Application Note No. LAN-052e

phyCAM Camera Modules Register Settings

Revision history:

Version	Changes	Date	Author
1.0	initial release	16.07.2009	H. Fendrich
1.1	WinCE added, new cameras added	20.01.2011	H. Fendrich
A2	added parameters for GStreamer plugin	14.04.2014	H. Fendrich

Inhaltsverzeichnis

1 Overview 2
2 Access to the camera registers 3
2.1 General register access (Linux) 3
2.2 General register access (WinCE) 4
3 Register Definition File 4
3.1 Register file syntax 4
3.2 Using the Register Definition File (Linux) 6
3.2.1 GStreamer Function 6
3.2.2 Requirements 7
3.2.3 Example of including a register definition (Linux) 7
3.3 Using the Register Definition File (WinCE) 8
3.3.1 WinCE Demo Software 8
3.3.2 Requirements 8
3.3.3 Example of uploading a register definition (WinCE) 8

1 Overview

The phyCAM digital camera modules are supported by the BSPs shipped with the Phytec microcontroller boards (SOMs).
The name of the corresponding drivers reflect the sensor chip of the camera module.

Examples:

- VM-006 series : Aptina mt9m001
- VM-007 series : Aptina mt9v022
- VM-009 series : Aptina mt9m131
- VM-010 series : Aptina mt9v024
- VM-011 series : Aptina mt9p031

Note that the camera modules supported for a given BSP might vary for different SOMs. Phytec is constantly improving the support for the camera modules with new BSP versions.
Please check if a certain camera module is supported by the BSP version before starting with your develoment.

Linux:
With Linux the cameras are accessed via the *Video4Linux 2* – interface (V4L2).
This allows access to image data directly from application software.Also multimedia frameworks like *GStreamer* can access the camera via the V4L2 interface.
Note that V4L2 implements many functions for different device types. Not all functions are available for the phyCAM camera modules. It's recommended to determine the supported functions by using the *Capability Querying*. This procedure and a detailed description of the V4L2 functions can be found in the *Video for Linux Two API Specification*.

WinCE:
The camera drivers for Windows Embedded are compatible to *DirectShow*.
Please see the demo application shipped with the development kit as a reference of how to access the camera data from your own application.
More informations can be found in the brief description of the corresponding drivers.

Note:
The camera sensors offer numerous options to control and fine-tune the image acquisition.
In general these settings are made by changing the values of the camera sensor's registers. The register contents are accessed by the I²C interface of the sensor.
The camera sensor driver already controls many register settings automatically when the camera device is opened or certain camera functions are invoked.
However, some special settings can not be accessed via the V4L2 driver (like, for examply, the settings for HDR mode that is provided by some cameras).
In this case the application software developer can use direct register access via the I²C driver to read, write and modify the settings of the camera's registers.

2 Access to the camera registers

This section describes how to change register settings of the camera sensor directly. This can be used to alter camera settings that are not supported by the V4L2 interface.
Direct access to the camera registers is aceivers by using APIs / function calls aside the V4L2 interface.

Important Note

The correct setting of some camera registers are fundamental for the function of the camera and an undisturbed image acquisition.
Changing the contents of these registers will result in image desturbances or an unstable system behavior.
Because of this Linux allows a manual modification of the registers **in debug mode only**.

Before changing any camera settings please verify carefully that the modifications will not harm the camera function nor affect the stability of the system.

Some registers are controlled by the V4L2 camera driver.
This means that changing these registers might affect the behavior of the driver.
On the other hand, some settings might be overwritten by the V4L2 driver, for examples when certain driver functions are executed. Thus these settings might not take any effect when followed by a driver call.

2.1 General register access (Linux)

To access the camera register with Linux the following V4L2 APIs can be used:

- *VIDIOC_DBG_G_REGISTER*
- *VIDIOC_DBG_S_REGISTER*

Important Note:
These functions can oly be used if the driver is compiled in **debug mode** (*CONFIG_VIDEO_ADV_DEBUG*)!

The camera drivers shipped with the Phytec distribution are already compiled in debug mode.

The functions for register access can be used in custom application software.
Before the integration of register settings into the application software, it might be a good idea to test the effects with simple tools. For this purpose Phytec ships a plugin for the popular *GStreamer* framework with the Embedded Imaging Kits.
The *GStreamer* framework can be configured by simple script writing. The example scripts that come with the kits use the *GStreamer* for demos like transmission and

viewing image streams or saving single shots and live streams to memory in various formats. See the phyCAM manual (L-748), section 3 for more information about the example scripts.

2.2 General register access (WinCE)

For Windows CE standard I²C routines can be used to access the camera registers. The Phytec BSP contains already these functions and these functions are also called by the camera driver.

The functions for register access can be used in custom application software. Before the integration of register settings into the application software, it might be a good idea to test the effects with simple tools. Phytec ships a demo application with the Embedded Imaging Kit which allows to alter register contents and to watch the changes in the live image.

3 Register Definition File

For adapting the camera characteristics to a certain use case it is helpful to alter several registers at once. For this Phytec provides a tool that loads the register map from a text file into the camera registers.
Also, predefined register files (*use cases*) are provided for many cameras. The register files can be edited by using a standard text editor.

3.1 Register file syntax

For each camera module an example for a register definition file comes with the development kits.
In the kits the following name convention is used, however, any file name can be used for custom register definition files:

- „register-settings-mt9v022.txt“ (VM-007)
- „register-settings-mt9m001.txt“ (VM-006)
- „register-settings-mt9m131.txt“ (VM-009)

The content of a register definition file must be in accordance with the following syntax:

chip-address, register-address, value, [mode]

chip-address =	I²C device address of the camera sensor format: 8 bit hex (examples MT9V022 = 0x48 / MT9M001 = 0x5D)
register-address =	Register address of the register that is to be written format: 8 bit hex
value =	value to be written to the register specified by *register-address* format: either 16 bit hex (leading 0x...) or decimal

mode = optional, allows a logical operation of *value* and the current register content.
parameters:
a (AND, bitwise AND of current content and *value*)
o (OR, bitwise AND of current content and *value*),
x (XOR, bitwise XOR of current content and *value*)

Comments can be included by a leading hash (#).
Note:
This is a simple way to switch a certain modification on or off. See example below.

The example definition files shipped with the BSPs already inculde many options with most of them set inactive by a leading **#**.
For testing a specific setting just remove the **#** at the beginning of the line.

Example:
For the MT9V022 sensor (VM-007) the AGC level (automatic gain control) is to be set to gain factor x2.
To acheive this, the # at the beginning of the corresponding line was removed (line printed bold in the example below):

```
---------------------------
#  set MT9V022 AGC
#  ===============
#  set AGC automatic
#0x48,0xAF,0x0002,o     # automatic = on

#  set AGC manual
0x48,0xAF,0xFFFD,a      # automatic = off
#0x48,0x35,0x0010 # set gain = 1.00
#0x48,0x35,0x0014 # set gain = 1.25
#0x48,0x35,0x0018 # set gain = 1.50
#0x48,0x35,0x001C # set gain = 1.75
0x48,0x35,0x0020  # set gain = 2.00
#0x48,0x35,0x0024 # set gain = 2.25
#0x48,0x35,0x0028 # set gain = 2.50
#0x48,0x35,0x002C # set gain = 2.75
#0x48,0x35,0x0030 # set gain = 3.00
…
---------------------------
```

After loading the register settings into the camera the image is captured with a fixed gain setting of 2. The predefined register definition files include more optional settings for the individual camera modules.

For more informations about the register settings of a camera sensor see the datasheet of the sensor. Datasheets for the sensor chip can be downloaded from the sensor manufacturer's website (for example for MT9M001, Mt9V022 see www.aptina.com).

We recommend using a FTP connection between the developent computer and the embedded system for transferring register definition files between PC and the target system. This allows a comfortable editing of the file.

The predefined example files for the camera sensors are located:

- for **Linux** in the same directory as the example scripts for the GStreamer …/gstreamer_examples/…

- for **WinCE** in the folder …/NandFlash/.. .
  If the register definition file in this location should be deleted or renamed the demo software will create a default file.

Note

If a register definition includes more than one definition for a dedicated register the definitions are written in the same order as they are listed in the file.
This means that the value of the last definition is in effect after the register upload has finished. However, all other write commands have been executed before (and might have had affected the sensor's behavior).
Take care that a desired setting is not overwritten by a later definition of the file, especially if you are using the comment (#) method.

In connection with the logical operands (AND, OR, XOR) consecutive write commands can be helpful to modify only certain parts of a register (bit masking).

Example:
Setting the lower byte of a register to 0x35 without altering the upper byte:

```
chip-address, register-address, 0xFF00,a # clear lower byte
chip-address, register-address, 0x0035,o # write 0x35 to low-byte
```

3.2 Using the Register Definition File (Linux)

3.2.1 GStreamer Function

The script examples shipped by Phytec show how the *GStreamer*-examples can be extended with regsister file upload.
In the first step, the desired camera configuration is set by editing the register definition file. This can be done by adding / removing the comment signs (#) from certain lines or adding new register definitions.
In the second step this file is uploaded by a GStreamer command added to the pipeline.

3.2.2 Requirements

- Make sure that the camera driver used has been compiled in debug mode (see section 2.1). The camera drives shipped with the Phytec BSPs are already compiled in debug mode.
- *GStreamer* - plug-in for register upload (part of the Phytec distribution).

3.2.3 Example of including a register definition (Linux)

This example shows how to use a register definition file with the *GStreamer* framework. For this demonstration we use the *bwcam-save_jpg_full_res* script for the VM-007-BW (mt9v022) camera module.
This script captures a single shot image from the VM-007-BW camera and saves it to a JPEG-file.
The image sensor of the VM-007-BW module (Aptina MT9V022 sensor) is operated with its default settings in automatic mode. Thise means AGC/AEC and atuomatic BLC are enabled.

The script looks like this:

```
--------------------------
#!/bin/sh

. `dirname $0`/func.sh

init_dev bw
[ $? -ne 0 ] && exit 1

guess_res

echo "starting gstreamer ..."
gst-launch \
      V4L2src num-buffers=1 ! \
      video/x-raw-gray$FRAME_SIZE ! \
      ffmpegcolorspace ! \
      video/x-raw-yuv$FRAME_SIZE ! \
      jpegenc ! \
      filesink location=bw_image.jpg
--------------------------
```

To upload register settings to the camera sensor, we have to add the plug-in ***i2c*** to this script. When processed by the GStreamer pipeline, this script uploads the register settings from a text file to the specified camera (see syntax in section 3.1).
The plug-in syntax needs the following parameters:
`i2c file=[name of register definition file] show=0`
It's important to place the plug-in at the right position in the script so that the settings are not overwritten by another process.

A good place for our example is between the initialization and the image capture call:

```
--------------------------
      …
      V4L2src num-buffers=1 ! \
      i2c file=register-settings-mt9v022.txt show=0 ! \
      video/x-raw-gray$FRAME_SIZE ! \
      …
--------------------------
```

The register definition file (here: *register-settings-mt9v022.txt*) is loacted in the working directory.
Setting of the parameter *show=1* echos the register access also on the serial interface (console output).

Type `gst-inspect i2c` to list more parameter options for the

Note:
If there is more than one camera present in a system, the parameter `addr=[i2c hex_address of the device]` allows to use the same definition file for more cameras of the same type but different I²C addresses.
Adding the `addr-` parameter will ignore the I²C addresses defined in the file and replace them with the address specified.
Example for video-device_0 at address 0x48 and video-device_1 at addresse0x4C:

```
i2c addr=0x48 file=register-settings-mt9v022.txt show=0  dev=/dev/video0 ! \
i2c addr=0x4c file=register-settings-mt9v022.txt show=0  dev=/dev/video1 ! \
```

3.3 Using the Register Definition File (WinCE)

3.3.1 WinCE Demo Software

The Demo Software for Windows CE / Embedded shipped with the Embedded Imaging Kit allows to upload register definition files to the camera.

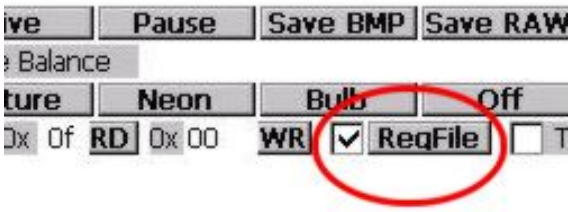
3.3.2 Requirements

A register definition file has to be present on the sytem.
For example in the folder `.../NandFlash/...` (part of the Phytec distribution).

3.3.3 Example of uploading a register definition (WinCE)

Start the demo application *CameraApp.exe* which is located in the Windows-Folder (*My Device/Windows/…*) on the embedded System.

Cick on the *RegFile* – button of the application to select a register definition file.
Activate the register upload by selecting the RegFile function checkbox: ☑



To make this settings permanent, call the *savereg* function after closing the demo application. By this, the contents of the register file will be uploaded to the camera automatically every time the demo application is started.
If the register definition file in this location `.../NandFlash/...` should be deleted or renamed the demo software will create a default file.