Movidius
an Intel company

MA2x5x FLIC Camera Application Note

v1.0 / November 2017

Movidius Confidential

Movidius
an Intel company

Copyright and Proprietary Information Notice

Intel Movidius
2200 Mission College Blvd
M/S SC11-201
Santa Clara, CA 95054
http://www.movidius.com/

Movidius
an Intel company

Revision History

Revision	Description	Author
1.0	First released version.	John Scott (Movidius)

Movidius
an Intel company

Table of Contents

1 Introduction

This application note describes how to build a basic streaming camera application using the FLIC framework.

This application processes raw pixel data is captured from the IMX208 sensor and processed line-by-line through the SIPP Image Signal Processing (ISP) engine, the processed YUV is then sent over a UVC class USB3 interface. The resulting YUV stream can be viewed on Linux using standard USB UVC class viewers such a cheese or guvcview.

1.1 Related documents and resources

Related MDK documentation can be obtained from http://www.movidius.org. If you do not have access to the documents below, you can request them. Relevant documents include:

1. Movidius MDK release (download from www.movidius.org).
2. FLIC Programmers Guide and API.
3. MA2x5x SIPP User Guide.

1.2 Abbreviations

Term	Definition
3A	Auto-White-Balance, Auto-Exposure, and Auto-Focus.
HDMI	High Definition TV interface.
ISP	Image Signal Processing.
UVC	USB Video Class.
YUV	Video format.

2 FLIC Camera Application

A basic streaming examples, **mdk/common/components/flic/examples/arch/ma2x5x/flic06LiveCam** has been added to demonstrate how to use the FLIC framework to create a simple single sensor, camera application. This is a basic camera as it does not have any 3A algorithms or camera control, these will be added in future releases.

NOTE: This pipeline should NOT be used for image quality evaluation. The pipeline uses a static set of parameters. There are no 3A (Auto-White-Balance, Auto-Exposure, Auto-Focus) algorithms integrated which are required to dynamically generate new parameters based on light conditions.

Raw pixel data is captured from the Sony IMX208 sensor and processed line-by-line through the SIPP Image Signal Processing (ISP) engine, and the processed YUV is then sent over HDMI interface to view on TV or HDMI capable monitor.

2.1 Constructing a Camera Application with FLIC

The camera application is a single FLIC pipeline containing the plugins shown in Figure 1.

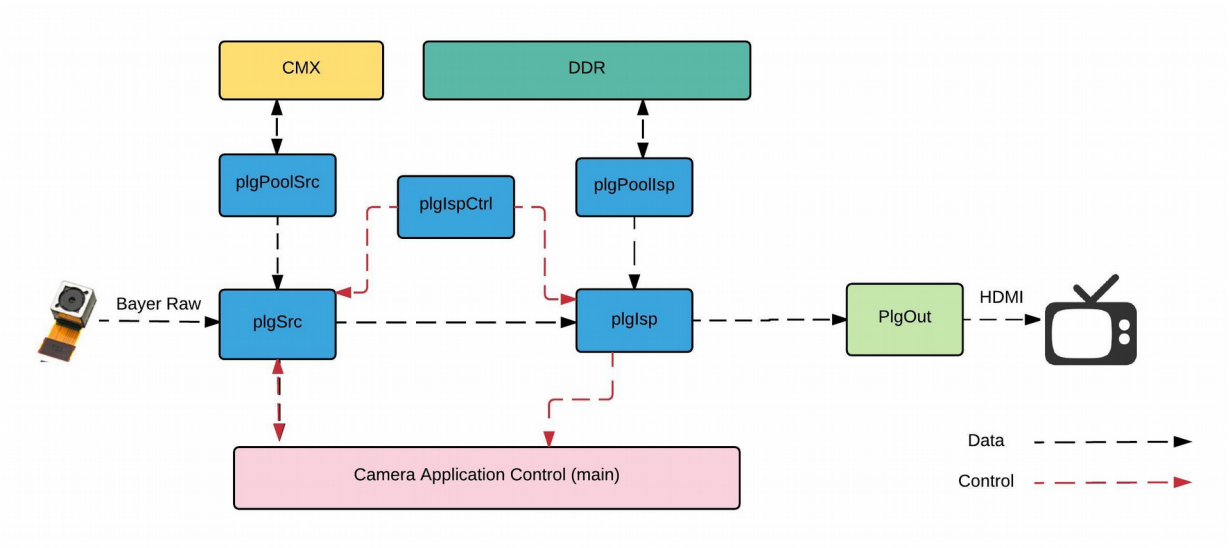

Figure 1: Camera application FLIC pipeline to HDMI

The camera pipeline is constructed from 3 main plugins:

1. plgSrc – receive frames from sensor.
2. plgIsp – ISP processing to generate YUV stream.
3. plgOut – transmit YUV stream over HDMI.

In addition, the following plugins are used to support the man pipeline:

1. PlgIspCtrl – provide, synchronize and construct composite Image Frame + ISP configuration input message for SIPP ISP.
2. PlgPoolSrc – frame pool for raw sensor frames.
3. PlgPoolIsp – frame pool for output YUV frames.

The plugins are located in mdk/common/components/flic/arch/ma2x5x.

The files for this application are located in common/components/flic/examples/arch/ma2x5x/flic06.

- main.cpp – main control application for the camera; create and start the pipeline.
- AppSpecInterface.h
- PipeIspSingle2.h

The main application control runs as a RTEMS threads on LeonOs

The function to create the pipeline, link the plugins and start the ISP processing:

```
//#################################################################
void pipeIspSingleCreate(GetSrcSzLimits getSrcSzLimits) {
```

```
    icSourceSetup srcSet;

    OpipeReset();  // reset the SIPP ISP Opipe component

    //Create plugins
    RgnAlloc.Create(RgnBuff, DEF_POOL_SZ);
    plgIspCtrl = PlgIspCtrl::instance();
    plgIspCtrl->Create();
    plgSrc.Create(IC_SOURCE_0);
    plgSrc.outFmt = SIPP_FMT_16BIT; // Two byte per pixel
    plgIsp.Create(IC_SOURCE_0);


    if(0 == getSrcSzLimits(IC_SOURCE_0, &srcSet)) {
        icSourceSetup* srcLimits = &srcSet;
        srcLimits->maxBpp = 16;

        plgPoolSrc.Create(&RgnAlloc,
              N_POOL_FRMS,
              ((srcLimits->maxPixels * srcLimits->maxBpp)) >> 3); //RAW

        plgPoolIsp.Create(&RgnAlloc,
               N_POOL_FRMS,
              (srcLimits->maxPixels * 3)>>1); //YUV420
    }
    else {
        // not define max size for initialized camera
        assert(0);
    }

    plgOut.Create();

    // add plugins to pipeline
    p.Add(plgIspCtrl);
    p.Add(&plgOut);
    p.Add(&plgSrc);
    p.Add(&plgIsp);
    p.Add(&plgPoolSrc);
    p.Add(&plgPoolIsp);


    // Link the plugins to create a path through the pipleine
    plgPoolSrc .out             .Link(&plgSrc.inO);
    plgSrc     .outCommand      .Link(&plgIspCtrl->inCtrlResponse);
    plgIspCtrl→outSrcCommand[IC_SOURCE_0].Link(&plgSrc.inCommand);

    plgPoolIsp .out             .Link(&plgIsp.inO);
    plgSrc     .out             .Link(&plgIsp.inI);
    plgIsp     .outF            .Link(&plgOut.in );
    plgIsp     .outE            .Link(&plgIspCtrl->inCtrlResponse);
    plgOut     .outCmd          .Link(&plgIspCtrl->inCtrlResponse);
    plgIspCtrl->outOutCmd       .Link(&plgOut.inCmd);

    // Start the pipeline
    p.Start();
}
```

When raw pixels arrive from the sensor, they are streamed directly into DDR from the PlgSrc plugin (the MipiRx hardware has dedicated DMA to DDR). Start-of-Frame events are generated for each new frame by

PlgSrc.

Similar, End-of-Frame events triggered by the PlgSrc are sent to PlgIspCtrl.

A DMA transfer complete IRQ, triggers the PlgISPCtrl to create a combined message, containing the raw frame and ISP configuration parameters (typically generated by 3A algorithms).

This combined message is sent to the PlgIsp, that will convert the raw pixels to processed YUV image.

The YUV image is transferred to PlgOut plugin for transmission over HDMI.

2.2 Common Types used in Plugins

A key design consideration for pipelines designed using the FLIC framework is to agree and use common (or application level) data types in the pipeline.

For raw and processed frames typically used in Camera or Computer Vision applications processing pixel date, the FLIC has pre-defined a number of useful Types common to these type of applications. These are defined in:

- flic/common/commonTypes/leon/FrameTypes.h
- flic/common/commonTypes/leon/ImgTypes.h

```
typedef struct {
    uint32_t    w; // Width
    uint32_t    h; // Height
} ImgSize;

typedef struct {
    int32_t     x1;
    int32_t     y1;
    int32_t     x2;
    int32_t     y2;
} ImgRect; // Rectangle coordinates


// MsgBuf data description structure
typedef struct FrameS {
    ImgFrmFmt  fmt; // raw, YUV, ..
    ImgFrmType type; //
    void *     fbPtr [IMG_MAX_PLANES];
    uint32_t   stride[IMG_MAX_PLANES];
    uint32_t   width [IMG_MAX_PLANES];
    uint32_t   height[IMG_MAX_PLANES];
    uint32_t   tSize [IMG_MAX_PLANES];
    uint32_t   nPlanes;
    uint64_t   ts     [IMG_MAX_TS];
    uint32_t   tsEvent[IMG_MAX_TS]; // app specific events associated
                                    // with timestamps
    uint32_t   tsNr;
    uint32_t   seqNo;     // index for the frame
    void       *userData; // associated app specific information
                          // with this frame
    uint32_t   instId;    // instance associated with this frame
} FrameT;
```

```
class IspInputMsg
{
  public:
    icIspConfig*  ispCfg;  // SIPP ISP kernel parameters
    ImgFramePtr   img;     // Input frame pointer
};



// mipiRx, receiver Configuration structure
typedef struct {
    IcMipiRxCtrlNoT     controllerNo;
    uint32_t            noLanes;
    uint32_t            laneRateMbps;
    IcMipiRxDataTypeT   dataType;
    uint32_t            dataMode;
    IcMipiRxCtrlRecNoT  recNrl;
} icMipiConfig;


typedef struct {
    ImgSize          cameraOutputSize;
    ImgRect          cropWindow;

    /*
     * Bayer Format - Raw, Demosaic and LSC blocks should be
programmed
     * to match the Bayer order specified here.
     */
    icBayerFormat   bayerFormat;
    uint32_t        bitsPerPixel;

    /* mipi RX data configuration     */
    icMipiConfig    mipiRxData;
    // used just for pc run version, and contain input file name
    const char *inFileName;
} icSourceConfig;
```

3 Building & Running the Camera Application on MV0212

3.1 Dependencies

The following items are required to run the Streaming Camera Application:

1. MV0212 development kit with MV0200 camera daughter card (Sony IMX208 sensor).
2. MDK release 17.11 or later.
3. Olimex JTAG connector and USB cable.
4. PC Linux.
5. HDMI cable and HDMI capable monitor or TV.

3.2 Building & Running the Camera Application

To Build and run the application on the MV0235:

1. `cd mdk/common/components/flic/examples/arch/ma2x5x/flic06LiveCam`
2. `make clean all -j4`
3. `make debug`

To view the ISP processed YUV frames transmitted over HDMI to TV or monitor from the HDMI interface on the Myriad development board.