE, FALSE, FALSE, XSYSMON_PS);
```

```
VccPSI01RawData = XSysMonPsu_GetAdcData(SysMonInstPtr, XSM_CH_SUPPLY7, XSYSMON_PS);  
if (VccPSI01RawData == 0U)  
    return XST_FAILURE;
```

```
VccPSI01Data = XSysMonPsu_RawToVoltage(VccPSI01RawData);
```

```
/* Set the sequencer in Single channel mode. */  
XSysMonPsu_SetSequencerMode(SysMonInstPtr, XSM_SEQ_MODE_SINGCHAN, XSYSMON_PS);
```

```
/* Clear any bits set in the Interrupt Status Register. */  
IntrStatus = XSysMonPsu_IntrGetStatus(SysMonInstPtr);  
XSysMonPsu_IntrClear(SysMonInstPtr, IntrStatus);
```


```
/*  
 * Set the configuration registers for single channel continuous mode  
 * of operation for the VCC_PSBATT channel.  
 */  
Status= XSysMonPsu_SetSingleChParams(SysMonInstPtr, XSM_CH_VCCAUX,  
                                     FALSE, FALSE, FALSE, XSYSMON_PS);
```

```
VccAuxRawData = XSysMonPsu_GetAdcData(SysMonInstPtr, XSM_CH_VCCAUX, XSYSMON_PS);  
if (VccAuxRawData == 0U)  
    return XST_FAILURE;
```

```
VccAuxData = XSysMonPsu_RawToVoltage(VccAuxRawData);
```

```
/* Set the sequencer in Single channel mode. */  
XSysMonPsu_SetSequencerMode(SysMonInstPtr, XSM_SEQ_MODE_SINGCHAN, XSYSMON_PS);
```

```
/* Clear any bits set in the Interrupt Status Register. */  
IntrStatus = XSysMonPsu_IntrGetStatus(SysMonInstPtr);  
XSysMonPsu_IntrClear(SysMonInstPtr, IntrStatus);
```



```
xil_printf("=====\n\n"  
          "Current Temperature: %0d.%03d Centigrades\n\n"  
          "PSI01: %0d.%03d Volts\n\n"  
          "VCCAUX: %0d.%03d Volts\n\n"  
          "VccPSDDR: %0d.%03d Volts\n\n"  
          "VccPSINTLP: %0d.%03d Volts\n\n"  
          "PLVPVN: %0d.%03d Volts\n\n",  
          (int)(TempData), SysMonPsuFractionToInt(TempData), (int)(VccPSI01Data),  
          SysMonPsuFractionToInt(VccPSI01Data),  
          (int)(VccAuxData), SysMonPsuFractionToInt(VccAuxData),  
          (int)(VccPSDDRData), SysMonPsuFractionToInt(VccPSDDRData),  
          (int)(VccPSINTLPData), SysMonPsuFractionToInt(VccPSINTLPData),  
          (int)(PLVPVNData), SysMonPsuFractionToInt(PLVPVNData));
```

# ZCU104(MPSoC) System Monitor Usage with Vitis

寫法二 - 第一種寫法在長時間執行會有 Crash 現象發生

```
ConfigPtr = XSysMonPsu_LookupConfig(SysMonDeviceId);

if (ConfigPtr == NULL) {
    return XST_FAILURE;
}

XSysMonPsu_CfgInitialize(SysMonInstPtr, ConfigPtr, ConfigPtr->BaseAddress);
```

➡ 一樣先做 initial 與 Config

```
XSysMonPsu_SetSequencerMode(SysMonInstPtr, XSM_SEQ_MODE_SAFE, XSYSMON_PS); //设置Sequence Mode为安全模式
```

```
XSysMonPsu_SetAlarmEnables(SysMonInstPtr, 0x0, XSYSMON_PS); //关闭寄存器1中指定信号的警报
```

```
XSysMonPsu_SetAvg(SysMonInstPtr, XSM_AVG_16_SAMPLES, XSYSMON_PS); //设置采样16次后计算平均值
```

```
XSysMonPsu_SetSeqAvgEnables(SysMonInstPtr,
    XSYSMONPSU_SEQ_CH0_TEMP_MASK | //使能Temp_LPD通道平均值测量
    XSYSMONPSU_SEQ_CH2_SUP7_MASK | //使能VCCO_PSI01通道平均值测量
    XSYSMONPSU_SEQ_CH0_SUP3_MASK | //使能VCC_PSAUX通道平均值测量
    XSYSMONPSU_SEQ_CH0_SUP4_MASK | //使能VCC_PSDDR通道平均值测量
    XSYSMONPSU_SEQ_CH0_SUP1_MASK | //使能VCC_PSINTLP通道平均值测量
    XSYSMONPSU_SEQ_CH0_VP_VN_MASK, //使能PL_VPVN通道平均值测量
    XSYSMON_PS);
```

```
XSysMonPsu_SetSeqChEnables(SysMonInstPtr,
    XSYSMONPSU_SEQ_CH0_TEMP_MASK | //使能Temp_LPD通道平均值测量
    XSYSMONPSU_SEQ_CH2_SUP7_MASK | //使能VCCO_PSI01通道平均值测量
    XSYSMONPSU_SEQ_CH0_SUP3_MASK | //使能VCC_PSAUX通道平均值测量
    XSYSMONPSU_SEQ_CH0_SUP4_MASK | //使能VCC_PSDDR通道平均值测量
    XSYSMONPSU_SEQ_CH0_SUP1_MASK | //使能VCC_PSINTLP通道平均值测量
    XSYSMONPSU_SEQ_CH0_VP_VN_MASK, //使能PL_VPVN通道平均值测量
    XSYSMON_PS);
```

```
IntrStatus = XSysMonPsu_IntrGetStatus(SysMonInstPtr); //读中断状态寄存器
```

```
XSysMonPsu_IntrClear(SysMonInstPtr, IntrStatus); //清除中断状态寄存器
```

```
XSysMonPsu_SetSequencerMode(SysMonInstPtr, XSM_SEQ_MODE_CONTINPASS, XSYSMON_PS); //设定Sequence Mode为通道循环模式
```

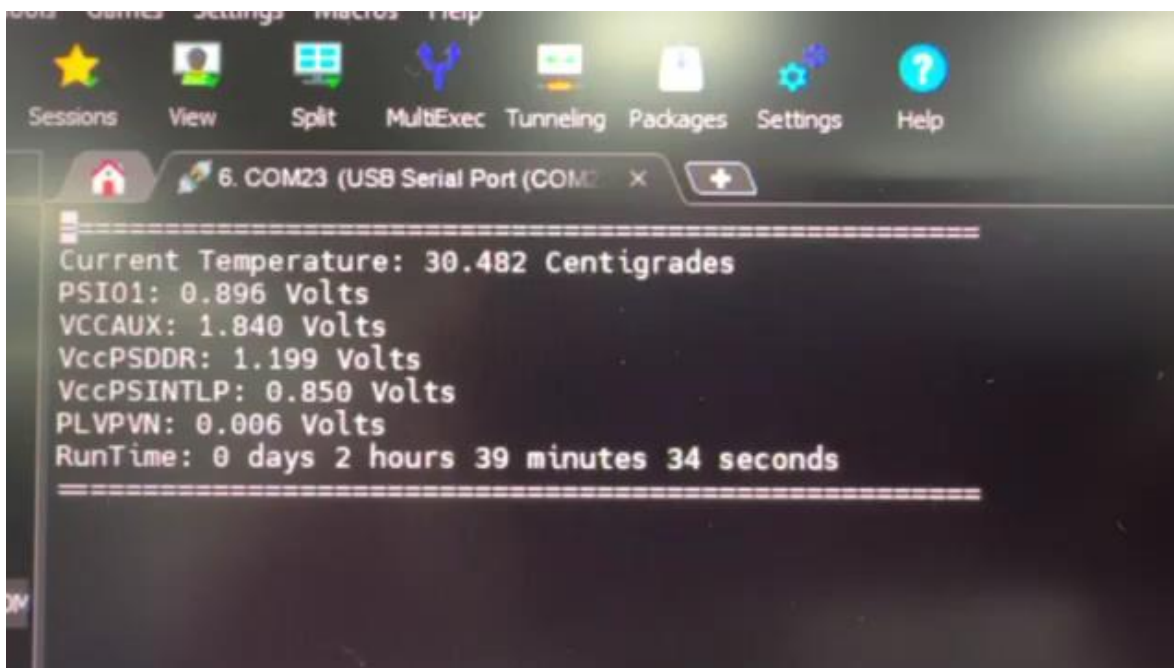
```
while ((XSysMonPsu_IntrGetStatus(SysMonInstPtr) & ((u64)XSYSMONPSU_ISR_1_EOS_MASK<<32))!= ((u64)XSYSMONPSU_ISR_1_EOS_MASK<< 32)); //等待EOS(end of sequence)发生
```

```
XTime_GetTime(&tStart);
```

# ZCU104(MPSoC) System Monitor Usage with Vitis

## Result

```
=====
Current Temperature: 29.953 Centigrades
PSI01: 0.896 Volts
VCCAUX: 1.840 Volts
VccPSDDR: 1.199 Volts
VccPSINTLP: 0.850 Volts
PLVPVN: 0.006 Volts
RunTime: 0 days 2 hours 39 minutes 0 secondss
=====
```



```
=====
Current Temperature: 30.482 Centigrades
PSI01: 0.896 Volts
VCCAUX: 1.840 Volts
VccPSDDR: 1.199 Volts
VccPSINTLP: 0.850 Volts
PLVPVN: 0.006 Volts
RunTime: 0 days 2 hours 39 minutes 34 seconds
=====
```

有加上運行時間顯示，使用 `XTime_GetTime`

```
#include "xtime_1.h"
#include "xil_io.h"

...
XTime tEnd, tStart;
...
XTime_GetTime(&tStart);
while(1){
    XTime_GetTime(&tEnd);
    ...
    pass_time = (int)((double)(tEnd - tStart) / (double)COUNTS_PER_SECOND);

    days = pass_time / (60 * 60 * 24);
    pass_time -= days * (60 * 60 * 24);
    hours = pass_time / (60 * 60);
    pass_time -= hours * (60 * 60);
    minutes = pass_time / 60;
    pass_time -= minutes * (60);
    seconds = pass_time;

    xil_printf("RunTime: %d days %d hours %d minutes %d seconds\n\r", days, hours,
    minutes, seconds);
}
```



# APPENDIX A: xil\_printf vs printf

可以參考以下這篇，簡單來說 **xil\_printf** 不支援 **float**

[Which printf or xil\\_print? \(xilinx.com\)](#)



barriet (Xilinx)

3 years ago

I'm not the expert here, but definitely also remember them NOT being the same.

A quick check of the current OS libraries reveals

/\*

xil\_printf

xil\_printf() is a light-weight implementation of printf. It is much smaller in size (only 1 Kb). It does not have support for floating point numbers. xil\_printf() also does not support printing of long (such as 64-bit) numbers.

...

\*/

from [https://www.xilinx.com/support/documentation/sw\\_manuals/xilinx2020\\_1/oslib\\_rm.pdf](https://www.xilinx.com/support/documentation/sw_manuals/xilinx2020_1/oslib_rm.pdf) page 10.

Cheers,

bt

Like • Reply • 2 likes

[oslib\\_rm-en-us-2023.2.pdf](#) • 查看器 • AMD 自适应计算文档门户 ([xilinx.com](#)) page 10.