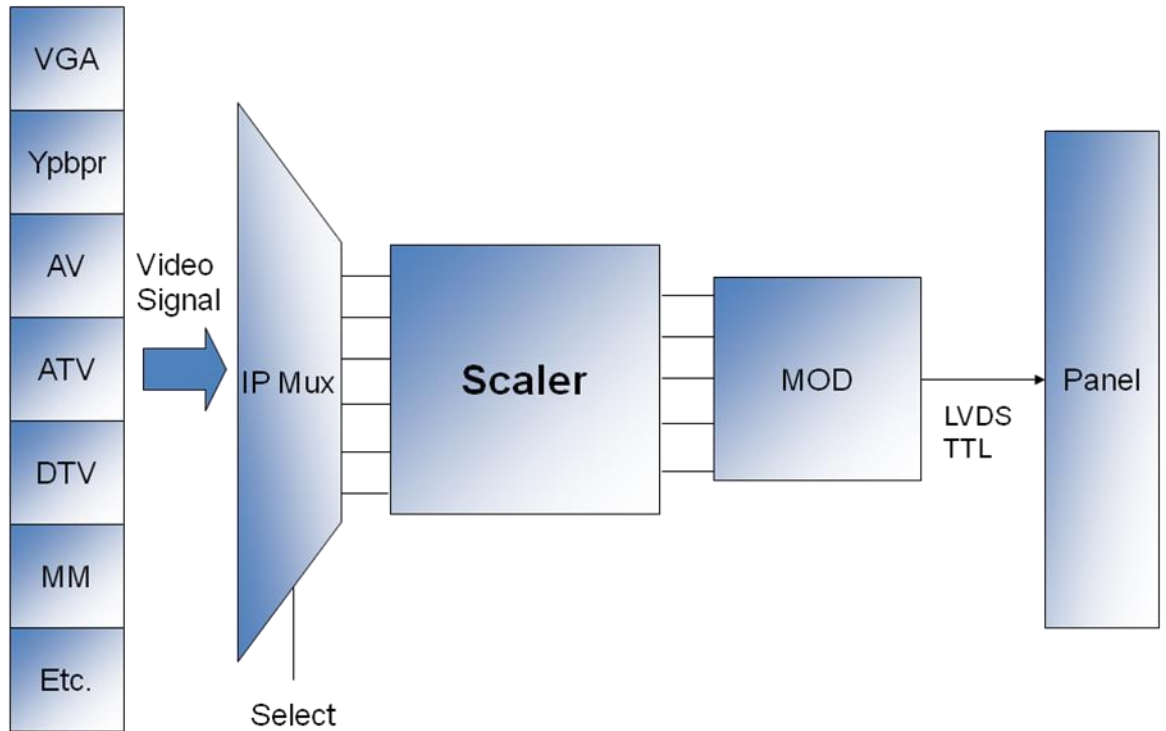
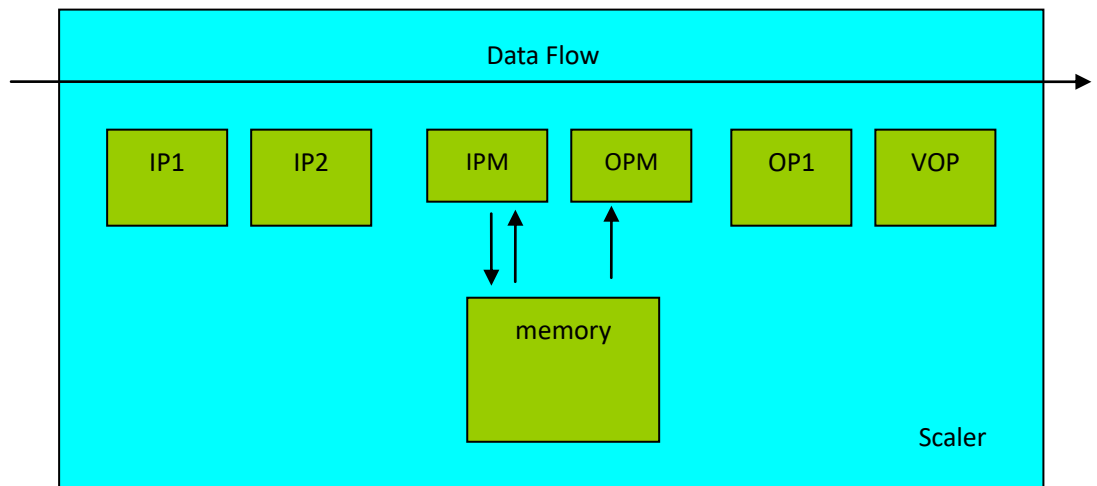


# Scaler register and utopia driver 实践

## 1 传统 xc hw 大致结构



XC 与周边 ip 关系图



XC 内部结构图

IPMUX: 切 source 的选择开关

IP1: sync detect, color space 转换

IP2: color space 转换; h/v prescaling down, 注意只能 down, 不能 up

DNR: 写 memory, 相关 PQ 图效操作

OPM: 读 memory, 相关 PQ 图效操作

OP1: post scaling

VOP: display window,color matrix

LPLL:frame lock 相关

MOD:屏相关设置

## 2 常用的 xc 术语介绍

Htotal: 2 个 hsync 之间的 pixel 数

HDE:horizontal active video,水平方向有效图像。

HFrontPorch: 见下图

HSyncWidth: 见下图

HBackPorch: 见下图

HBlanking: 见下图

Vtotal:2 个 vsync 之间的 hsync 数(线数)

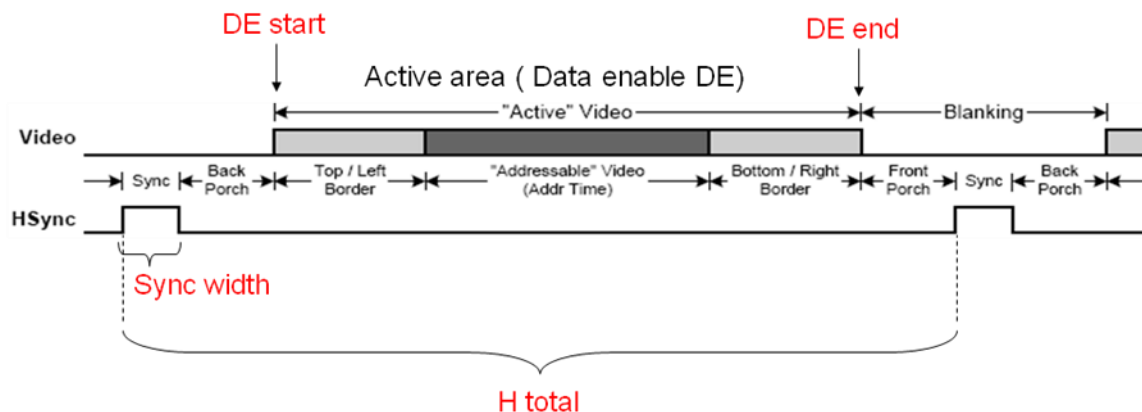
VDE:vertical active video,水平方向有效图像

VFrontPorch: 见下图

VSynWidth: 见下图

VBackPorch: 见下图

VBlanking: 见下图

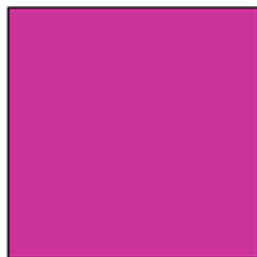


- For example, this diagram is about 1920x1080 signal.
  - H total will be 2200 pixel (VESA spec).
  - Active area is 1920 pixels
  - Blanking area is equal to 280 pixels (2200 – 1920)

VFreq:output:每秒 panel 刷新次数, input:每秒 frame 数

DCLK/Pixel CLK:htotal\*vtotal\*vfreq

Progressive:



Even field

Odd field

Interlace:

Deinterlace: 将 interlace 的 even field 和 odd field 组成一个 frame 的动作

NR: noise reduction

UCNR: ultra clear noise reduction

Frame Lock: output 端信号跟 input 端同步

FreeRun: output 端不跟 input 端信号同步, 直接以固定频率输出

Overscan: **ATV** 还有其他的模拟输入比如 **AV/SV**, 影像的上下左右边缘通常会有

garbage, 所以 **Scaler** 在抓取影像的时候会把這些边缘裁掉, 称之为

overscan

N Frame Mode: xc memory 中存 N 张 frame, 针对 progressive video

N Field Mode: xc memory 中存 N 张 field, 针对 interlace video

Frame Buffer mode: video 资料进 xc memory, 通常情况

Frame Buffer Less mode: 没有 xc memory, ip 端送进来的资料直接通过 op 输出, 通常是为了省 memory 或者节省 bw.

R Frame Buffer less mode: 仅有非常少量的 xc memory, 通常是为了省 memory 或者节省 bw.

HDuplicate/HSampling: ip 前端做的多倍 sampling, 比如 adc, mvop

Prescaling: xc 有 2 个子 ip 可以做缩放。Ip2 和 op1. 通常 prescaling 指的是在 ip2 做的缩放。对 prescaling 而言, xc ip2 只支援 prescaling down, 而不能 prescaling up.

Postscaling: op1 做的 scaling. 既可以 post scaling up 也可以 post scaling down.

Mirror: 为适应有些倒置的 panel, video image 需要旋转 180 度, xc hw 支持这个功能

Capture Window: src window, 一般情况下是信号源大小

Crop Window: clip window, 裁剪 capture window, 最终需要 xc 输出的 src 内容大小.

Display Window: dst window, xc 最终输出的大小。

Note: Display Window 跟 Panel DE 本身没关系, display window 是 xc 输出的范围大小, panel de 是 panel 接收的信号大小

Frame Color: xc 有效区域(display window)之外的 color

NoSignal Display Color: xc 有效区域(display window)之内的 screen mute 时的 Color

16 bit address vs 8 bit address: 16bit address 每个 address 代表 2byte 的地址空间; 而 8bit address 每个 address 只占有 1byte address 空间。所以地址空间转换是 2 倍的关系。

### 3 Driver 总体架构

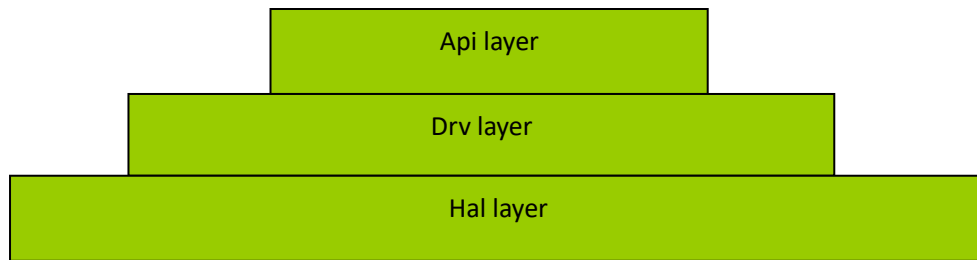
分层设计, 分 Api layer, Drv layer, Hal layer.

Api layer: interface

Drv layer: driver 主要 logic

Hal layer: 各个 chip 相关的 register 设置

Note:随着 utopia 维护的 chip 越来越多, 要 driver develop 人员在 hal 层维护不同的 chip 变的越来越复杂。由于很多 chip 部分 hw ip 的相似性, 因此未来开发中, 我们建议在 drv layer 控制 register 设置, 用 macro 来区别不同 feature。



## 4 常用的 API 简介

- **MS\_BOOL MApi\_XC\_SetWindow(XC\_SETWIN\_INFO \*pstXC\_SetWin\_Info, MS\_U32 u32InitDataLen, SCALER\_WIN eWindow)**  
设置 xc window, hw 处理 timing, 比如设置 crop window, display window 等, xc 最重要的函数。  
pstXC\_SetWin\_Info: 指向 XC\_SETWIN\_INFO 的指针。XC\_SETWIN\_INFO 中包含设置 xc hw 的所有信息  
u32InitDataLen: XC\_SETWIN\_INFO 的 size  
eWindow: 指定具体哪个 window  
返回值: TRUE, 设置成功; FALSE, 设置不成功。
- **void MApi\_XC\_SetPanelTiming(XC\_SetTiming\_Info \*pTimingInfo, SCALER\_WIN eWindow)**  
设置 xc panel timing, 处理 xc output frame lock。  
pTimingInfo: 指向 XC\_SetTiming\_Info 的指针。XC\_SetTiming\_Info 中包含设置 xc framelock 的所有信息。  
eWindow: 指定具体哪个 window
- **MS\_U16 MApi\_XC\_GetStatusEx(XC\_ApiStatusEx \*pDrvStatusEx, SCALER\_WIN eWindow)**  
返回 xc 当前的 status info. 比如 display 信息, 当前 input timing 的信息等等。  
pDrvStatusEx: 指向 XC\_ApiStatusEx 的指针。  
eWindow: 指定具体哪个 window  
返回值: 当前返回 byte 数
- **MS\_BOOL MApi\_XC\_SetDbgLevel(MS\_U16 u16DbgSwitch)**  
通过设置不同 xc dbg level, print xc msg.  
u16DbgSwitch: dbg level  

```
#define XC_DBGLEVEL_OFF          (0x0000)    ///< Turn off
#define XC_DBGLEVEL_SETWINDOW    (0x0001)    ///< Set Window
```

```

#define XC_DBGLEVEL_SETTIMING      (0x0002)    ///< LPLL / SetPanelTiming
#define XC_DBGLEVEL_SYNCDETECT    (0x0004)    ///< Sync detection
//=====>Not using it by now

#define XC_DBGLEVEL_MUX            (0x0008)    ///< Mux / Path Creation
#define XC_DBGLEVEL_MODEPARSE     (0x0010)    ///< Mode Parse
//=====>Not using it by now

#define XC_DBGLEVEL_HDMIMONITOR   (0x0020)    ///< HDMI and Package Monitor
//====>Not using it by now

#define XC_DBGLEVEL_AUTOGEOMETRY  (0x0040)    ///< Auto Geometry / Auto
//Calibration

#define XC_DBGLEVEL_CROPCALC      (0x0080)    ///< Crop Window / Memory / OPM
//IPM calculation

#define XC_DBGLEVEL_SETTIMING_ISR (0x0100)    ///< ISR / SetPanelTiming
#define XC_DBGLEVEL_DWIN_SETWINDOW (0x0200)   ///< DWIN / Set Window
#define XC_DBGLEVEL_FUNCTION_TRACE (0x0400)   ///< Trace XC function using
//status

#define XC_DBGLEVEL_PARAMETER_TRACE (0x0800)  ///< Trace XC function
//parameter

```

返回值: TRUE,设置成功; FALSE,设置不成功。

- **E\_APIXC\_ReturnValue MApi\_XC\_GetChipCaps(E\_XC\_CAPS eCapType, MS\_U32\* pRet, MS\_U32 ret\_size)**

chip 是否支援指定 hw feature

eCapType: hw feature

pRet:返回信息

ret\_size:返回 size

返回值:设置成功与否 s

- **void MApi\_XC\_SetInputSource( INPUT\_SOURCE\_TYPE\_t enInputSourceType, SCALER\_WIN eWindow )**

设置 input source

enInputSourceType: input source type

eWindow: 指定具体哪个 window

- **MS\_BOOL MApi\_XC\_EnableMirrorModeEx( MirrorMode\_t eMirrorMode, SCALER\_WIN eWindow)**

Enable xc mirror

eMirrorMode: mirror type,有 h mirror, v mirror, hv mirror

eWindow: 指定具体哪个 window

返回值: TRUE,设置成功; FALSE,设置不成功。

- **MirrorMode\_t MApi\_XC\_GetMirrorModeTypeEx(SCALER\_WIN eWindow)**

返回当前 xc 的 mirror status

eWindow: 指定具体哪个 window

返回值: mirror type

- **void MApi\_XC\_SetHdmiSyncMode(E\_HDMI\_SYNC\_TYPE esynctype)**  
设置 hdmi detect mode  
esynctype: DE mode: HDMI\_SYNC\_DE HV mode: HDMI\_SYNC\_HV
- **E\_HDMI\_SYNC\_TYPE MApi\_XC\_GetHdmiSyncMode(void)**  
返回当前的 hdmi detect mode  
返回值: hdmi detect mode, hv mode or de mode
- **void MApi\_XC\_SetDispWinToReg(MS\_WINDOW\_TYPE \*pstDspwin, SCALER\_WINDOW eWindow)**  
直接根据指定的 display window,设置 xc display window reg.  
pstDspwin:需要设置的 display window  
eWindow: 指定具体哪个 window
- **void MApi\_XC\_GetDispWinFromReg(MS\_WINDOW\_TYPE \*pstDspwin, SCALER\_WINDOW eWindow)**  
直接从 display window reg 中读取 display window 值  
pstDspwin:从 reg 读到的 display window 值  
eWindow: 指定具体哪个 window
- **void MApi\_XC\_FreezeImg(MS\_BOOL bEnable, SCALER\_WINDOW eWindow)**  
freeze image,使 video 画面静止  
bEnable: freeze image or not  
eWindow: 指定具体哪个 window
- **MS\_BOOL MApi\_XC\_IsFreezeImg(SCALER\_WINDOW eWindow)**  
xc image is freezed or not  
eWindow: 指定具体哪个 window  
返回值: TRUE: freezed FALSE: not freezed
- **void MApi\_XC\_GenerateBlackVideo(MS\_BOOL bEnable, SCALER\_WINDOW eWindow)**  
遮解屏  
bEnable: TRUE, 遮屏; FALSE, 解屏  
eWindow: 指定具体哪个 window
- **MS\_BOOL MApi\_XC\_IsBlackVideoEnable(SCALER\_WINDOW eWindow)**  
check 是否有遮屏  
eWindow: 指定具体哪个 window  
返回值: TRUE:当前是遮屏状态 FALSE: 当前在 video 正常播放状态, 屏没有遮起来
- **void MApi\_XC\_EnableFrameBufferLess(MS\_BOOL bEnable)**  
enable FBL or not

bEnable: TRUE: enable FBL mode FALSE: enable FB mode

- **MS\_BOOL MApi\_XC\_IsCurrentFrameBufferLessMode(void)**

Check 当前是否 FBL mode

返回值: TRUE: FBL mode FALSE: FB mode

- **void MApi\_XC\_EnableRequest\_FrameBufferLess(MS\_BOOL bEnable)**

enable RFBL or not

bEnable: TRUE: enable RFBL mode FALSE: enable FB mode

- **MS\_BOOL MApi\_XC\_IsCurrentRequest\_FrameBufferLessMode(void)**

Check 当前是否 RFBL mode

返回值: TRUE: RFBL mode FALSE: FB mode

- **MS\_BOOL MApi\_XC\_Set\_3D\_Mode(E\_XC\_3D\_INPUT\_MODE e3dInputMode,  
E\_XC\_3D\_OUTPUT\_MODE e3dOutputMode,  
E\_XC\_3D\_PANEL\_TYPE e3dPanelType,  
SCALER\_WIN eWindow)**

Enable 3D 的主要 function,这个 function 主要是保存 3D 信息,然后通过 SetWindow 进行 3D 转换。

e3dInputMode: 输入, input 3d format

e3dOutputMode:输入, output 3d format

e3dPanelType:输入,3d panel type

eWindow:设置具体哪个 window 的 3d type

返回值: TRUE,设置成功; FALSE,设置不成功。

- **MS\_U16 MApi\_XC\_Get\_3D\_HW\_Version()**

返回 hw 3d 版本号

返回值:3d 版本号

- **MS\_BOOL MApi\_XC\_Get\_3D\_IsSupportedHW2DTo3D(void)**

Check 是否当前 ic 支持 hw 2d to 3d

返回值: TRUE, 支持 hw 2d to 3d; FALSE,不支持 hw 2d to 3d

- **E\_XC\_3D\_INPUT\_MODE MApi\_XC\_Get\_3D\_Input\_Mode(SCALER\_WIN eWindow)**

Get 当前各 window 对应的 input 3d format

eWindow: 指定具体哪个 window

返回值: input 3d format

- **E\_XC\_3D\_OUTPUT\_MODE MApi\_XC\_Get\_3D\_Output\_Mode(void)**

Get 当前的 output 3d format

返回值:output 3d format

- **MS\_BOOL MApi\_XC\_Set\_3D\_LR\_Frame\_Exchg(SCALER\_WIN eWindow)**

Do LR exchange for 3D

eWindow: 指定具体哪个 window

返回值: TRUE,设置成功; FALSE,设置不成功。

- **MS\_BOOL MApi\_XC\_3D\_Is\_LR\_Frame\_Exchged(SCALER\_WIN eWindow)**  
Check 当前是否 LR 还是 RL  
eWindow: 指定具体哪个 window  
返回值: TRUE,RL; FALSE,LR。
- **MS\_BOOL MApi\_XC\_Set\_3D\_HShift(MS\_U16 u16HShift)**  
Set sw 2d to 3d 以及真 3D 的景深  
u16HShift:景深 value  
返回值: TRUE,设置成功; FALSE,设置不成功。
- **MS\_U16 MApi\_XC\_Get\_3D\_HShift(void)**  
Get sw 2d to 3d 以及真 3D 的景深  
返回值: 景深 value
- **MS\_BOOL MApi\_XC\_Set\_3D\_HW2DTo3D\_Parameters(MS\_XC\_3D\_HW2DTO3D\_PARA st3DHw2DTo3DPara)**  
Set hw 2d to 3d parameters for user adjustment  
st3DHw2DTo3DPara:hw 2d to 3d parameters,具体字段含义详见前面数据结构解释  
返回值: TRUE,设置成功; FALSE,设置不成功。
- **MS\_BOOL MApi\_XC\_Get\_3D\_HW2DTo3D\_Parameters(MS\_XC\_3D\_HW2DTO3D\_PARA \*pst3DHw2DTo3DPara)**  
Get hw 2d to 3d parameters  
pst3DHw2DTo3DPara: pointer to hw 2d to 3d parameters  
返回值: TRUE,设置成功; FALSE,设置不成功。
- **MS\_BOOL MApi\_XC\_Set\_3D\_HW2DTo3D\_Buffer(MS\_PHYADDR u32HW2DTO3D\_DD\_Buf, MS\_PHYADDR u32HW2DTO3D\_DR\_Buf)**  
设置 hw 2d to 3d 所要使用的 Depth Detection Buffer 以及 Depth Render Buffer  
u32HW2DTO3D\_DD\_Buf: Depth Detection Buffer address  
u32HW2DTO3D\_DR\_Buf: Depth Render Buffer address  
返回值: TRUE,设置成功; FALSE,设置不成功。
- **MS\_BOOL MApi\_XC\_Set\_3D\_Detect3DFormat\_Parameters(MS\_XC\_3D\_DETECT3DFORMAT\_PARA \*pstDetect3DFormatPara)**  
Set sw 3D 自动识别参数  
pstDetect3DFormatPara: sw 3d 自动识别参数, 具体字段含义详见前面数据结构解释  
返回值: TRUE,设置成功; FALSE,设置不成功。



- **MS\_BOOL MApi\_XC\_Get\_3D\_Detect3DFormat\_Parameters(MS\_XC\_3D\_DETECT3DFORMAT\_PARA \*pstDetect3DFormatPara)**  
Get sw 3D 自动识别参数  
pstDetect3DFormatPara: sw 3d 自动识别参数，具体字段含义详见前面数据结构解释  
返回值: TRUE,设置成功; FALSE,设置不成功。
- **E\_XC\_3D\_INPUT\_MODE MApi\_XC\_Detect3DFormatByContent(SCALER\_WIN eWindow)**  
开始做 3D 自动识别。  
eWindow: 指定具体哪个 window  
返回值: 识别到的 3D 格式
- **MS\_BOOL MApi\_XC\_Is3DFormatSupported(E\_XC\_3D\_INPUT\_MODE e3dInputMode, E\_XC\_3D\_OUTPUT\_MODE e3dOutputMode)**  
Check 当前 driver 支持的 3D 输入输出格式  
e3dInputMode: input 3d format  
e3dOutputMode: output 3d format  
返回值: TRUE,支持; FALSE,不支持。
- **MS\_U32 MApi\_XC\_GetAccurateVFreqx1K(SCALER\_WIN eWindow)**  
返回 input vfreq. 特别注意，返回的 vfreq 有 x1000。  
eWindow: 指定具体哪个 window  
返回值: vfreq
- **void MApi\_XC\_DisableInputSource(MS\_BOOL bDisable, SCALER\_WIN eWindow)**  
disable input source or not  
bDisable: TRUE,disable input source FALSE,enable input source  
eWindow: 指定具体哪个 window
- **MS\_BOOL MApi\_XC\_IsInputSourceDisabled(SCALER\_WIN eWindow)**  
Check Input source 是否有 disable  
eWindow: 指定具体哪个 window  
返回值: TRUE: input source is disabled FALSE: input source is enabled
- **void MApi\_XC\_SetFrameColor(MS\_U32 u32aRGB)**  
设置 frame color  
u32aRGB: 需要设置的顏色 the 4 bytes in aRGB are "a: no use, R 23:16, G 15:8, B 7:0" (B is in LSB)
- **void MApi\_XC\_SetDispWindowColor(MS\_U8 u8Color, SCALER\_WIN eWindow)**  
设置无信号 display window 的 color  
u8Color: 需要设置的顏色 format in a byte => R 7:5, G 4:2, B 1:0  
eWindow: 指定具体哪个 window

- **MS\_U16 MApi\_XC\_GetOutputVFreqX100(void)**  
返回 output vfreq, 注意, 返回的 output vfreq 是 x100 的。  
返回值: output vfreq
- **void MApi\_SC\_ForceFreerun(MS\_BOOL bEnable)**  
force freerun 输出。有些需要固定输出频率的 panel 需要使用这个 function  
bEnable: TRUE: force freerun FALSE: frame lock
- **MS\_BOOL MApi\_SC\_IsForceFreerun(void)**  
返回 force freerun 状态, 是否 force freerun  
返回值: TRUE: force freerun FALSE: 没有设置 force freerun
- **void MApi\_SC\_SetFreerunVFreq(E\_VFREQ\_SEL VFreq)**  
设置 force freerun 时的 vfreq  
VFreq: 可供选择的 vfreq
- **void MApi\_XC\_EnableWindow(MS\_BOOL bEnable, SCALER\_WIN eWindow)**  
enable 指定的 xc Window  
bEnable: enable window or not  
eWindow: 指定具体哪个 window

## 5 编译 utopia 方法

- **编译 Utopia2:**
  - 1 cd ./build
  - 2 cp napoli/.config\_napoli\_linux\_arm\_hardfloat\_dynamic\_general .config(复制需要 build chip 的相应 config 到.config)
  - 3 make menuconfig -->进去后选 exit --> 选 yes
  - 4 make xc bsp
  - 5 libapiXC.so 会产生在./bsp/lib
- **编译 utopia:**
  - 1 cd ./project/edison\_linux\_arm (具体 project 的目录)
  - 2 make xc bsp
  - 3 libapiXC.so 会产生在./bsp/lib

pnl, ace, dlc 等 xc 相关 lib 编译方法类似, 只不过把 make xc bsp 换成 make pnl bsp, make ace bsp, make dlc bsp

## 6 如何打开 utopia xc msg 功能

### ■ 之前 project 的方法,需要修改 utopia driver:

- mxlib\drv\xc\include\mvideo\_context.h 中:

```
//#define MS_DEBUG 1
```

改为

```
#define MS_DEBUG 1
```

```
if( (_dbgSwitch_ & _u16XCdbgSwitch_InternalUseOnly) != 0 ) \
```

改为

```
if( (_dbgSwitch_ & 0x183) != 0 ) \
```

注意,很老的 ic 比如 t4 可能没有, 那就不要做这步

- **mxlib\drv\xc\mvideo.c 中:**

```
#define msg_video(x) //x
```

改为

```
#define msg_video(x) x
```

注意,老 ic 才有这样的 code,如果你没找到这样的 code, 也不管。

- **mxlib\drv\xc\mdrv\_sc\_scaling.c 中:**

```
#define SC_DBG(x) //x
```

```
#define SC_FNLDBG(x) //x
```

```
#define SC_RWBANK_DBG(x) //x
```

改为

```
#define SC_DBG(x) x
```

```
#define SC_FNLDBG(x) x
```

```
#define SC_RWBANK_DBG(x) x
```

注意,老 ic 才有这样的 code,如果你没找到这样的 code, 也不管。

- **mdrv\_sc\_3d.c 中:**

```
#define SC_3D_DBG(x) //x
```

改为

```
#define SC_3D_DBG(x) x
```

注意,老 ic 才有这样的 code,如果你没找到这样的 code, 也不管。

修改完 utopia code,后 make clean, make xc bsp 就 ok 了.

## ■ 最新方法, 不需要修改 utopia:

Xc init(比如 SN 位置在 mapi\_video::SysInitXC)后 call MApi\_XC\_SetDbgLevel, 参数是给定需要打印 log 的值, 这些 enum 可以在 apiXC.h 中找到:

```
#define XC_DBGLEVEL_OFF (0x0000) ///< Turn off
#define XC_DBGLEVEL_SETWINDOW (0x0001) ///< Set Window
#define XC_DBGLEVEL_SETTIMING (0x0002) ///< LPLL / SetPanelTiming
#define XC_DBGLEVEL_SYNCDETECT (0x0004) ///< Sync detection
//=====>Not using it by now
#define XC_DBGLEVEL_MUX (0x0008) ///< Mux / Path Creation
#define XC_DBGLEVEL_MODEPARSE (0x0010) ///< Mode Parse
```

```

//=====>Not using it by now
#define XC_DBGLEVEL_HDMIMONITOR    (0x0020)  ///< HDMI and Package Monitor
//=====>Not using it by now
#define XC_DBGLEVEL_AUTOGEOMETRY    (0x0040)  ///< Auto Geometry / Auto
//Calibration
#define XC_DBGLEVEL_CROPCALC        (0x0080)  ///< Crop Window / Memory / OPM
//IPM calculation
#define XC_DBGLEVEL_SETTIMING_ISR    (0x0100)  ///< ISR / SetPanelTiming
#define XC_DBGLEVEL_DWIN_SETWINDOW  (0x0200)  ///< DWIN / Set Window
#define XC_DBGLEVEL_FUNCTION_TRACE  (0x0400)  ///< Trace XC function using
//status
#define XC_DBGLEVEL_PARAMETER_TRACE (0x0800)  ///< Trace XC function
//parameter

```

比如最常用的是把这个参数设置成 **0x183**,即 **enable** XC\_DBGLEVEL\_SETTIMING\_ISR, XC\_DBGLEVEL\_CROPCALC, XC\_DBGLEVEL\_SETTIMING, XC\_DBGLEVEL\_SETWINDOW

## 7 如何查看 xc msg 得到有用的信息

注意打印信息会随着 code 更新,有稍许不同,最好的办法是大致看一下 code. 打印出来的 dbg msg 会像如下这样:

```

[XC,SetWindow][MApi_XC_SetWindow, 1965] ===== SetWindow Start (Window : 0, src:
34) =====
[XC,SetWindow][MApi_XC_SetWindow, 1966] CapWin  x: 152  y: 6  w: 1920  h: 1080
[XC,SetWindow][MApi_XC_SetWindow, 1967] CropWin x: 0  y: 0  w: 1920  h: 1080
[XC,SetWindow][MApi_XC_SetWindow, 1968] DispWin x: 60  y: 0  w: 1366  h: 768
[XC,SetWindow][MApi_XC_SetWindow, 1969] Panel   x: 60  y: 0  w: 1366  h: 768
[XC,SetWindow][MApi_XC_SetWindow, 1970] Mirror/Interlace/Hdup = 0/1/0
[XC,SetWindow][MApi_XC_SetWindow, 1971] H/V total = (0, 1125), VFreq = 500
[XC,SetWindow][MApi_XC_SetWindow, 1972] Pre   H cus scaling 0 (0 -> 0)
[XC,SetWindow][MApi_XC_SetWindow, 1973] Pre   V cus scaling 0 (0 -> 0)
[XC,SetWindow][MApi_XC_SetWindow, 1974] Post H cus scaling 0 (0 -> 0)
[XC,SetWindow][MApi_XC_SetWindow, 1975] Post V cus scaling 0 (0 -> 0)
[XC,SetWindow][MApi_XC_Set_PQ_SourceData, 1622] PQSetSource:Input src=34, win=0,
[XC,SetWindow][MApi_XC_Set_PQ_SourceData, 1623] PQSetSource:   PQ src=9, win=0
[XC,SetWindow][msAPI_Scaler_SetMode, 1445] MemFmt422=1
[XC,SetWindow][msAPI_Scaler_SetMode, 1468] bMemYUVFmt=1, RGBin=0
省略 10000 字.....

```

**Capture window** 的尺寸(一般情况下可以认为是原始 video 的尺寸):

width: msg 中的 w; height: msg 中的 h

Capture window 的 x,y 通常不是 care 的重点。

```

[XC,SetWindow][MApi_XC_SetWindow, 1966] CapWin  x: 152  y: 6  w: 1920  h:
1080

```

**Crop window** 的尺寸:

解释跟 catpure window 一样

[XC,SetWindow][MApi\_XC\_SetWindow, 1967] CropWin x: 0 y: 0 w: 1920 h: 1080

#### Display window 的尺寸:

解释跟 catpure window 一样,注意屏幕上 x,y 坐标要减去 Panel 的尺寸中的 x,y, 不然是有加上 panel 本身的 hstart 和 vstart

[XC,SetWindow][MApi\_XC\_SetWindow, 1968] DispWin x: 60 y: 0 w: 1366 h: 768

#### Panel 的尺寸:

解释跟 catpure window 一样

[XC,SetWindow][MApi\_XC\_SetWindow, 1969] Panel x: 60 y: 0 w: 1366 h: 768

#### 是否 mirror:

msg 中的 Mirror

#### Input Timing 是否 interlace:

msg 中的 Interlace

#### 是否有做 Hduplicate(H 方向前端有 double size or \*4 size 等):

msg 中 Hdup

[XC,SetWindow][MApi\_XC\_SetWindow, 1970] Mirror/Interlace/Hdup = 0/1/0

#### Input Vtotal:

#### Input VFreq:

[XC,SetWindow][MApi\_XC\_SetWindow, 1971] H/V total = (0, 1125), VFreq = 500

#### 是否 AP 有指定 H 方向 Custom prescaling,并且指定从多少缩放到多少:

[XC,SetWindow][MApi\_XC\_SetWindow, 1972] Pre H cus scaling 0 (0 -> 0)

#### 是否 AP 有指定 V 方向 Custom prescaling,并且指定从多少缩放到多少:

[XC,SetWindow][MApi\_XC\_SetWindow, 1973] Pre V cus scaling 0 (0 -> 0)

#### 是否 AP 有指定 H 方向 Custom postscaling,并且指定从多少缩放到多少:

[XC,SetWindow][MApi\_XC\_SetWindow, 1974] Post H cus scaling 0 (0 -> 0)

#### 是否 AP 有指定 V 方向 Custom postscaling,并且指定从多少缩放到多少:

[XC,SetWindow][MApi\_XC\_SetWindow, 1975] Post V cus scaling 0 (0 -> 0)

#### 当前 input source:

Msg 中的 Input src,具体 value 代表哪个端口, 可以查看 enum: INPUT\_SOURCE\_TYPE\_t

#### 当前 window:

Win=0 代表 main window; win=1 代表 sub window

[XC,SetWindow][MApi\_XC\_Set\_PQ\_SourceData, 1622] PQSetSource:Input src=34, win=0,

#### 422 or not:

[XC,SetWindow][msAPI\_Scaler\_SetMode, 1445] MemFmt422=1

#### Yuv or not:

[XC,SetWindow][msAPI\_Scaler\_SetMode, 1468] bMemYUVFmt=1, RGBin=0

#### UCNR 是否有开:

[XC,SetWindow][MApi\_XC\_SetWindow, 2047] UCNr enabled: NO

#### V 方向 Scaler 如何前缩:

前缩前尺寸: VSrc 前缩后尺寸:VDst

[XC,SetWindow][MDrv\_SC\_set\_prescaling\_ratio, 973] [PRE]eWindow[0] VSrc=1080,

VDst=1080, bBilinear=0

#### H 方向 Scaler 如何前缩:

前缩前尺寸: HSrc 前缩后尺寸: HDst

[XC,SetWindow][MDrv\_SC\_set\_prescaling\_ratio, 1450] [PRE]eWindow[0] HSrc=1920,  
HDst=1366, bAdvMode=3

#### V 方向 Scaler 如何后缩放:

后缩放前尺寸: VSrc 后缩放后尺寸: VDst

[XC,SetWindow][MDrv\_SC\_set\_postscaling\_ratio, 1692] [PST]VSrc=1080, VDst=768,  
eDeInterlaceMode=3, bFBL=0, bInterlace=1

#### H 方向 Scaler 如何后缩放:

后缩放前尺寸: HSrc 后缩放后尺寸: HDst

[XC,SetWindow][MDrv\_SC\_set\_postscaling\_ratio, 1609] [PST]HSrc=1366, HDst=1366

#### 是否 FBL:

msg 中的 bFBL

[XC,SetWindow][MDrv\_SC\_set\_postscaling\_ratio, 1692] [PST]VSrc=1080, VDst=768,  
eDeInterlaceMode=3, bFBL=0, bInterlace=1

#### 每个 pixel 占几个 byte:

Msg 中的 BytesPer2Pixel,注意这里是 2 个 pixel 占几个 byte

[XC,Cropping][MDrv\_SC\_set\_fetch\_number\_limit, 2556] =>Framesize(4458240)=  
VSizeAfterPreSD(1080) \* DNROffset(1376) \* BytesPer2Pixel(6)/2

#### Xc total memory size:

Msg 中的 DNRBufSize,单位 byte

[XC,Cropping][MDrv\_SC\_set\_fetch\_number\_limit, 2669] u32Offset :8388608,  
DNRBufSize=16777216

#### 当前是几 frame mode:

Msg 中的 pFrameStoreNumber,具体数字看 XC\_FRAME\_STORE\_NUMBER 定义

[XC,Cropping][\_MDrv\_XC\_Refine\_FrameNum, 1997] pFrameStoreNumber: 2 , corresponding  
factor is: 2

#### Output htotal:

#### Output Vtotal:

[XC,LPLL][MDrv\_SC\_set\_Htt\_Vtt, 1549] Htt\_out, Vtt\_out= 1560 , 967

#### Output vfreq:

u16OutputVFreqAfterFRC,数值有\*10

[XC,LPLL][MApi\_XC\_SetPanelTiming\_FSM, 2394] FSM: \_u16OutputVFreqAfterFRC = 500

#### 是否有做 frame lock:

[XC,LPLL][MDrv\_SC\_Cal\_FRC\_Output\_Vfreq, 529] Frame Lock = True

#### 是否有锁住:

FB 下看是否有一下 msg,如果有, 代表锁住:

[XC,Unknown][\_MApi\_XC\_FPLL\_FSM\_ISR, 2284] Locked, the using SET is 68ba8c

FBL 下:没锁住,会滚屏

#### 3D input format:

msg 中的 e3dInputMode,具体 value 含义请参见 E\_XC\_3D\_INPUT\_MODE 定义

#### 3D output format:

Msg 中的 e3dOutputMode,具体 value 含义请参见 E\_XC\_3D\_OUTPUT\_MODE。有一点例外是,

在 3D version1

版本中 line alternative 输出的 E\_XC\_3D\_OUTPUT\_MODE 跟 top bottom 输出一样, 也是 E\_XC\_3D\_OUTPUT\_TOP\_BOTTOM,这两种输出的不同需要通过 register 来区分,subbank0x23 的 16bit address 0x53 的 bit15=1,则是 line alternative 输出, 否则是 top bottom output [XC,SetWindow][MApi\_XC\_Set\_3D\_Mode, 4136] e3dInputMode=8, e3dOutputMode=2

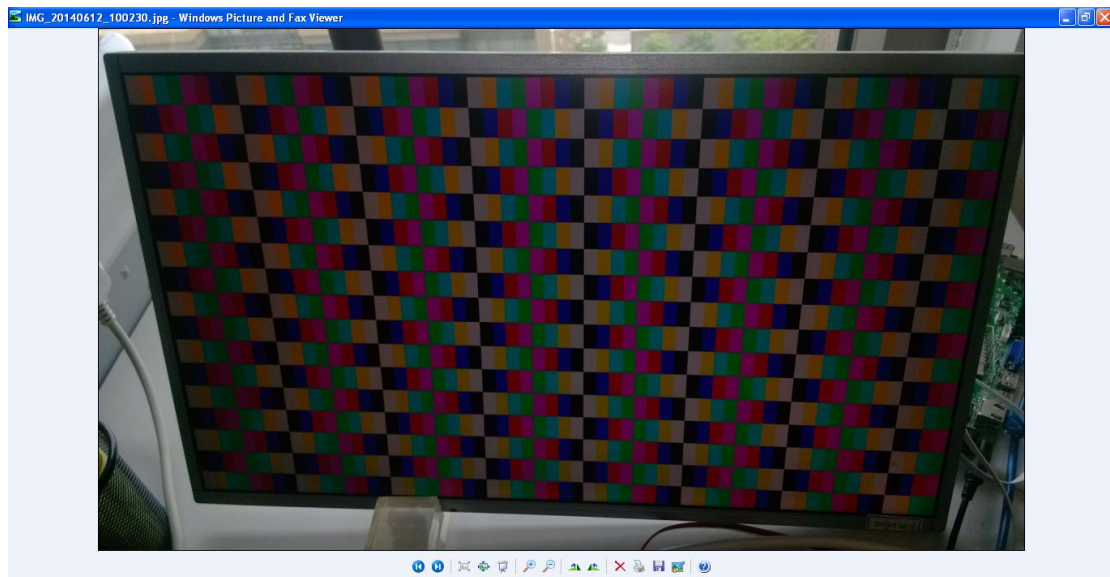
## 8 xc 相关 pattern 介绍

打 Pattern 的意思是从 show 出该 pattern 的 ip 所在位置开始, 原始的 miu data 会被 pattern 取代后显示在 panel 位置上, 用来理清到底是否是 scaler 问题以及 scaler 哪个阶段的问题。如果 pattern 是好的, 基本上可以排除之后的 hw ip 都没有问题, 除了某些用 pattern 不易察觉的 bug 外(打个比方, 比如细微线条, 跟 pattern 类似等);如果 pattern 也是坏的, 说明出问题的 hw ip 在这个 pattern 的 ip 之后。

以下描述一些跟 xc 相关的比较常用的 test pattern:

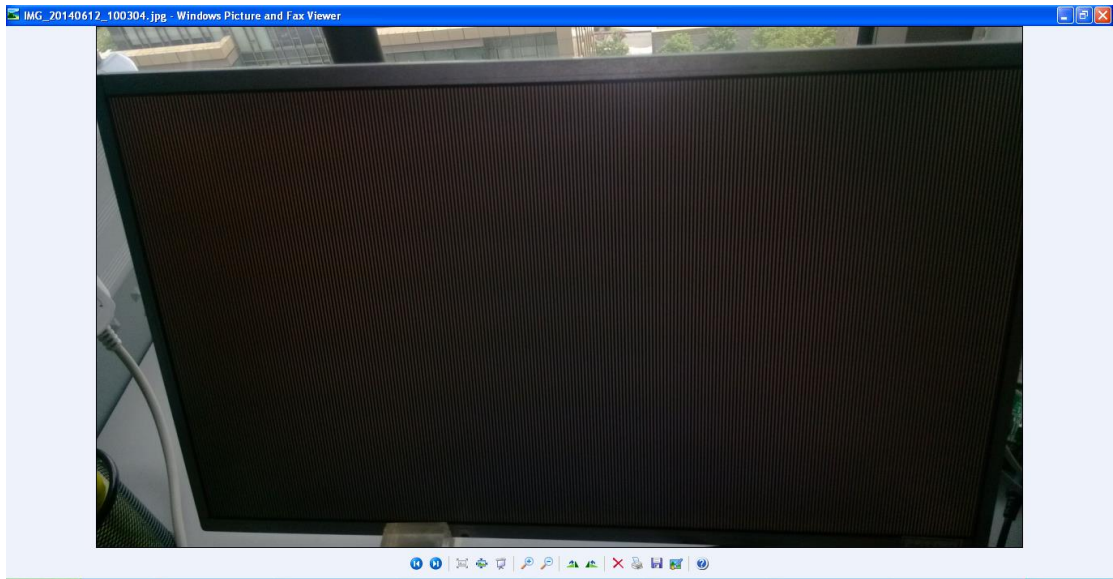
MVOP test pattern:

101400BK\_12[0]



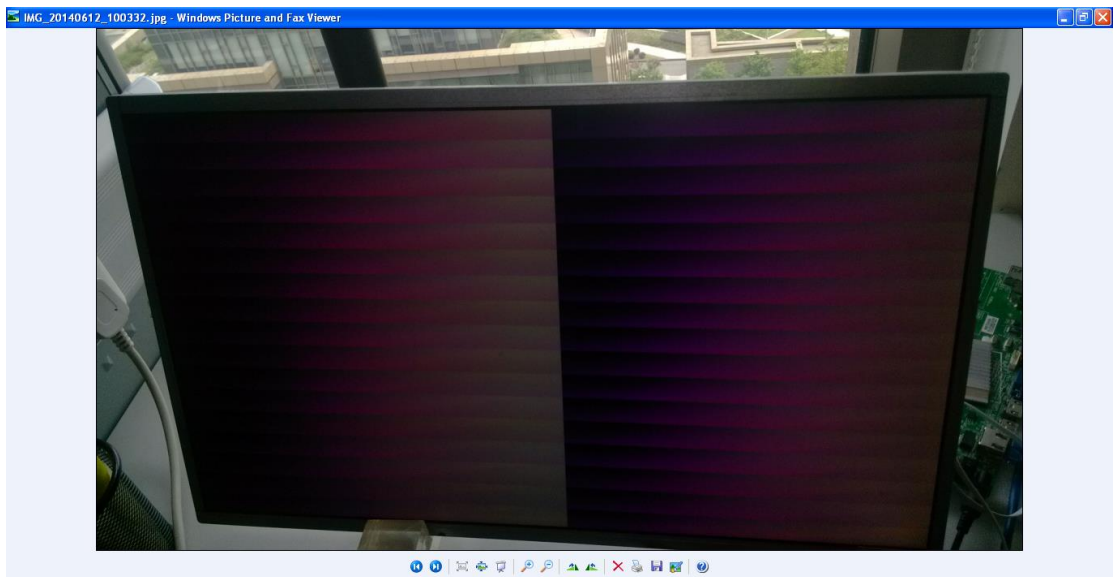
IP1 test pattern :

BK01_40[0]	H test pattern enable
BK01_40[1]	V test pattern enable
BK01_40[3]	gray bar enable
BK01_40[10:8]	Color bar sequence select
BK01_40[13:11]	Color bar sequence select
BK01_42[7:0]	Color bar H width
BK01_42[15:8]	Color bar V width
BK01_40[2]=1	make test pattern moving automatically



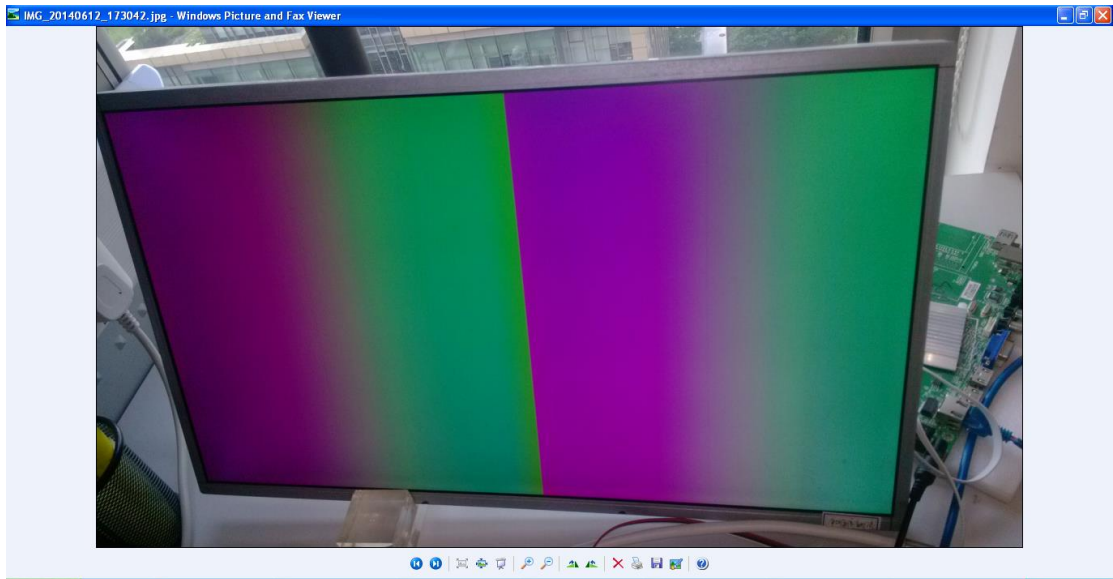
OP test pattern :

BK20\_10[15]: the test pattern between OPM & HVSP(post\_scaling)



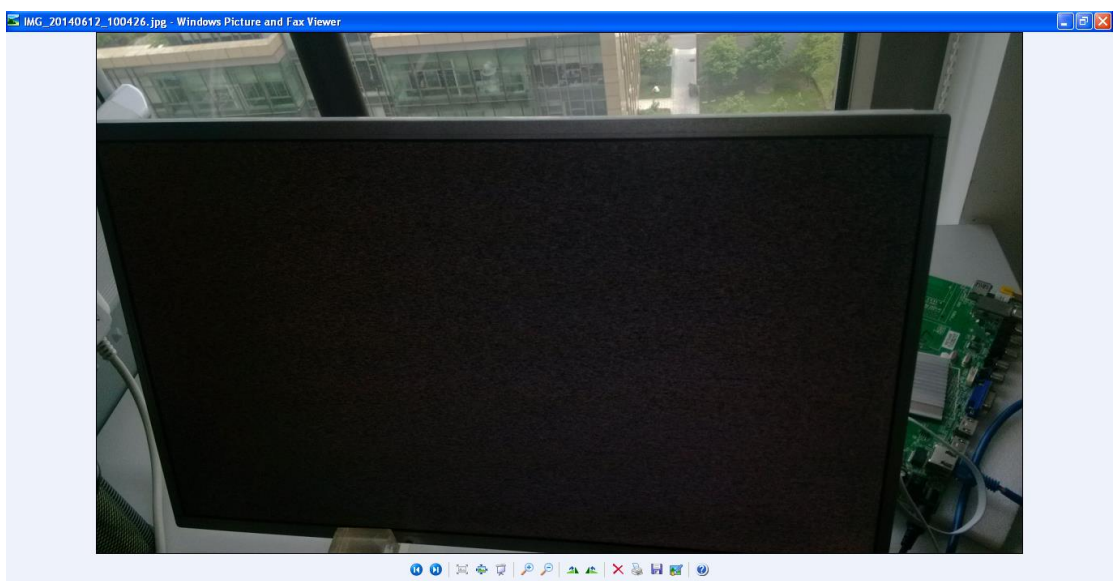
BK20\_1F[10]: the test pattern between HVSP & Display Line Buffer





VOP test pattern :

BK10\_13[8] : enable



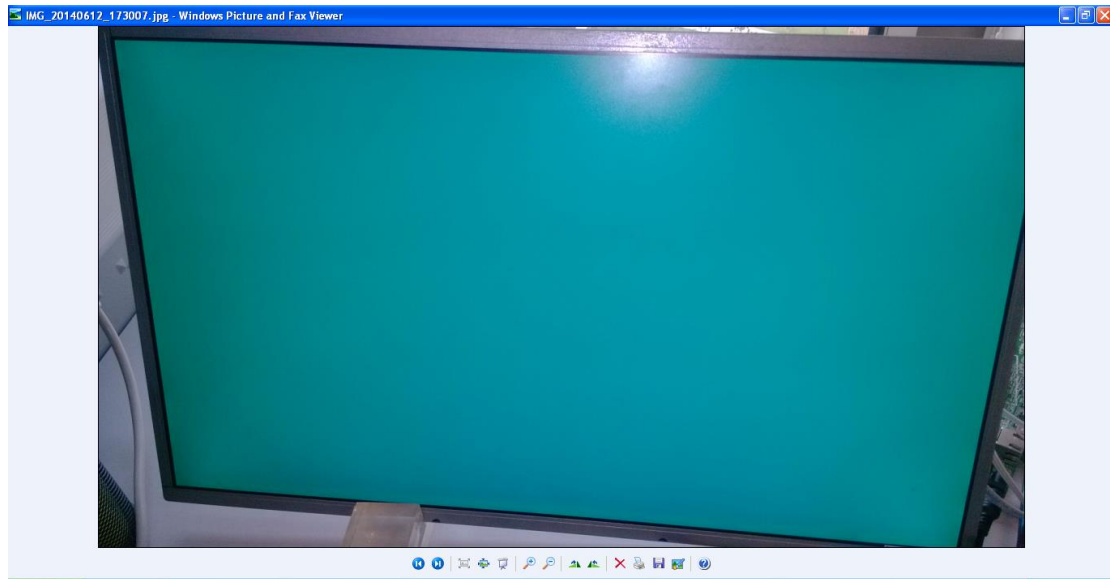
MOD test pattern : 0x3200

BK00\_01[15] : enable

BK00\_02[9:0]: R data

BK00\_03[9:0]: G data

BK00\_04[9:0]: B data



## 9 xc 主要 bank 介绍

Xc 内部有分成了很多个 subbank. 每个 subbank 有 256 个字节的地址空间. 这里主要介绍一下常用的 subbank.

**Bank0:** 主要处理 scaler reset, interrupt, 以及 gop 的 blending

**Bank1:** main window ip1, 主要处理 sync detect, color space 转换

**Bank2:** main window ip2, 主要处理 color space 转换; h/v prescaling down, 注意只能 down, 不能 up

**Bank3:** sub window ip1, 主要处理 sync detect, color space 转换

**Bank4:** sub window ip2, 主要处理 color space 转换; h/v prescaling down, 注意只能 down, 不能 up

**BankA:** 主要处理 Film detect

**BankF:** 主要处理 sub window 相关的 vop 设置, display window, color matrix

**Bank10:** VOP 相关设置, display window, color matrix

**Bank12:** DNR OPM memory setting

**Bank13:** offline detect 相关设置

**Bank15:** t3d depth detection 相关设置

**Bank18:** ACE 相关设置

**Bank19:** PQ Peaking 相关设置

**Bank1A:** DLC 相关设置

**Bank1F:** Dynamic Scaling 相关设置

**Bank20:** main sub op enable/control

**Bank23:** post scaling ratio 以及 factor 相关设置

**Bank2A:** MCNR 相关设置

**Bank62:** t3d depth render pixel 部分设置

**Bank63:** t3d depth render block 部分设置

**102E00 bank:** ipmux bank, 切 source 的选择开关

**103100 bank:** LPLL bank, 处理 xc frame lock

**103200 bank:** mod 屏相关设置

# 10 如何读取 scaler register

读 register 之前记得先停一下 uart(00112233 或者 silientme)以及停 cpu(除了读取 scaler 的第 4 种方法不需要停 cpu 以外):

Mips 和 arm 在 clkgen(0x0B00 bank 的 16bit address 0x11 的 bit8),如下图:

MStar TV System Tool v4.113.239013

File Edit View Chip Select I2C Configuration Debug Help

Mstar Semiconductor, Inc

Reset Run Pause ISP

MStar Controller Video Decoder EEPROM TV Tuner I2C Devices

Read Bank

Read All Bank

VD Bank (MCU 32)

Normal XDATA

HK FRC

DMD

Bank (MCU32)

1st Base Address

0x 2 E 0 0

Access 1st Area

NONPM 0

2nd Base Address

0x 2 F 0 0

Access 2nd Area

3rd Base Address

0x 0 B 0 0

Access 3rd Area

Addr Unit

8 bit

16 bit

Clear

MCLK Freq (MHz) 100

Aeon

TSP

Address

22

Value

0x 8100

(Dec) 33024

Bit Edit

7 6 5 4 3 2 1 0

F E D C B A 9 8

☒ ☐ ☐ ☐ ☐ ☐ ☐ ☒

	00 / 08	01 / 09	02 / 0A	03 / 0B	04 / 0C	05 / 0D	06 / 0E	07 / 0F
00	0000	0007	0000	0001	0000	0002	0000	2800
08	0000	0000	0000	0000	0000	0000	0000	0000
10	0001	8100	0088	0C00	0C0C	0000	0014	0024
18	0000	0000	0000	0000	0000	0000	0100	000C
20	0100	0011	0501	0101	0901	000D	0101	0000
28	0006	0000	0000	0000	0000	0000	0101	0101
30	0005	0005	0000	0000	0000	000D	0000	0000
38	0000	0501	0101	0101	0101	0000	0001	0000
40	0400	0400	0000	0000	0000	0000	0000	0000
48	0000	0000	0000	0000	0100	0101	0000	0000
50	0000	0404	0000	1C1C	2010	000C	0000	0100
58	0001	0001	0001	0000	0000	0000	0000	0000
60	0000	0000	0001	0000	005C	0000	0009	0000
68	0001	0000	0000	0000	0000	0000	0000	0000
70	0000	0000	0000	0000	0000	0000	0000	0000
78	0000	0000	0000	0000	0000	0000	0001	0006

MStar Controller

I2C mode (USB port , CH 4)

如果是 m10 等 r2 cpu, 则在 clockgen 0x0B00 bank 的 16bit address 0x12 的 bit8

读取 scaler register 共有 4 种方法:

- 通过 Color Bank 读取 scaler register

如下图:

MStar Semiconductor, Inc

MStar Controller | Video Decoder | EEPROM | TV Tuner | I2C Devices

Read Bank

VD Bank (MCU 32) | HK FRC | DMD

Normal | XDATA | Color | Bank (MCU32)

☐ BK0 ☐ BKC ☐ BK18 ☐ BK24 ☐ All Bank

☐ BK1 ☐ BKD ☐ BK19 ☐ BK25

☐ BK2 ☐ BKE ☐ BK1A ☐ BK26

☐ BK3 ☐ BKF ☐ BK1B ☐ BK27

☐ BK4 ☒ BK10 ☐ BK1C ☐ BK28

☐ BK5 ☐ BK11 ☐ BK1D ☐ BK29

☐ BK6 ☐ BK12 ☐ BK1E ☐ BK2A ☐ BK25 ADC\_A

☐ BK7 ☐ BK13 ☐ BK1F ☐ BK2B ☐ BK26 ADC\_D

☐ BK8 ☐ BK14 ☐ BK20 ☐ BK2C ☐ BK35 AFEC

Address: 22 Value: 0x00 (Dec) 0

Bit Edit: 7 6 5 4 3 2 1 0

	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
00	10	00	13	F0	00	03	03	03	3C	00	91	05	00	00	FF	02
10	5A	03	20	05	24	01	DB	01	17	06	C6	03	17	06	00	00
20	00	40	00	00	20	08	00	10	00	00	00	00	00	19	00	00
30	00	00	20	00	00	00	00	2D	00	01	00	00	17	00	00	00
40	00	14	00	00	00	00	A0	0F	00	00	DB	08	EF	05	FE	03
50	44	10	EA	11	00	04	F7	10	2D	00	FE	03	B6	07	35	00
60	06	03	00	80	C6	03	00	00	64	00	00	00	07	00	00	00
70	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
80	00	00	00	00	50	00	00	00	00	00	00	00	00	40	00	00
90	00	00	FF	20	00	00	44	55	24	00	44	55	24	00	00	00
A0	11	00	00	00	02	00	00	00	00	00	00	00	00	00	00	00
B0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	14	00
C0	00	00	00	00	00	00	00	00	07	00	00	00	00	00	00	00
D0	00	00	00	00	00	00	00	00	FF	00	00	00	F0	0F	F0	0F
E0	F0	0F	00	00	00	00	00	00	00	00	00	00	00	00	00	00
F0	00	00	00	00	F0	0F	F0	0F	F0	0F	F0	0F	F0	0F	F0	0F

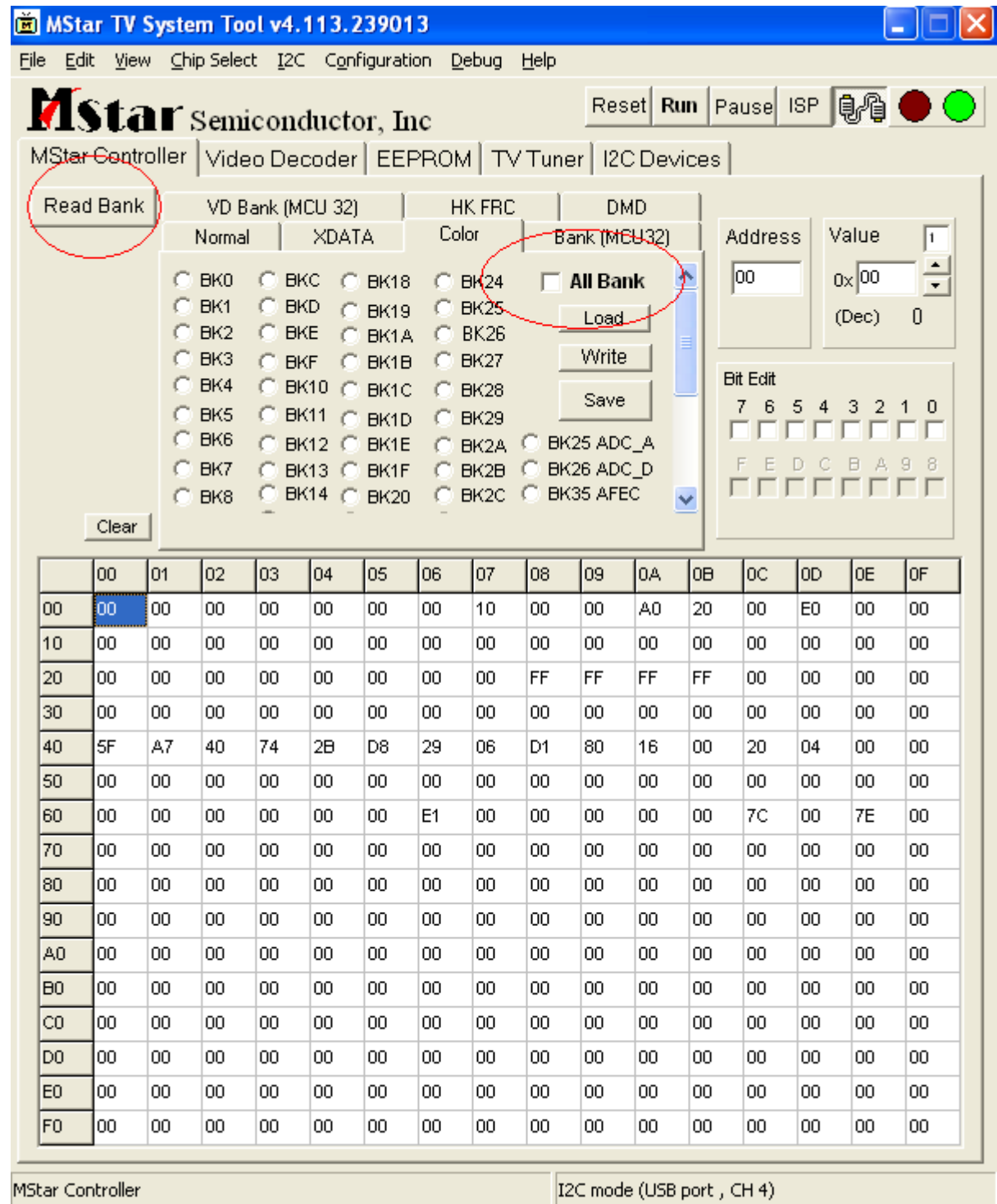
MStar Controller I2C mode (USB port , CH 4)

这里看到的都是 8bit address.如何知道读出来的 register 对不对? 可以通过点到具体某一个 BK,最左上角

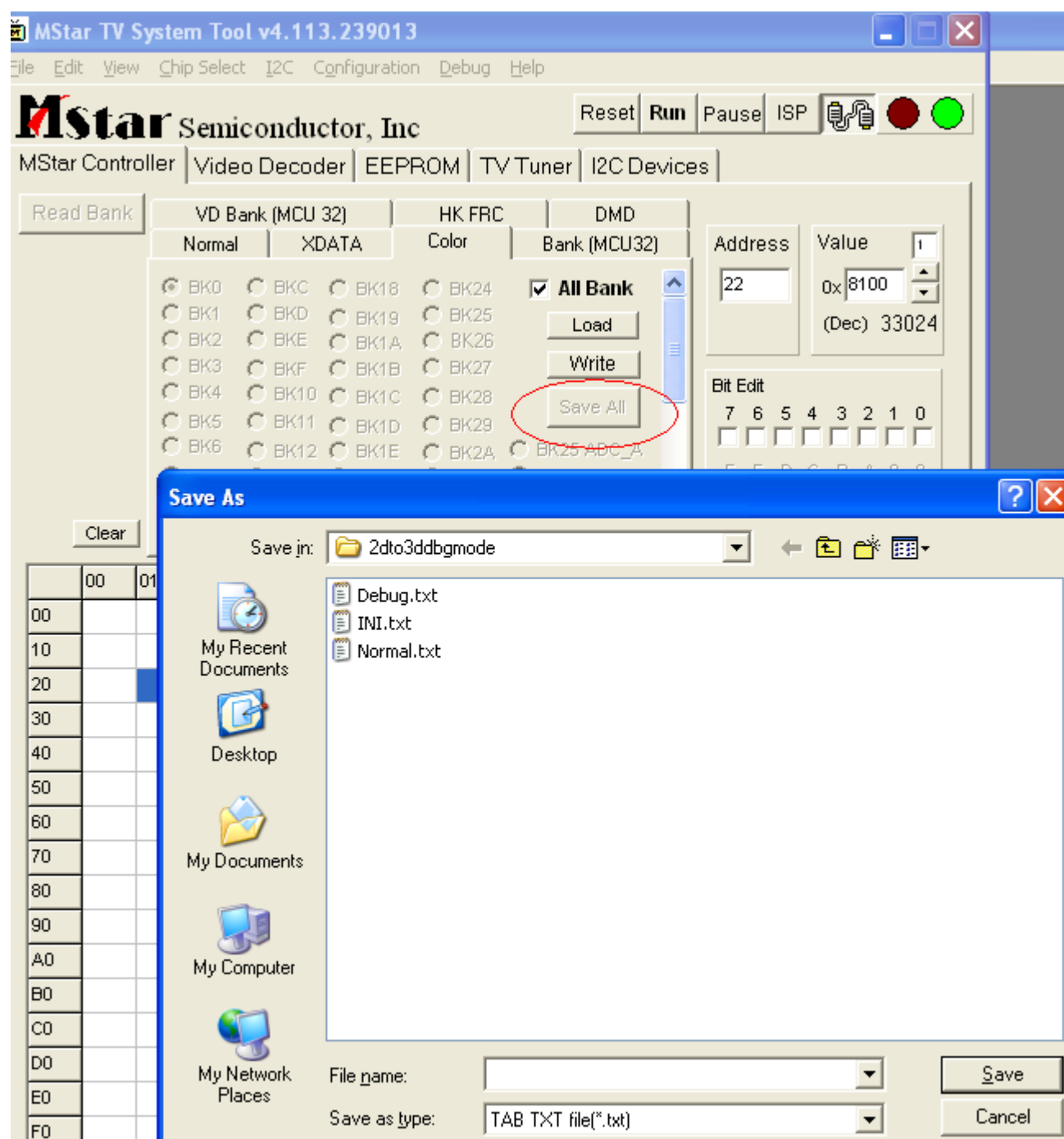
Address 0x00 位置应该是该 bank 号(如上图红圈所示), 如果不是的话, 代表有读取错。如果读取错, 首先要

检查有没有停住 cpu,然后再先点一下其他 BK,再点需要读取的 BK(如上图中 BKF,再点 BK10,这是 mstv tool 的问题),通常都可以读取正确值了。

如果要把 color bank 存到文件里给 scaler team debug:可以先勾选 All Bank,然后 read bank,把整个 color bank 读出来。如下图所示:



然后再点选 Save,保存到本地。如下图所示:



- 通过 XDATA 读取 scaler register

直接在 XDATA 输入 2 F 0 0, 0x2F00 代表 scaler 的 base address,在最左上角那个位置输入 subbank 号,代表具体是 scaler 的哪个 subbank,见下图,蓝色位置,目前查看的是 subbank0x12

**MStar TV System Tool v4.113.239013**

File Edit View Chip Select I2C Configuration Debug Help

**Mstar Semiconductor, Inc** Reset Run Pause ISP

MStar Controller Video Decoder EEPROM TV Tuner I2C Devices

Read Bank  
Read All Bank

VD Bank (MCU 32) Normal XDATA HK FRC Color Bank (MCU32) DMD

1st Base Address 0x 3 2 0 0 Access 1st Area NONPM 0

2nd Base Address 0x 2 F 0 0 Access 2nd Area

3rd Base Address 0x 0 B 0 0 Access 3rd Area

Addr Unit  
8 bit  
16 bit

Clear MCLK Freq (MHz) 100 Aeon TSP

Address Value  
00 0x 0012 (Dec) 18

Bit Edit  
7 6 5 4 3 2 1 0  
F E D C B A 9 8

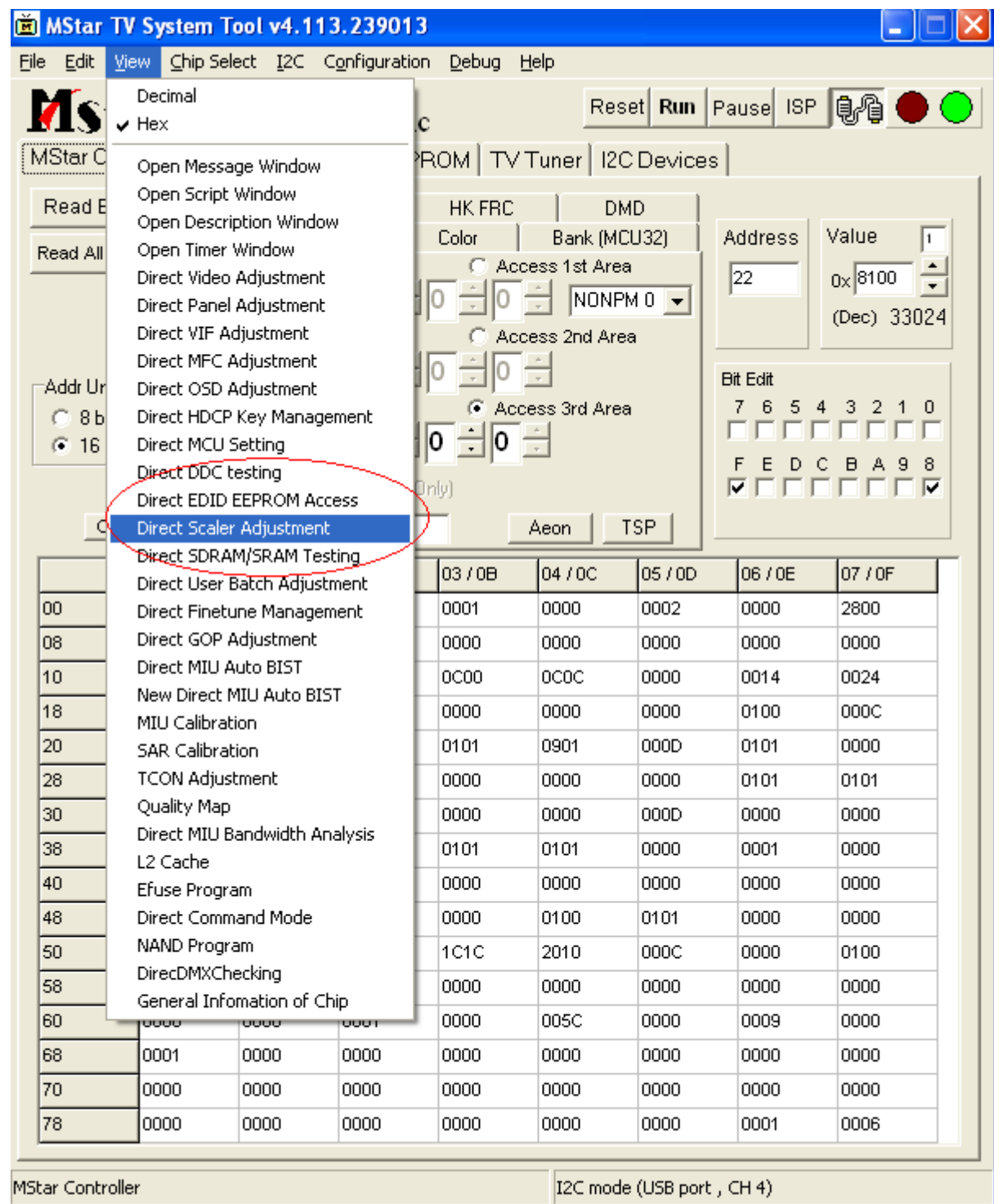
	00 / 08	01 / 09	02 / 0A	03 / 0B	04 / 0C	05 / 0D	06 / 0E	07 / 0F
00	0012	2100	07AA	0800	0001	0000	0000	0088
08	0D00	006C	0D00	0074	0D00	006C	0780	0780
10	0D00	006C	0D00	0074	0D00	006C	0780	03C0
18	1438	0003	0CFF	027C	0000	0000	0000	0000
20	2020	1818	3030	3030	0060	2828	0000	0000
28	0000	0000	0000	0000	0000	0400	5000	00FF
30	1000	0000	0000	0000	0000	0000	0000	0000
38	0000	A000	0702	C712	00BA	AA00	8FFF	0000
40	000B	2100	07AA	0800	0000	0000	0000	008B
48	0D00	006C	0D00	0074	0D00	006C	0780	0780
50	0E68	006C	0E68	0074	0E68	006C	0780	03C0
58	1438	0003	0CFF	027C	0000	0000	0000	0000
60	1010	1010	1010	0000	0000	2020	0000	0000
68	0000	0000	0000	0000	0000	0000	0000	0000
70	0000	0000	0000	0000	0000	0000	0000	0000
78	0000	0000	0002	0000	0000	AA00	0000	0000

MStar Controller I2C mode (USB port , CH 4)

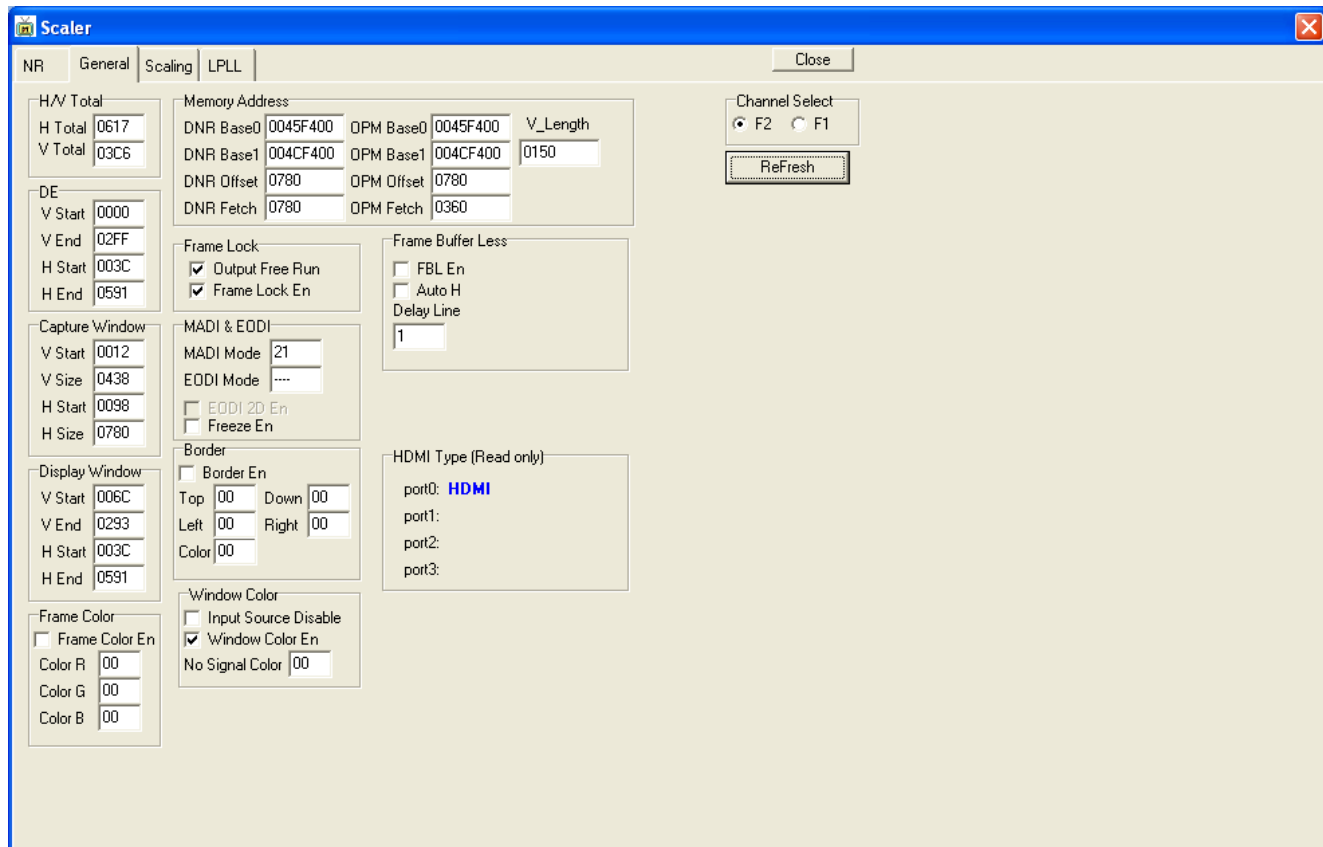
这里可以由用户选择用 8bit address 还是 16bit address 来看 register.

- 通过 Direct Scaler Adjustment page 读取 scaler information



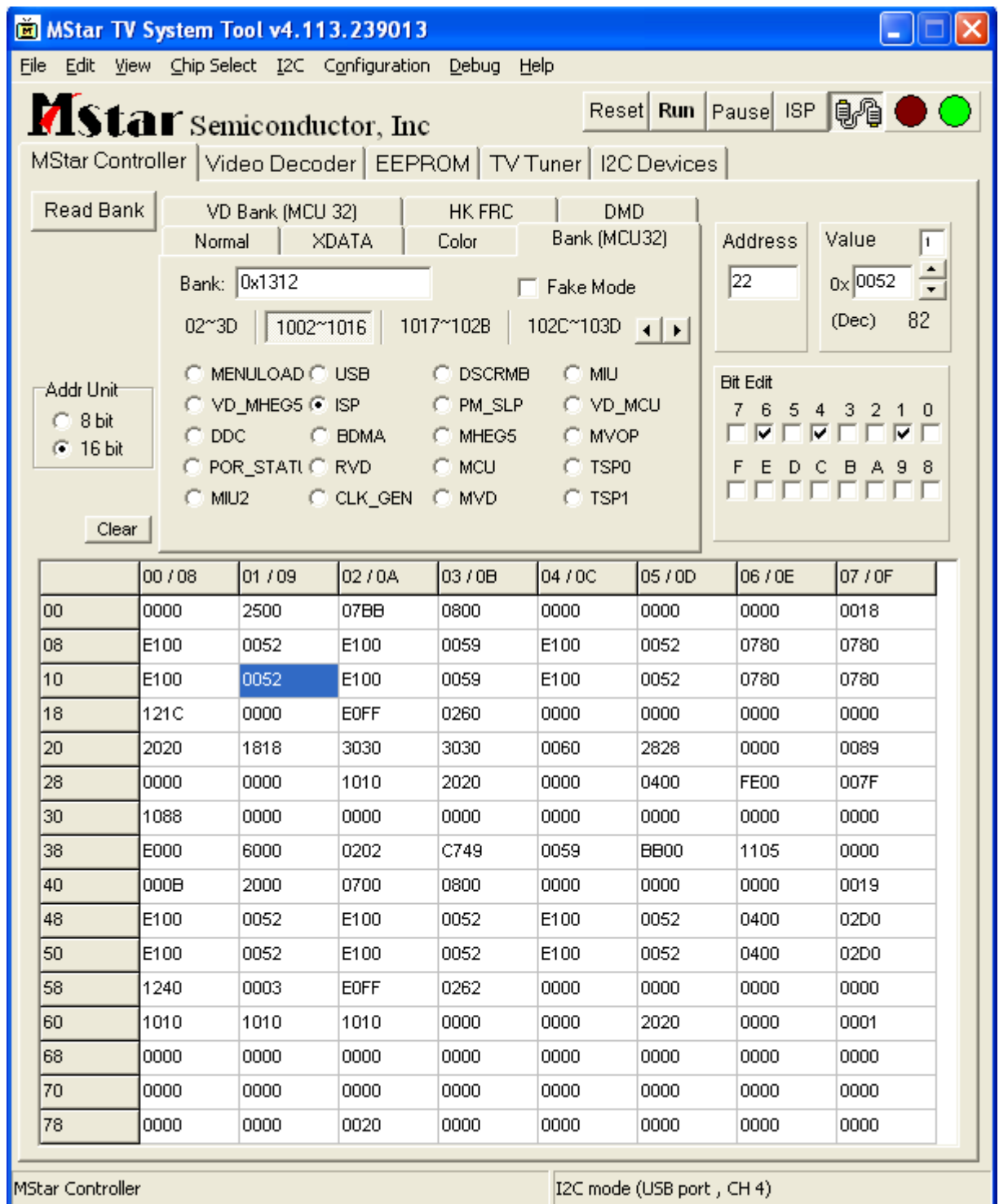






### ● Spread register 读取方法

前面几种读取 scaler register 的方式其实都要通过切 subbank 才能达成，但是这种方式不需要，直接通过另外一个 mapped base address(0x1300)进行直接访问。注意目前 tv ic 从 t8 开始支持这个功能,但 j2,a6,eden 没有这种 spread 读取方式。注意因为 spread 读取方法不需要切 subbank,因此可以不用停 cpu,具体读取方法是 Bank 栏输入:0x1300+subbank 号，如希望读取 scaler subbank0x12，则输入 0x1312，如下图:



## 11 如何通过 scaler register 了解 scaler information

### Input Timing width(capture window width):

Main window:subbank 0x1 16bit address 0x7

Sub window: subbank 0x3 16bit address 0x7

### Input Timing height(capture window height):

Main window: subbank 0x1 16bit address 0x6

Sub window: subbank 0x3 16bit address 0x6

**Input Timing 是否 interlace:**

Main Window:subbank 0x1 16bit address 0x1E bit11, 1 的话为 interlace,否则是 progressive

Sub Window:subbank 0x3 16bit address 0x1E bit11, 1 的话为 interlace,否则是 progressive

**Input vtotal:**

Main Window: subbank 0x1 16bit address 0x1F

Sub Window: subbank 0x3 16bit address 0x1F

**lp Input vfreq:**

Main Window: 12000000/(subbank0x1 16bit address 0x3B~0x3C 的 value)

Sub Window: 12000000/(subbank0x3 16bit address 0x3B~0x3C 的 value)

**Scaler input 端 source 是否有打开:**

Main Window: subbank 0x1 16bit address 0x2 的 bit7, bit7=0,代表 scaler input 有打开, 可以接收信号; bit7=1, 代表 scaler input 端是关闭的, 不可以接收信号, 没有工作。

Sub Window: subbank 0x3 16bit address 0x2 的 bit7, bit7=0,代表 scaler input 有打开, 可以接收信号; bit7=1, 代表 scaler input 端是关闭的, 不可以接收信号, 没有工作。

**是否有做 mirror:**

Main Window: subbank0x12 的 16bit address 0x3 的 bit12,代表 h mirror; bit13,代表 v mirror

Sub Window: subbank0x12 的 16bit address 0x43 的 bit12,代表 h mirror; bit13,代表 v mirror

**Output htotal:**

subbank 0x10 16bit address 0xC

**Output Vtotal:**

subbank 0x10 16bit address 0xD

**Output display window:**

Main Window: subbank 0x10 16bit address 0x8(HStart), 0x9(HEnd), 0xA(VStart), 0xB(VEnd)

Sub Window: subbank 0xF 16bit address 0x7(HStart), 0x8(HEnd), 0x9(VStart), 0xA(VEnd)

**是否有开 ucnr/ucdi:**

Subbank 0x2A 16bit address 0x2 的 bit15,bit13 以及 bit7 其中一个是否有 enable,enable 代表有开。或者 subbank 0x12 的 16bit address 0x27 的 bit0 是否=0, 如果=0, 代表有开

**几 frame mode check:**

3D version2 及以上,

Main Window:通过 subbank0x12 16bit address 0x19 的 bit0~bit4,具体数值:progrssive 代表几 frame mode; field,代表几 field mode

Sub Window:通过 subbank0x12 16bit address 0x59 的 bit0~bit4,具体数值:progrssive 代表几 frame mode; field,代表几 field mode

3D version0 以及 1,

Main Window: 4frame mode flag,在 subbank0x12 的 16bit address 0x7 的 bit13

3frame mode flag,在 subbank0x12 的 16bit address 0x4 的 bit7

Sub Window: 4frame mode flag,在 subbank0x12 的 16bit address 0x47 的 bit13

3frame mode flag,在 subbank0x12 的 16bit address 0x44 的 bit7

**是否 FBL:**

Main Window:subbank 0x12 16bit address 0x1 的 bit7,如果为 1, 则为 FBL,否则是 FB

Sub Window: subbank 0x12 16bit address 0x41 的 bit7,如果为 1, 则为 FBL,否则是 FB

### 是否画面 Freeze:

Main Window: subbank 0x12 16bit address 0x1 的 bit11, 如果为 1, 则画面 freeze; 否则画面不 freeze

Sub Window: subbank 0x12 16bit address 0x41 的 bit11, 如果为 1, 则画面 freeze; 否则画面不 freeze

### 是否有开屏:

Main window: subbank 0x10 的 16bit address 0x19 的 bit1, bit1=0, 代表开屏; bit1=1, 代表关屏

Sub window: subbank 0x10 的 16bit address 0x19 的 bit5, bit5=0, 代表开屏; bit5=1, 代表关屏

### Scaler op 是否 enable:

Main Window: subbank 0x20 的 16bit address 0x10 的 bit0=1

Sub Window: subbank 0x20 的 16bit address 0x10 的 bit1=1

### 是否有做 frame lock 标志:

BK31(注意不是 scaler subbank), 16bit address 0xC 的 bit3 是否为 1, 如果为 1, 代表有做 frame lock; 0, 代表 freerun

### 是否锁住:

BK31 16bit address 0x2A 的 bit8=1

### Fpll Input vfreq :

$12000000 / (0x21 \sim 0x22 \text{ 的 value}) * ((0x0C \text{ 的 bit8} \sim \text{bit11 的 value}) + 1)$ , 跟 ip input vfreq 一样

### Fpll Output vfreq:

$12000000 / (0x23 \sim 0x24 \text{ 的 value}) * ((0x0C \text{ 的 bit12} \sim \text{bit15 的 value}) + 1)$ , fpll 后的 output vfreq

一般情况下只有 3D 以及 PIP 下才需要看 sub window 的 register.

## 12 Basic debug skills

### ● Check msg 和 color bank 中的异常

通过上一段讲述的方法, 读取到 scaler information。可以根据已知的一些 timing 信息跟目前从 msg 和 register 得到的 scaler information 去比对, 检查不一致异常的地方, 以次为突破口, 继续 dbg 下去。纯粹 3D 的 register 基本上类似于 load table, 实际中不太会出问题, 如果不放心, 可以拿 ap note 中的 3D reg table 跟 mstv tool 读出来的进行比较。

### ● 如果有涉及到 sub window, 单独 disable main window 和 sub window 去 check

具体 disable 方法, 可以参见上一段的” Scaler op 是否 enable”。注意如果 main window disable 后, subwindow 会黑屏的话, 请把 sub window 的 display window 调成全屏, 具体调整 register 请参见上一段的” Output display window”

如果只有一个 window 有问题, 就可以:

- 检查 register 把好 ok 的 window 的相关 register 踩到 ng 的 window 上。常见的几个容易踩的 register bank 是:

Subbank 0x12: main 部分 register(0x1 开始)对应踩到 sub 部分 register(0x41 开始)

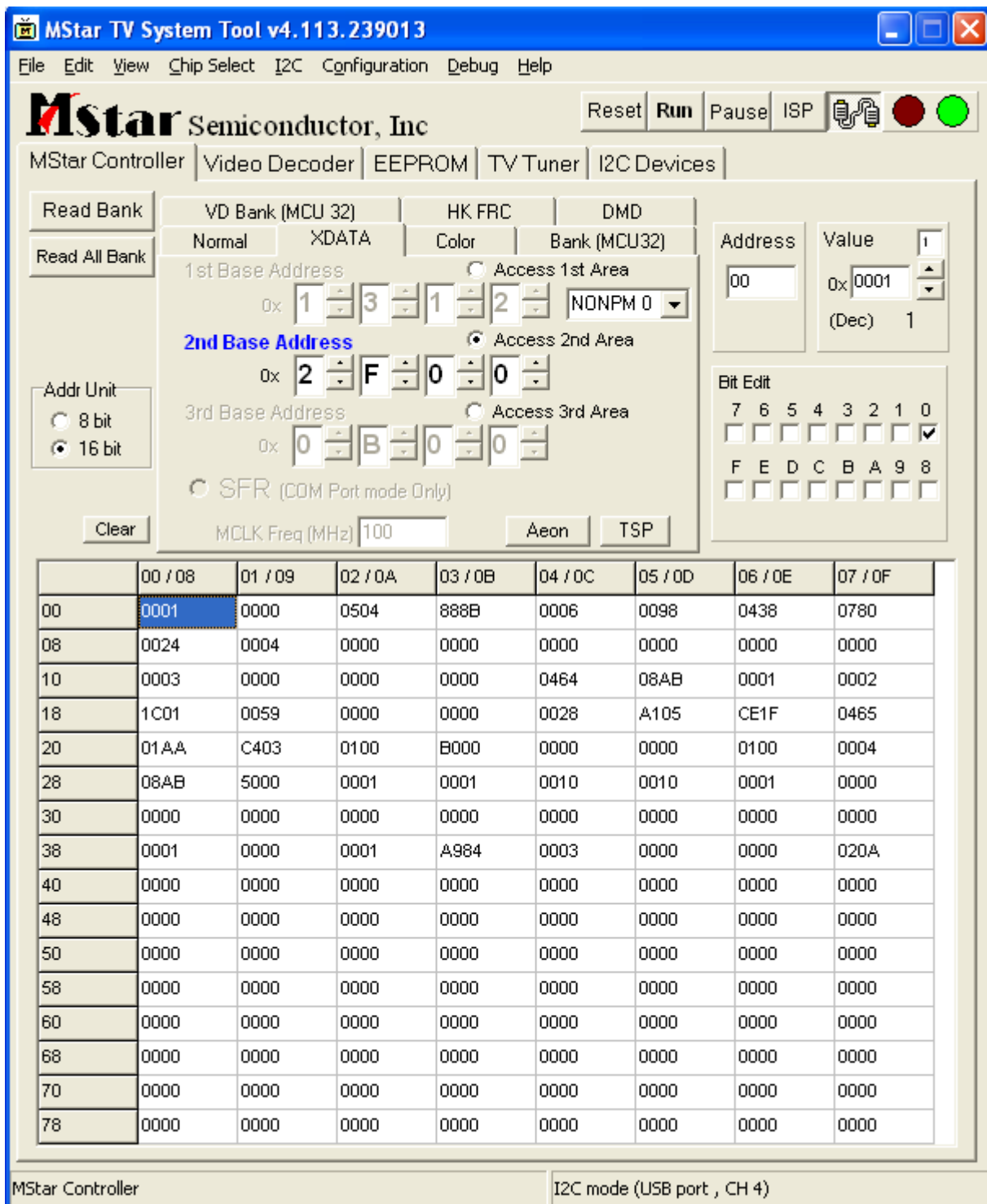
Subbank 0x23: main 部分 register(0x1 开始)对应踩到 sub 部分 register(0x21 开始)

Subbank 0x1: main ip1 bank 对应 踩到 Subbank 0x3: sub ip1 bank

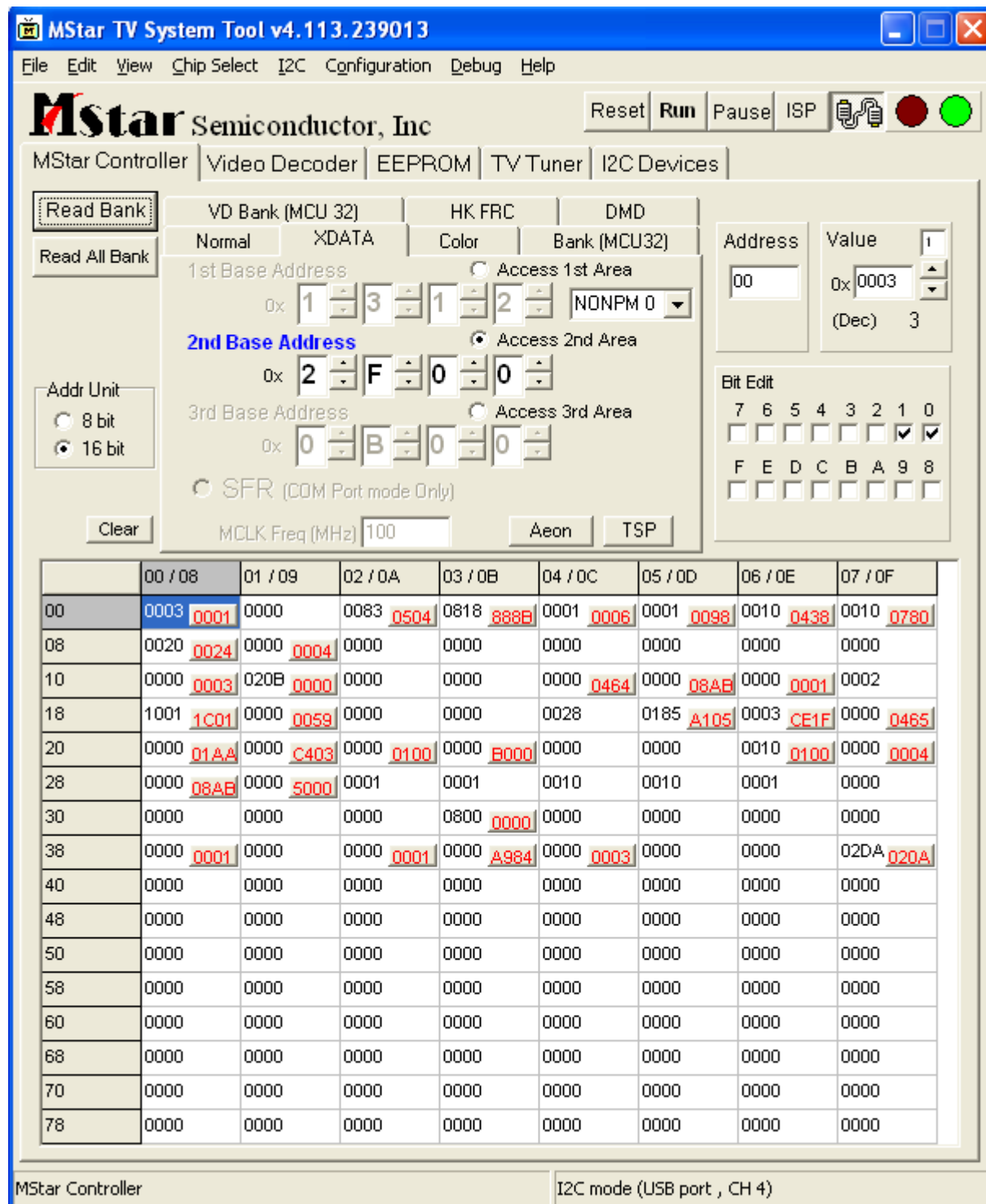
Subbank 0x2: main ip2 bank 对应踩到 Subbank 0x4: sub ip2 bank

subbank 0x1 踩到 subbank 3 可以这样踩:

1 读取 subbank 0x1



- 2 Debug 菜单中选 Store to Memory, compare with Memory,
- 3 切换到 subbank0x3,0x00 位置输入 0x3,如下图:



- 4 把红色部分 1 个个手工踩进去，注意不要踩 0x00 位置。
- 检查 sub window 和 main window msg 是否一样
- review code 是否 main/sub 处理差异
- 重点检查 sram3 跟 sram1 是否一致，sram4 跟 sram2 是否一致，具体方法见” **sram bypass 方法以及如何 dump sram**”
- 找到一个 ok 的 case
  - 1 最好是找到一个环境类似的 ok case,比如 timing, input 尺寸等一样。不然退而求其次，找稍微有点不一样的 case，看是否 ok or NG.

2 Check 公版

3 老 code

一旦找到一个 ok 的 case,就可以通过下面讲述的踩 register 或者夹 code 方法来做 debug.

- **踩 register**

这招是最容易也最经常要使用的 debug skill.

➤ 把好的 color bank 读取出来

勾选 All Bank, 然后 Read Bank,这样会把整个 color bank 的值读出来

**MStar TV System Tool v4.113.239013**

File Edit View Chip Select I2C Configuration Debug Help

**MStar Semiconductor, Inc**

MStar Controller | Video Decoder | EEPROM | TV Tuner | I2C Devices

Reset Run Pause ISP

**Read Bank**

VD Bank (MCU 32) | HK FRC | DMD

Normal | XDATA | Color | Bank (MCU32)

☐ BK0 ☐ BKC ☐ BK18 ☐ BK24 ☐ All Bank  
☐ BK1 ☐ BKD ☐ BK19 ☐ BK25  
☐ BK2 ☐ BKE ☐ BK1A ☐ BK26  
☐ BK3 ☐ BKF ☐ BK1B ☐ BK27  
☐ BK4 ☐ BK10 ☐ BK1C ☐ BK28  
☐ BK5 ☐ BK11 ☐ BK1D ☐ BK29  
☐ BK6 ☐ BK12 ☐ BK1E ☐ BK2A ☐ BK25 ADC\_A  
☐ BK7 ☐ BK13 ☐ BK1F ☐ BK2B ☐ BK26 ADC\_D  
☐ BK8 ☐ BK14 ☐ BK20 ☐ BK2C ☐ BK35 AFEC

Load Write Save

Clear

Address: 00 Value: 0x00 (Dec) 0

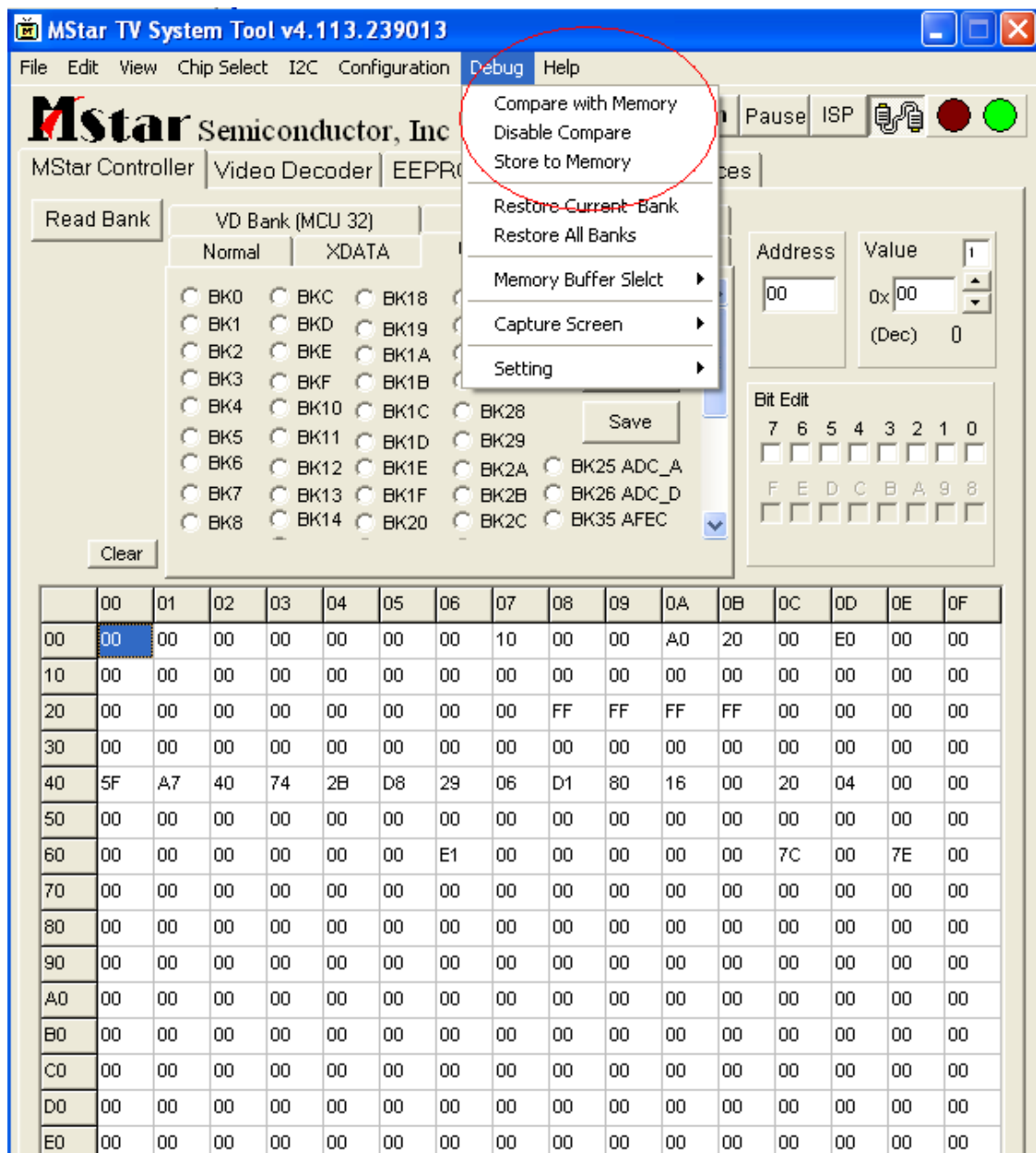
Bit Edit

	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
00	00	00	00	00	00	00	00	10	00	00	A0	20	00	E0	00	00
10	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
20	00	00	00	00	00	00	00	00	FF	FF	FF	FF	00	00	00	00
30	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
40	5F	A7	40	74	2B	D8	29	06	D1	80	16	00	20	04	00	00
50	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
60	00	00	00	00	00	00	E1	00	00	00	00	00	7C	00	7E	00
70	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
80	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
90	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
B0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
C0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
D0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
F0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

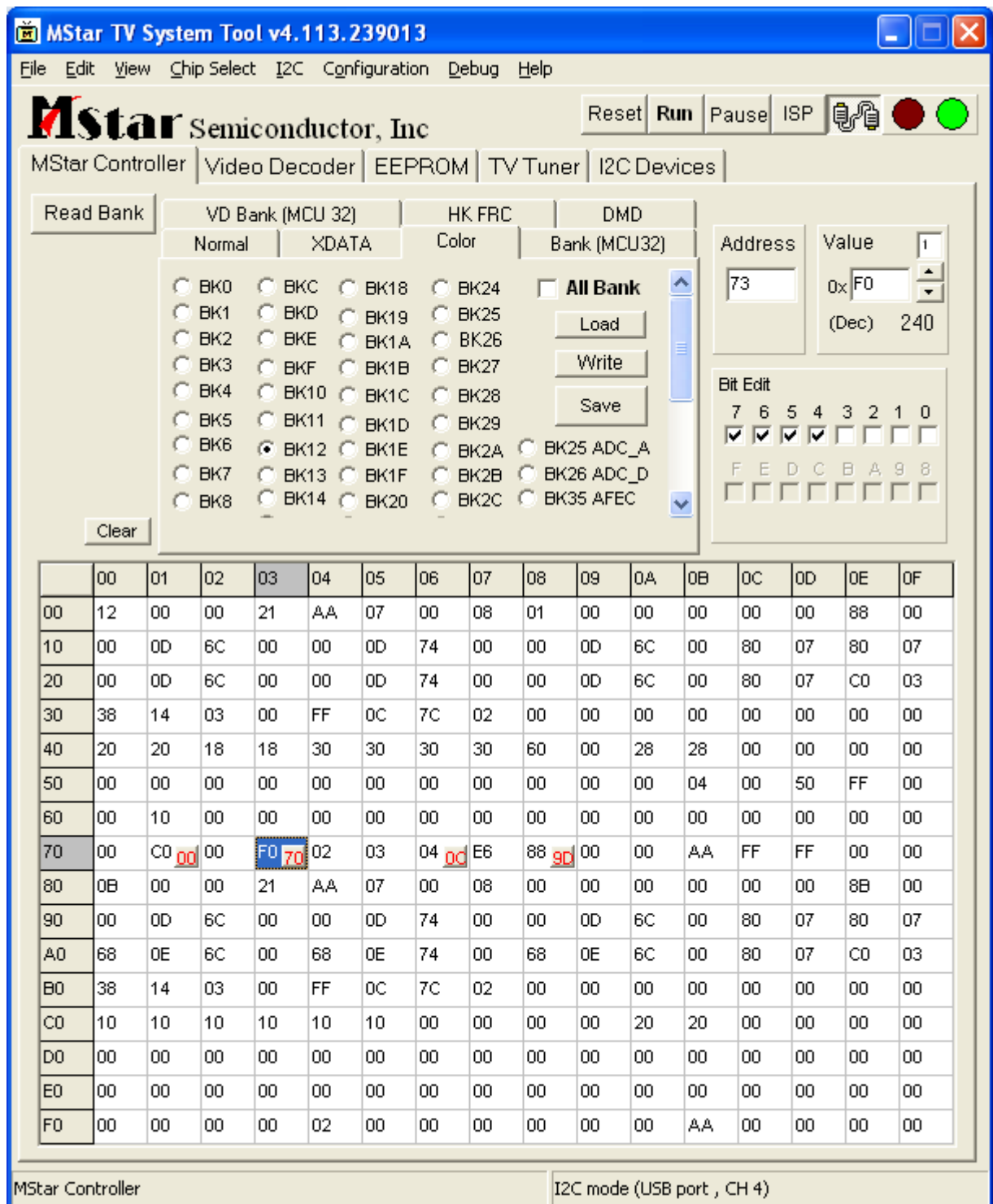
MStar Controller I2C mode (USB port , CH 4)

- 保存当前读出来的 color bank 然后在 Debug 菜单中先勾选 Store to Memory, 然后 compare with Memory

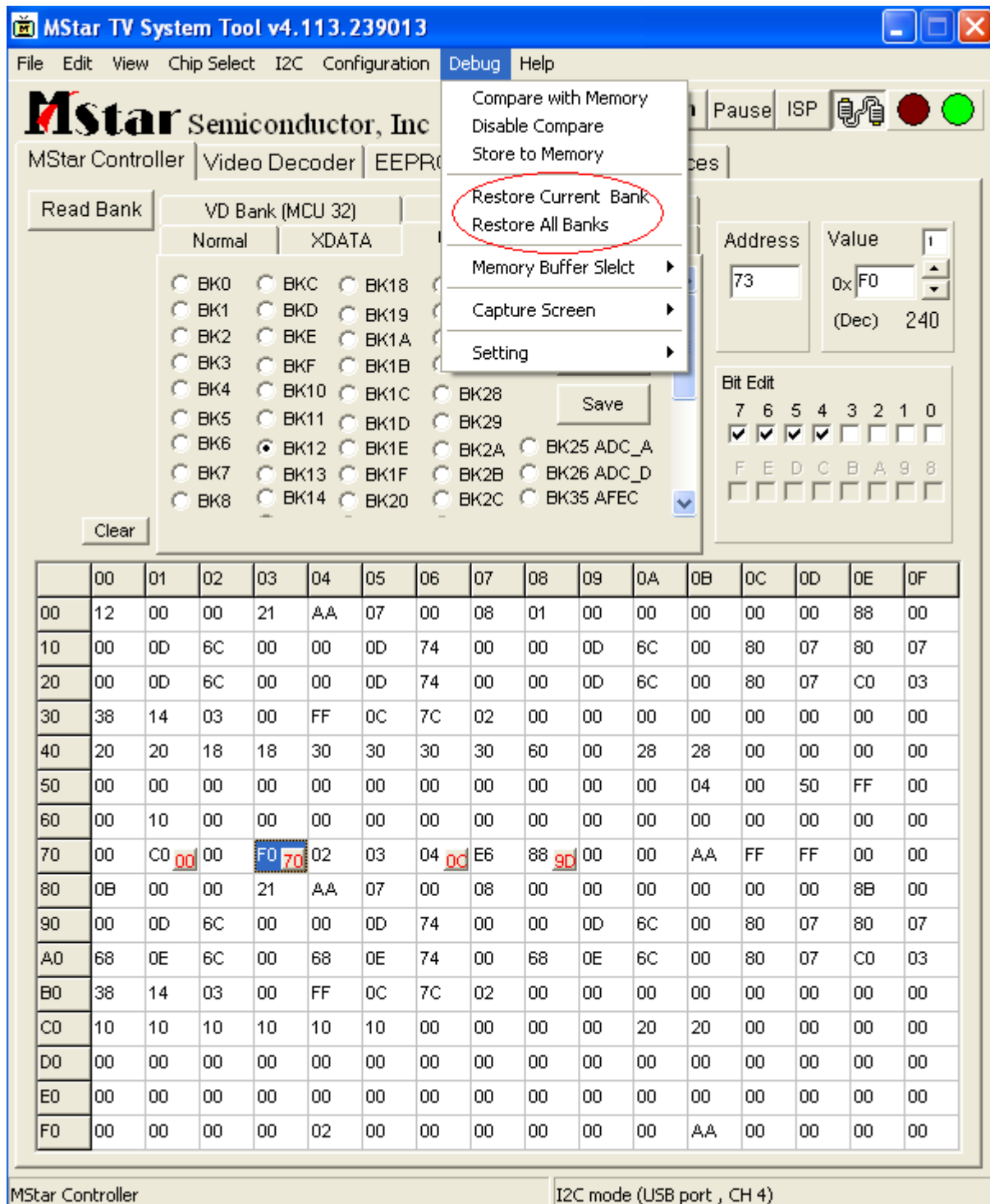




- 读取 NG 时的 color bank，跟读 ok color bank 方法一样，此时应该会看到红色的 register,那代表之前存进 memory 的 ok 的 color bank



- 手工点选红色按钮 resotre 或者 Debug 菜单中 Restore Current Bank(恢复当前 bank),Restore All Banks(恢复所有 Color Bank)



## ● 夹娃娃

夹娃娃也分好几种：

- 如果以前 code 是好的，就可以夹 CL#
- 如果已经知道某个 register 能改好，但是不知道 code 里是哪里把它改坏的。就可以通过 2 分法在可疑 code 中打印从 hw 读取的 register 值进行夹 code
- 通过 code 里加 while 死循环以及 dump register+mstv tool 等夹 register

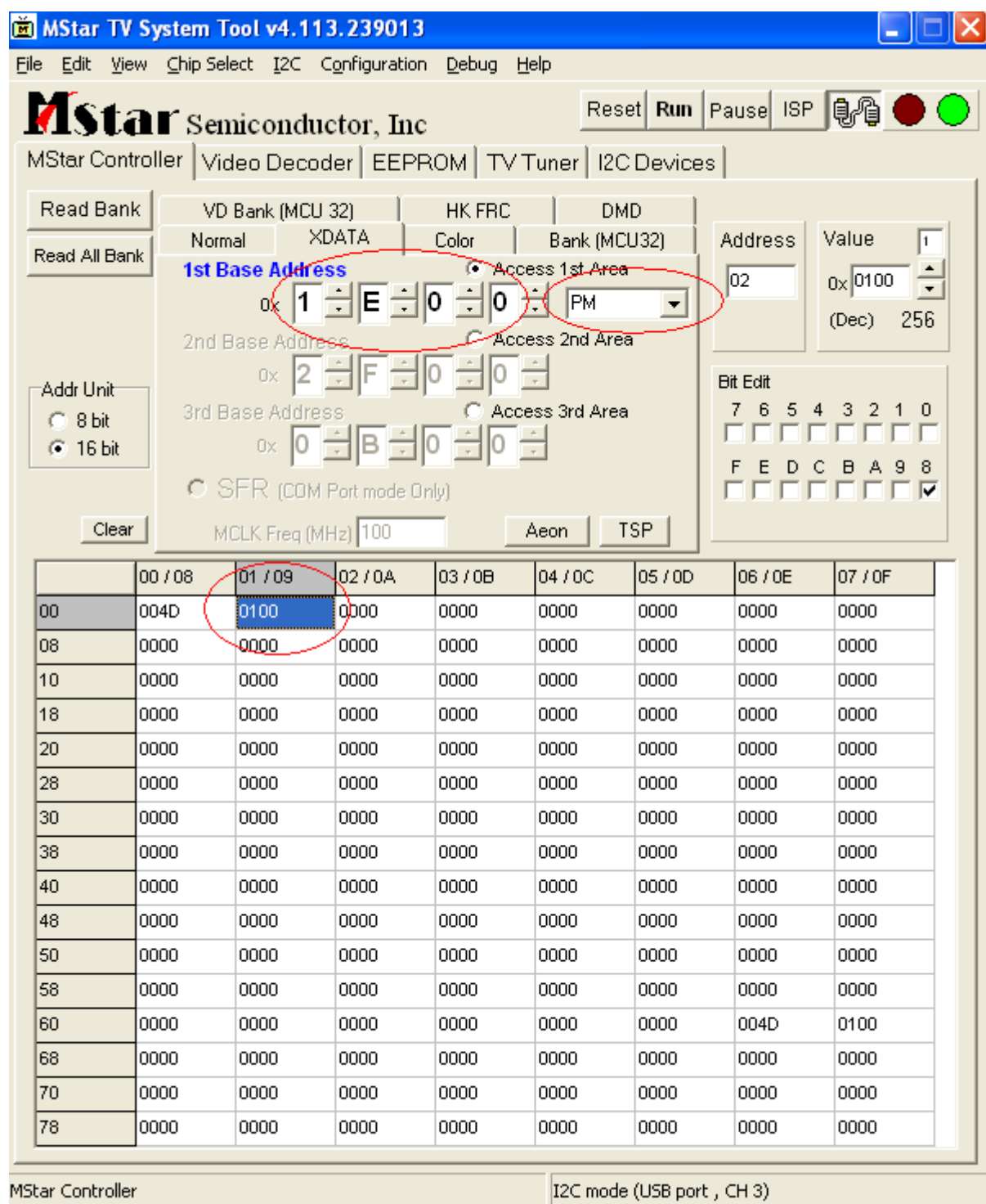
➤ 交叉验证

Ok 和 ng 的平台 Ic, 板子, code, panel, mboot, chip revision, qmap, 信号源, 仪器, timing, 操作方式等是否一样。

如果不一样, 可以把 ok 的模块替换到 ng, 或者也可以反过来, 把 ng 的模块替换到 ok 的模块进行交叉验证。

这里讲一个 chip revision 的读法, 在实际使用中经常可以用到:

PM bank 0x1E00 16bit address 0x1 的高 8bit value, 具体 u0 几是该数值+1, 如下图, 是 u02



**MStar TV System Tool v4.113.239013**

File Edit View Chip Select I2C Configuration Debug Help

MStar Semiconductor, Inc

MStar Controller | Video Decoder | EEPROM | TV Tuner | I2C Devices

Read Bank | Read All Bank

VD Bank (MCU 32) | HK FRC | DMD

Normal | XDATA | Color | Bank (MCU32)

1st Base Address: 0x 1 E 0 0 PM

2nd Base Address: 0x 2 F 0 0

3rd Base Address: 0x 0 B 0 0

Addr Unit: 8 bit | 16 bit

Clear | MCLK Freq (MHz) 100 | Aeon | TSP

Address: 02 | Value: 0x 0100 (Dec) 256

Bit Edit: 7 6 5 4 3 2 1 0

	00 / 08	01 / 09	02 / 0A	03 / 0B	04 / 0C	05 / 0D	06 / 0E	07 / 0F
00	004D	0100	0000	0000	0000	0000	0000	0000
08	0000	0000	0000	0000	0000	0000	0000	0000
10	0000	0000	0000	0000	0000	0000	0000	0000
18	0000	0000	0000	0000	0000	0000	0000	0000
20	0000	0000	0000	0000	0000	0000	0000	0000
28	0000	0000	0000	0000	0000	0000	0000	0000
30	0000	0000	0000	0000	0000	0000	0000	0000
38	0000	0000	0000	0000	0000	0000	0000	0000
40	0000	0000	0000	0000	0000	0000	0000	0000
48	0000	0000	0000	0000	0000	0000	0000	0000
50	0000	0000	0000	0000	0000	0000	0000	0000
58	0000	0000	0000	0000	0000	0000	0000	0000
60	0000	0000	0000	0000	0000	0000	004D	0100
68	0000	0000	0000	0000	0000	0000	0000	0000
70	0000	0000	0000	0000	0000	0000	0000	0000
78	0000	0000	0000	0000	0000	0000	0000	0000

MStar Controller | I2C mode (USB port , CH 3)

- **sram bypass 方法以及如何 dump sram**

画面不良问题，通常如果踩 register 踩不好,可能就要怀疑是否 sram 有问题.通常判断 sram 问题可以去

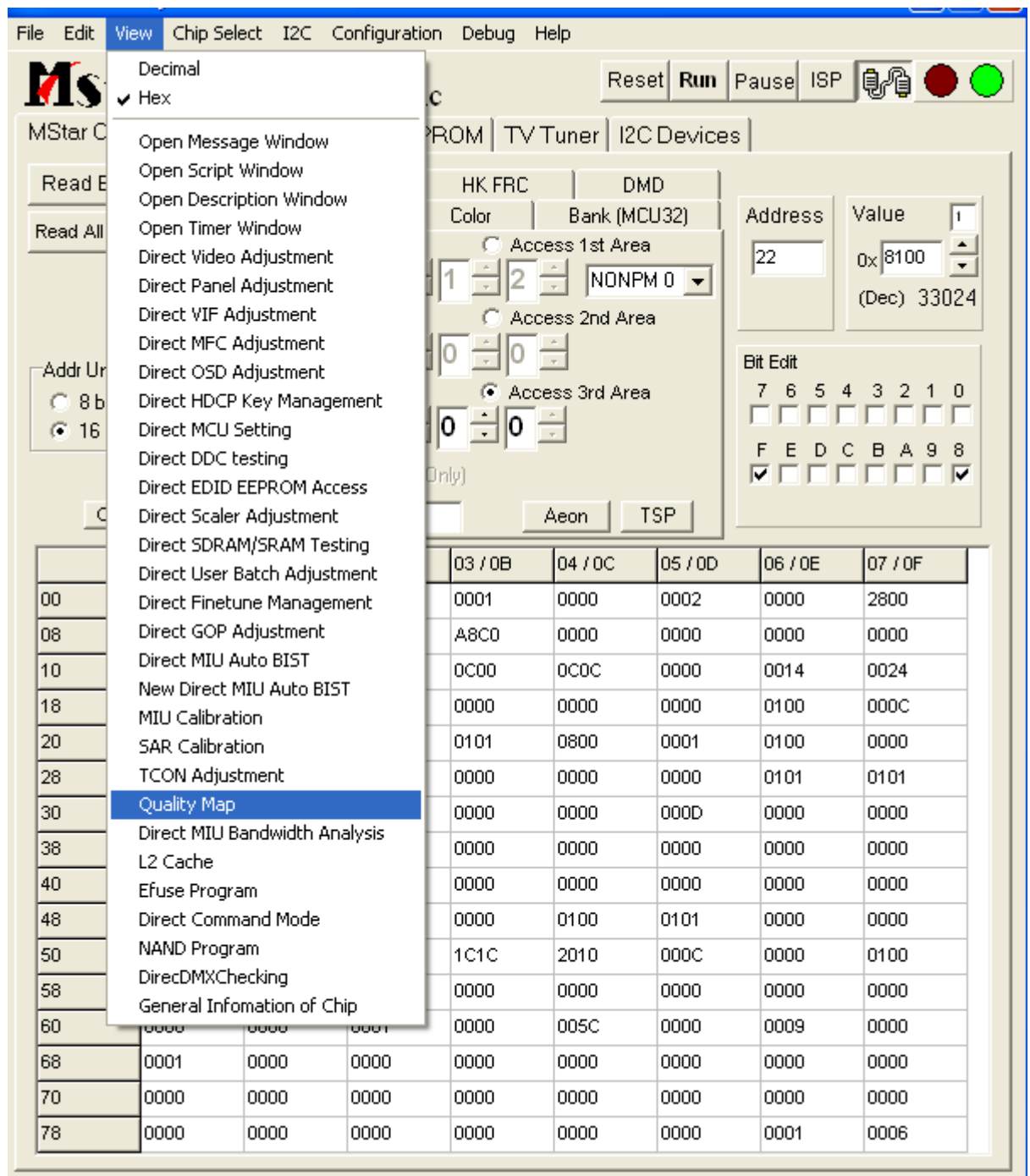
bypass sram，看画面是否正常。如果正常，基本可以确定 sram 有问题，或者 sram 选的 filter 有问题。具体 register 如下：

Main window: scaler subbank0x23 的 16bit address 0x0B 写成 0x0

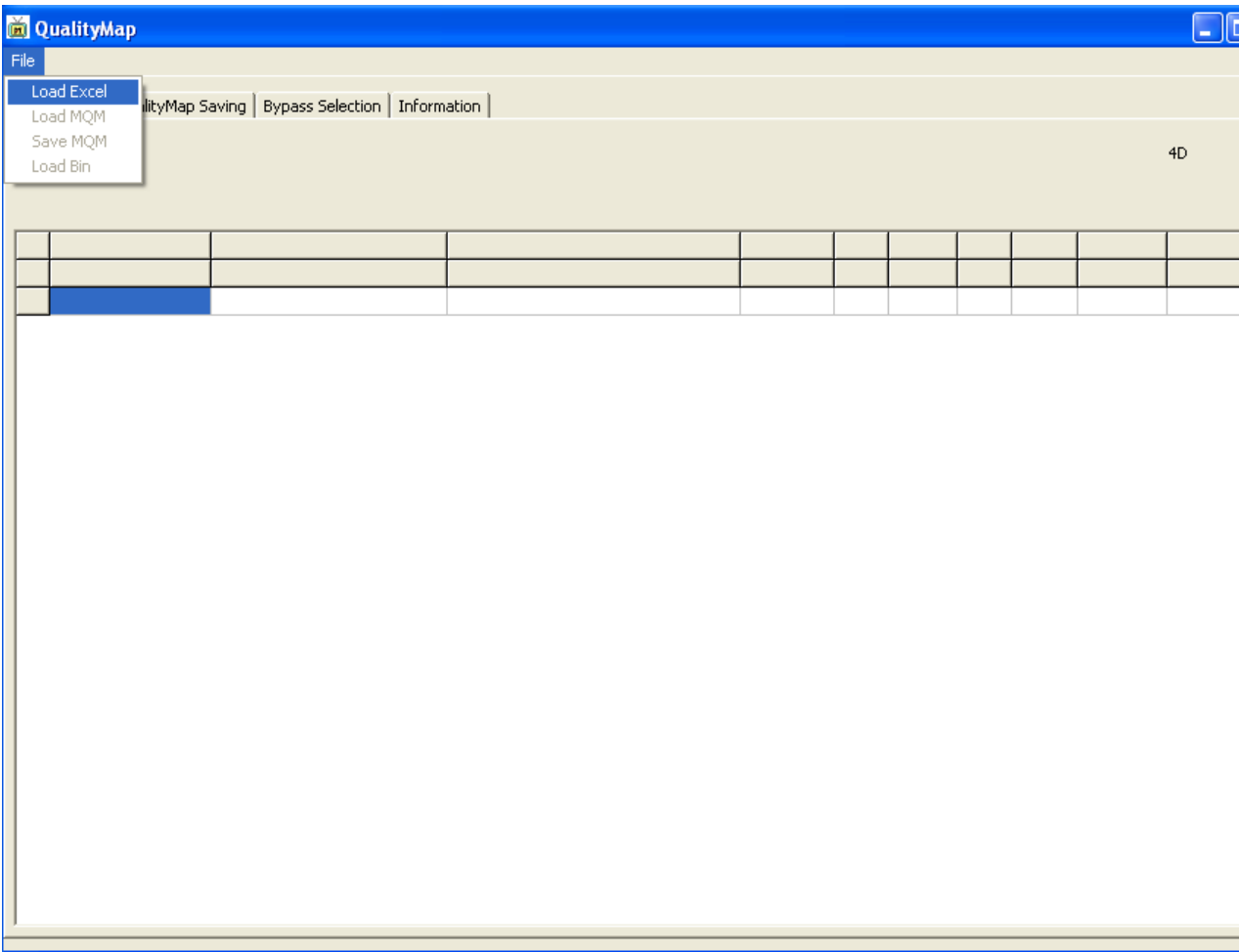
Sub window window: scaler subbank0x23 的 16bit address 0x2B 写成 0x0

一旦确定是 sram 问题，就要开始着手 dump sram:

1 View 中选择 Quality Map

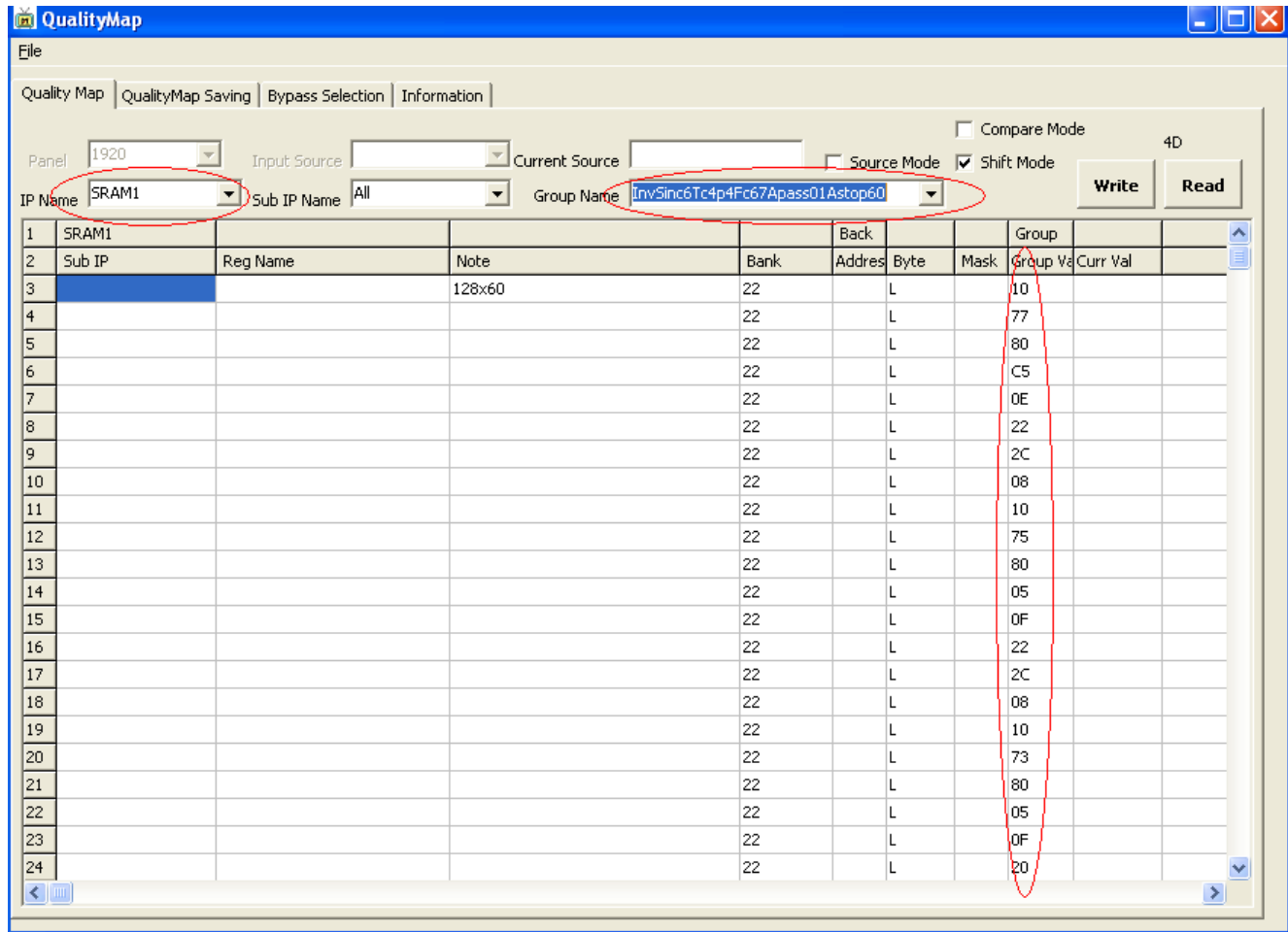


2 File 中 Load Excel



3 check sram

IP name 中有 4 个 item: SRAM1, SRAM2, SRAM3, SRAM4. 其中 SRAM1, SRAM2 是属于 main window 的, SRAM3 和 SRAM4 属于 sub window. 选择某一项 SRAM, 在 Group Name 中选择具体某一项 index, 在 table Group Val 列中就可以列出这项 index 对应的所有 sram value, 如下图:



4 通过上图的 Read 按钮, 读到当前 hw 中的 sram 值显示在 Curr Val 中。通过这个 sram 值可以比对 Group Val, 确定当前 hw 选用的是哪个 sram index. 如果没有一个对不少, 就要考虑 sram load function 是否写的有问题或者有多进程冲突导致写 sram 异常问题。

5 然后再根据这个 sram index, 从 qmap 和代码分析是否合理。

6 另外也可以把选中的那组 Group Val 通过 Write 按钮写到 hw 去, 已理清问题。

### ● 尝试公版是否能复制出问题

其实跟前夹娃娃和踩 register 一样, 就是设法找到一版 ok 的版本, 然后就可以应用上面段落中的方法。

另外公版复制问题还有一个好处是, 即使 NG, 也没关系, 在没有线索的情况下, 就可以直接找 scaler team 寻求帮助, 因为一般情况下 scaler team 也可以复制出问题了。

### ● 提供有效信息找 scaler team 寻求 help

Issue 请上 mantis, 为了减少沟通以及 sync 成本, 尽可能详细的提供如下信息:

#### ➤ 平台信息

是否 mirror panel, 2d or 3d, 具体 ic, 板号, 什么 platform(non os, supernova, android), 1ddr or 2ddr, panel 分辨率, 有无接其他 ic 等重要信息,

#### ➤ Timing 信息

具体 Input source, 分辨率, progressive or not, vfreq

- Sw 信息
  - utopia/ap code 位置, 以及 CL#,修改文件,build code,烧写 code 方法
- debug 信息
  - 提供 scaler msg 和 color bank

## 13 XC Issue Debug 基本流程

### 1 复制 issue

尽量 sync 报 issue 的环境, 重点看以下这些内容是否一样:

image 版本, mboot,panel,板子型号, chip revision,仪器, mirror or non-mirror,具体 timing info, 复制的步骤等。

如果复制不到, 找 report issue 的 owner sync 环境, 必要时进行交叉验证来 sync issue 状况。

### 2 打 pattern 确认 issue 位置

打 Pattern 的意思是从 show 出该 pattern 的 ip 所在位置开始, 原始的 miu data 会被 pattern 取代后显示在 panel 位置上, 用来理清到底是否是 scaler 问题以及 scaler 哪个阶段的问题。如果 pattern 是好的, 基本上可以排除之后的 hw ip 都没有问题, 除了某些用 pattern 不易察觉的 bug 外(打个比方, 比如细微线条, 跟 pattern 类似等);如果 pattern 也是坏的, 说明出问题的 hw ip 在这个 pattern 的 ip 之后。

### 3 log xc msg 以及 color bank 分析

这个分析需要经验,建立在对 xc 比较了解的基础上。可以根据已知的一些 timing 信息跟目前从 msg 和 register 得到的 scaler information 去比对, 检查不一致异常的地方, 以次为突破口, 继续 dbg 下去。

### 4 比较 msg 和踩 color bank

这步不需要太多 xc 的专业技能, 重点是找到一个 ok 的 case 进行对比。可从以下几个方面着手:

- 找到一个环境类似的 ok case, 比如仅仅是 ic 不一样, 或者仪器不一样, panel 不一样 s 等的 ok case. 尽量保持差异最小, 这样才具有可踩性。
- Check 公版, 很有可能公版是 ok 的 case
- 之前的 code
- 第 5 步夹娃娃确定 ok, 但还不知道具体 reg 以及进一步信息的 case

### 5 夹娃娃

跟第 4 步一样, 如果找到一个 ok 的 case, 但是通过第 4 步无法解决, 这时可以尝试夹娃娃的方式。

- 如果以前 code 是好的, 就可以夹 CL#
- 如果已经知道某个 register 能改好, 但是不知道 code 里是哪里把它改坏的。就可以通过 2 分法在可疑 code 中打印从 hw 读取的 register 值进行夹 code
- 通过 code 里加 while 死循环以及 dump register+mstv tool 等夹 register
- 交叉验证
  - Ok 和 ng 的平台 Ic, 板子, code, panel, mboot, chip revision, qmap, 信号源, 仪器,



timing, 操作方式等是否一样。

如果不一样, 可以把 ok 的模块替换到 ng, 或者也可以反过来, 把 ng 的模块替换到 ok 的模块进行交叉验证。如果可以做到 ok, 反过来, 可以继续用第 4 步比较 msg 和踩 color bank。

## 6 查 function or flow

通过第 4, 5 步, 一般情况已经确定是具体哪个 reg 或者哪样的异常动作引起了问题。这时应该可以通过搜索 reg 查到 function, 或者通过 gdb, 打印等方式查看具体 flow, 这已经跟普通非 xc issue 没有太大区别, 不再进行详述。

注意, 第 3 步跟第 4 步的顺序不是固定不变的, 具体 issue, 看哪个步骤比较方便, 方便的步骤可以先进行。