MID::DVT１モジュール/IMX500 Driver修正（２）

Raspberry Pi Imx500 Driver Modify 2 2024.11.25

Nov 21, 2024

Nov 25, 2024 更新

更新内容

2024/11/25

- 使用するlinuxのsourceを現状installされる最新版(6.6.51)に変更してbuildし直し
- 生成されたimx500.ko.xzを含むzipファイルに更新
- 上記に合わせてドキュメントを修正
- v4l2-ctrlのリスト出力の結果を追加

まとめ

- 目的はregisterのwrtiteをできるようにすること
- V4L2のcustom controlとして"imx500_reg_io"を追加した
- python scriptも作成した(複数パラメータがあるのでコマンドラインの"v4l2-ctl"から実行はできない)
- Read/Writeの両方に対応
- Address範囲にはlimitをかけていないので使用者が注意すること
- Input Tenosr表示やimx500内蔵ISPによるAe/Awb制御の動作確認用に修正したpicamera2 scriptを準備（修正内容は別資料）
- 同じkernel versionであれば、他のRaspberry Piでも".ko.xz"をcopyすることでテストは可能
- 動作確認はRaspberry Pi 5のみで実施

imx500_reg_io.pyの仕様

```
usage: imx500_reg_io.py [-h] [-a ADDRESS] [-b BIT_WIDTH] [-m MASK] [-d DATA]
                        [-w] [-n NUMBER_OF_BYTES] [-i INPUT_FILE]
                        [-o OUTPUT_FILE] [-l]
```

short format	long fomat	description	default
-h	--help	show this help message and exit	
-a ADDRESS	--address ADDRESS	read/write address	0

-b BIT_WIDTH	--bit-width BIT_WIDTH	bit width 8/16/32	8
-m MASK	--mask MASK	mask	no mask
-d DATA	--data DATA	write data	0
-w	--write	enable write	False
-n NUMBER_OF_BYTES	--number-of-bytes NUMBER_OF_BYTES	read bytes, max=2048	1/2/4
-i INPUT_FILE	--input-file INPUT_FILE	input file name <32bit,big-endian>	
-o OUTPUT_FILE	--output-file OUTPUT_FILE	output file name <32bit,big-endian>	
-l	--little-endian	print little endian like	False

使用例

1. レジスタからデータを読む (Address: 0xd000 の場合）

```
$ python imx500_reg_io.py -a 0xd000
```

- デフォルトは8bit長

2. レジスタからデータを16bit長で読む (Address: 0xd000 の場合）

```
$ python imx500_reg_io.py -a 0xd000 -b 16
```

3. レジスタから複数データを32bit長で読む (Address: 0xd000, 12Byte の場合）

```
$ python imx500_reg_io.py -a 0xd000 -n 12 -b 32
```

- 32bit(4byte)表示で12バイトなので3行表示される

4. レジスタにデータを書く (Address: 0xd000, Data: 0x10 の場合）

```
$ python imx500_reg_io.py -a 0xd000 -d 0x10 -w
```

- dataは10進でもよい（例えば上記では16）

5. レジスタにマスクを使ってデータを書く (Address: 0xd000, Data: 0x0f, Mask: 0x02 の場合）

```
$ python imx500_reg_io.py -a 0xd000 -d 0x0f -m 0x02 -w
```

- maskとdataのANDがregisterに書かれる（上記では0x2が書かれる）
- maskを省略または0とすると0xff(bit長による)となりマスクはなしになる

file 一覧 (圧縮ファイルあり）(2024/11/25更新)

- **20241122_imx500_driver_modify.zip** 20241122_imx500_driver_modify.zip 🆕

```
├── imx500.c
├── imx500_diff_20241122.patch
├── imx500.ko.xz
├── picamera2
│   ├── imx500_classification_demo.py
│   ├── imx500_object_detection_demo.py
│   ├── imx500_pose_estimation_higherhrnet_demo.py
│   └── imx500_segmentation_demo.py
├── python
│   ├── device_id.py
│   ├── imx500_reg_io.py
│   ├── readme.txt
│   ├── read_register.py
│   └── wb_gains.py
└── readme.txt
```

Driver binaryの適用 (Buildしない場合)

- Raspberry Pi 5で確認
- 同じkernel Versionであればimx500.ko.xzのみのcopyで動作可能

```
$ uname -a
Linux raspberrypi 6.6.51+rpt-rpi-2712 #1 SMP PREEMPT Debian 1:6.6.51-1+rpt3 (2024-10-08) aarch64 GNU/Linux
```

手順

1．backup

```
$ mkdir original
$ cp -p /lib/modules/6.6.51+rpt-rpi-2712/kernel/drivers/media/i2c/imx500.ko.xz original
```

2. copy

```
$ sudo cp imx500.ko.xz /lib/modules/6.6.51+rpt-rpi-2712/kernel/drivers/media/i2c/
```

3. reboot

```
$ sudo reboot
```

- 動作確認へすすむ

Build手順 (Source CodeからBuildする場合)

Kernel Source Code情報(2024/11/25更新)

- tag: stable_20241008
- Tag: stable_20241008

```
$ cd ~/git
$ git clone https://github.com/raspberrypi/linux -b stable_20241008 --depth=1
$ cd linux
```

1. 修正内容の適用 (2024/11/25更新)

```
$ cd ~/git/linux
$ cp ~/Downloads/20241122_imx500_driver_modify/imx500.c driver/media/i2c/
```

またはpatch適用

```
$ cd ~/git/linux
$ patch -p1 < ~/Downloads/20241122_imx500_driver_modify/imx500_diff_20241122.patch
```

2. Build

- Raspberry Pi 5で実行
- 環境準備

```
$ sudo apt install git bc bison flex libssl-dev make
```

- 初回実行時には2時間弱時間がかかりますのでご注意ください
- ２回目以降は差分のみのmakeなので早いです
- 以下基本的な手順ですので環境に合わせて適時修正ください

```
$ cd ~/git/linux
$ KERNEL=kernel_2712
$ make bcm2712_defconfig
$ sudo make -j6 Image.gz modules dtbs
```

3. Install

- Raspberry Pi 5で実行
- 以下基本的な手順ですので環境に合わせて適時修正ください

```
$ cd ~/git/linux
$ KERNEL=kernel_2712
$ sudo make -j6 modules_install
$ sudo cp /boot/firmware/kernel_2712.img /boot/firmware/kernel_2712-backup.img
$ sudo cp arch/arm64/boot/Image.gz /boot/firmware/kernel_2712.img
$ sudo cp arch/arm64/boot/dts/broadcom/*.dtb /boot/firmware/
$ sudo cp arch/arm64/boot/dts/overlays/*.dtb* /boot/firmware/overlays/
$ sudo cp arch/arm64/boot/dts/overlays/README /boot/firmware/overlays/
```

4. reboot

```
$ sudo reboot
```

5. v4l2 control確認 (2024/11/25追記)

- 正しくloadされると　"User Controls"に”imx500_read_regist”と"imx500_register_io"が表示される

```
$ v4l2-ctl -d /dev/v4l-subdev2 --list-ctrls
User Controls

                     exposure 0x00980911 (int)    : min=8 max=65478 step=1 default=1600 value=1600
              horizontal_flip 0x00980914 (bool)   : default=0 value=0 flags=modify-layout
                vertical_flip 0x00980915 (bool)   : default=0 value=0 flags=modify-layout
    imx500_inference_windows 0x00982900 (u32)    : min=0 max=4032 step=1 default=0 dims=[4] flags=has-payload, execute-on-write
imx500_network_firmware_file_fd 0x00982901 (int)    : min=-1 max=2147483647 step=1 default=-1 value=0 flags=write-only, execute-on-write
         imx500_read_register 0x00982902 (int)    : min=0 max=65535 step=1 default=53314 value=53314 flags=execute-on-write
           imx500_register_io 0x00982903 (u32)    : min=0 max=4294967295 step=1 default=0 dims=[3] flags=has-payload

Camera Controls

           camera_orientation 0x009a0922 (menu)   : min=0 max=2 default=2 value=2 (External) flags=read-only
       camera_sensor_rotation 0x009a0923 (int)    : min=0 max=0 step=1 default=0 value=0 flags=read-only

Image Source Controls

```

```
              vertical_blanking 0x009e0901 (int)    : min=1117 max=8380960 step=1 default=1117 value=1117
            horizontal_blanking 0x009e0902 (int)    : min=13844 max=13844 step=1 default=13844 value=13844
                analogue_gain 0x009e0903 (int)    : min=0 max=978 step=1 default=0 value=0
                 notify_gains 0x009e0909 (int)    : min=1 max=4095 step=1 default=256 dims=[4] flags=has-payload

Image Processing Controls

                link_frequency 0x009f0901 (intmenu): min=0 max=0 default=0 value=0 (444000000 0x1a76e700) flags=read-only
                    pixel_rate 0x009f0902 (int64)  : min=744000000 max=744000000 step=1 default=744000000 value=744000000 flags=read-only
```

動作確認方法と結果

0. 推論動作の開始（動作確認の準備のため）

- Raspberry Pi 5で実行
- 前提条件としてimx500 cameraで推論をしている必要があるので下記の例のように実行します
- 実行していなくても値が表示されますが、正しい値ではない場合があるのでご注意ください
  - rpicamの場合

```
$ rpicam-hello -t 0 --post-process-file /usr/share/rpi-camera-assets/imx500_mobilenet_ssd.json
```

  - picamera2の場合 (Input Tensorを表示できるようにした修正版を利用した場合。修正内容は別資料参照)

```
$ python imx500_classification_demo.py --model /usr/share/imx500-models/imx500_network_efficientnet_bo.rpk -i -n
```

- **3.以降は、imx500内蔵のISPのregisterを操作しているのでinput_tensorをpicamera2のscriptで表示しています**

1. addressを指定してregister値を読み出す

```
$ python imx500_reg_io.py -a 0xd040 -b 16
R: address(h): 0000d040, data(h): 100b, data(d):  4107, bit width: 16
```

- 結果はread_register.pyと一致

```
$ python read_register.py 0xd040
ADDRESS:D040, VAL(d): 4107, VAL(h):100B
```

2. device idを読み出す

```
$ python imx500_reg_io.py -a 0xd040 -b 32 -n 16
R: address(h): 0000d040, data(h): 100b5050, data(d):  269176912, bit width: 32
R: address(h): 0000d044, data(h): 0a212106, data(d):  169943302, bit width: 32
R: address(h): 0000d048, data(h): 54640130, data(d): 1415840048, bit width: 32
R: address(h): 0000d04c, data(h): 00000000, data(d):          0, bit width: 32
```

- 赤瀬さんの方法で読み出した値"100B50500A2121065464013000000000"と一致

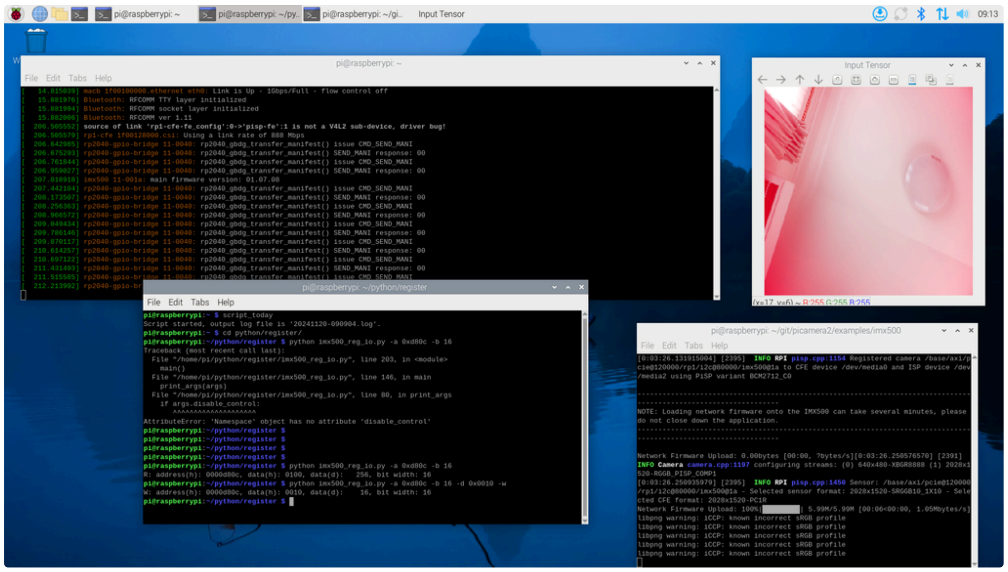
3. addressからdataを読む 16bit幅 (例:0xD80C:ISP_OPB_VAL_R[11:0])

```
$ python imx500_reg_io.py -a 0xd80c -b 16
R: address(h): 0000d80c, data(h): 0100, data(d):   256, bit width: 16
```

4. addressにdataを書く(例:0xD80C:ISP_OPB_VAL_R[11:0]を256から16にする)

```
$ python imx500_reg_io.py -a 0xd80c -b 16 -d 0x0010 -w
W: address(h): 0000d80c, data(h): 0010, data(d):    16, bit width: 16
```

- 画面が赤くなる

- 

```
pi@raspberrypi: ~
[   14.815039] macb 1f00100000.ethernet eth0: Link is Up - 1Gbps/Full - flow control off
[   15.881976] Bluetooth: RFCOMM TTY layer initialized
[   15.881994] Bluetooth: RFCOMM socket layer initialized
[   15.882006] Bluetooth: RFCOMM ver 1.11
[  206.505552] source of link 'rp1-cfe-fe_config':0->'pisp-fe':1 is not a V4L2 sub-device, driver bug!
[  206.505579] rp1-cfe 1f00128000.csi: Using a link rate of 888 Mbps
[  206.642985] rp2040-gpio-bridge 11-0040: rp2040_gbdg_transfer_manifest() issue CMD_SEND_MANI
[  206.675293] rp2040-gpio-bridge 11-0040: rp2040_gbdg_transfer_manifest() SEND_MANI response: 00
[  206.761844] rp2040-gpio-bridge 11-0040: rp2040_gbdg_transfer_manifest() issue CMD_SEND_MANI
[  206.959027] rp2040-gpio-bridge 11-0040: rp2040_gbdg_transfer_manifest() SEND_MANI response: 00
[  207.018918] imx500 11-001a: main firmware version: 01.07.08
[  207.442104] rp2040-gpio-bridge 11-0040: rp2040_gbdg_transfer_manifest() issue CMD_SEND_MANI
[  208.173507] rp2040-gpio-bridge 11-0040: rp2040_gbdg_transfer_manifest() SEND_MANI response: 00
[  208.256363] rp2040-gpio-bridge 11-0040: rp2040_gbdg_transfer_manifest() issue CMD_SEND_MANI
[  208.966572] rp2040-gpio-bridge 11-0040: rp2040_gbdg_transfer_manifest() SEND_MANI response: 00
[  209.049434] rp2040-gpio-bridge 11-0040: rp2040_gbdg_transfer_manifest() issue CMD_SEND_MANI
[  209.786146] rp2040-gpio-bridge 11-0040: rp2040_gbdg_transfer_manifest() SEND_MANI response: 00
[  209.870117] rp2040-gpio-bridge 11-0040: rp2040_gbdg_transfer_manifest() issue CMD_SEND_MANI
[  210.614257] rp2040-gpio-bridge 11-0040: rp2040_gbdg_transfer_manifest() SEND_MANI response: 00
[  210.697122] rp2040-gpio-bridge 11-0040: rp2040_gbdg_transfer_manifest() issue CMD_SEND_MANI
[  211.431493] rp2040-gpio-bridge 11-0040: rp2040_gbdg_transfer_manifest() SEND_MANI response: 00
[  211.515585] rp2040-gpio-bridge 11-0040: rp2040_gbdg_transfer_manifest() issue CMD_SEND_MANI
[  212.213992] rp2040-gpio-br
```

```
pi@raspberrypi: ~/python/register
pi@raspberrypi:~ $ script_today
Script started, output log file is '20241120-090904.log'.
pi@raspberrypi:~ $ cd python/register/
pi@raspberrypi:~/python/register $ python imx500_reg_io.py -a 0xd80c -b 16
Traceback (most recent call last):
  File "/home/pi/python/register/imx500_reg_io.py", line 203, in <module>
    main()
  File "/home/pi/python/register/imx500_reg_io.py", line 146, in main
    print_args(args)
  File "/home/pi/python/register/imx500_reg_io.py", line 80, in print_args
    if args.disable_control:
       ^^^^^^^^^^^^^^^^^^^^^
AttributeError: 'Namespace' object has no attribute 'disable_control'
pi@raspberrypi:~/python/register $
pi@raspberrypi:~/python/register $
pi@raspberrypi:~/python/register $
pi@raspberrypi:~/python/register $
pi@raspberrypi:~/python/register $ python imx500_reg_io.py -a 0xd80c -b 16
R: address(h): 0000d80c, data(h): 0100, data(d):   256, bit width: 16
pi@raspberrypi:~/python/register $ python imx500_reg_io.py -a 0xd80c -b 16 -d 0x0010 -w
W: address(h): 0000d80c, data(h): 0010, data(d):    16, bit width: 16
pi@raspberrypi:~/python/register $
```

```
pi@raspberrypi: ~/git/picamera2/examples/imx500
[0:03:26.131915004] [2395]  INFO RPI pisp.cpp:1154 Registered camera /base/axi/p
cie@120000/rp1/i2c@80000/imx500@1a to CFE device /dev/media0 and ISP device /dev
/media2 using PiSP variant BCM2712_C0

--------------------------------------------------------------------------------
------------------------------
NOTE: Loading network firmware onto the IMX500 can take several minutes, please
do not close down the application.
--------------------------------------------------------------------------------
------------------------------

Network Firmware Upload: 0.00bytes [00:00, ?bytes/s][0:03:26.250576570] [2391]
INFO Camera camera.cpp:1197 configuring streams: (0) 640x480-XBGR8888 (1) 2028x1
520-RGGB_PISP_COMP1
[0:03:26.250935979] [2395]  INFO RPI pisp.cpp:1450 Sensor: /base/axi/pcie@120000
/rp1/i2c@80000/imx500@1a - Selected sensor format: 2028x1520-SRGGB10_1X10 - Sele
cted CFE format: 2028x1520-PC1R
Network Firmware Upload: 100%|██████████| 5.99M/5.99M [00:06<00:00, 1.05Mbytes/s]
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
```

5. addressにdataを書く(例:0xD80C:ISP_OPB_VAL_R[11:0]を16から256に戻す)

```
$ python imx500_reg_io.py -a 0xd80c -b 16 -d 0x0010 -w
W: address(h): 0000d80c, data(h): 0100, data(d):   256, bit width: 16
```

- 画面が戻る

6. addressから複数バイト(8バイト)を読む (例:0xD80C:ISP_OPB_VAL_R[11:0]/_GR/_GB/_B)

```
$ python imx500_reg_io.py -a 0xd80c -b 16 -n 8
R: address(h): 0000d80c, data(h): 0100, data(d):   256, bit width: 16
R: address(h): 0000d80e, data(h): 0100, data(d):   256, bit width: 16
R: address(h): 0000d810, data(h): 0100, data(d):   256, bit width: 16
R: address(h): 0000d812, data(h): 0100, data(d):   256, bit width: 16
```

7. addressからdataを読む 8bit幅(例:0xD822:LSC_CRCT_ENABALE[0])

```
$ python imx500_reg_io.py -a 0xd822 -b 8
R: address(h): 0000d822, data(h): 01, data(d):   1, bit width:  8
```

8. addressからdataを読む 32bit幅 (例:0xD75C:DNN_SPI_SCLK_FREQUEN CY [31:0])

```
$ python imx500_reg_io.py -a 0xd75c -b 32
R: address(h): 0000d75c, data(h): 00bebc20, data(d):   12500000, bit width: 32
```

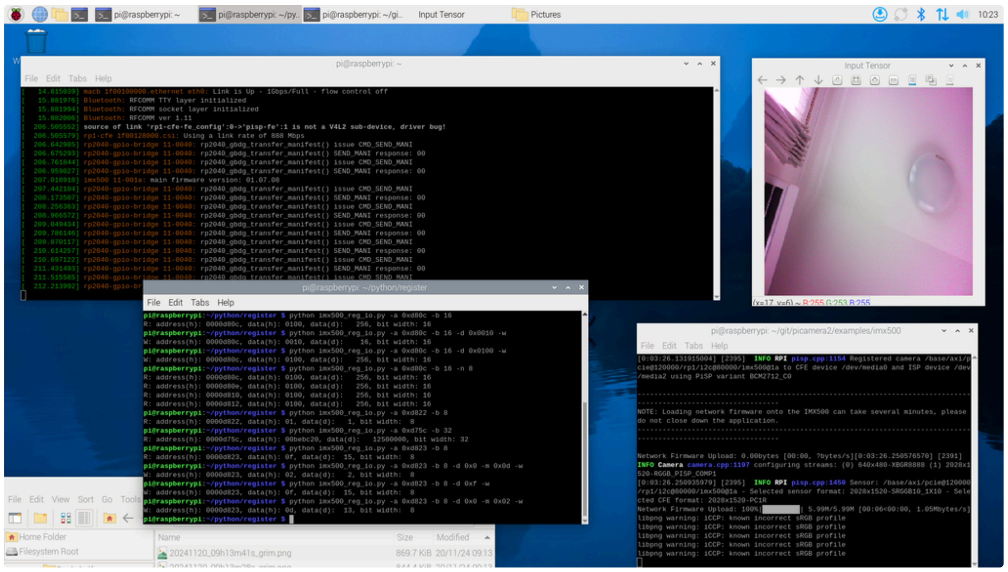
9. addressからdataを読む 8bit幅(例:0xD823:ISP_LSC_EN_EACH_ COLOR [3:0])

```
$ python imx500_reg_io.py -a 0xd75c -b 8
R: address(h): 0000d823, data(h): 0f, data(d):  15, bit width:  8
```

10. addresにdataをmaskして書く 8bit幅(例:0xD823:ISP_LSC_EN_EACH_ COLOR [3:0])

```
$ python imx500_reg_io.py -a 0xd823 -b 8 -d 0x0 -m 0x02 -w
W: address(h): 0000d823, data(h): 0d, data(d):  13, bit width:  8
```

- maskが1のbitのみデータが反映される
- 画面がマゼンタになる



```
[   14.815039] macb 1f00100000.ethernet eth0: Link is Up - 1Gbps/Full - flow control off
[   15.881976] Bluetooth: RFCOMM TTY layer initialized
[   15.881994] Bluetooth: RFCOMM socket layer initialized
[   15.882006] Bluetooth: RFCOMM ver 1.11
[  206.505552] source of link 'rp1-cfe-fe_config':0->'pisp-fe':1 is not a V4L2 sub-device, driver bug!
[  206.505579] rp1-cfe 1f00128000.csi: Using a link rate of 888 Mbps
[  206.642985] rp2040-gpio-bridge 11-0040: rp2040_gbdg_transfer_manifest() issue CMD_SEND_MANI
[  206.675293] rp2040-gpio-bridge 11-0040: rp2040_gbdg_transfer_manifest() SEND_MANI response: 00
[  206.761844] rp2040-gpio-bridge 11-0040: rp2040_gbdg_transfer_manifest() issue CMD_SEND_MANI
[  206.959027] rp2040-gpio-bridge 11-0040: rp2040_gbdg_transfer_manifest() SEND_MANI response: 00
[  207.018918] imx500 11-001a: main firmware version: 01.07.08
[  207.442104] rp2040-gpio-bridge 11-0040: rp2040_gbdg_transfer_manifest() issue CMD_SEND_MANI
[  208.173507] rp2040-gpio-bridge 11-0040: rp2040_gbdg_transfer_manifest() SEND_MANI response: 00
[  208.256363] rp2040-gpio-bridge 11-0040: rp2040_gbdg_transfer_manifest() issue CMD_SEND_MANI
[  208.966572] rp2040-gpio-bridge 11-0040: rp2040_gbdg_transfer_manifest() SEND_MANI response: 00
[  209.049434] rp2040-gpio-bridge 11-0040: rp2040_gbdg_transfer_manifest() issue CMD_SEND_MANI
[  209.786146] rp2040-gpio-bridge 11-0040: rp2040_gbdg_transfer_manifest() SEND_MANI response: 00
[  209.870117] rp2040-gpio-bridge 11-0040: rp2040_gbdg_transfer_manifest() issue CMD_SEND_MANI
[  210.614257] rp2040-gpio-bridge 11-0040: rp2040_gbdg_transfer_manifest() SEND_MANI response: 00
[  210.697122] rp2040-gpio-bridge 11-0040: rp2040_gbdg_transfer_manifest() issue CMD_SEND_MANI
[  211.431493] rp2040-gpio-bridge 11-0040: rp2040_gbdg_transfer_manifest() SEND_MANI response: 00
[  211.515585] rp2040-gpio-bridge 11-0040: rp2040_gbdg_transfer_manifest() issue CMD_SEND_MANI
[  212.213992] rp2040-gpio-br
```

```
pi@raspberrypi:~/python/register $ python imx500_reg_io.py -a 0xd80c -b 16
R: address(h): 0000d80c, data(h): 0100, data(d):   256, bit width: 16
pi@raspberrypi:~/python/register $ python imx500_reg_io.py -a 0xd80c -b 16 -d 0x0010 -w
W: address(h): 0000d80c, data(h): 0010, data(d):    16, bit width: 16
pi@raspberrypi:~/python/register $ python imx500_reg_io.py -a 0xd80c -b 16 -d 0x0100 -w
W: address(h): 0000d80c, data(h): 0100, data(d):   256, bit width: 16
pi@raspberrypi:~/python/register $ python imx500_reg_io.py -a 0xd80c -b 16 -n 8
R: address(h): 0000d80c, data(h): 0100, data(d):   256, bit width: 16
R: address(h): 0000d80e, data(h): 0100, data(d):   256, bit width: 16
R: address(h): 0000d810, data(h): 0100, data(d):   256, bit width: 16
R: address(h): 0000d812, data(h): 0100, data(d):   256, bit width: 16
pi@raspberrypi:~/python/register $ python imx500_reg_io.py -a 0xd822 -b 8
R: address(h): 0000d822, data(h): 01, data(d):   1, bit width:  8
pi@raspberrypi:~/python/register $ python imx500_reg_io.py -a 0xd75c -b 32
R: address(h): 0000d75c, data(h): 00bebc20, data(d):   12500000, bit width: 32
pi@raspberrypi:~/python/register $ python imx500_reg_io.py -a 0xd823 -b 8
R: address(h): 0000d823, data(h): 0f, data(d):  15, bit width:  8
pi@raspberrypi:~/python/register $ python imx500_reg_io.py -a 0xd823 -b 8 -d 0x0 -m 0x0d -w
W: address(h): 0000d823, data(h): 02, data(d):   2, bit width:  8
pi@raspberrypi:~/python/register $ python imx500_reg_io.py -a 0xd823 -b 8 -d 0xf -w
W: address(h): 0000d823, data(h): 0f, data(d):  15, bit width:  8
pi@raspberrypi:~/python/register $ python imx500_reg_io.py -a 0xd823 -b 8 -d 0x0 -m 0x02 -w
W: address(h): 0000d823, data(h): 0d, data(d):  13, bit width:  8
pi@raspberrypi:~/python/register $
```

```
[0:03:26.131915004] [2395]  INFO RPI pisp.cpp:1154 Registered camera /base/axi/p
cie@120000/rp1/i2c@80000/imx500@1a to CFE device /dev/media0 and ISP device /dev
/media2 using PiSP variant BCM2712_C0

--------------------------------------------------------------------------------
----------------------------------
NOTE: Loading network firmware onto the IMX500 can take several minutes, please
do not close down the application.
--------------------------------------------------------------------------------
----------------------------------

Network Firmware Upload: 0.00bytes [00:00, ?bytes/s][0:03:26.250576570] [2391]
INFO Camera camera.cpp:1197 configuring streams: (0) 640x480-XBGR8888 (1) 2028x1
520-RGGB_PISP_COMP1
[0:03:26.250935979] [2395]  INFO RPI pisp.cpp:1450 Sensor: /base/axi/pcie@120000
/rp1/i2c@80000/imx500@1a - Selected sensor format: 2028x1520-SRGGB10_1X10 - Sele
cted CFE format: 2028x1520-PC1R
Network Firmware Upload: 100%|██████████| 5.99M/5.99M [00:06<00:00, 1.05Mbytes/s]
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
libpng warning: iCCP: known incorrect sRGB profile
```

•

11. addressからdataを読む 16bit幅 上位下位バイト入れ替え表示(例:0xD822:LSC_CRCT_ENABALE[0], 0xD823:ISP_LSC_EN_EACH_ COLOR [3:0])

- 通常 (big endian)

```
$ python imx500_reg_io.py -a 0xd822 -b 16
R: address(h): 0000d822, data(h): 010f, data(d):   271, bit width: 16
```

- -l オプション (little endian)
- little endian 様表示：16bit以上の数字の場合はそのままの表示が正しいが、8bitづつ関連性のないバイトの場合はこの形式がわかりやすいので準備した

```
$ python imx500_reg_io.py -a 0xd822 -b 16 -l
R: address(h): 0000d822, data(h): 0f01, data(d):  3841, bit width: 16
```

12. 複数アドレスの結果をファイルに出力する　（例：0xD850　G_KNOT_R000[8:0]-0xD8A4:G_KNOT_R3FF[8:0]）

```
$ python imx500_reg_io.py -a 0xd850 -n 88 -o g_knot_r.txt

R: address(h): 0000d850, data(h): 00010000, data(d):     65536, bit width: 32
R: address(h): 0000d854, data(h): 00040002, data(d):    262146, bit width: 32
R: address(h): 0000d858, data(h): 00060005, data(d):    393221, bit width: 32
R: address(h): 0000d85c, data(h): 00090008, data(d):    589832, bit width: 32
R: address(h): 0000d860, data(h): 000c000a, data(d):    786442, bit width: 32
R: address(h): 0000d864, data(h): 0012000f, data(d):   1179663, bit width: 32
R: address(h): 0000d868, data(h): 001c0014, data(d):   1835028, bit width: 32
R: address(h): 0000d86c, data(h): 002a0024, data(d):   2752548, bit width: 32
R: address(h): 0000d870, data(h): 00360030, data(d):   3538992, bit width: 32
R: address(h): 0000d874, data(h): 0046003c, data(d):   4587580, bit width: 32
R: address(h): 0000d878, data(h): 005a0051, data(d):   5898321, bit width: 32
R: address(h): 0000d87c, data(h): 00750064, data(d):   7667812, bit width: 32
R: address(h): 0000d880, data(h): 00920084, data(d):   9568388, bit width: 32
R: address(h): 0000d884, data(h): 00a9009e, data(d):  11075742, bit width: 32
R: address(h): 0000d888, data(h): 00ba00b2, data(d):  12189874, bit width: 32
R: address(h): 0000d88c, data(h): 00c700c1, data(d):  13041857, bit width: 32
R: address(h): 0000d890, data(h): 00d100cd, data(d):  13697229, bit width: 32
R: address(h): 0000d894, data(h): 00de00d6, data(d):  14549206, bit width: 32
R: address(h): 0000d898, data(h): 00e900e4, data(d):  15270116, bit width: 32
```

```
R: address(h): 0000d89c, data(h): 00f300ee, data(d):   15925486, bit width: 32
R: address(h): 0000d8a0, data(h): 00fb00f7, data(d):   16449783, bit width: 32
R: address(h): 0000d8a4, data(h): 00ff0000, data(d):   16711680, bit width: 32
```

- フォーマットは32bit固定

```
$ cat g_knot_r.txt
0000d850, 00010000
0000d854, 00040002
0000d858, 00060005
0000d85c, 00090008
0000d860, 000c000a
0000d864, 0012000f
0000d868, 001c0014
0000d86c, 002a0024
0000d870, 00360030
0000d874, 0046003c
0000d878, 005a0051
0000d87c, 00750064
0000d880, 00920084
0000d884, 00a9009e
0000d888, 00ba00b2
0000d88c, 00c700c1
0000d890, 00d100cd
0000d894, 00de00d6
0000d898, 00e900e4
0000d89c, 00f300ee
0000d8a0, 00fb00f7
0000d8a4, 00ff0000
```

13. ファイルにあるアドレスのデータを読む

- address列のアドレスから読み出す
- data列のデータは使用されない

```
$ python imx500_reg_io.py -i g_knot_r.txt

R: address(h): 0000d850, data(h): 00010000, data(d):      65536, bit width: 32
R: address(h): 0000d854, data(h): 00040002, data(d):     262146, bit width: 32
R: address(h): 0000d858, data(h): 00060005, data(d):     393221, bit width: 32
R: address(h): 0000d85c, data(h): 00090008, data(d):     589832, bit width: 32
R: address(h): 0000d860, data(h): 000c000a, data(d):     786442, bit width: 32
R: address(h): 0000d864, data(h): 0012000f, data(d):    1179663, bit width: 32
R: address(h): 0000d868, data(h): 001c0014, data(d):    1835028, bit width: 32
R: address(h): 0000d86c, data(h): 002a0024, data(d):    2752548, bit width: 32
R: address(h): 0000d870, data(h): 00360030, data(d):    3538992, bit width: 32
R: address(h): 0000d874, data(h): 0046003c, data(d):    4587580, bit width: 32
R: address(h): 0000d878, data(h): 005a0051, data(d):    5898321, bit width: 32
R: address(h): 0000d87c, data(h): 00750064, data(d):    7667812, bit width: 32
R: address(h): 0000d880, data(h): 00920084, data(d):    9568388, bit width: 32
R: address(h): 0000d884, data(h): 00a9009e, data(d):   11075742, bit width: 32
R: address(h): 0000d888, data(h): 00ba00b2, data(d):   12189874, bit width: 32
R: address(h): 0000d88c, data(h): 00c700c1, data(d):   13041857, bit width: 32
R: address(h): 0000d890, data(h): 00d100cd, data(d):   13697229, bit width: 32
R: address(h): 0000d894, data(h): 00de00d6, data(d):   14549206, bit width: 32
R: address(h): 0000d898, data(h): 00e900e4, data(d):   15270116, bit width: 32
R: address(h): 0000d89c, data(h): 00f300ee, data(d):   15925486, bit width: 32
R: address(h): 0000d8a0, data(h): 00fb00f7, data(d):   16449783, bit width: 32
```

```
R: address(h): 0000d8a4, data(h): 00ff0000, data(d):   16711680, bit width: 32
```

14. ファイルにあるアドレスのデータをレジスタに書く

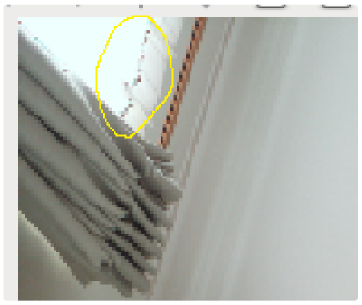
- ガンマデータを一部書き換えた“g_knot_r_1.txt”を準備

```
$ diff g_knot_r.txt g_knot_r_1.txt
21,22c21,22
< 0000d8a0, 00fb00f7
< 0000d8a4, 00ff0000
---
> 0000d8a0, 00f000f0
> 0000d8a4, 00f00000
```

- address列のアドレスから読み出す
- data列のデータを書き込む

```
$ python imx500_reg_io -i g_knot_r_1.txt -w
```

- 左のカーテンの明るい部分に黒いラインが出る

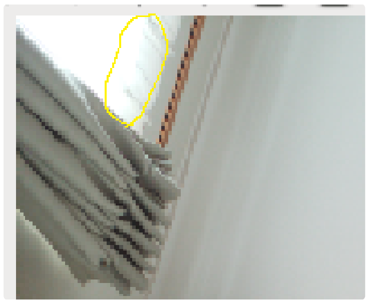


-

15. ファイルのアドレスとデータをレジスタに書いて別ファイルに保存する

```
$ python imx500_reg_io.py -i g_knot_r.txt -o g_knot_r_2.txt -w

W: address(h): 0000d850, data(h): 00010000, data(d):      65536, bit width: 32
W: address(h): 0000d854, data(h): 00040002, data(d):     262146, bit width: 32
W: address(h): 0000d858, data(h): 00060005, data(d):     393221, bit width: 32
W: address(h): 0000d85c, data(h): 00090008, data(d):     589832, bit width: 32
W: address(h): 0000d860, data(h): 000c000a, data(d):     786442, bit width: 32
W: address(h): 0000d864, data(h): 0012000f, data(d):    1179663, bit width: 32
W: address(h): 0000d868, data(h): 001c0014, data(d):    1835028, bit width: 32
W: address(h): 0000d86c, data(h): 002a0024, data(d):    2752548, bit width: 32
W: address(h): 0000d870, data(h): 00360030, data(d):    3538992, bit width: 32
W: address(h): 0000d874, data(h): 0046003c, data(d):    4587580, bit width: 32
W: address(h): 0000d878, data(h): 005a0051, data(d):    5898321, bit width: 32
W: address(h): 0000d87c, data(h): 00750064, data(d):    7667812, bit width: 32
W: address(h): 0000d880, data(h): 00920084, data(d):    9568388, bit width: 32
W: address(h): 0000d884, data(h): 00a9009e, data(d):   11075742, bit width: 32
W: address(h): 0000d888, data(h): 00ba00b2, data(d):   12189874, bit width: 32
W: address(h): 0000d88c, data(h): 00c700c1, data(d):   13041857, bit width: 32
W: address(h): 0000d890, data(h): 00d100cd, data(d):   13697229, bit width: 32
W: address(h): 0000d894, data(h): 00de00d6, data(d):   14549206, bit width: 32
W: address(h): 0000d898, data(h): 00e900e4, data(d):   15270116, bit width: 32
W: address(h): 0000d89c, data(h): 00f300ee, data(d):   15925486, bit width: 32
W: address(h): 0000d8a0, data(h): 00fb00f7, data(d):   16449783, bit width: 32
W: address(h): 0000d8a4, data(h): 00ff0000, data(d):   16711680, bit width: 32
```

- 元のガンマテーブルのデータを書く
- 左のカーテンの明るい部分に黒いラインが消える
- 

driver 修正内容 (git diff)

- file: **driver/media/i2c/imx500.c**

```
diff --git a/drivers/media/i2c/imx500.c b/drivers/media/i2c/imx500.c
index 15e7d3f7d..c7ea7f8de 100644
--- a/drivers/media/i2c/imx500.c
+++ b/drivers/media/i2c/imx500.c
@@ -235,6 +235,10 @@ enum pad_types { IMAGE_PAD, METADATA_PAD, NUM_PADS };

 #define V4L2_CID_USER_IMX500_INFERENCE_WINDOW (V4L2_CID_USER_IMX500_BASE + 0)
 #define V4L2_CID_USER_IMX500_NETWORK_FW_FD (V4L2_CID_USER_IMX500_BASE + 1)
+//added->
+#define V4L2_CID_USER_IMX500_READ_REGISTER (V4L2_CID_USER_IMX500_BASE + 2)
+#define V4L2_CID_USER_IMX500_REGISTER_IO (V4L2_CID_USER_IMX500_BASE + 3)
+//added<-

 #define ONE_MIB (1024 * 1024)

@@ -1364,6 +1368,10 @@ struct imx500 {

    struct v4l2_rect inference_window;

+   //added->
+   u32 reg_io[3];
+   //added<-
+
    /* Current mode */
    const struct imx500_mode *mode;

@@ -1575,6 +1583,69 @@ static int imx500_set_inference_window(struct imx500 *imx500)
                  ARRAY_SIZE(window_regs), NULL);
 }

+//added->
+static int imx500_set_reg_io(struct imx500 *imx500)
+{
+   int ret = 0;
+   u64 tmp, val, mask, data;
+   u32 addr;
+   int bitw, io;
+   //printk(KERN_INFO "%s r[0](h)=%08x r[1](h)=%08x r[2](h)=%08x\n", __func__, imx500->reg_io[0], imx500->reg_io[1], imx500->reg_io[2]);
+   //set data
+   addr = imx500->reg_io[0] & 0x0000FFFF;
+   bitw = imx500->reg_io[0]>>16 & 0x000000FF;
+   io = imx500->reg_io[0]>>24 & 0x00000001;
+  val = imx500->reg_io[1];
+  mask = imx500->reg_io[2];
+  //printk(KERN_INFO "%s address(h):%04x bitw:%2d io:%1d value(h):%08llx mask(h):%08llx\n", __func__, addr, bitw, io, val, mask);
+  //0 is no mask
+  if(mask == 0)
+      mask = ~mask;
+  switch (bitw) {
+      case 16:   //16bit (2byte)
+          ret = cci_read(imx500->regmap, CCI_REG16(addr), &tmp, NULL);
+          mask = 0x0000FFFF & mask;
+          if(ret>=0){
+              if(io>0){
+                  data = (val & mask) | (tmp & ~mask);
+                  ret = cci_write(imx500->regmap, CCI_REG16(addr), data, NULL);
+              } else
+                  data = tmp & mask;
+              //printk(KERN_INFO "%s mask(h):%08llx val(h):%08llx tmp(h):%08llx data(h):%08llx ret:%d\n", __func__, mask, val, tmp, data, ret);
+          }
+          break;
+      case 32:   //32bit (4byte)
+          ret = cci_read(imx500->regmap, CCI_REG32(addr), &tmp, NULL);
+          mask = 0xFFFFFFFF & mask;
+          if(ret>=0){
+              if(io>0){
+                  data = (val & mask) | (tmp & ~mask);
+                  ret = cci_write(imx500->regmap, CCI_REG32(addr), data, NULL);
+              } else
+                  data = tmp & mask;
+              //printk(KERN_INFO "%s mask(h):%08llx val(h):%08llx tmp(h):%08llx data(h):%08llx ret:%d\n", __func__, mask, val, tmp, data, ret);
+          }
+          break;
+      default:    //8bit (1byte)
+          ret = cci_read(imx500->regmap, CCI_REG8(addr), &tmp, NULL);
+          mask = 0x000000FF & mask;
+          if(ret>=0){
+              if(io>0){
+                  data = (val & mask) | (tmp & ~mask);
+                  ret = cci_write(imx500->regmap, CCI_REG8(addr), data, NULL);
+              } else
+                  data = tmp & mask;
+              //printk(KERN_INFO "%s mask(h):%08llx val(h):%08llx tmp(h):%08llx data(h):%08llx ret:%d\n", __func__, mask, val, tmp, data, ret);
+          }
+  }
+  if(ret>=0){
+      imx500->reg_io[1] = data;
+      imx500->reg_io[2] = mask;
+  }
+return ret;
+}
+//added<-
+
 static int imx500_reg_val_write_cbk(void *arg,
                     const struct cci_reg_sequence *reg)
 {
@@ -1874,6 +1945,9 @@ static int imx500_set_ctrl(struct v4l2_ctrl *ctrl)
        container_of(ctrl->handler, struct imx500, ctrl_handler);
    struct i2c_client *client = v4l2_get_subdevdata(&imx500->sd);
    int ret = 0;
```

```
+ //added->
+ u64 tmp;
+ //added<-

    if (ctrl->id == V4L2_CID_USER_IMX500_NETWORK_FW_FD) {
        /* Reset state of the control. */
@@ -1974,6 +2048,20 @@ static int imx500_set_ctrl(struct v4l2_ctrl *ctrl)
               sizeof(struct v4l2_rect));
        ret = imx500_set_inference_window(imx500);
        break;
+ //added->
+ case V4L2_CID_USER_IMX500_READ_REGISTER:
+     ret = cci_read(imx500->regmap, CCI_REG16(ctrl->val), &tmp, NULL);
+     //printk(KERN_INFO "%s address(h):%04x value(d):%llu (h):%04llx\n", __func__, ctrl->val, tmp, tmp);
+     if(ret>=0)
+         ctrl->val = tmp;
+     break;
+ case V4L2_CID_USER_IMX500_REGISTER_IO:
+     memcpy(&imx500->reg_io, ctrl->p_new.p_u32, sizeof(u32) * 3);
+     ret = imx500_set_reg_io(imx500);
+     if(ret>=0)
+         memcpy(ctrl->p_new.p_u32, &imx500->reg_io, sizeof(u32) * 3);
+     break;
+ //added<-
    default:
        dev_info(&client->dev,
             "ctrl(id:0x%x,val:0x%x) is not handled\n", ctrl->id,
@@ -2828,6 +2916,34 @@ static const struct v4l2_ctrl_config network_fw_fd = {
    .def      = -1,
 };

+//added ->
+/* Custom control for camera address read */
+static const struct v4l2_ctrl_config cam_read_register = {
+ .name      = "IMX500 Read Register",
+ .id        = V4L2_CID_USER_IMX500_READ_REGISTER,
+ .ops       = &imx500_ctrl_ops,
+ .type      = V4L2_CTRL_TYPE_INTEGER,
+ .flags     = V4L2_CTRL_FLAG_EXECUTE_ON_WRITE,
+ .min       = 0,
+ .max       = U16_MAX,
+ .step      = 1,
+ .def       = 53314,
+};
+/* Custom control for camera address io */
+static const struct v4l2_ctrl_config cam_register_io = {
+ .name      = "IMX500 Register IO",
+ .id        = V4L2_CID_USER_IMX500_REGISTER_IO,
+ .dims      = {3},
+ .ops       = &imx500_ctrl_ops,
+ .type      = V4L2_CTRL_TYPE_U32,
+ .elem_size   = sizeof(u32),
+ .min       = 0x00,
+ .max       = U32_MAX,
+ .step      = 1,
+ .def       = 0,
+};
+//added <-
```

```
+
 /* Initialize control handlers */
 static int imx500_init_controls(struct imx500 *imx500)
 {
@@ -2858,6 +2974,12 @@ static int imx500_init_controls(struct imx500 *imx500)
    if (imx500->link_freq)
        imx500->link_freq->flags |= V4L2_CTRL_FLAG_READ_ONLY;

+  //added->
+  //init parameter
+  imx500->reg_io[0]=0;
+  imx500->reg_io[1]=0;
+  imx500->reg_io[2]=0;
+  //added<-
    /*
     * Create the controls here, but mode specific limits are setup
     * in the imx500_set_framing_limits() call below.
@@ -2891,7 +3013,11 @@ static int imx500_init_controls(struct imx500 *imx500)
    v4l2_ctrl_new_custom(ctrl_hdlr, &inf_window_ctrl, NULL);
    imx500->network_fw_ctrl =
        v4l2_ctrl_new_custom(ctrl_hdlr, &network_fw_fd, NULL);
-
+  //added ->
+  //imx500->imx500_read_register = v4l2_ctrl_new_custom(ctrl_hdlr, &cam_read_register, NULL);
+  v4l2_ctrl_new_custom(ctrl_hdlr, &cam_read_register, NULL);
+  v4l2_ctrl_new_custom(ctrl_hdlr, &cam_register_io, NULL);
+  //added <-
    if (ctrl_hdlr->error) {
        ret = ctrl_hdlr->error;
        dev_err(&client->dev, "%s control init failed (%d)\n", __func__,
```

EOF