SDK RELEASE NOTES

VGA2USB, DVI2USB
VGA2USB LR, VGA2USB HR, VGA2USB PRO
DVI2USB SOLO, DVI2USB DUO
KVM2USB, KVM2USB LR, KVM2USB PRO
VGA2ETHERNET, KVM2ETHERNET
VGA2PCI, DVI2PCI

EPIPHAN SYSTEMS INC.

CONTENTS

INTRODUCTION

SDK contains interface definition files (.h), libraries and sample code for Epiphan System's family of VGA/DVI frame grabbers.

QUICK START

Good place to start is to open the solution file for Microsoft Visual Studio 2005 (in SDK\epiphan\samples\v2u folder). Solution includes all other project provided in the SDK.

The heart of the SDK is SDK\epiphan\frmgrab\include\frmgrab.h file describing the interface to Epiphan's frmgrab library that allows you to work with Epiphan System's frame grabbers including USB and network grabbers.

SDK\epiphan\samples\v2u_lib demonstrates how to use the ioctls to control USB based frame grabbers.

SDK\epiphan\samples\v2u demonstrates how to use frmgrab API.

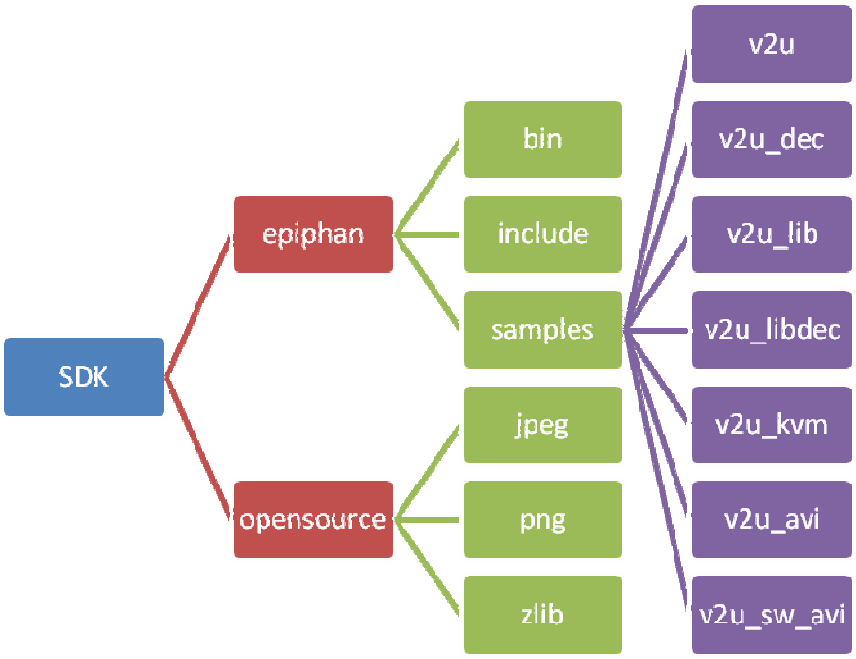
SDK\epiphan\samples\v2u_dec demonstrate how to work with on-board compression (raw format container).

SDK\epiphan\samples\v2u_libdec contains dll required to decompress frames.

SDK\epiphan\samples\v2u_avi demonstrate how to encapsulate on-board compression into AVI file container (v2u_ds_decoder.ax is required to work with such AVI files).

SDK LAYOUT

The SDK consists of the following files



SDK\OPENSOURCE

This directory contains open source libraries used to save captured frames

SDK\EPIPHAN\BIN

This directory contains ready to use precompiled examples.

- v2u.exe – utility performing simple operations like capture and save frame; detect VGA mode; adjust capture parameters such as contrast and brightness.
- v2u_dec.exe – utility demonstrating saving and decoding frames using Epiphan Systems' on board compression. This utility works only with VGA2USB LR/HR/Pro and DVI2USB Duo/Solo devices.
- v2u_libdec.dll – decompression algorithm library. Required for v2u_dec.exe.
- frmgrab.dll – unified frame grabber API for network and local frame grabber access. Required for v2u.exe
- v2u_avi.exe – utility demonstrating saving frames using Epiphan Systems' on board compression into AVI file container. This utility works only with VGA2USB LR/HR/Pro and DVI2USB Duo/Solo devices.
- v2u_ds_decoder.ax – DirectShow codec to process AVI files (created by v2u_avi) containing on-board compressed imaged in third party DirectShow compatible application.

- v2u_kvm.exe – utility demonstrating mouse and keyboard operations via KVM2USB device. NOTE: KVM protocol has changed in version 3.16.0. See source code for details.

SDK\EPIPHAN\INCLUDE

This directory contains interface definition files (.h) for Epiphan System's family of USB-based VGA/DVI frame grabber.

SDK\EPIPHAN\FRMGRAB

This directory contains interface definition files (.h) and binaries for Epiphan System's unified frame grabber interface which supports local and network frame grabbers. Windows applications using this interface need frmgrab.dll at runtime. This shared library is redistributable. Also included are static libraries for Mac OS X and Linux operating systems.

SDK\EPIPHAN\SAMPLES

Source code for examples

ONBOARD COMPRESSION (AVI FORMAT)

This section explains how to record AVI files directly from Epiphan LR/HR/Pro and DVI2USB Duo/Solo frame grabbers using on-board compression. This approach does not require any significant involvement of CPU into the capturing process, thus it's mainly targeted for solutions involving low-performance CPU platforms or otherwise CPU-bound applications.

RECORD AVI FILE

v2u_avi.exe utility can be used to directly record frames compressed by the frame grabber to an AVI file. The utility takes name of the AVI files as an argument. By default, the utility records frames using compressed RGB24 colorspace. If desired, optional '-p' argument may be specified to record compressed YUY2 colorspace.

Example:

```
v2u_avi.exe test.avi
```

INSTALL EPIPHAN DECODER DIRECTSHOW FILTER

In order to play AVI files recorded directly from Epiphan VGA2USB LR/HR/Pro and DVI2USB Duo/Solo hardware you need to install Epiphan Decoder DirectShow filter.

The following sequence describes installation process:

1. Unplug VGA frame grabber and close all application that may possibly use it.

2. Uninstall previous version of decoder by running
   regsvr32 /u <full path to previous location of v2u_ds_decoder.ax>
3. Remove previous version of decoder from the hard driver if required
4. Unpack new version of the decoder from the archive
5. Register DirectShow filter by running
   regsvr32 <full path to previous location of v2u_ds_decoder.ax>

PLAY AVI FILE

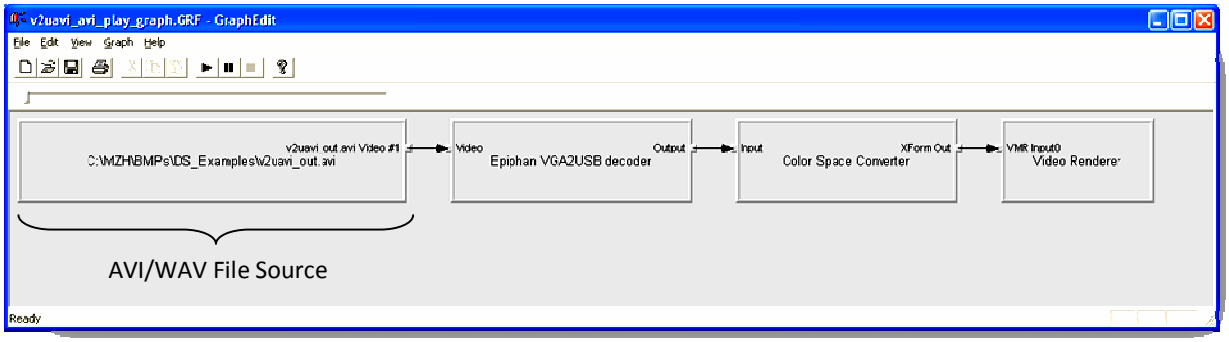
MICRSOFT MEDIAPLAYER

After installing the Epiphan Decoder DirectShow Filter, DirectShow compatible applications such as Microsoft Media Player will be able to play AVI files recorded using Epiphan on-board compression.

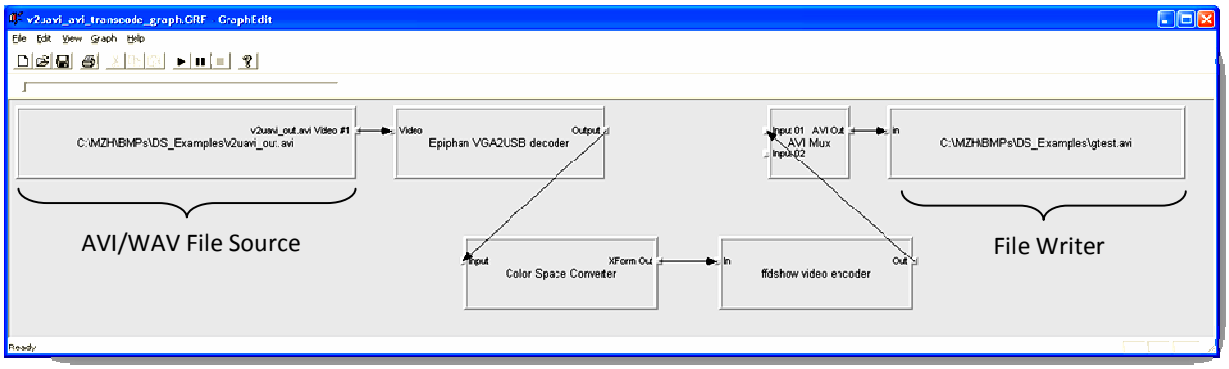
CUSTOM APPLICATIONS

AVI files containing Epiphan on-board compression can be dealt with using de following DirectShow graphs.

Playing:



Transcoding:



ONBOARD COMPRESSION (RAW FORMAT)

V2u_dec.exe utility demonstrates operations with on-board compression on lower level. Before starting please make sure that

1. You have VGA2USB LR/HR/Pro or DVI2USB Duo/Solo device as only these devices support on-board compression.

2. Version of the installed driver is at least 3.7.0.0000

SAVE COMPRESSED FRAMES

To capture compressed frames please execute v2u_dec.exe with the following parameters:

```
v2u_dec.exe 100 test.epm
```

The utility will capture 100 frames and store them in test.epm files in linear form (one compressed frame followed by another without any additional paketization and/or meta-information).

DECOMPRESS STORED FRAMES

To decompress stored frames please execute v2u_dec.exe with the following parameters:

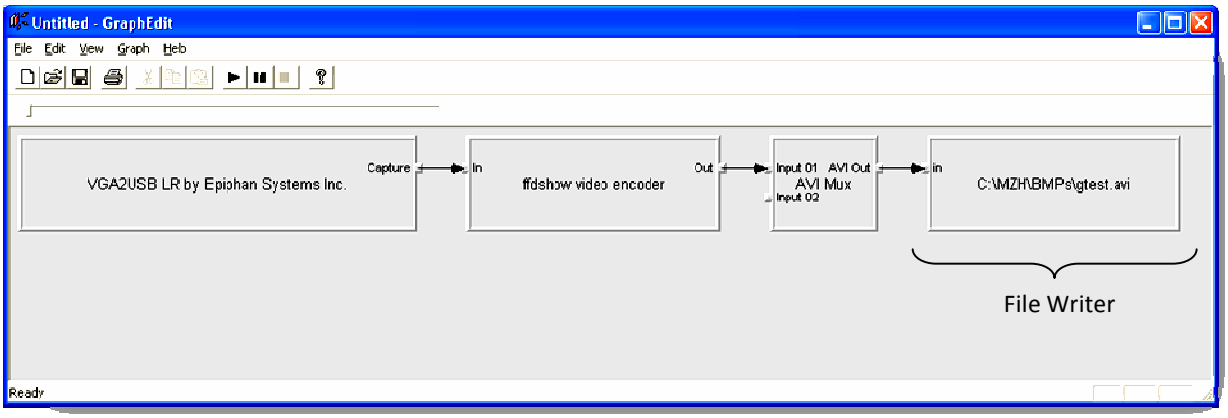
```
v2u_dec.exe x test.epm
```

The utility will extract all stored frames from test.epm file, decompress them and saves back to the disk with the following names: test.epm.NNNN.bmp, where NNNN is the sequential number of the frame.

RECORDING AVI WITH SOFTWARE COMPRESSION

USING DIRECTSHOW

All Epiphan frame grabbers support DirectShow API. The following graph demonstrates recording of an AVI file.



The v2u_dshow sample demonstrates how to detect Epiphan grabber and programmatically adjust grabber parameters such as frame rate, fixed resolution and so on.

FRMGRAB API

FrmGrab is the library that provides unified API to access USB and network frame grabbers. This section briefly describes FrmGrab API defined in epiphan\frmgrab\include\frmgrab.h

void **FrmGrab_Init**(void)
void **FrmGrabNet_Init**(void)

These functions initialize internal data structures of FrmGrab library. You may but DO NOT have to call these functions on Windows. That's because Windows version of FrmGrab is distributed as a DLL and it can initialize itself when it's being loaded. However, you DO have to call these functions if you are using FrmGrab on Mac OS X or Linux. You have to call FrmGrabNet_Init of you are planning to access network frame grabbers, otherwise you may call FrmGrab_Init. These functions must be called before any other FrmGrab functions.

void **FrmGrab_Deinit**(void)
void **FrmGrabNet_Deinit**(void)

These functions are opposite to FrmGrab_Init and FrmGrabNet_Init. Similarly, you may but DO NOT have to call these functions on Windows. You have to call one of these functions on Mac OS X or Linux in order to deallocate memory used by FrmGrab internal data structures. Call FrmGrab_Deinit if you called FrmGrab_Init in the beginning of your program, or call FrmGrabNet_Deinit if you called FrmGrabNet_Init.

FrmGrabber* **FrmGrab_Open**(const char* location)

FrmGrab_Open opens a frame grabber described by the location parameter which has the following syntax:

`local:[SERIAL]`	Specifies a local frame grabber. Optionally, the serial number can be specified.
`net:[ADDRESS[:PORT]]`	Specifies a network frame grabber at the specified address/port. If no address is specified, then FrmGrab_Open attempts to find and open a random network frame grabber on your local subnet.
`sn:SERIAL`	Specifies a local or network frame grabber with the specified serial number. FrmGrab_Open checks the local frame grabbers first then goes to the network.
`id:INDEX`	Specifies a local frame grabber with the specified index.

FrmGrabber* **FrmGrab_Dup**(FrmGrabber* fg)

FrmGrab_Dup function duplicates handle to the frame grabber. Returns a new independent FrmGrabber instance pointing to the same piece of hardware.

const char* **FrmGrab_GetSN**(FrmGrabber* fg)

FrmGrab_GetSN function returns frame grabber's serial number. The pointer is valid for the entire lifetime of the FrmGrabber instance.

int FrmGrab_**GetProductId**(FrmGrabber* fg)

FrmGrab_GetProductId function returns the unique product id. Includes product type bits (lower 16 bits) OR'ed with type-specific product id.

const char* **FrmGrab_GetProductName**(FrmGrabber* fg)

FrmGrab_GetProductName function returns the product description string ("VGA2USB", "VGA2Ethernet", etc)

const char* **FrmGrab_GetLocation**(FrmGrabber* fg)

FrmGrab_GetLocation function returns a string that describes the location of the grabber ("USB, "192.168.0.122", etc).

V2U_BOOL **FrmGrab_DetectVideoMode**(FrmGrabber* fg, V2U_VideoMode* vm)

FrmGrab_DetectVideoMode function returns the video mode detected by the frame grabber. It returns V2U_TRUE on success, V2U_FALSE otherwise. The vm parameter points to V2U_VideoMode structure defined in v2u_defs.h:

```
typedef struct ioctl_videomode {
    V2U_INT32  width;       /* screen width, pixels */
    V2U_INT32  height;      /* screen height, pixels */
    V2U_INT32  vfreq;       /* vertical refresh rate, mHz */
} V2U_VideoMode;
```

If no signal is detected, all fields are set to zero. Example:

```
    /* Detect video mode */
    V2U_VideoMode vm;
    if (FrmGrab_DetectVideoMode(fg,&vm) && vm.width && vm.height) {

        printf("Detected %dx%d %d.%d Hz\n",vm.width,vm.height,
            (vm.vfreq+50)/1000, ((vm.vfreq+50)%1000)/100);

    } else {
        printf("No signal detected\n");

    }
```

V2U_BOOL **FrmGrab_GetGrabParams**(FrmGrabber* fg, V2U_GrabParameters* gp)

FrmGrab_GetGrabParams function queries the current VGA capture parameters. It returns V2U_TRUE on success, V2U_FALSE otherwise.

V2U_BOOL **FrmGrab_SetGrabParams**(FrmGrabber* fg, const V2U_GrabParameters* gp)

FrmGrab_SetGrabParams function sets VGA capture parameters. It returns V2U_TRUE on success, V2U_FALSE otherwise.

V2U_BOOL **FrmGrab_GetProperty**(FrmGrabber* fg, V2U_Property* prop)

FrmGrab_GetProperty function queries the device property. It returns V2U_TRUE on success, V2U_FALSE otherwise. The caller sets prop->key field to one of the V2UPropertyKey enum values defined in v2u_defs.h. Upon successful return, the value of the property can be found in prop->value. Example:

```
    /* Check if KVM functionality is supported */
    V2U_Property p;
    p.key = V2UKey_KVMCapable;
    if (FrmGrab_GetProperty(fg, &p)) {
        if (p.value.boolean) {

            ... // do something KVM specific

        } else {
            printf("Frame grabber doesn't support KVM functionality\n");
        }

    }
```

V2U_BOOL **FrmGrab_SetProperty**(FrmGrabber* fg, const V2U_Property* prop)

FrmGrab_SetProperty sets the device property. It returns V2U_TRUE on success, V2U_FALSE otherwise.

V2U_BOOL **FrmGrab_SendPS2**(FrmGrabber* fg, const V2U_SendPS2* ps2)

FrmGrab_SendPS2 function sends PS/2 events to a KVM-capable frame grabber, like KVM2USB or KVM2Ethernet. It returns V2U_TRUE on success, V2U_FALSE otherwise. You can use V2UKey_KVMCapable property to check whether frame grabber supports KVM functionality (see the example above).

void **FrmGrab_Start**(FrmGrabber* fg)

FrmGrab_Start function signals the frame grabber to prepare for capturing frames with maximum frame rate. Currently, it doesn't matter for local grabbers, however it's really important for network grabbers. For network

grabbers, this function turns streaming on, otherwise FrmGrab_Frame will have to work on request/response basis, which is much slower.

void **FrmGrab_Stop**(FrmGrabber* fg)

FrmGrab_Stop signals the grabber to stop capturing frames with maximum frame rate.

V2U_BOOL **FrmGrab_SetMaxFps**(FrmGrabber* fg, double maxFps)

FrmGrab_SetMaxFps function sets intended frame rate limit (average number of FrmGrab_Frame calls per second). The frame grabber may use this hint to reduce resource usage, especially in low fps case. For example, it may reduce the network bandwidth utilization if you are using a network frame grabber.

V2U_GrabFrame2* **FrmGrab_Frame**(FrmGrabber* fg, V2U_UINT32 format, const V2URect* crop)

FrmGrab_Frame function grabs one frame. The caller doesn't have to call FrmGrab_Start first, but it's recommended in order to achieve maximum possible frame rate. The second parameter is the capture format, i.e. one of V2U_GRABFRAME_FORMAT_* constants defined in v2u_defs.h. The last parameter is a pointer to the requested crop rectangle. Pass NULL if you need the whole frame. Note that you have to release the frame when you no longer need it by calling FrmGrab_Release.

void **FrmGrab_Release**(FrmGrabber* fg, V2U_GrabFrame2* frame)

FrmGrab_Release function releases the frame previously returned by FrmGrab_Frame

void **FrmGrab_Close**(FrmGrabber* fg)

FrmGrab_Close function closes the frame grabber and invalidates the handle. All frames returned by FrmGrab_Frame must be released prior to calling FrmGrab_Close.

USB SPECIFIC FUNCTIONS

FrmGrabber* **FrmGrabLocal_Open**(void)

FrmGrabLocal_Open function opens the default USB frame grabber. If you have several USB frame grabbers attached to your system, it’s not guaranteed which one of them will be opened. This function returns NULL if it doesn’t find any frame grabbers attached to you computer.

FrmGrabber* **FrmGrabLocal_OpenSN** (const char* sn)

FrmGrabLocal_OpenSN function opens a USB frame grabber with the specified serial number. This function returns NULL if it doesn’t find the requested frame grabber.

int **FrmGrabLocal_Count**(void)

This function returns the number of USB frame grabbers connected to your system.

int **FrmGrabLocal_OpenAll**(FrmGrabber* grabbers[], int maxcount)

FrmGrabLocal_OpenAll opens multiple USB frame grabbers at once. It returns the number of actually opened frame grabbers.

NETWORK SPECIFIC FUNCTIONS

FrmGrabber* **FrmGrabNet_Open**(void)

FrmGrabNet_Open function attempts to find and open a network grabber on your subnet. It returns the frame grabber handle on success, NULL on failure.

FrmGrabber* **FrmGrabNet_OpenSN**(const char* sn)

FrmGrabNet_OpenSN function attempts to find and open a network grabber on your subnet with the specified serial number.

FrmGrabber* **FrmGrabNet_OpenLocation**(const char* location)

FrmGrabNet_OpenAddress function connects to the frame grabber at specified location (host name or IP address).

FrmGrabber* **FrmGrabNet_OpenAddress**(V2U_UINT32 ipaddr, V2U_UINT16 port)

FrmGrabNet_OpenAddress function connects to the frame grabber at specified IP address. Pass zero as the port number to connect to the default port.

FrmGrabber* **FrmGrabNet_OpenAddress2**(V2U_UINT32 ipaddr, V2U_UINT16 port, FrmGrabAuthProc authproc, void* param, FrmGrabConnectStatus* status)

FrmGrabNet_OpenAddress function connects to the frame grabber at specified IP address. Pass zero as the port number to connect to the default port. If the frame grabber requires authentication, calls the provided FrmGrabAuthProc callback to get username and password. FrmGrabAuthProc callback has the following prototype:

```
typedef V2U_BOOL (*FrmGrabAuthProc)(char* user, char* pass, void* param);
```

The maximum size of username is FG_USERNAME_SIZE (32), the maximum password size is FG_PASSWORD_SIZE (64) bytes. Both username and password must be encoded in UTF-8. The password is never sent over the network. If username is not required, the password pointer is NULL.

This is a convenience function that combines FrmGrabNet_OpenAddress and, if necessary, FrmGrabNet_Auth functionality.

V2U_BOOL **FrmGrabNet_IsProtected**(FrmGrabber* fg)

FrmGrabNet_IsProtected function checks if the network frame grabber is password protected. FrmGrabNet_Open, FrmGrabNet_OpenSN and FrmGrabNet_OpenLocation function may return a frame grabber handle that does not allow pretty much anything until it gets “unprotected”. In other words, before the client can do anything, it must first prove that it knows the credentials. You have to use FrmGrabNet_Auth function below to pass authentication.

FrmGrabConnectStatus **FrmGrabNet_Auth**(FrmGrabber* fg, FrmGrabAuthProc authproc, void* param)

FrmGrabNet_Auth authenticates the client if necessary. See documentation for FrmGrabNet_OpenAddress2 function for the description of FrmGrabAuthProc.

V2U_BOOL **FrmGrabNet_GetStat**(FrmGrabber* fg, FrmGrabNetStat* netstat)

FrmGrabNet_GetStat function returns the network statistics, i.e. number of bytes sent and received over the network.

V2U_BOOL **FrmGrabNet_GetRemoteAddr**(FrmGrabber* fg, struct sockaddr_in* addr)

FrmGrabNet_GetRemoteAddr function returns the network address of the frame grabber.