Arduino Camera Driver

Generated by Doxygen 1.8.9.1

Wed Aug 19 2015 01:06:25

Contents

1 Hierarchical Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

2 Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

3 File Index

3.1 File List

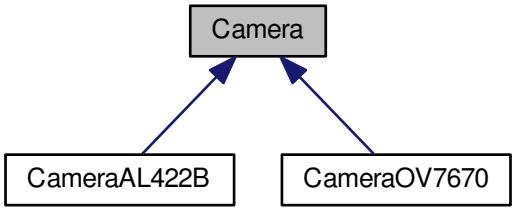
Here is a list of all files with brief descriptions:

4 Class Documentation

4.1 Camera Class Reference

`#include <Camera.h>`

Inheritance diagram for Camera:



Public Member Functions

- virtual bool capture ()=0
- Camera ()
- void Begin ()
- void Begin (uint8_t address)
- void Begin (int address)
- bool Reset ()
- void Init ()
- void ColorBar (bool On)
- void Power (bool On)
- void Mirror (bool On)
- bool Capture ()
- void Dump (bool Hex)
- void DumpConfig ()
- uint8_t ReadConfigByte (uint8_t MemAddr)
- uint8_t ReadNextVideoByte ()
- void DumpVideoByte (uint8_t pixel, uint8_t *count)
- void ResetVideoPointer ()

Static Public Member Functions

- static void DebugPrintValue (uint8_t Value)
- static void UYV2RGB (uint8_t U, uint8_t Y, uint8_t V, uint8_t *R, uint8_t *G, uint8_t *B)

Static Private Member Functions

- static uint8_t Clip (float Value)

Private Attributes

- int _addr

4.1.1 Detailed Description

Arduino - Camera interface.

Camera.h

The abstract class for a Camera.

Author

Dalmir da Silva dalmirdasilva@gmail.com

Definition at line 14 of file Camera.h.

4.1.2 Constructor & Destructor Documentation

4.1.2.1 Camera::Camera ( )

Definition at line 29 of file MIN_at_Camera.cpp.

4.1.3 Member Function Documentation

4.1.3.1 void Camera::Begin ( )

Definition at line 37 of file MIN_at_Camera.cpp.

4.1.3.2 void Camera::Begin (uint8_t *address*)

4.1.3.3 void Camera::Begin (int *address*)

Definition at line 47 of file MIN_at_Camera.cpp.

4.1.3.4 virtual bool Camera::capture ( ) [pure virtual]

Captures a frame.

Implemented in CameraAL422B.

4.1.3.5 bool Camera::Capture ( )

Definition at line 150 of file MIN_at_Camera.cpp.

4.1.3.6 uint8_t Camera::Clip (float *Value*) [static], [private]

Definition at line 341 of file MIN_at_Camera.cpp.

4.1.3.7 void Camera::ColorBar (bool *On*)

Definition at line 135 of file MIN_at_Camera.cpp.

4.1.3.8 void Camera::DebugPrintValue (uint8_t *Value*) [static]

Definition at line 318 of file MIN_at_Camera.cpp.

4.1.3.9 void Camera::Dump (bool *Hex*)

Definition at line 189 of file MIN_at_Camera.cpp.

4.1.3.10 void Camera::DumpConfig ( )

Definition at line 214 of file MIN_at_Camera.cpp.

4.1.3.11 void Camera::DumpVideoByte (uint8_t *pixel*, uint8_t * *count*)

Definition at line 286 of file MIN_at_Camera.cpp.

4.1.3.12 void Camera::Init ( )

Definition at line 93 of file MIN_at_Camera.cpp.

4.1.3.13 void Camera::Mirror (bool *On*)

Definition at line 145 of file MIN_at_Camera.cpp.

4.1.3.14 void Camera::Power (bool *On*)

Definition at line 140 of file MIN_at_Camera.cpp.

4.1.3.15 byte Camera::ReadConfigByte (uint8_t *MemAddr*)

Definition at line 264 of file MIN_at_Camera.cpp.

4.1.3.16 uint8_t Camera::ReadNextVideoByte ( )

Definition at line 273 of file MIN_at_Camera.cpp.

4.1.3.17 bool Camera::Reset ( )

Definition at line 61 of file MIN_at_Camera.cpp.

4.1.3.18 void Camera::ResetVideoPointer ( )

Definition at line 299 of file MIN_at_Camera.cpp.

4.1.3.19 void Camera::UYV2RGB (uint8_t *U*, uint8_t *Y*, uint8_t *V*, uint8_t * *R*, uint8_t * *G*, uint8_t * *B*) [static]

Definition at line 326 of file MIN_at_Camera.cpp.

4.1.4 Member Data Documentation

4.1.4.1 int Camera::_addr [private]

Definition at line 111 of file MIN_at_Camera.h.

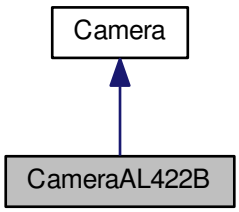
The documentation for this class was generated from the following files:

- Camera.h
- MIN_at_Camera.h
- MIN_at_Camera.cpp

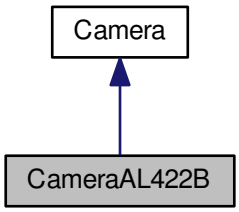
4.2 CameraAL422B Class Reference

`#include <CameraAL422B.h>`

Inheritance diagram for CameraAL422B:



Collaboration diagram for CameraAL422B:



Classes

- union MVFPbits

Public Types

- enum Mask {
  STR_OPT_MODE = 0x03, STR_OPT_REQUEST = 0x80, STR_OPT_GAIN = 0x40, MVFP_MIRROR = 0x20,
  MVFP_FLIP = 0x10, COM7_RESET = 0x80, COM7_RESOLUTION = 0x38, COM7_FORMAT = 0x05,
  COM7_COLOR_BAR = 0x02, COM2_SSLEEP = 0x10, _ = 0x00, COM1_CCIR656 = 0x40,
  COM3_SWAP = 0x40, COM3_SCALEEN = 0x08, COM3_DCWEN = 0x04, CLKRC_EXT = 0x40,
  CLKRC_SCALE = 0x3f, COM8_FASTAEC = 0x80, COM8_AECSTEP = 0x40, COM8_BFILT = 0x20,
  COM8_AGC = 0x04, COM8_AWB = 0x02, COM8_AEC = 0x01, COM10_HSYNC = 0x40,
  COM10_PCLK_HB = 0x20, COM10_HREF_REV = 0x08, COM10_VS_LEAD = 0x04, COM10_VS_NEG = 0x02,
  COM10_HS_NEG = 0x01, TSLB_YLAST = 0x04, COM11_NIGHT = 0x80, COM11_NMFR = 0x60,
  COM11_HZAUTO = 0x10, COM11_50HZ = 0x08, COM11_EXP = 0x02, COM12_HREF = 0x80,
  COM13_GAMMA = 0x80, COM13_UVSAT = 0x40, COM13_UVSWAP = 0x01, COM14_DCWEN = 0x10,
  COM15_R10F0 = 0x00, COM15_R01FE = 0x80, COM15_R00FF = 0xc0, COM15_RGB = 0x30,
  COM16_AWBGAIN = 0x08, COM17_AECWIN = 0xc0, COM17_CBAR = 0x08, R76_WHTPCOR = 0x40,
  REG76_BLKPCOR = 0x20, REG76_EDGE = 0x1f }

- enum Register {
  GAIN = 0x00, BLUE = 0x01, RED = 0x02, VREF = 0x03,
  COM1 = 0x04, BAVE = 0x05, GBAVE = 0x06, AECHH = 0x07,
  RAVE = 0x08, COM2 = 0x09, PID = 0x0a, VER = 0x0b,
  COM3 = 0x0c, COM4 = 0x0d, COM5 = 0x0e, COM6 = 0x0f,
  AECH = 0x10, CLKRC = 0x11, COM7 = 0x12, COM8 = 0x13,
  COM9 = 0x14, COM10 = 0x15, HSTART = 0x17, HSTOP = 0x18,
  VSTART = 0x19, VSTOP = 0x1a, PSHFT = 0x1b, MIDH = 0x1c,
  MIDL = 0x1d, MVFP = 0x1e, AEW = 0x24, AEB = 0x25,
  VPT = 0x26, BBIAS = 0x27, GBBIAS = 0x28, EXHCH = 0x2a,
  EXHCL = 0x2b, RBIAS = 0x2c, ADVFL = 0x2d, ADVFH = 0x2e,
  YAVE = 0x2f, HSYST = 0x30, HSYEN = 0x31, HREF = 0x32,
  CHLF = 0x33, ARBLM = 0x34, ADC_CONTROL = 0x37, ACOM = 0x38,
  OFON = 0x39, TSLB = 0x3a, COM11 = 0x3b, COM12 = 0x3c,
  COM13 = 0x3d, COM14 = 0x3e, EDGE = 0x3f, COM15 = 0x40,
  COM16 = 0x41, COM17 = 0x42, AWBC1 = 0x43, AWBC2 = 0x44,
  AWBC3 = 0x45, AWBC4 = 0x46, AWBC5 = 0x47, AWBC6 = 0x48,
  REG4B = 0x4b, DNSTH = 0x4c, DM_POS = 0x4d, MTX1 = 0x4f,
  MTX2 = 0x50, MTX3 = 0x51, MTX4 = 0x52, MTX5 = 0x53,
  MTX6 = 0x54, BRIGHT = 0x55, CONTRAS = 0x56, CONTRAS_CENTER = 0x57,
  MTXS = 0x58, AWBC7 = 0x59, AWBC8 = 0x5a, AWBC9 = 0x5b,
  AWBC10 = 0x5c, AWBC11 = 0x5d, AWBC12 = 0x5e, B_LMT = 0x5f,
  R_LMT = 0x60, G_LMT = 0x61, LCC1 = 0x62, LCC2 = 0x63,
  LCC3 = 0x64, LCC4 = 0x65, LCC5 = 0x66, MANU = 0x67,
  MANV = 0x68, GFIX = 0x69, GGAIN = 0x6a, DBLV = 0x6b,
  AWBCTR3 = 0x6c, AWBCTR2 = 0x6d, AWBCTR1 = 0x6e, AWBCTR0 = 0x6f,
  SCALING_XSC = 0x70, SCALING_YSC = 0x71, SCALING_DCWCTR = 0x72, SCALING_PCLK_DIV = 0x73,
  REG74 = 0x74, REG75 = 0x75, REG76 = 0x76, REG77 = 0x77,
  SLOP = 0x7a, GAM1 = 0x7b, GAM2 = 0x7c, GAM3 = 0x7d,
  GAM4 = 0x7e, GAM5 = 0x7f, GAM6 = 0x80, GAM7 = 0x81,
  GAM8 = 0x82, GAM9 = 0x83, GAM10 = 0x84, GAM11 = 0x85,
  GAM12 = 0x86, GAM13 = 0x87, GAM14 = 0x88, GAM15 = 0x89,
  DM_LNL = 0x92, DM_LNH = 0x93, LCC6 = 0x94, LCC7 = 0x95,
  BD50ST = 0x9d, BD60ST = 0x9e, HRL = 0x9f, LRL = 0xa0,
  DSPC3 = 0xa1, SCALING_PCLK_DELAY = 0xa2, NT_CTRL = 0xa4, AECGMAX = 0xa5,
  LPH = 0xa6, UPL = 0xa7, TPL = 0xa8, TPH = 0xa9,
  NALG = 0xaa, STR_OPT = 0xac, STR_R = 0xad, STR_G = 0xae,
  STR_B = 0xaf, ABLC1 = 0xb1, THL_ST = 0xb3, THL_DLT = 0xb5,
  AD_CHB = 0xbe, AD_CHR = 0xbf, AD_CHGB = 0xc0 }
- enum FlashlightModeSelect { XENON = 0x00, LED1 = 0x01, LED2 = 0x02 }
- enum OutputFormat { YUV = 0x00, RGB = 0x04, RAW_BAYER_RGB = 0x01, PROCESSED_BAYER_RGB = 0x05 }
- enum OutputResolution { VGA = 0x00, CIF = 0x20, QVGA = 0x10, QCIF = 0x08 }
- enum RGBOutput { RGB_NORMAL = 0x00, RGB_565 = 0x10, RGB_555 = 0x30 }

Public Member Functions

- CameraAL422B (unsigned char(∗read)(), unsigned char vsyncPin, unsigned char writeEnPin, unsigned char readClockPin, unsigned char readResetPin)
- void begin ()
- virtual bool capture ()
- int readFrame (OutputStream ∗out)
- void setHorizontalMirror (bool mirror)
- void setVerticalFlip (bool flip)
- void setFlashlightModeSelect (FlashlightModeSelect mode)
- void setStrobeRequest (bool request)
- void setColorGainControlEnable (bool enable)

- void setOutputFormat (OutputFormat format)
- void setOutputResolution (OutputResolution resolution)
- void setRGBOutput (RGBOutput output)
- void enableWrite ()
- void disableWrite ()
- void resetReadPointer ()
- void configureRegisterBits (Register reg, Mask mask, unsigned char v)
- void writeRegister (Register reg, unsigned char v)
- unsigned char readRegister (Register reg)

Private Member Functions

- void resetRegisters ()
- int readRow (OutputStream *out)

Private Attributes

- unsigned char(* read)()
- unsigned char vsyncPin
- unsigned char writeEnPin
- unsigned char readClockPin
- unsigned char readResetPin
- unsigned char address
- int width
- int height

Additional Inherited Members

4.2.1 Detailed Description

Definition at line 41 of file CameraAL422B.h.

4.2.2 Member Enumeration Documentation

4.2.2.1 enum CameraAL422B::FlashlightModeSelect

Enumerator

XENON

LED1

LED2

Definition at line 693 of file CameraAL422B.h.

4.2.2.2 enum CameraAL422B::Mask

Enumerator

STR_OPT_MODE

STR_OPT_REQUEST

STR_OPT_GAIN

MVFP_MIRROR

MVFP_FLIP

COM7_RESET

COM7_RESOLUTION
COM7_FORMAT
COM7_COLOR_BAR
COM2_SSLEEP

_
COM1_CCIR656
COM3_SWAP
COM3_SCALEEN
COM3_DCWEN
CLKRC_EXT
CLKRC_SCALE
COM8_FASTAEC
COM8_AECSTEP
COM8_BFILT
COM8_AGC
COM8_AWB
COM8_AEC
COM10_HSYNC
COM10_PCLK_HB
COM10_HREF_REV
COM10_VS_LEAD
COM10_VS_NEG
COM10_HS_NEG
TSLB_YLAST
COM11_NIGHT
COM11_NMFR
COM11_HZAUTO
COM11_50HZ
COM11_EXP
COM12_HREF
COM13_GAMMA
COM13_UVSAT
COM13_UVSWAP
COM14_DCWEN
COM15_R10F0
COM15_R01FE
COM15_R00FF
COM15_RGB
COM16_AWBGAIN
COM17_AECWIN
COM17_CBAR
R76_WHTPCOR
REG76_BLKPCOR
REG76_EDGE

Definition at line 81 of file CameraAL422B.h.

4.2.2.3 enum CameraAL422B::OutputFormat

Enumerator

YUV

RGB

RAW_BAYER_RGB

PROCESSED_BAYER_RGB

Definition at line 697 of file CameraAL422B.h.

4.2.2.4 enum CameraAL422B::OutputResolution

Enumerator

VGA

CIF

QVGA

QCIF

Definition at line 712 of file CameraAL422B.h.

4.2.2.5 enum CameraAL422B::Register

Enumerator

GAIN

BLUE

RED

VREF

COM1

BAVE

GBAVE

AECHH

RAVE

COM2

PID

VER

COM3

COM4

COM5

COM6

AECH

CLKRC

COM7

COM8

COM9

COM10

HSTART

HSTOP

VSTART

VSTOP
PSHFT
MIDH
MIDL
MVFP
AEW
AEB
VPT
BBIAS
GBBIAS
EXHCH
EXHCL
RBIAS
ADVFL
ADVFH
YAVE
HSYST
HSYEN
HREF
CHLF
ARBLM
ADC_CONTROL
ACOM
OFON
TSLB
COM11
COM12
COM13
COM14
EDGE
COM15
COM16
COM17
AWBC1
AWBC2
AWBC3
AWBC4
AWBC5
AWBC6
REG4B
DNSTH
DM_POS
MTX1
MTX2
MTX3

GAM7
GAM8
GAM9
GAM10
GAM11
GAM12
GAM13
GAM14
GAM15
DM_LNL
DM_LNH
LCC6
LCC7
BD50ST
BD60ST
HRL
LRL
DSPC3
SCALING_PCLK_DELAY
NT_CTRL
AECGMAX
LPH
UPL
TPL
TPH
NALG
STR_OPT
STR_R
STR_G
STR_B
ABLC1
THL_ST
THL_DLT
AD_CHB
AD_CHR
AD_CHGB

Definition at line 234 of file CameraAL422B.h.

4.2.2.6 enum CameraAL422B::RGBOutput

Enumerator

RGB_NORMAL
RGB_565
RGB_555

Definition at line 727 of file CameraAL422B.h.

4.2.3 Constructor & Destructor Documentation

4.2.3.1 CameraAL422B::CameraAL422B (unsigned char(∗)() *read,* unsigned char *vsyncPin,* unsigned char *writeEnPin,* unsigned char *readClockPin,* unsigned char *readResetPin*)

Public constructor.

Parameters

read	The reader function.
vsyncPin	The vertical sync pin number.
hsyncPin	The horizontal sync pin number.
pclkPin	The clock pin number.

Definition at line 16 of file CameraAL422B.cpp.

4.2.4 Member Function Documentation

4.2.4.1 void CameraAL422B::begin ( )

Initializes the camera.

Definition at line 29 of file CameraAL422B.cpp.

4.2.4.2 bool CameraAL422B::capture ( ) [virtual]

Captures a frame.

Implements Camera.

Definition at line 40 of file CameraAL422B.cpp.

4.2.4.3 void CameraAL422B::configureRegisterBits (Register *reg*, Mask *mask*, unsigned char *v*)

Configures a registers inside the camera.

Parameters

reg	The register number.
mask	The mask to be used.
v	The value to be used.

Definition at line 138 of file CameraAL422B.cpp.

4.2.4.4 void CameraAL422B::disableWrite ( ) [inline]

Disables write to the FIFO.

Definition at line 128 of file CameraAL422B.cpp.

4.2.4.5 void CameraAL422B::enableWrite ( ) [inline]

Enables write to the FIFO.

Definition at line 124 of file CameraAL422B.cpp.

4.2.4.6 int CameraAL422B::readFrame (OutputStream * *out*)

Returns a frame.

Returns

A frame.

Definition at line 49 of file CameraAL422B.cpp.

4.2.4.7 unsigned char CameraAL422B::readRegister (Register *reg*)

Reades a value from a register.

Parameters

reg	The register number.

Returns

The register value.

Definition at line 154 of file CameraAL422B.cpp.

4.2.4.8 int CameraAL422B::readRow (OutputStream * *out*) [private]

Reads a row.

Parameters

out	The output stream to be read into.

Definition at line 58 of file CameraAL422B.cpp.

4.2.4.9 void CameraAL422B::resetReadPointer ( ) [inline]

Resets the FIFO internal read pointer.

Definition at line 132 of file CameraAL422B.cpp.

4.2.4.10 void CameraAL422B::resetRegisters ( ) [inline], [private]

Resets all register to default value.

Definition at line 90 of file CameraAL422B.cpp.

4.2.4.11 void CameraAL422B::setColorGainControlEnable (bool *enable*) [inline]

Enable/Disable color gain.

Parameters

enable	Enable or disable.

Definition at line 85 of file CameraAL422B.cpp.

4.2.4.12 void CameraAL422B::setFlashlightModeSelect (FlashlightModeSelect *mode*) [inline]

Select the flashlight mode.

Parameters

mode	The FlashlightModeSelect to be used.

Definition at line 76 of file CameraAL422B.cpp.

4.2.4.13 void CameraAL422B::setHorizontalMirror (bool *mirror*) [inline]

En/disable horizontal mirror.

Parameters

mirror	The mirror option.

Definition at line 68 of file CameraAL422B.cpp.

4.2.4.14 void CameraAL422B::setOutputFormat (OutputFormat *format*)

Sets the output format.

Parameters

format	The output format.

Definition at line 94 of file CameraAL422B.cpp.

4.2.4.15 void CameraAL422B::setOutputResolution (OutputResolution *resolution*)

Sets predefined output resolution.

Parameters

resolution	The output resolution.

Definition at line 98 of file CameraAL422B.cpp.

4.2.4.16 void CameraAL422B::setRGBOutput (RGBOutput *output*)

Sets the RGB output.

Parameters

output	The RGB output.

Definition at line 120 of file CameraAL422B.cpp.

4.2.4.17 void CameraAL422B::setStrobeRequest (bool *request*) [inline]

Exit/Enter strobe mode.

Parameters

request	Enter or exit.

Definition at line 80 of file CameraAL422B.cpp.

4.2.4.18 void CameraAL422B::setVerticalFlip (bool *flip*) [inline]

En/disable vertical flip.

Parameters

mirror	The vertical flip.

Definition at line 72 of file CameraAL422B.cpp.

4.2.4.19 void CameraAL422B::writeRegister (Register *reg,* unsigned char *v*)

Writes a value to a register.

Parameters

reg	The register number.
v	The value to be used.

Definition at line 147 of file CameraAL422B.cpp.

4.2.5 Member Data Documentation

4.2.5.1 unsigned char CameraAL422B::address [private]

Definition at line 54 of file CameraAL422B.h.

4.2.5.2 int CameraAL422B::height [private]

Height in pixels.

Definition at line 64 of file CameraAL422B.h.

4.2.5.3 unsigned char(* CameraAL422B::read) () [private]

Definition at line 44 of file CameraAL422B.h.

4.2.5.4 unsigned char CameraAL422B::readClockPin [private]

Definition at line 50 of file CameraAL422B.h.

4.2.5.5 unsigned char CameraAL422B::readResetPin [private]

Definition at line 52 of file CameraAL422B.h.

4.2.5.6 unsigned char CameraAL422B::vsyncPin [private]

Definition at line 46 of file CameraAL422B.h.

4.2.5.7 int CameraAL422B::width [private]

Width in pixels.

Definition at line 59 of file CameraAL422B.h.

4.2.5.8 unsigned char CameraAL422B::writeEnPin [private]

Definition at line 48 of file CameraAL422B.h.

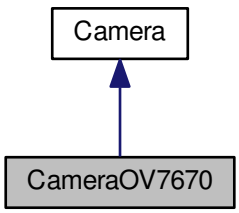
The documentation for this class was generated from the following files:

- CameraAL422B.h
- CameraAL422B.cpp

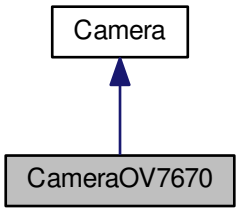
4.3 CameraOV7670 Class Reference

`#include <CameraOV7670.h>`

Inheritance diagram for CameraOV7670:



Collaboration diagram for CameraOV7670:



Public Types

- enum Register {
  REG_GAIN = 0x00, REG_BLUE = 0x01, REG_RED = 0x02, REG_VREF = 0x03,
  REG_COM1 = 0x04, REG_BAVE = 0x05, REG_GBAVE = 0x06, REG_AECHH = 0x07,
  REG_RAVE = 0x08, REG_COM2 = 0x09, REG_PID = 0x0a, REG_VER = 0x0b,
  REG_COM3 = 0x0c, REG_COM4 = 0x0d, REG_COM5 = 0x0e, REG_COM6 = 0x0f,
  REG_AECH = 0x10, REG_CLKRC = 0x11, REG_COM7 = 0x12, REG_COM8 = 0x13,
  REG_COM9 = 0x14, REG_COM10 = 0x15, REG_HSTART = 0x17, REG_HSTOP = 0x18,
  REG_VSTART = 0x19, REG_VSTOP = 0x1a, REG_PSHFT = 0x1b, REG_MIDH = 0x1c,
  REG_MIDL = 0x1d, REG_MVFP = 0x1e, REG_AEW = 0x24, REG_AEB = 0x25,
  REG_VPT = 0x26, REG_HSYST = 0x30, REG_HSYEN = 0x31, REG_HREF = 0x32,
  REG_TSLB = 0x3a, REG_COM11 = 0x3b, REG_COM12 = 0x3c, REG_COM13 = 0x3d,
  REG_COM14 = 0x3e, REG_EDGE = 0x3f, REG_COM15 = 0x40, REG_COM16 = 0x41,
  REG_COM17 = 0x42, REG_CMATRIX_BASE = 0x4f, REG_CMATRIX_SIGN = 0x58, REG_BRIGHT = 0x55,
  REG_CONTRAS = 0x56, REG_GFIX = 0x69, REG_R76 = 0x76, REG_RGB444 = 0x8c,
  REG_HAECC1 = 0x9f, REG_HAECC2 = 0xa0, REG_BD50MAX = 0xa5, REG_HAECC3 = 0xa6,
  REG_HAECC4 = 0xa7, REG_HAECC5 = 0xa8, REG_HAECC6 = 0xa9, REG_HAECC7 = 0xaa,
  REG_BD60MAX = 0xab }

Public Member Functions

- CameraOV7670 (unsigned char(∗read)(), unsigned char vsyncPin, unsigned char hsyncPin)
- void clearBuffers ()
- int readFrame (OutputStream ∗out)

Private Attributes

- unsigned char(∗ read)()
- unsigned char vsyncPin
- unsigned char hsyncPin
- unsigned char address

Additional Inherited Members

4.3.1 Detailed Description

Definition at line 183 of file CameraOV7670.h.

4.3.2 Member Enumeration Documentation

4.3.2.1 enum CameraOV7670::Register

Enumerator

REG_GAIN
REG_BLUE
REG_RED
REG_VREF
REG_COM1
REG_BAVE
REG_GBAVE
REG_AECHH
REG_RAVE
REG_COM2
REG_PID
REG_VER
REG_COM3
REG_COM4
REG_COM5
REG_COM6
REG_AECH
REG_CLKRC
REG_COM7
REG_COM8
REG_COM9
REG_COM10
REG_HSTART
REG_HSTOP
REG_VSTART
REG_VSTOP
REG_PSHFT
REG_MIDH
REG_MIDL
REG_MVFP
REG_AEW
REG_AEB
REG_VPT
REG_HSYST
REG_HSYEN
REG_HREF
REG_TSLB
REG_COM11
REG_COM12
REG_COM13
REG_COM14

REG_EDGE
REG_COM15
REG_COM16
REG_COM17
REG_CMATRIX_BASE
REG_CMATRIX_SIGN
REG_BRIGHT
REG_CONTRAS
REG_GFIX
REG_R76
REG_RGB444
REG_HAECC1
REG_HAECC2
REG_BD50MAX
REG_HAECC3
REG_HAECC4
REG_HAECC5
REG_HAECC6
REG_HAECC7
REG_BD60MAX

Definition at line 195 of file CameraOV7670.h.

4.3.3 Constructor & Destructor Documentation

4.3.3.1 CameraOV7670::CameraOV7670 (unsigned char(∗)() *read,* unsigned char *vsyncPin,* unsigned char *hsyncPin*)

Public constructor.

Parameters

read	The reader function.
vsyncPin	The vertical sync pin number.
hsyncPin	The horizontal sync pin number.
pclkPin	The clock pin number.

Definition at line 16 of file CameraOV7670.cpp.

4.3.4 Member Function Documentation

4.3.4.1 void CameraOV7670::clearBuffers ( )

Clears the buffers.

4.3.4.2 int CameraOV7670::readFrame (OutputStream ∗ *out*)

Returns a frame.

Parameters

out	The frame out.

Returns

The frame size.

Definition at line 26 of file CameraOV7670.cpp.

4.3.5 Member Data Documentation

4.3.5.1 unsigned char CameraOV7670::address [private]

Definition at line 192 of file CameraOV7670.h.

4.3.5.2 unsigned char CameraOV7670::hsyncPin [private]

Definition at line 190 of file CameraOV7670.h.

4.3.5.3 unsigned char(* CameraOV7670::read) () [private]

Definition at line 186 of file CameraOV7670.h.

4.3.5.4 unsigned char CameraOV7670::vsyncPin [private]

Definition at line 188 of file CameraOV7670.h.

The documentation for this class was generated from the following files:

- CameraOV7670.h
- CameraOV7670.cpp

4.4 CameraVC0706 Class Reference

`#include <CameraVC0706.h>`

Public Types

- enum DownSize { NO_ZOON = 0x00, HALF_SIZE = 0x01, QUARTER_SIZE = 0x02 }
- enum ControlBy { GPIO = 0x00, UART = 0x01 }
- enum MotionControl { MOTION_CONTROL = 0, ALARM_ATTRIBUTE = 1, ALARM_ENABLING = 2, ALARM_CONTROL = 3 }
- enum ColorControlMode { AUTO_STEP_BLACK_WHITE = 0, MANUAL_STEP_SELECT_COLOR = 1, MANUAL_STEP_SELECT_BLACK_WHITE = 2 }
- enum Command {
  GEN_VERSION = 0x11, SET_SERIAL_NUMBER = 0x21, SET_PORT = 0x24, SYSTEM_RESET = 0x26,
  READ_DATA = 0x30, WRITE_DATA = 0x31, READ_FBUF = 0x32, WRITE_FBUF = 0x33,
  GET_FBUF_LEN = 0x34, SET_FBUF_LEN = 0x35, FBUF_CTRL = 0x36, COMM_MOTION_CTRL = 0x37,
  COMM_MOTION_STATUS = 0x38, COMM_MOTION_DETECTED = 0x39, MIRROR_CTRL = 0x3A, MIRROR_STATUS = 0x3B,
  COLOR_CTRL = 0x3C, COLOR_STATUS = 0x3D, POWER_SAVE_CTRL = 0x3E, POWER_SAVE_STATUS = 0x3F,
  AE_CTRL = 0x40, AE_STATUS = 0x41, MOTION_CTRL = 0x42, MOTION_STATUS = 0x43,
  TV_OUT_CTRL = 0x44, OSD_ADD_CHAR = 0x45, DOWNSIZE_CTRL = 0x54, DOWNSIZE_STATUS = 0x55,
  GET_FLASH_SIZE = 0x60, ERASE_FLASH_SECTOR = 0x61, ERASE_FLASH_ALL = 0x62, READ_LOGO

= 0x70,
SET_BITMAP = 0x71, BATCH_WRITE = 0x80 }

- enum OutputResolution { RES_640X480 = 0x00, RES_320X240 = 0x11, RES_160X120 = 0x22 }
- enum BufferControl { STOP_CURRENT_FRAME = 0x00, STOP_NEXT_FRAME = 0x01, RESUME_FRAME = 0x03, STEP_FRAME = 0x03 }
- enum BaudRate {
B_9600 = 0xaec8, B_19200 = 0x56e4, B_38400 = 0x2af2, B_57600 = 0x1c4c,
B_115200 = 0x0da6 }

Public Member Functions

- CameraVC0706 (HardwareSerial *serial, Stream *debug)
- bool begin (long baud)
- bool close ()
- bool capture ()
- bool resume ()
- bool setDownSize (unsigned char widthDownSize, unsigned char heightDownSize)
- unsigned char getDownSize ()
- unsigned int getFrameLength ()
- unsigned int readFrame (unsigned char *buf, unsigned int frameOffset, unsigned int bufferOffset, unsigned int len)
- bool setHorizontalMirror (unsigned char by, unsigned char mirrorMode)
- unsigned char getHorizontalMirrorStatus ()
- bool setColorControl (unsigned char by, unsigned char colorControlMode)
- unsigned char getColorControlStatus ()
- bool setOutputResolution (unsigned char resolution)
- bool setMotionMonitoring (bool monitor)
- bool getMotionMonitoringStatus ()
- bool setMotionControl (unsigned char motionControl, unsigned char param0, unsigned char param1)
- bool pollMotionMonitoring (unsigned int timeout, void(*callback)(void *))
- bool setOsdCharacters (unsigned char x, unsigned char y, unsigned char *str, unsigned char len)
- bool setCompression (unsigned char compression)
- unsigned char getCompression ()
- float getVersion ()
- bool reset ()
- bool executeBufferControl (unsigned char control)
- bool setTVOutput (unsigned char onOff)
- bool setBoudRate (long baudRate)
- bool executeCommand (unsigned char cmd, unsigned char *args, unsigned char argc, unsigned int responseLength)

Private Member Functions

- void printBuff (unsigned char *buf, unsigned int c)
- unsigned int write (unsigned char *buf, unsigned int size)
- unsigned int read (unsigned char *buf, unsigned int size)
- unsigned int sendCommand (unsigned char cmd, unsigned char *args, unsigned int argc)
- bool verifyResponse (unsigned char cmd)
- unsigned int readResponse (unsigned int length)

Private Attributes

- unsigned char rxBuffer [VC0760_RX_BUFFER_SIZE]
- unsigned int rxBufferPointer
- unsigned char serialNumber
- unsigned int framePointer
- unsigned int baudRate
- HardwareSerial * serial
- Stream * debug

4.4.1 Detailed Description

Definition at line 25 of file CameraVC0706.h.

4.4.2 Member Enumeration Documentation

4.4.2.1 enum CameraVC0706::BaudRate

Enumerator

B_9600

B_19200

B_38400

B_57600

B_115200

Definition at line 204 of file CameraVC0706.h.

4.4.2.2 enum CameraVC0706::BufferControl

Enumerator

STOP_CURRENT_FRAME

STOP_NEXT_FRAME

RESUME_FRAME

STEP_FRAME

Definition at line 189 of file CameraVC0706.h.

4.4.2.3 enum CameraVC0706::ColorControlMode

Enumerator

AUTO_STEP_BLACK_WHITE

MANUAL_STEP_SELECT_COLOR

MANUAL_STEP_SELECT_BLACK_WHITE

Definition at line 68 of file CameraVC0706.h.

4.4.2.4 enum CameraVC0706::Command

Enumerator

GEN_VERSION

SET_SERIAL_NUMBER

SET_PORT

SYSTEM_RESET
READ_DATA
WRITE_DATA
READ_FBUF
WRITE_FBUF
GET_FBUF_LEN
SET_FBUF_LEN
FBUF_CTRL
COMM_MOTION_CTRL
COMM_MOTION_STATUS
COMM_MOTION_DETECTED
MIRROR_CTRL
MIRROR_STATUS
COLOR_CTRL
COLOR_STATUS
POWER_SAVE_CTRL
POWER_SAVE_STATUS
AE_CTRL
AE_STATUS
MOTION_CTRL
MOTION_STATUS
TV_OUT_CTRL
OSD_ADD_CHAR
DOWNSIZE_CTRL
DOWNSIZE_STATUS
GET_FLASH_SIZE
ERASE_FLASH_SECTOR
ERASE_FLASH_ALL
READ_LOGO
SET_BITMAP
BATCH_WRITE

Definition at line 80 of file CameraVC0706.h.

4.4.2.5 enum CameraVC0706::ControlBy

Enumerator

GPIO
UART

Definition at line 49 of file CameraVC0706.h.

4.4.2.6 enum CameraVC0706::DownSize

Enumerator

NO_ZOON
HALF_SIZE
QUARTER_SIZE

Definition at line 45 of file CameraVC0706.h.

4.4.2.7 enum CameraVC0706::MotionControl

Enumerator

MOTION_CONTROL

ALARM_ATTRIBUTE

ALARM_ENABLING

ALARM_CONTROL

Definition at line 53 of file CameraVC0706.h.

4.4.2.8 enum CameraVC0706::OutputResolution

Enumerator

RES_640X480

RES_320X240

RES_160X120

Definition at line 185 of file CameraVC0706.h.

4.4.3 Constructor & Destructor Documentation

4.4.3.1 CameraVC0706::CameraVC0706 (HardwareSerial * *serial,* Stream * *debug*)

Public constructor on debug mode.

Parameters

debug	Stream pointer.

Definition at line 4 of file CameraVC0706.cpp.

4.4.4 Member Function Documentation

4.4.4.1 bool CameraVC0706::begin (long *baud*)

Initializes the camera.

Definition at line 18 of file CameraVC0706.cpp.

4.4.4.2 bool CameraVC0706::capture ( )

Captures a frame.

Definition at line 28 of file CameraVC0706.cpp.

4.4.4.3 bool CameraVC0706::close ( )

Closes the camera.

Definition at line 23 of file CameraVC0706.cpp.

4.4.4.4 bool CameraVC0706::executeBufferControl (unsigned char *control*)

Execute a buffer control issue.

Command function :control frame buffer register Command format :0x56+serial number+0x36+0x01+control flag(1 byte) control flag:

```
0:stop current frame
```

```
1:stop next frame
2:resume frame
3:step frame
```

Return format : OK:0x76+serial number+0x36+0x00+0x00

Parameters

control	The buffer control.

Definition at line 36 of file CameraVC0706.cpp.

4.4.4.5 bool CameraVC0706::executeCommand (unsigned char *cmd*, unsigned char * *args*, unsigned char *argc*, unsigned int *responseLength*)

Runs a command.

Parameters

cmd	The command to be runned.
args	Buffer of the command params.
argc	How many bytes the buffer has (the command args size).
responseLength	The expected response length.

Definition at line 221 of file CameraVC0706.cpp.

4.4.4.6 unsigned char CameraVC0706::getColorControlStatus ( )

Command function : get color control mode and show mode.

Command format :0x56+serial number+0x3D+0x00

```
Control mode:
     0:control color by GPIO.
     1:control color by UART.

Show mode:show current color by UART.
     0:automatically step black-white and color.
     1:manual step color, select color.
     2:manual step color, select black-white.
```

Return format :

0x76+serial number+0x3D+0x00+0x03+control mode(1 byte)+show mode(1 byte)+current color(1 byte)

Returns

status bit0 Control mode (GPIO = 0, UART = 1) bit[1,2] Show mode 00:automatically step black-white and color. 01:manual step color, select color. 10:manual step color, select black-white.

Definition at line 117 of file CameraVC0706.cpp.

4.4.4.7 unsigned char CameraVC0706::getCompression ( )

Get image compression.

Returns

The compression.

Definition at line 340 of file CameraVC0706.cpp.

4.4.4.8 unsigned char CameraVC0706::getDownSize ( )

Command function : get downsize status.

Command format : 0x56+serial number+0x54+0x00 control item:zooming image proportion

```
Bit[1:0]:width zooming proportion
     2b'00:1:1, no zoom
     2b'01:1:2, the proportion is 1/2.
     2b'10:1:4, the proportion is 1/4.
     2b'11:reservation

Bit[3:2]:height zooming proportion
     2b'00:1:1, no zoom
     2b'01:1:2, the proportion is 1/2.
     2b'10:1:4, the proportion is 1/4.
     2b'11:reservation
```

Return format : 0x76+serial number+0x54+0x00+0x01+control item(1 byte)

Definition at line 71 of file CameraVC0706.cpp.

4.4.4.9 unsigned int CameraVC0706::getFrameLength ( )

Gets the frame length.

Command function :get byte-lengths inFBUF Command format :0x56+serial number+0x34+0x01+FBUF type(1 byte)

```
FBUF type:current frame or next frame
        0:current frame
        1:next frame
```

Return format : OK:0x76+serial number+0x34+0x00+0x04+FBUF data-lengths(4 bytes)

Returns

The frame length.

Definition at line 80 of file CameraVC0706.cpp.

4.4.4.10 unsigned char CameraVC0706::getHorizontalMirrorStatus ( )

Command function : get show status of sensor mirror.

Command format :0x56+serial number+0x3B+0x00

Control mode: 0:control mirror by GPIO. 1:control mirror by UART.

Mirror mode:whether show mirror by UART, it is effective only with UART. It needs GPIO value to set with GPIO control. 0:do not show mirror 1:show mirror

Return format : 0x76+serial number+0x3B+0x00+0x02+control mode(1 byte)+Mirror mode(1 byte)

Returns

status bit0 is the control mode, and the bit1 is the mirror mode.

Definition at line 100 of file CameraVC0706.cpp.

4.4.4.11 bool CameraVC0706::getMotionMonitoringStatus ( )

Gets the motion status.

Command function :get motion monitoring status in communication interface. Command format :0x56+serial number+0x38+0x00

Return format :

OK:0x76+serial number+0x38+0x00+0x01+control flag(1 byte)

```
control flag:
        0:stop motion monitoring
        1:start motion monitoring
```

Error:0x76+serial number+0x37+0x03+0x00

Parameters

return	The flag.

Definition at line 137 of file CameraVC0706.cpp.

4.4.4.12 float CameraVC0706::getVersion ( )

Get the camera version.

Command function :Get Firmware version information Command format :0x56+Serial number+0x11+0x00 Return format :0x76+Serial number+0x11+0x00+0x0B+"VC0706 1.00"

Returns

The float version.

Definition at line 309 of file CameraVC0706.cpp.

4.4.4.13 bool CameraVC0706::pollMotionMonitoring (unsigned int *timeout*, void(∗)(void ∗) *callback*)

Polling for motion detection.

Command function : detect motion

Command format : After starting motion monitoring, once system detects motion, it will send the command. Return format :0x76+serial number+0x39+0x00

E.g. 0x76+0x00+0x39+0x00 detect motion

It is an active command that system send to control terminal.

Parameters

timeout	The timeout to wait.
callback	Function pointer.

Definition at line 143 of file CameraVC0706.cpp.

4.4.4.14 void CameraVC0706::printBuff (unsigned char ∗ *buf*, unsigned int *c*) [private]

Utility function.

Parameters

buf	The buffer to be debugged.
c	How many bytes will be printed.

Definition at line 285 of file CameraVC0706.cpp.

4.4.4.15 unsigned int CameraVC0706::read (unsigned char ∗ *buf*, unsigned int *size*) [private]

Reads from UART.

Parameters

buf	Buffer where data will be read to.
size	How many bytes will tried to read.

Definition at line 191 of file CameraVC0706.cpp.

4.4.4.16 unsigned int CameraVC0706::readFrame (unsigned char * *buf*, unsigned int *frameOffset*, unsigned int *bufferOffset*, unsigned int *len*)

Returns a frame.

Command function :read image data from FBUF. Command format :0x56+serial number+0x32+0x0C+FBUF type(1 byte)+control mode(1 byte) +starting address(4 bytes)+data-length(4 bytes)+delay(2 bytes)

```
FBUF type:current frame or next frame
        0:current frame
        1:next frame

Control mode:the mode by which image data transfer
        Bit0:0:data transfer by MCU mode
             1:data transfer by DMA mode
        Bit[2:1]:2'b11
        Bit3: 1'b11
```

Starting address: the address in fbuf to store the image data. Data-length:the byte number ready to read, it must be the multiple of 4. Delay:the delay time between command and data, the unit is 0.01 millisecond. Return format : Ok:if execute right, return 0x76+serial number+0x32+0x00+0x00, the following is image data, at last, return 0x76+serial number+0x32+0x00+0x00 again.

Returns

A frame.

Definition at line 41 of file CameraVC0706.cpp.

4.4.4.17 unsigned int CameraVC0706::readResponse (unsigned int *length*) [private]

Reads data and put into the rxBuffer.

Adjust the current rxBuffer pointer.

Parameters

length	How many data will be read.

Returns

How many data was actually read.

Definition at line 276 of file CameraVC0706.cpp.

4.4.4.18 bool CameraVC0706::reset ( )

Reset the camera.

Definition at line 295 of file CameraVC0706.cpp.

4.4.4.19 bool CameraVC0706::resume ( )

Resumes the camera.

Definition at line 32 of file CameraVC0706.cpp.

4.4.4.20 unsigned int CameraVC0706::sendCommand (unsigned char *cmd*, unsigned char * *args*, unsigned int *argc*) [private]

Receive command format :

Protocol sign(1byte)+Serial number(1byte)+Command(1byte)+Data-lengths(1byte)+Data(0∼16bytes)

Parameters

cmd	The command.
args	The command data array.
argc	The command data length.

Definition at line 237 of file CameraVC0706.cpp.

4.4.4.21 bool CameraVC0706::setBoudRate (long *baudRate*)

Configures the baud rate.

Command function :Set the property of communication interface Command format :0x56+Serial number+0x24+↩ Data-length+interface type1byte)+configuration data

Such as set MCU UART: 0x56+Serial number+0x24+0x03+0x01+S1RELH(1byte)+S1RELL(1byte) interface type:

```
0x01:MCU UART
```

Return format : OK: 0x76+Serial number+0x24+0x00+0x00

Parameters

baudRate	The baud rate.

Definition at line 355 of file CameraVC0706.cpp.

4.4.4.22 bool CameraVC0706::setColorControl (unsigned char *by*, unsigned char *colorControlMode*)

Command function : color control mode and show mode Command format :0x56+serial number+0x3↩ C+0x02+control mode(1 byte)+show mode(1 byte)

```
Control mode:
        0:control color by GPIO.
        1:control color by UART.
```

Show mode:show different color by UART, it is effective only with UART.

```
It needs Mirror value to set with GPIO control.
        0:automatically step black-white and color.
        1:manually step color, select color.
        2:manually step color, select black-white.
```

Return format : OK:0x76+serial number+0x3C+0x00+0x00

Parameters

by	The color control (UART or GPIO).
colorControl↩ *Mode*	The color control mode.

Definition at line 110 of file CameraVC0706.cpp.

4.4.4.23 bool CameraVC0706::setCompression (unsigned char *compression*)

Set image compression.

Parameters

compression	The compression.

Definition at line 335 of file CameraVC0706.cpp.

4.4.4.24 bool CameraVC0706::setDownSize (unsigned char *widthDownSize,* unsigned char *heightDownSize*)

Command function : control downsize attribute.

Command format :0x56+serial number+0x53+0x01+control item(1 byte)control item:zooming image proportion

```
Bit[1:0]:width zooming proportion
     2b'00:1:1, no zoom
     2b'01:1:2, the proportion is 1/2.
     2b'10:1:4, the proportion is 1/4.
     2b'11:reservation

Bit[3:2]:height zooming proportion
     2b'00:1:1, no zoom
     2b'01:1:2, the proportion is 1/2.
     2b'10:1:4, the proportion is 1/4.
     2b'11:reservation
```

Notice:

1. The image width must be the multiple of 16 in FBUF, image height is the multiple of 8, so the configuration information could satisfy the condition.
2. The zooming proportion of image height is not more than the zooming proportion of width. Return format : 0x76+serial number+0x53+0x00+0x00

Parameters

widthDownSize	The width downsize.
heightDownSize	The height downsize.

Definition at line 64 of file CameraVC0706.cpp.

4.4.4.25 bool CameraVC0706::setHorizontalMirror (unsigned char *by,* unsigned char *mirrorMode*)

En/disable horizontal mirror.

Command function : control show status of sensor mirror. Command format :0x56+serial number+0x3↩A+0x02+control mode(1 byte)+Mirror mode(1 byte)

```
  Control mode:
        0:control mirror by GPIO.
        1:control mirror by UART.
```

Mirror mode:whether show mirror by UART, it is effective only with UART.It needs GPIO value to set with GPIO control.

```
        0:do not show mirror
        1:show mirror
*
```

Parameters

by	The mirror control.
mirrorMode	The mirror mode.

Definition at line 93 of file CameraVC0706.cpp.

4.4.4.26 bool CameraVC0706::setMotionControl (unsigned char *motionControl,* unsigned char *param0,* unsigned char *param1*)

Command function : motion control.

Command format :0x56+serial number+0x42+data-lengths+motion attribute+control item

```
motion attribute:

        0:motion control and enabling control
        1:alarm-output attribute
        2:alarm-output enabling control
        3:alarm-output control


control item:
        > motion control and enabling control
        The first byte:
            0:GPIO
            1:UART
        The second byte:
            0:forbid motion monitoring
            1:start motion monitoring
    > alarm-output attribute
        The first byte:
            bit0:alarm type
            0:stop alarming at a certain time.
            1:alarm at all times.
            Bit1:alarm electrical level
            0:it is low level until alarm.
            1:it is high level until alarm.
        The second and third byte mean the alarm time, the lower byte follows the higher byte, the
        > alarm-output enabling control
            The first byte:
            0:forbid alarm-output
            1:enable alarm-output
        > alarm-output control
            The first byte:
            0:stop alarm-output
            1:start alarm-output
```

Return format : OK: 0x76+serial number+0x42+0x00+0x00 Error:0x76+serial number+0x42+0x03+0x00 E.g.

0x56+0x00+0x42+0x03+0x00+0x01+0x01

Enable motion monitoring by MCU UART, and open it.

0x56+0x00+0x42+0x03+0x00+0x01+0x00

Enable motion monitoring by MCU UART, and stop it.

0x56+0x00+0x42+0x03+0x00+0x00+0x00

Enable motion monitoring by GPIO.

0x56+0x00+0x42+0x04+0x01+0x02+0x00+0x64

Set alarm-output attribute.

0x56+0x00+0x42+0x02+0x02+0x01

Enable alarm-output control.

0x56+0x00+0x42+0x02+0x02+0x00

Disallow alarm-output control.

0x56+0x00+0x42+0x02+0x03+0x01

Start alarm-output.

0x56+0x00+0x42+0x02+0x03+0x00

Stop alarm-output.

Definition at line 159 of file CameraVC0706.cpp.

4.4.4.27 bool CameraVC0706::setMotionMonitoring (bool *monitor*)

Sets the motion detection.

Command function :motion detect on or off in communication interface Command format :0x56+serial number+0x37+0x01+control flag(1 byte)

```
control flag:
        0:stop motion monitoring
        1:start motion monitoring
```

Error:0x76+serial number+0x37+0x03+0x00

Parameters

monitor	The flag.

Definition at line 132 of file CameraVC0706.cpp.

4.4.4.28 bool CameraVC0706::setOsdCharacters (unsigned char *x*, unsigned char *y*, unsigned char * *str*, unsigned char *len*)

Command function : add OSD characters to channels(channel 1)

Command format :0x56+serial number+0x45+data-length+character number(1 byte)+starting address(1 byte)+characters(n characters)character number: the number of characters which continuously are written to channels, the most is 14.

```
starting address:the starting place from which characters show. The format is as follows.
        Bit[4-0]:Y-coordinate
        Bit[6-5]:X-coordinate
```

Characters:the characters ready to show. It is VC0706 OSD characters. Return format : OK: 0x76+serial number+0x45+0x00+0x00

Parameters

x	The x position.
y	The y position.
str	The string to be used.
len	How many char to use.

Definition at line 323 of file CameraVC0706.cpp.

4.4.4.29 bool CameraVC0706::setOutputResolution (unsigned char *resolution*)

Sets predefined output resolution.

Parameters

resolution	The output resolution.

Definition at line 127 of file CameraVC0706.cpp.

4.4.4.30 bool CameraVC0706::setTVOutput (unsigned char *onOff*)

Set TV output.

Parameters

onOff	TV output flag.

Definition at line 350 of file CameraVC0706.cpp.

4.4.4.31 bool CameraVC0706::verifyResponse (unsigned char *cmd*) [private]

Protocol sign(1byte)+Serial number(1byte)+Command(1byte)+Status(1byte)+Data-lengths(1byte)+Data(0∼16bytes)

Parameters

cmd	The command to check the response.

Returns

True if there is a correct response, false otherwise.

Definition at line 267 of file CameraVC0706.cpp.

4.4.4.32 unsigned int CameraVC0706::write (unsigned char * *buf*, unsigned int *size*) [private]

Writes to UART.

Parameters

buf	Buffer from data will come from.
size	How many bytes will tried to write.

Definition at line 165 of file CameraVC0706.cpp.

4.4.5 Member Data Documentation

4.4.5.1 unsigned int CameraVC0706::baudRate [private]

Definition at line 35 of file CameraVC0706.h.

4.4.5.2 Stream* CameraVC0706::debug [private]

Definition at line 40 of file CameraVC0706.h.

4.4.5.3 unsigned int CameraVC0706::framePointer [private]

Definition at line 33 of file CameraVC0706.h.

4.4.5.4 unsigned char CameraVC0706::rxBuffer[VC0760_RX_BUFFER_SIZE] [private]

Definition at line 27 of file CameraVC0706.h.

4.4.5.5 unsigned int CameraVC0706::rxBufferPointer [private]

Definition at line 29 of file CameraVC0706.h.

4.4.5.6 HardwareSerial* CameraVC0706::serial [private]

Definition at line 37 of file CameraVC0706.h.

4.4.5.7 unsigned char CameraVC0706::serialNumber [private]

Definition at line 31 of file CameraVC0706.h.

The documentation for this class was generated from the following files:

- CameraVC0706.h
- CameraVC0706.cpp

4.5 DS1307 Class Reference

```
#include <MIN_at_DS1307.h>
```

Public Member Functions

- DS1307 ()
- void Begin ()
- void Begin (uint8_t address)
- void Begin (int address)
- void Reset ()
- uint8_t ReadConfigByte ()
- void WriteConfigByte (uint8_t value)
- bool ReadTime ()
- void WriteTime ()
- void WriteTimeArray (uint8_t Array[ ])

Public Attributes

- uint8_t _rtc_sec
- uint8_t _rtc_min
- uint8_t _rtc_hour
- uint8_t _rtc_wday
- uint8_t _rtc_day
- uint8_t _rtc_mon
- uint8_t _rtc_year

Private Attributes

- int _addr

4.5.1 Detailed Description

Definition at line 38 of file MIN_at_DS1307.h.

4.5.2 Constructor & Destructor Documentation

4.5.2.1 DS1307::DS1307 ( )

Definition at line 29 of file MIN_at_DS1307.cpp.

4.5.3 Member Function Documentation

4.5.3.1 void DS1307::Begin ( )

Definition at line 37 of file MIN_at_DS1307.cpp.

4.5.3.2 void DS1307::Begin (uint8_t *address*)

4.5.3.3 void DS1307::Begin (int *address*)

Definition at line 47 of file MIN_at_DS1307.cpp.

4.5.3.4 uint8_t DS1307::ReadConfigByte ( )

Definition at line 60 of file MIN_at_DS1307.cpp.

4.5.3.5 bool DS1307::ReadTime ( )

Definition at line 72 of file MIN_at_DS1307.cpp.

4.5.3.6 void DS1307::Reset ( )

Definition at line 52 of file MIN_at_DS1307.cpp.

4.5.3.7 void DS1307::WriteConfigByte (uint8_t *value*)

Definition at line 67 of file MIN_at_DS1307.cpp.

4.5.3.8 void DS1307::WriteTime ( )

Definition at line 124 of file MIN_at_DS1307.cpp.

4.5.3.9 void DS1307::WriteTimeArray (uint8_t *Array[ ]*)

Definition at line 131 of file MIN_at_DS1307.cpp.

4.5.4 Member Data Documentation

4.5.4.1 int DS1307::_addr [private]

Definition at line 41 of file MIN_at_DS1307.h.

4.5.4.2 uint8_t DS1307::_rtc_day

Definition at line 49 of file MIN_at_DS1307.h.

4.5.4.3 uint8_t DS1307::_rtc_hour

Definition at line 47 of file MIN_at_DS1307.h.

4.5.4.4 uint8_t DS1307::_rtc_min

Definition at line 46 of file MIN_at_DS1307.h.

4.5.4.5 uint8_t DS1307::_rtc_mon

Definition at line 50 of file MIN_at_DS1307.h.

4.5.4.6 uint8_t DS1307::_rtc_sec

Definition at line 45 of file MIN_at_DS1307.h.

4.5.4.7 uint8_t DS1307::_rtc_wday

Definition at line 48 of file MIN_at_DS1307.h.

4.5.4.8 uint8_t DS1307::_rtc_year

Definition at line 51 of file MIN_at_DS1307.h.

The documentation for this class was generated from the following files:

- MIN_at_DS1307.h
- MIN_at_DS1307.cpp

4.6 CameraAL422B::MVFPbits Union Reference

`#include <CameraAL422B.h>`

Public Attributes

- struct {
  unsigned char:2
  unsigned char BLACK_SUN_EN:1
  unsigned char VFLIP:1
  unsigned char MIRROR:1
  };
- unsigned char value

4.6.1 Detailed Description

Definition at line 68 of file CameraAL422B.h.

4.6.2 Member Data Documentation

4.6.2.1 struct { ... }

4.6.2.2 unsigned char CameraAL422B::MVFPbits::BLACK_SUN_EN

Definition at line 72 of file CameraAL422B.h.

4.6.2.3 unsigned CameraAL422B::MVFPbits::char

Definition at line 71 of file CameraAL422B.h.

4.6.2.4 unsigned char CameraAL422B::MVFPbits::MIRROR

Definition at line 75 of file CameraAL422B.h.

4.6.2.5 unsigned char CameraAL422B::MVFPbits::value

Definition at line 78 of file CameraAL422B.h.

4.6.2.6 unsigned char CameraAL422B::MVFPbits::VFLIP

Definition at line 74 of file CameraAL422B.h.

The documentation for this union was generated from the following file:

- CameraAL422B.h

4.7 ov7670_control Struct Reference

`#include <from_kernel.h>`

Public Attributes

- struct v4l2_queryctrl qc
- int(∗ query)(struct i2c_client ∗c, __s32 ∗value)
- int(∗ tweak)(struct i2c_client ∗c, int value)

4.7.1 Detailed Description

Definition at line 1063 of file from_kernel.h.

4.7.2 Member Data Documentation

4.7.2.1 struct v4l2_queryctrl ov7670_control::qc

Definition at line 1064 of file from_kernel.h.

4.7.2.2 int(∗ ov7670_control::query) (struct i2c_client ∗c, __s32 ∗value)

Definition at line 1065 of file from_kernel.h.

4.7.2.3 int(∗ ov7670_control::tweak) (struct i2c_client ∗c, int value)

Definition at line 1066 of file from_kernel.h.

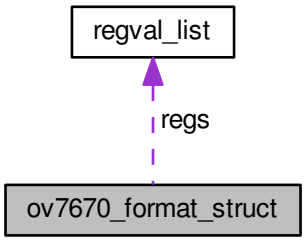
The documentation for this struct was generated from the following file:

- from_kernel.h

4.8 ov7670_format_struct Struct Reference

`#include <from_kernel.h>`

Collaboration diagram for ov7670_format_struct:



Public Attributes

- __u8 * desc
- __u32 pixelformat
- struct regval_list * regs
- int cmatrix [CMATRIX_LEN]

4.8.1 Detailed Description

Definition at line 481 of file from_kernel.h.

4.8.2 Member Data Documentation

4.8.2.1 int ov7670_format_struct::cmatrix[CMATRIX_LEN]

Definition at line 485 of file from_kernel.h.

4.8.2.2 __u8* ov7670_format_struct::desc

Definition at line 482 of file from_kernel.h.

4.8.2.3 __u32 ov7670_format_struct::pixelformat

Definition at line 483 of file from_kernel.h.

4.8.2.4 struct regval_list* ov7670_format_struct::regs

Definition at line 484 of file from_kernel.h.

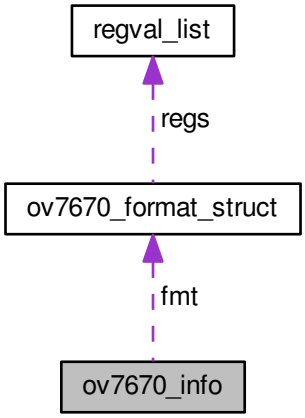
The documentation for this struct was generated from the following file:

- from_kernel.h

4.9 ov7670_info Struct Reference

```
#include <from_kernel.h>
```

Collaboration diagram for ov7670_info:



Public Attributes

- struct ov7670_format_struct * fmt
- unsigned char sat
- int hue

4.9.1 Detailed Description

Definition at line 185 of file from_kernel.h.

4.9.2 Member Data Documentation

4.9.2.1 struct ov7670_format_struct* ov7670_info::fmt

Definition at line 186 of file from_kernel.h.

4.9.2.2 int ov7670_info::hue

Definition at line 188 of file from_kernel.h.

4.9.2.3 unsigned char ov7670_info::sat

Definition at line 187 of file from_kernel.h.

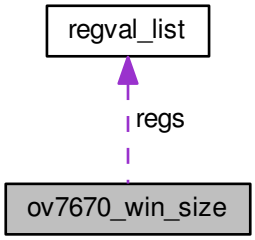
The documentation for this struct was generated from the following file:

- from_kernel.h

4.10 ov7670_win_size Struct Reference

`#include <from_kernel.h>`

Collaboration diagram for ov7670_win_size:



Public Attributes

- int width
- int height
- unsigned char com7_bit
- int hstart
- int hstop
- int vstart
- int vstop
- struct regval_list * regs

4.10.1 Detailed Description

Definition at line 542 of file from_kernel.h.

4.10.2 Member Data Documentation

4.10.2.1 unsigned char ov7670_win_size::com7_bit

Definition at line 545 of file from_kernel.h.

4.10.2.2 int ov7670_win_size::height

Definition at line 544 of file from_kernel.h.

4.10.2.3 int ov7670_win_size::hstart

Definition at line 546 of file from_kernel.h.

4.10.2.4 int ov7670_win_size::hstop

Definition at line 547 of file from_kernel.h.

4.10.2.5 struct regval_list* ov7670_win_size::regs

Definition at line 550 of file from_kernel.h.

4.10.2.6 int ov7670_win_size::vstart

Definition at line 548 of file from_kernel.h.

4.10.2.7 int ov7670_win_size::vstop

Definition at line 549 of file from_kernel.h.

4.10.2.8 int ov7670_win_size::width

Definition at line 543 of file from_kernel.h.

The documentation for this struct was generated from the following file:

- from_kernel.h

4.11 regval_list Struct Reference

`#include <from_kernel.h>`

Public Attributes

- unsigned char reg_num
- unsigned char value

4.11.1 Detailed Description

Definition at line 202 of file from_kernel.h.

4.11.2 Member Data Documentation

4.11.2.1 unsigned char regval_list::reg_num

Definition at line 203 of file from_kernel.h.

4.11.2.2 unsigned char regval_list::value

Definition at line 204 of file from_kernel.h.

The documentation for this struct was generated from the following file:

- from_kernel.h

4.12 TC74 Class Reference

`#include <MIN_at_TC74.h>`

Public Member Functions

- TC74 ()
- void Begin ()
- void Begin (uint8_t address)
- void Begin (int address)
- void Standby (bool Value)
- uint8_t ReadConfigByte ()
- void WriteConfigByte (uint8_t value)
- int8_t ReadTemperature ()

Private Attributes

- int _addr

4.12.1 Detailed Description

Definition at line 41 of file MIN_at_TC74.h.

4.12.2 Constructor & Destructor Documentation

4.12.2.1 TC74::TC74 ( )

Definition at line 29 of file MIN_at_TC74.cpp.

4.12.3 Member Function Documentation

4.12.3.1 void TC74::Begin ( )

Definition at line 37 of file MIN_at_TC74.cpp.

4.12.3.2 void TC74::Begin (uint8_t *address*)

4.12.3.3 void TC74::Begin (int *address*)

Definition at line 47 of file MIN_at_TC74.cpp.

4.12.3.4 uint8_t TC74::ReadConfigByte ( )

Definition at line 58 of file MIN_at_TC74.cpp.

4.12.3.5 int8_t TC74::ReadTemperature ( )

Definition at line 70 of file MIN_at_TC74.cpp.

4.12.3.6 void TC74::Standby (bool *Value*)

Definition at line 52 of file MIN_at_TC74.cpp.

4.12.3.7 void TC74::WriteConfigByte (uint8_t *value*)

Definition at line 65 of file MIN_at_TC74.cpp.

4.12.4 Member Data Documentation

4.12.4.1 int TC74::_addr [private]

Definition at line 44 of file MIN_at_TC74.h.

The documentation for this class was generated from the following files:

- MIN_at_TC74.h
- MIN_at_TC74.cpp

4.13 Tools Class Reference

```
#include <MIN_at_Tools.h>
```

Public Member Functions

- Tools ()

Static Public Member Functions

- static int ReadDec (uint8_t MaxLen, uint8_t Flags, bool *Valid)
- static char * FormatHEX (uint8_t Value, uint8_t Prefix)
- static char * FormatHEX16 (int Value, uint8_t Prefix)
- static char * FormatBIN (uint8_t Value)
- static void I2C_Write (uint8_t I2cAddr, uint16_t MemAddr, uint8_t UseLongAddr)
- static void I2C_WriteValue (uint8_t I2cAddr, uint16_t MemAddr, uint8_t UseLongAddr, uint8_t Value, int Delay)
- static bool I2C_ReadByte (uint8_t I2cAddr, uint16_t MemAddr, uint8_t UseLongAddr, uint8_t *Value)
- static bool I2C_ReadByteDefault (uint8_t I2cAddr, uint16_t MemAddr, uint8_t UseLongAddr, uint8_t *Value, uint8_t DefaultValue)
- static void I2C_SetBitAt (uint8_t I2cAddr, uint16_t MemAddr, uint8_t UseLongAddr, uint8_t BitNum, bool Value, int Delay)
- static void I2C_EEWriteBuffer (uint8_t I2cAddr, uint16_t MemAddr, byte *Data, byte Length)
- static void I2C_EEReadBuffer (uint8_t I2cAddr, uint16_t MemAddr, byte *Data, int Length)
- static uint8_t bcdToDec (uint8_t Value)
- static uint8_t dec2bcd (uint8_t num)

4.13.1 Detailed Description

Definition at line 41 of file MIN_at_Tools.h.

4.13.2 Constructor & Destructor Documentation

4.13.2.1 Tools::Tools ( )

Definition at line 25 of file MIN_at_Tools.cpp.

4.13.3 Member Function Documentation

4.13.3.1 uint8_t Tools::bcdToDec (uint8_t *Value*) [static]

Definition at line 232 of file MIN_at_Tools.cpp.

4.13.3.2 uint8_t Tools::dec2bcd (uint8_t *num*) [static]

Definition at line 238 of file MIN_at_Tools.cpp.

4.13.3.3 char * Tools::FormatBIN (uint8_t *Value*) [static]

Definition at line 113 of file MIN_at_Tools.cpp.

4.13.3.4 char * Tools::FormatHEX (uint8_t *Value,* uint8_t *Prefix*) [static]

Definition at line 99 of file MIN_at_Tools.cpp.

4.13.3.5 char * Tools::FormatHEX16 (int *Value,* uint8_t *Prefix*) [static]

Definition at line 106 of file MIN_at_Tools.cpp.

4.13.3.6 void Tools::I2C_EEReadBuffer (uint8_t *I2cAddr,* uint16_t *MemAddr,* byte * *Data,* int *Length*) [static]

Definition at line 218 of file MIN_at_Tools.cpp.

4.13.3.7 void Tools::I2C_EEWriteBuffer (uint8_t *I2cAddr,* uint16_t *MemAddr,* byte * *Data,* byte *Length*) [static]

Definition at line 205 of file MIN_at_Tools.cpp.

4.13.3.8 bool Tools::I2C_ReadByte (uint8_t *I2cAddr,* uint16_t *MemAddr,* uint8_t *UseLongAddr,* uint8_t * *Value*) [static]

Definition at line 156 of file MIN_at_Tools.cpp.

4.13.3.9 bool Tools::I2C_ReadByteDefault (uint8_t *I2cAddr,* uint16_t *MemAddr,* uint8_t *UseLongAddr,* uint8_t * *Value,* uint8_t *DefaultValue*) [static]

Definition at line 171 of file MIN_at_Tools.cpp.

4.13.3.10 void Tools::I2C_SetBitAt (uint8_t *I2cAddr,* uint16_t *MemAddr,* uint8_t *UseLongAddr,* uint8_t *BitNum,* bool *Value,* int *Delay*) [static]

Definition at line 183 of file MIN_at_Tools.cpp.

4.13.3.11 void Tools::I2C_Write (uint8_t *I2cAddr,* uint16_t *MemAddr,* uint8_t *UseLongAddr*) [static]

Definition at line 126 of file MIN_at_Tools.cpp.

4.13.3.12 void Tools::I2C_WriteValue (uint8_t *I2cAddr,* uint16_t *MemAddr,* uint8_t *UseLongAddr,* uint8_t *Value,* int *Delay*) [static]

Definition at line 139 of file MIN_at_Tools.cpp.

4.13.3.13 int Tools::ReadDec (uint8_t *MaxLen,* uint8_t *Flags,* bool * *Valid*) [static]

Definition at line 33 of file MIN_at_Tools.cpp.

The documentation for this class was generated from the following files:

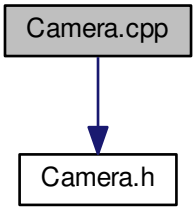
- MIN_at_Tools.h
- MIN_at_Tools.cpp

5 File Documentation

5.1 Camera.cpp File Reference

```
#include "Camera.h"
```

Include dependency graph for Camera.cpp:



Macros

- #define __ARDUINO_DRIVER_CAMERA_CPP__ 1

5.1.1 Macro Definition Documentation

5.1.1.1 #define __ARDUINO_DRIVER_CAMERA_CPP__ 1

Arduino - Camera interface.

Camera.cpp

The abstract class for a Camera.

Author

Dalmir da Silva dalmirdasilva@gmail.com

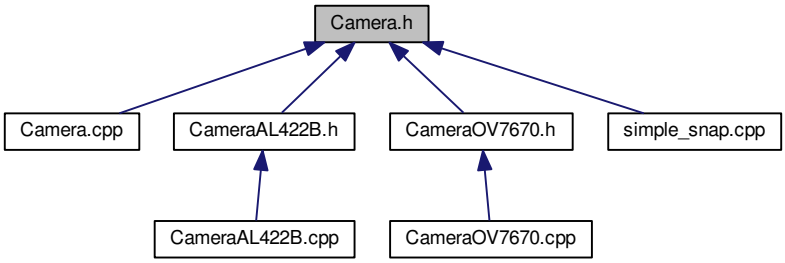
Definition at line 12 of file Camera.cpp.

5.2 Camera.cpp

```
00001
00011 #ifndef __ARDUINO_DRIVER_CAMERA_CPP__
00012 #define __ARDUINO_DRIVER_CAMERA_CPP__ 1
00013
00014 #include "Camera.h"
00015
00016 #endif /* __ARDUINO_DRIVER_CAMERA_CPP__ */
```

5.3 Camera.h File Reference

This graph shows which files directly or indirectly include this file:



Classes

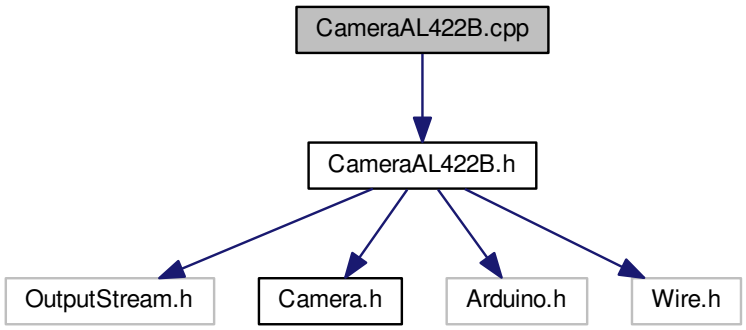
- class Camera

5.4 Camera.h

```
00001
00011 #ifndef __ARDUINO_DRIVER_CAMERA_H__
00012 #define __ARDUINO_DRIVER_CAMERA_H__ 1
00013
00014 class Camera {
00015 public:
00016
00020     virtual bool capture() = 0;
00021 };
00022
00023 #endif /* __ARDUINO_DRIVER_CAMERA_H__ */
```

5.5 CameraAL422B.cpp File Reference

`#include "CameraAL422B.h"`
Include dependency graph for CameraAL422B.cpp:



Macros

- #define __ARDUINO_DRIVER_CAMERA_AL422B_CPP__ 1

5.5.1 Macro Definition Documentation

5.5.1.1 #define __ARDUINO_DRIVER_CAMERA_AL422B_CPP__ 1

Arduino - CameraAL422B implementation.

CameraAL422B.cpp

The class CameraAL422B.

Author

Dalmir da Silva dalmirdasilva@gmail.com

Definition at line 12 of file CameraAL422B.cpp.

5.6 CameraAL422B.cpp

```
00001
00011 #ifndef __ARDUINO_DRIVER_CAMERA_AL422B_CPP__
00012 #define __ARDUINO_DRIVER_CAMERA_AL422B_CPP__ 1
00013
00014 #include "CameraAL422B.h"
00015
00016 CameraAL422B::CameraAL422B(unsigned char (*read)(), unsigned char vsyncPin,
00017         unsigned char writeEnPin, unsigned char readClockPin,
00018         unsigned char readResetPin) :
00019         Camera(), read(read) {
00020     this->vsyncPin = vsyncPin;
00021     this->writeEnPin = writeEnPin;
00022     this->readClockPin = readClockPin;
00023     this->readResetPin = readResetPin;
00024     address = 0x42;
00025     width = 640;
00026     height = 480;
00027 }
00028
00029 void CameraAL422B::begin() {
00030     Wire.begin();
00031     pinMode(vsyncPin, INPUT);
00032     pinMode(writeEnPin, OUTPUT);
00033     pinMode(readClockPin, OUTPUT);
00034     pinMode(readResetPin, OUTPUT);
00035     resetRegisters();
00036     disableWrite();
00037     delayMicroseconds(100);
00038 }
00039
00040 bool CameraAL422B::capture() {
00041     while (digitalReadFast(vsyncPin));
00042     while (!digitalReadFast(vsyncPin));
00043     enableWrite();
00044     while (digitalReadFast(vsyncPin));
00045     disableWrite();
00046     return true;
00047 }
00048
00049 int CameraAL422B::readFrame(OutputStream *out) {
00050     int i, n = 0;
00051     resetReadPointer();
00052     for (i = 0; i < height; i++) {
00053         n += readRow(out);
00054     }
00055     return n;
00056 }
00057
00058 int CameraAL422B::readRow(OutputStream *out) {
00059     int i;
00060     for (i = 0; i < width; i++) {
00061         digitalWriteHighFast(readClockPin);
00062         out->write(read());
00063         digitalWriteLowFast(readClockPin);
00064     }
```

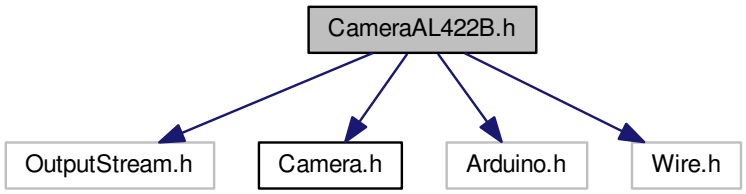
```
00065     return i;
00066 }
00067
00068 void CameraAL422B::setHorizontalMirror(bool mirror) {
00069     configureRegisterBits(MVFP, MVFP_MIRROR, (mirror) ?
      MVFP_MIRROR : 0x00);
00070 }
00071
00072 void CameraAL422B::setVerticalFlip(bool flip) {
00073     configureRegisterBits(MVFP, MVFP_FLIP, (flip) ?
      MVFP_FLIP : 0x00);
00074 }
00075
00076 void CameraAL422B::setFlashlightModeSelect(
      FlashlightModeSelect mode) {
00077     configureRegisterBits(STR_OPT, STR_OPT_MODE, (unsigned char)
      mode);
00078 }
00079
00080 void CameraAL422B::setStrobeRequest(bool request) {
00081     configureRegisterBits(STR_OPT, STR_OPT_REQUEST,
00082             (request) ? STR_OPT_REQUEST : 0x00);
00083 }
00084
00085 void CameraAL422B::setColorGainControlEnable(bool enable) {
00086     configureRegisterBits(STR_OPT, STR_OPT_GAIN,
00087             (enable) ? STR_OPT_GAIN : 0x00);
00088 }
00089
00090 void CameraAL422B::resetRegisters() {
00091     configureRegisterBits(COM7, COM7_RESET, 0xff);
00092 }
00093
00094 void CameraAL422B::setOutputFormat(OutputFormat format) {
00095     configureRegisterBits(COM7, COM7_FORMAT, (unsigned char) format);
00096 }
00097
00098 void CameraAL422B::setOutputResolution(
      OutputResolution resolution) {
00099     configureRegisterBits(COM7, COM7_RESOLUTION, (unsigned char)
      resolution);
00100     switch (resolution) {
00101         case VGA:
00102             width = 640;
00103             height = 480;
00104             break;
00105         case QVGA:
00106             width = 320;
00107             height = 240;
00108             break;
00109         case CIF:
00110             width = 352;
00111             height = 288;
00112             break;
00113         case QCIF:
00114             width = 176;
00115             height = 144;
00116             break;
00117     }
00118 }
00119
00120 void CameraAL422B::setRGBOutput(RGBOutput output) {
00121     configureRegisterBits(COM15, COM15_RGB, (unsigned char) output);
00122 }
00123
00124 void CameraAL422B::enableWrite() {
00125     digitalWriteLowFast(writeEnPin);
00126 }
00127
00128 void CameraAL422B::disableWrite() {
00129     digitalWriteHighFast(writeEnPin);
00130 }
00131
00132 void CameraAL422B::resetReadPointer() {
00133     digitalWriteLowFast(readResetPin);
00134     delayMicroseconds(100);
00135     digitalWriteHighFast(readResetPin);
00136 }
00137
00138 void CameraAL422B::configureRegisterBits(
      Register reg, Mask mask,
00139         unsigned char v) {
00140     unsigned char n;
00141     n = readRegister(reg);
00142     n &= ~((unsigned char) mask);
00143     n |= v & ((unsigned char) mask);
00144     writeRegister(reg, n);
```

```
00145 }
00146
00147 void CameraAL422B::writeRegister(Register reg, unsigned char v) {
00148     Wire.beginTransmission(address);
00149     Wire.write((unsigned char) reg);
00150     Wire.write(v);
00151     Wire.endTransmission();
00152 }
00153
00154 unsigned char CameraAL422B::readRegister(Register reg) {
00155     Wire.beginTransmission(address);
00156     Wire.write((unsigned char) reg);
00157     Wire.endTransmission(false);
00158     Wire.requestFrom(address, (unsigned char) 1);
00159     while (!Wire.available()) {
00160         delay(10);
00161     }
00162     return Wire.read();
00163 }
00164
00165 #endif /* __ARDUINO_DRIVER_CAMERA_AL422B_CPP__ */
```

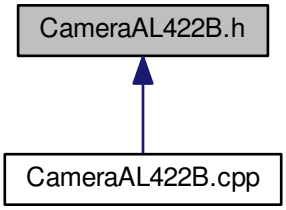
5.7 CameraAL422B.h File Reference

```
#include <OutputStream.h>
#include <Camera.h>
#include <Arduino.h>
#include <Wire.h>
```

Include dependency graph for CameraAL422B.h:



This graph shows which files directly or indirectly include this file:



Classes

- class CameraAL422B
- union CameraAL422B::MVFPbits

Macros

- #define digitalWriteFast(pin, state) ((tate) == LOW) ? digitalWriteLowFast((pin)) : digitalWriteHighFast((pin))
- #define digitalWriteLowFast(pin) ((pin) < 8) ? PORTD &= ~(1 << (pin)) : ((pin) > 13) ? PORTC &= ~(1 << ((pin) - 14)) : PORTB &= ~(1 << ((pin) - 8))
- #define digitalWriteHighFast(pin) ((pin) < 8) ? PORTD |= (1 << (pin)) : ((pin) > 13) ? PORTC |= (1 << ((pin) - 14)) : PORTB |= (1 << ((pin) - 8))
- #define digitalReadFast(pin) ((pin) < 8) ? (PORTD & (1 << (pin))) : ((pin) > 13) ? (PORTC & (1 << ((pin) - 14))) : (PORTB & (1 << ((pin) - 8)))

5.7.1 Macro Definition Documentation

5.7.1.1 #define digitalReadFast(*pin*) ((pin) < 8) ? (PORTD & (1 << (pin))) : ((pin) > 13) ? (PORTC & (1 << ((pin) - 14))) : (PORTB & (1 << ((pin) - 8)))

Definition at line 18 of file CameraAL422B.h.

5.7.1.2 #define digitalWriteFast(*pin, state*) ((tate) == LOW) ? digitalWriteLowFast((pin)) : digitalWriteHighFast((pin))

Arduino - CameraAL422B implementation.

CameraAL422B.h

The class CameraAL422B.

Author

Dalmir da Silva dalmirdasilva@gmail.com

Definition at line 15 of file CameraAL422B.h.

5.7.1.3 #define digitalWriteHighFast(*pin*) ((pin) < 8) ? PORTD |= (1 << (pin)) : ((pin) > 13) ? PORTC |= (1 << ((pin) - 14)) : PORTB |= (1 << ((pin) - 8))

Definition at line 17 of file CameraAL422B.h.

5.7.1.4 #define digitalWriteLowFast(*pin*) ((pin) < 8) ? PORTD &= ~(1 << (pin)) : ((pin) > 13) ? PORTC &= ~(1 << ((pin) - 14)) : PORTB &= ~(1 << ((pin) - 8))

Definition at line 16 of file CameraAL422B.h.

5.8 CameraAL422B.h

```
00001
00011 #ifndef __ARDUINO_DRIVER_CAMERA_AL422B_H__
00012 #define __ARDUINO_DRIVER_CAMERA_AL422B_H__ 1
00013
00014 // Fast IO
00015 #define digitalWriteFast(pin, state)    ((tate) == LOW) ? digitalWriteLowFast((pin)) :
       digitalWriteHighFast((pin))
00016 #define digitalWriteLowFast(pin)        ((pin) < 8) ? PORTD &= ~(1 << (pin)) : ((pin) > 13) ? PORTC &= ~(1
       << ((pin) - 14)) : PORTB &= ~(1 << ((pin) - 8))
00017 #define digitalWriteHighFast(pin)       ((pin) < 8) ? PORTD |= (1 << (pin)) : ((pin) > 13) ? PORTC |= (1 <<
       ((pin) - 14)) : PORTB |= (1 << ((pin) - 8))
00018 #define digitalReadFast(pin)            ((pin) < 8) ? (PORTD & (1 << (pin))) : ((pin) > 13) ? (PORTC & (1
       << ((pin) - 14))) : (PORTB & (1 << ((pin) - 8)))
00019
00020 #include <OutputStream.h>
```

```
00021 #include <Camera.h>
00022 #include <Arduino.h>
00023 #include <Wire.h>
00024
00025 /*
00026 1121 static unsigned char ov7670_sm_to_abs(unsigned char v)
00027 1122 {
00028 1123         if ((v & 0x80) == 0)
00029 1124                 return v + 128;
00030 1125         return 128 - (v & 0x7f);
00031 1126 }
00032 1127
00033 1128
00034 1129 static unsigned char ov7670_abs_to_sm(unsigned char v)
00035 1130 {
00036 1131         if (v > 127)
00037 1132                 return v & 0x7f;
00038 1133         return (128 - v) | 0x80;
00039 1134 }*/
00040
00041 class CameraAL422B : public Camera {
00042 private:
00043
00044     unsigned char (*read)();
00045
00046     unsigned char vsyncPin;
00047
00048     unsigned char writeEnPin;
00049
00050     unsigned char readClockPin;
00051
00052     unsigned char readResetPin;
00053
00054     unsigned char address;
00055
00059     int width;
00060
00064     int height;
00065
00066 public:
00067
00068     union MVFPbits {
00069
00070         struct {
00071             unsigned char :2;
00072             unsigned char BLACK_SUN_EN :1;
00073             unsigned char :1;
00074             unsigned char VFLIP :1;
00075             unsigned char MIRROR :1;
00076             unsigned char :2;
00077         };
00078         unsigned char value;
00079     };
00080
00081     enum Mask {
00082
00083         // Flashlight Mode Select
00084         STR_OPT_MODE = 0x03,
00085
00086         // Strobe Request
00087         STR_OPT_REQUEST = 0x80,
00088
00089         // Color Gain Control Enable
00090         STR_OPT_GAIN = 0x40,
00091
00092         // Horizontal mirror
00093         MVFP_MIRROR = 0x20,
00094
00095         // Vertical flip
00096         MVFP_FLIP = 0x10,
00097
00098         // Reset
00099         COM7_RESET = 0x80,
00100
00101         // Output resolution.
00102         COM7_RESOLUTION = 0x38,
00103
00104         // Output format.
00105         COM7_FORMAT = 0x05,
00106
00107         // Color bar
00108         COM7_COLOR_BAR = 0x02,
00109
00110         // Soft sleep mode
00111         COM2_SSLEEP = 0x10,
00112
00113         // Separator
```

```
00114         _ = 0x00,
00115
00116         // CCIR656 enable
00117         COM1_CCIR656 = 0x40,
00118
00119         // Byte swap
00120         COM3_SWAP = 0x40,
00121
00122         // Enable scaling
00123         COM3_SCALEEN = 0x08,
00124
00125         // Enable downsamp/crop/window
00126         COM3_DCWEN = 0x04,
00127
00128         // Use external clock directly
00129         CLKRC_EXT = 0x40,
00130
00131         // Mask for internal clock scale
00132         CLKRC_SCALE = 0x3f,
00133
00134         // Enable fast AGC/AEC
00135         COM8_FASTAEC = 0x80,
00136
00137         // Unlimited AEC step size
00138         COM8_AECSTEP = 0x40,
00139
00140         // Band filter enable
00141         COM8_BFILT = 0x20,
00142
00143         // Auto gain enable
00144         COM8_AGC = 0x04,
00145
00146         // White balance enable
00147         COM8_AWB = 0x02,
00148
00149         // Auto exposure enable
00150         COM8_AEC = 0x01,
00151
00152         // HSYNC instead of HREF
00153         COM10_HSYNC = 0x40,
00154
00155         // Suppress PCLK on horiz blank
00156         COM10_PCLK_HB = 0x20,
00157
00158         // Reverse HREF
00159         COM10_HREF_REV = 0x08,
00160
00161         // VSYNC on clock leading edge
00162         COM10_VS_LEAD = 0x04,
00163
00164         // VSYNC negative
00165         COM10_VS_NEG = 0x02,
00166
00167         // HSYNC negative
00168         COM10_HS_NEG = 0x01,
00169
00170         // UYVY or VYUY - see com13
00171         TSLB_YLAST = 0x04,
00172
00173         // Night mode enable
00174         COM11_NIGHT = 0x80,
00175
00176         // Two bit NM frame rate
00177         COM11_NMFR = 0x60,
00178
00179         // Auto detect 50/60 Hz
00180         COM11_HZAUTO = 0x10,
00181
00182         // Manual 50Hz select
00183         COM11_50HZ = 0x08,
00184
00185         // Exp
00186         COM11_EXP = 0x02,
00187
00188         // HREF always
00189         COM12_HREF = 0x80,
00190
00191         // Gamma enable
00192         COM13_GAMMA = 0x80,
00193
00194         // UV saturation auto adjustment
00195         COM13_UVSAT = 0x40,
00196
00197         // V before U - w/TSLB
00198         COM13_UVSWAP = 0x01,
00199
00200         // DCW/PCLK-scale enable
```

```
00201         COM14_DCWEN = 0x10,
00202
00203         // Data range 10 to F0
00204         COM15_R10F0 = 0x00,
00205
00206         // Data range 01 to FE
00207         COM15_R01FE = 0x80,
00208
00209         // Data range 00 to FF
00210         COM15_R00FF = 0xc0,
00211
00212         // RGB options
00213         COM15_RGB = 0x30,
00214
00215         // AWB gain enable
00216         COM16_AWBGAIN = 0x08,
00217
00218         // AEC window - must match COM4
00219         COM17_AECWIN = 0xc0,
00220
00221         // DSP Color bar
00222         COM17_CBAR = 0x08,
00223
00224         // White pixel correction enable
00225         R76_WHTPCOR = 0x40,
00226
00227         // Black pixel correction enable
00228         REG76_BLKPCOR = 0x20,
00229
00230         // Edge enhancement higher limit
00231         REG76_EDGE = 0x1f
00232     };
00233
00234     enum Register {
00235
00236         // Gain lower 8 bits (rest in vref)
00237         GAIN = 0x00,
00238
00239         // blue gain
00240         BLUE = 0x01,
00241
00242         // red gain
00243         RED = 0x02,
00244
00245         // Pieces of GAIN, VSTART, VSTOP
00246         VREF = 0x03,
00247
00248         // Control 1
00249         COM1 = 0x04,
00250
00251         // U/B Average level
00252         BAVE = 0x05,
00253
00254         // Y/Gb Average level
00255         GBAVE = 0x06,
00256
00257         // AEC MS 5 bits
00258         AECHH = 0x07,
00259
00260         // V/R Average level
00261         RAVE = 0x08,
00262
00263         // Control 2
00264         COM2 = 0x09,
00265
00266         // Product ID MSB
00267         PID = 0x0a,
00268
00269         // Product ID LSB
00270         VER = 0x0b,
00271
00272         // Control 3
00273         COM3 = 0x0c,
00274
00275         // Control 4
00276         COM4 = 0x0d,
00277
00278         // All "reserved"
00279         COM5 = 0x0e,
00280
00281         // Control 6
00282         COM6 = 0x0f,
00283
00284         // More bits of AEC value
00285         AECH = 0x10,
00286
00287         // Clocl control
```

```
00288         CLKRC = 0x11,
00289
00290         // Control 7
00291         COM7 = 0x12,
00292
00293         // Control 8
00294         COM8 = 0x13,
00295
00296         // Control 9  - gain ceiling
00297         COM9 = 0x14,
00298
00299         // Control 10
00300         COM10 = 0x15,
00301
00302         // Horiz start high bits
00303         HSTART = 0x17,
00304
00305         // Horiz stop high bits
00306         HSTOP = 0x18,
00307
00308         // Vert start high bits
00309         VSTART = 0x19,
00310
00311         // Vert stop high bits
00312         VSTOP = 0x1a,
00313
00314         // Pixel delay after HREF
00315         PSHFT = 0x1b,
00316
00317         // Manuf. ID high
00318         MIDH = 0x1c,
00319
00320         // Manuf. ID low
00321         MIDL = 0x1d,
00322
00323         // Mirror / vflip
00324         MVFP = 0x1e,
00325
00326         // AGC upper limit.
00327         AEW = 0x24,
00328
00329         // AGC lower limit.
00330         AEB = 0x25,
00331
00332         // AGC/AEC fast mode op region.
00333         VPT = 0x26,
00334
00335         // B Channel Signal Output Bias (effective only when COM6[3] = 1).
00336         BBIAS = 0x27,
00337
00338         // Gb Channel Signal Output Bias (effective only when COM6[3] = 1).
00339         GBBIAS = 0x28,
00340
00341         // Dummy Pixel Insert MSB.
00342         EXHCH = 0x2a,
00343
00344         // Dummy Pixel Insert LSB.
00345         EXHCL = 0x2b,
00346
00347         // R Channel Signal Output Bias (effective only when COM6[3] = 1).
00348         RBIAS = 0x2c,
00349
00350         // LSB of insert dummy rows in vertical direction (1 bit equals 1 row).
00351         ADVFL = 0x2d,
00352
00353         // MSB of insert dummy rows in vertical direction.
00354         ADVFH = 0x2e,
00355
00356         // Y/G Channel Average Value.
00357         YAVE = 0x2f,
00358
00359         // HSYNC rising edge delay
00360         HSYST = 0x30,
00361
00362         // HSYNC falling edge delay
00363         HSYEN = 0x31,
00364
00365         // HREF pieces
00366         HREF = 0x32,
00367
00368         // Array Current Control.
00369         CHLF = 0x33,
00370
00371         // Array Reference Control.
00372         ARBLM = 0x34,
00373
00374         // ADC Control.
```

```
00375         ADC_CONTROL = 0x37,
00376
00377         // ADC and Analog Common Mode Control.
00378         ACOM = 0x38,
00379
00380         // ADC Offset Control.
00381         OFON = 0x39,
00382
00383         // Line Buffer Test Option.
00384         TSLB = 0x3a,
00385
00386         // Control 11
00387         COM11 = 0x3b,
00388
00389         // Control 12
00390         COM12 = 0x3c,
00391
00392         // Control 13
00393         COM13 = 0x3d,
00394
00395         // Control 14
00396         COM14 = 0x3e,
00397
00398         // Edge enhancement factor
00399         EDGE = 0x3f,
00400
00401         // Control 15
00402         COM15 = 0x40,
00403
00404         // Control 16
00405         COM16 = 0x41,
00406
00407         // Control 17
00408         COM17 = 0x42,
00409
00410         // WB Control 1.
00411         AWBC1 = 0x43,
00412
00413         // WB Control 2.
00414         AWBC2 = 0x44,
00415
00416         // WB Control 3.
00417         AWBC3 = 0x45,
00418
00419         // WB Control 4.
00420         AWBC4 = 0x46,
00421
00422         // WB Control 5.
00423         AWBC5 = 0x47,
00424
00425         // WB Control 6.
00426         AWBC6 = 0x48,
00427
00428         // UV average.
00429         REG4B = 0x4b,
00430
00431         // De-noise Strength.
00432         DNSTH = 0x4c,
00433
00434         // Dummy row position.
00435         DM_POS = 0x4d,
00436
00437         // Matrix Coefficient 1.
00438         MTX1 = 0x4f,
00439
00440         // Matrix Coefficient 2.
00441         MTX2 = 0x50,
00442
00443         // Matrix Coefficient 3.
00444         MTX3 = 0x51,
00445
00446         // Matrix Coefficient 4.
00447         MTX4 = 0x52,
00448
00449         // Matrix Coefficient 5.
00450         MTX5 = 0x53,
00451
00452         // Matrix Coefficient 6.
00453         MTX6 = 0x54,
00454
00455         // Brightness Control.
00456         BRIGHT = 0x55,
00457
00458         // Contrast Control.
00459         CONTRAS = 0x56,
00460
00461         // Contrast Center.
```

```
00462         CONTRAS_CENTER = 0x57,
00463
00464         // Matrix Coefficient Sign for coefficient 5 to 0.
00465         MTXS = 0x58,
00466
00467         // AWB Control 7.
00468         AWBC7 = 0x59,
00469
00470         // AWB Control 8.
00471         AWBC8 = 0x5a,
00472
00473         // AWB Control 9.
00474         AWBC9 = 0x5b,
00475
00476         // AWB Control 10.
00477         AWBC10 = 0x5c,
00478
00479         // AWB Control 11.
00480         AWBC11 = 0x5d,
00481
00482         // AWB Control 12.
00483         AWBC12 = 0x5e,
00484
00485         // AWB B Gain Range.
00486         B_LMT = 0x5f,
00487
00488         // AWB R Gain Range.
00489         R_LMT = 0x60,
00490
00491         // AWB G Gain Range.
00492         G_LMT = 0x61,
00493
00494         // Lens Correction Option 1 - X Coordinate of Lens Correction Center Relative to Array Center.
00495         LCC1 = 0x62,
00496
00497         // Lens Correction Option 2 - Y Coordinate of Lens Correction Center Relative to Array Center.
00498         LCC2 = 0x63,
00499
00500         // Lens Correction Option 3
00501         LCC3 = 0x64,
00502
00503         // Lens Correction Option 4 - Radius of the circular section where no compensation applies.
00504         LCC4 = 0x65,
00505
00506         // Lens Correction Control.
00507         LCC5 = 0x66,
00508
00509         // Manual U Value (effective only when register TSLB[4] is high).
00510         MANU = 0x67,
00511
00512         // Manual V Value (effective only when register TSLB[4] is high).
00513         MANV = 0x68,
00514
00515         // Fix gain control
00516         GFIX = 0x69,
00517
00518         // G Channel AWB Gain.
00519         GGAIN = 0x6a,
00520
00521         // PLL Control.
00522         DBLV = 0x6b,
00523
00524         // AWB Control 3.
00525         AWBCTR3 = 0x6c,
00526
00527         // AWB Control 2.
00528         AWBCTR2 = 0x6d,
00529
00530         // AWB Control 1.
00531         AWBCTR1 = 0x6e,
00532
00533         // AWB Control 0.
00534         AWBCTR0 = 0x6f,
00535
00536         // Test_pattern[0] - works with test_pattern[1] test_pattern.
00537         SCALING_XSC = 0x70,
00538
00539         // Test_pattern[1] - works with test_pattern[0] test_pattern (SCALING_XSC[7], SCALING_YSC[7]).
00540         SCALING_YSC = 0x71,
00541
00542         // DCW Control
00543         SCALING_DCWCTR = 0x72,
00544
00545         // Clock.
00546         SCALING_PCLK_DIV = 0x73,
00547
00548         // Gain control.
```

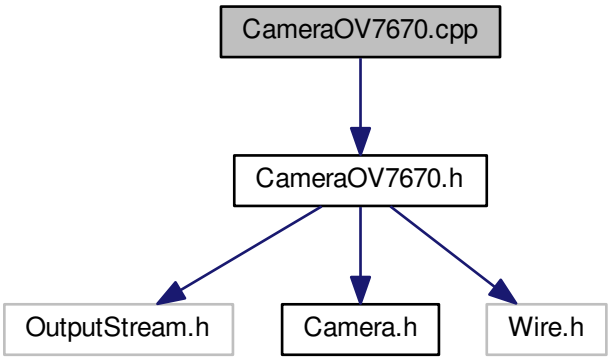
```
00549         REG74 = 0x74,
00550
00551         // Edge enhancement.
00552         REG75 = 0x75,
00553
00554         // Pixel correction.
00555         REG76 = 0x76,
00556
00557         // Offset, de-noise range control.
00558         REG77 = 0x77,
00559
00560         // Gamma Curve Highest Segment Slop.
00561         SLOP = 0x7a,
00562
00563         // Gamma Curve 1st Segment Input End Point 0x04 Output Value.
00564         GAM1 = 0x7b,
00565
00566         // Gamma Curve 2nd Segment Input End Point 0x08 Output Value.
00567         GAM2 = 0x7c,
00568
00569         // Gamma Curve 3rd Segment Input End Point 0x10 Output Value.
00570         GAM3 = 0x7d,
00571
00572         // Gamma Curve 4th Segment Input End Point 0x20 Output Value.
00573         GAM4 = 0x7e,
00574
00575         // Gamma Curve 5th Segment Input End Point 0x28 Output Value.
00576         GAM5 = 0x7f,
00577
00578         // Gamma Curve 6th Segment Input End Point 0x30 Output Value.
00579         GAM6 = 0x80,
00580
00581         // Gamma Curve 7th Segment Input End Point 0x38 Output Value.
00582         GAM7 = 0x81,
00583
00584         // Gamma Curve 8th Segment Input End Point 0x40 Output Value.
00585         GAM8 = 0x82,
00586
00587         // Gamma Curve 9th Segment Input End Point 0x48 Output Value.
00588         GAM9 = 0x83,
00589
00590         // Gamma Curve 10th Segment Input End Point 0x50 Output Value.
00591         GAM10 = 0x84,
00592
00593         // Gamma Curve 11th Segment Input End Point 0x60 Output Value.
00594         GAM11 = 0x85,
00595
00596         // Gamma Curve 12th Segment Input End Point 0x70 Output Value.
00597         GAM12 = 0x86,
00598
00599         // Gamma Curve 13th Segment Input End Point 0x90 Output Value.
00600         GAM13 = 0x87,
00601
00602         // Gamma Curve 14th Segment Input End Point 0xB0 Output Value.
00603         GAM14 = 0x88,
00604
00605         // Gamma Curve 15th Segment Input End Point 0xD0 Output Value.
00606         GAM15 = 0x89,
00607
00608         // RGB 444 control
00609 //        RGB444 = 0x8c,
00610
00611 // Dummy Row low 8 bits.
00612         DM_LNL = 0x92,
00613
00614         // Dummy Row high 8 bits.
00615         DM_LNH = 0x93,
00616
00617         // Lens Correction Option 6 (effective only when LCC5[2] is high).
00618         LCC6 = 0x94,
00619
00620         // Lens Correction Option 7 (effective only when LCC5[2] is high).
00621         LCC7 = 0x95,
00622
00623         // 50 Hz Banding Filter Value (effective only when COM8[5] is high and COM11[3] is high).
00624         BD50ST = 0x9d,
00625
00626         // 60 Hz Banding Filter Value (effective only when COM8[5] is high and COM11[3] is low).
00627         BD60ST = 0x9e,
00628
00629         // High Reference Luminance.
00630         HRL = 0x9f,
00631
00632         // Low Reference Luminance.
00633         LRL = 0xa0,
00634
00635         // DSP Control 3.
```

```
00636         DSPC3 = 0xa1,
00637
00638         // DSP Control 3.
00639         SCALING_PCLK_DELAY = 0xa2,
00640
00641         // Frame rate adjustment.
00642         NT_CTRL = 0xa4,
00643
00644         // Maximum Banding Filter Step.
00645         AECGMAX = 0xa5,
00646
00647         // Lower Limit of Probability for HRL, after exposure/gain stabilizes.
00648         LPH = 0xa6,
00649
00650         // Upper Limit of Probability for LRL, after exposure/gain stabilizes.
00651         UPL = 0xa7,
00652
00653         // Probability Threshold for LRL to control AEC/AGC speed.
00654         TPL = 0xa8,
00655
00656         // Probability Threshold for HRL to control AEC/AGC speed.
00657         TPH = 0xa9,
00658
00659         // AEC algorithm selection.
00660         NALG = 0xaa,
00661
00662         // Strobe
00663         STR_OPT = 0xac,
00664
00665         // Red gain for strobe.
00666         STR_R = 0xad,
00667
00668         // Green gain for strobe.
00669         STR_G = 0xae,
00670
00671         // Blue gain for strobe.
00672         STR_B = 0xaf,
00673
00674         // ABLC function.
00675         ABLC1 = 0xb1,
00676
00677         // ABLC Target.
00678         THL_ST = 0xb3,
00679
00680         // ABLC Stable Range.
00681         THL_DLT = 0xb5,
00682
00683         // Blue Channel Black Level Compensation.
00684         AD_CHB = 0xbe,
00685
00686         // Red Channel Black Level Compensation.
00687         AD_CHR = 0xbf,
00688
00689         // Gb Channel Black Level Compensation.
00690         AD_CHGB = 0xc0
00691     };
00692
00693     enum FlashlightModeSelect {
00694         XENON = 0x00, LED1 = 0x01, LED2 = 0x02
00695     };
00696
00697     enum OutputFormat {
00698
00699         // YUV format
00700         YUV = 0x00,
00701
00702         // RGB format
00703         RGB = 0x04,
00704
00705         // Raw bayer RGB format
00706         RAW_BAYER_RGB = 0x01,
00707
00708         // Processed bayer RGB format
00709         PROCESSED_BAYER_RGB = 0x05
00710     };
00711
00712     enum OutputResolution {
00713
00714         // VGA format.
00715         VGA = 0x00,
00716
00717         // CIF format
00718         CIF = 0x20,
00719
00720         // QVGA format
00721         QVGA = 0x10,
00722
```

```
00723         // QCIF format
00724         QCIF = 0x08
00725     };
00726
00727     enum RGBOutput {
00728
00729         // Normal RGB
00730         RGB_NORMAL = 0x00,
00731
00732         // RGB 565
00733         RGB_565 = 0x10,
00734
00735         // RGB 555
00736         RGB_555 = 0x30
00737     };
00738
00748     CameraAL422B(unsigned char (*read)(), unsigned char vsyncPin,
00749             unsigned char writeEnPin, unsigned char readClockPin,
00750             unsigned char readResetPin);
00751
00755     void begin();
00756
00760     virtual bool capture();
00761
00767     int readFrame(OutputStream *out);
00768
00774     void inline setHorizontalMirror(bool mirror);
00775
00781     void inline setVerticalFlip(bool flip);
00782
00788     void inline setFlashlightModeSelect(
    FlashlightModeSelect mode);
00789
00795     void inline setStrobeRequest(bool request);
00796
00802     void inline setColorGainControlEnable(bool enable);
00803
00809     void setOutputFormat(OutputFormat format);
00810
00816     void setOutputResolution(OutputResolution resolution);
00817
00823     void setRGBOutput(RGBOutput output);
00824
00828     void inline enableWrite();
00829
00833     void inline disableWrite();
00834
00838     void inline resetReadPointer();
00839
00847     void configureRegisterBits(Register reg, Mask mask, unsigned char v);
00848
00855     void writeRegister(Register reg, unsigned char v);
00856
00863     unsigned char readRegister(Register reg);
00864
00865 private:
00866
00870     void inline resetRegisters();
00871
00877     int readRow(OutputStream *out);
00878 };
00879
00880 #endif /* __ARDUINO_DRIVER_CAMERA_AL422B_H__ */
```

5.9 CameraOV7670.cpp File Reference

`#include "CameraOV7670.h"`
Include dependency graph for CameraOV7670.cpp:



Macros

- #define __ARDUINO_DRIVER_CAMERA_OV7670_CPP__ 1

5.9.1 Macro Definition Documentation

5.9.1.1 #define __ARDUINO_DRIVER_CAMERA_OV7670_CPP__ 1

Arduino - CameraOV7670 implementation.

CameraOV7670.cpp

The class CameraOV7670.

Author

Dalmir da Silva dalmirdasilva@gmail.com

Definition at line 12 of file CameraOV7670.cpp.

5.10 CameraOV7670.cpp

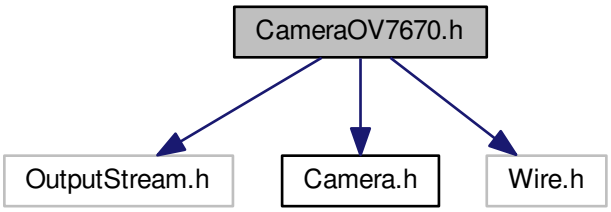
```
00001
00011 #ifndef __ARDUINO_DRIVER_CAMERA_OV7670_CPP__
00012 #define __ARDUINO_DRIVER_CAMERA_OV7670_CPP__ 1
00013
00014 #include "CameraOV7670.h"
00015
00016 CameraOV7670::CameraOV7670(unsigned char (*read)(), unsigned char vsyncPin,
00017         unsigned char hsyncPin) :
00018         Camera() {
00019     this->read = read;
00020     this->vsyncPin = vsyncPin;
00021     this->hsyncPin = hsyncPin;
00022     address = 0x42;
00023     Wire.begin();
00024 }
00025
```

```
00026 int CameraOV7670::readFrame(OutputStream *out) {
00027     return 0;
00028 }
00029
00030 #endif /* __ARDUINO_DRIVER_CAMERA_OV7670_CPP__ */
```

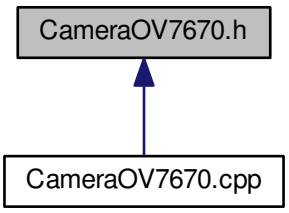
5.11 CameraOV7670.h File Reference

```
#include <OutputStream.h>
#include <Camera.h>
#include <Wire.h>
```

Include dependency graph for CameraOV7670.h:



This graph shows which files directly or indirectly include this file:



Classes

- class CameraOV7670

Macros

- #define OV7670_COM1_CCIR656 0x40
- #define OV7670_COM2_SSLEEP 0x10
- #define OV7670_COM3_SWAP 0x40
- #define OV7670_COM3_SCALEEN 0x08
- #define OV7670_COM3_DCWEN 0x04
- #define OV7670_CLKRC_EXT 0x40

- #define OV7670_CLKRC_SCALE 0x3f
- #define OV7670_COM7_RESET 0x80
- #define OV7670_COM7_FMT_MASK 0x38
- #define OV7670_COM7_FMT_VGA 0x00
- #define OV7670_COM7_FMT_CIF 0x20
- #define OV7670_COM7_FMT_QVGA 0x10
- #define OV7670_COM7_FMT_QCIF 0x08
- #define OV7670_COM7_RGB 0x04
- #define OV7670_COM7_YUV 0x00
- #define OV7670_COM7_BAYER 0x01
- #define OV7670_COM7_PBAYER 0x05
- #define OV7670_COM8_FASTAEC 0x80
- #define OV7670_COM8_AECSTEP 0x40
- #define OV7670_COM8_BFILT 0x20
- #define OV7670_COM8_AGC 0x04
- #define OV7670_COM8_AWB 0x02
- #define OV7670_COM8_AEC 0x01
- #define OV7670_COM10_HSYNC 0x40
- #define OV7670_COM10_PCLK_HB 0x20
- #define OV7670_COM10_HREF_REV 0x08
- #define OV7670_COM10_VS_LEAD 0x04
- #define OV7670_COM10_VS_NEG 0x02
- #define OV7670_COM10_HS_NEG 0x01
- #define OV7670_MVFP_MIRROR 0x20
- #define OV7670_MVFP_FLIP 0x10
- #define OV7670_TSLB_YLAST 0x04
- #define OV7670_COM11_NIGHT 0x80
- #define OV7670_COM11_NMFR 0x60
- #define OV7670_COM11_HZAUTO 0x10
- #define OV7670_COM11_50HZ 0x08
- #define OV7670_COM11_EXP 0x02
- #define OV7670_COM12_HREF 0x80
- #define OV7670_COM13_GAMMA 0x80
- #define OV7670_COM13_UVSAT 0x40
- #define OV7670_COM13_UVSWAP 0x01
- #define OV7670_COM14_DCWEN 0x10
- #define OV7670_COM15_R10F0 0x00
- #define OV7670_COM15_R01FE 0x80
- #define OV7670_COM15_R00FF 0xc0
- #define OV7670_COM15_RGB565 0x10
- #define OV7670_COM15_RGB555 0x30
- #define OV7670_COM16_AWBGAIN 0x08
- #define OV7670_COM17_AECWIN 0xc0
- #define OV7670_COM17_CBAR 0x08
- #define OV7670_CMATRIX_LEN 0x06
- #define OV7670_R76_BLKPCOR 0x80
- #define OV7670_R76_WHTPCOR 0x40
- #define OV7670_RGB444_ENABLE 0x02
- #define OV7670_RGB444_RGBX 0x01

5.11.1 Macro Definition Documentation

5.11.1.1 #define OV7670_CLKRC_EXT 0x40

Definition at line 30 of file CameraOV7670.h.

5.11.1.2 #define OV7670_CLKRC_SCALE 0x3f

Definition at line 33 of file CameraOV7670.h.

5.11.1.3 #define OV7670_CMATRIX_LEN 0x06

Definition at line 165 of file CameraOV7670.h.

5.11.1.4 #define OV7670_COM10_HREF_REV 0x08

Definition at line 90 of file CameraOV7670.h.

5.11.1.5 #define OV7670_COM10_HS_NEG 0x01

Definition at line 99 of file CameraOV7670.h.

5.11.1.6 #define OV7670_COM10_HSYNC 0x40

Definition at line 84 of file CameraOV7670.h.

5.11.1.7 #define OV7670_COM10_PCLK_HB 0x20

Definition at line 87 of file CameraOV7670.h.

5.11.1.8 #define OV7670_COM10_VS_LEAD 0x04

Definition at line 93 of file CameraOV7670.h.

5.11.1.9 #define OV7670_COM10_VS_NEG 0x02

Definition at line 96 of file CameraOV7670.h.

5.11.1.10 #define OV7670_COM11_50HZ 0x08

Definition at line 120 of file CameraOV7670.h.

5.11.1.11 #define OV7670_COM11_EXP 0x02

Definition at line 123 of file CameraOV7670.h.

5.11.1.12 #define OV7670_COM11_HZAUTO 0x10

Definition at line 117 of file CameraOV7670.h.

5.11.1.13 #define OV7670_COM11_NIGHT 0x80

Definition at line 111 of file CameraOV7670.h.

5.11.1.14 #define OV7670_COM11_NMFR 0x60

Definition at line 114 of file CameraOV7670.h.

5.11.1.15 #define OV7670_COM12_HREF 0x80

Definition at line 126 of file CameraOV7670.h.

5.11.1.16 #define OV7670_COM13_GAMMA 0x80

Definition at line 129 of file CameraOV7670.h.

5.11.1.17 #define OV7670_COM13_UVSAT 0x40

Definition at line 132 of file CameraOV7670.h.

5.11.1.18 #define OV7670_COM13_UVSWAP 0x01

Definition at line 135 of file CameraOV7670.h.

5.11.1.19 #define OV7670_COM14_DCWEN 0x10

Definition at line 138 of file CameraOV7670.h.

5.11.1.20 #define OV7670_COM15_R00FF 0xc0

Definition at line 147 of file CameraOV7670.h.

5.11.1.21 #define OV7670_COM15_R01FE 0x80

Definition at line 144 of file CameraOV7670.h.

5.11.1.22 #define OV7670_COM15_R10F0 0x00

Definition at line 141 of file CameraOV7670.h.

5.11.1.23 #define OV7670_COM15_RGB555 0x30

Definition at line 153 of file CameraOV7670.h.

5.11.1.24 #define OV7670_COM15_RGB565 0x10

Definition at line 150 of file CameraOV7670.h.

5.11.1.25 #define OV7670_COM16_AWBGAIN 0x08

Definition at line 156 of file CameraOV7670.h.

5.11.1.26 #define OV7670_COM17_AECWIN 0xc0

Definition at line 159 of file CameraOV7670.h.

5.11.1.27 #define OV7670_COM17_CBAR 0x08

Definition at line 162 of file CameraOV7670.h.

5.11.1.28 #define OV7670_COM1_CCIR656 0x40

Arduino - CameraOV7670 implementation.

CameraOV7670.h

The class CameraOV7670.

Author

Dalmir da Silva dalmirdasilva@gmail.com

Definition at line 15 of file CameraOV7670.h.

5.11.1.29 #define OV7670_COM2_SSLEEP 0x10

Definition at line 18 of file CameraOV7670.h.

5.11.1.30 #define OV7670_COM3_DCWEN 0x04

Definition at line 27 of file CameraOV7670.h.

5.11.1.31 #define OV7670_COM3_SCALEEN 0x08

Definition at line 24 of file CameraOV7670.h.

5.11.1.32 #define OV7670_COM3_SWAP 0x40

Definition at line 21 of file CameraOV7670.h.

5.11.1.33 #define OV7670_COM7_BAYER 0x01

Definition at line 60 of file CameraOV7670.h.

5.11.1.34 #define OV7670_COM7_FMT_CIF 0x20

Definition at line 45 of file CameraOV7670.h.

5.11.1.35 #define OV7670_COM7_FMT_MASK 0x38

Definition at line 39 of file CameraOV7670.h.

5.11.1.36 #define OV7670_COM7_FMT_QCIF 0x08

Definition at line 51 of file CameraOV7670.h.

5.11.1.37 #define OV7670_COM7_FMT_QVGA 0x10

Definition at line 48 of file CameraOV7670.h.

5.11.1.38 #define OV7670_COM7_FMT_VGA 0x00

Definition at line 42 of file CameraOV7670.h.

5.11.1.39 #define OV7670_COM7_PBAYER 0x05

Definition at line 63 of file CameraOV7670.h.

5.11.1.40 #define OV7670_COM7_RESET 0x80

Definition at line 36 of file CameraOV7670.h.

5.11.1.41 #define OV7670_COM7_RGB 0x04

Definition at line 54 of file CameraOV7670.h.

5.11.1.42 #define OV7670_COM7_YUV 0x00

Definition at line 57 of file CameraOV7670.h.

5.11.1.43 #define OV7670_COM8_AEC 0x01

Definition at line 81 of file CameraOV7670.h.

5.11.1.44 #define OV7670_COM8_AECSTEP 0x40

Definition at line 69 of file CameraOV7670.h.

5.11.1.45 #define OV7670_COM8_AGC 0x04

Definition at line 75 of file CameraOV7670.h.

5.11.1.46 #define OV7670_COM8_AWB 0x02

Definition at line 78 of file CameraOV7670.h.

5.11.1.47 #define OV7670_COM8_BFILT 0x20

Definition at line 72 of file CameraOV7670.h.

5.11.1.48 #define OV7670_COM8_FASTAEC 0x80

Definition at line 66 of file CameraOV7670.h.

5.11.1.49 #define OV7670_MVFP_FLIP 0x10

Definition at line 105 of file CameraOV7670.h.

5.11.1.50 #define OV7670_MVFP_MIRROR 0x20

Definition at line 102 of file CameraOV7670.h.

5.11.1.51 #define OV7670_R76_BLKPCOR 0x80

Definition at line 168 of file CameraOV7670.h.

5.11.1.52 #define OV7670_R76_WHTPCOR 0x40

Definition at line 171 of file CameraOV7670.h.

5.11.1.53 #define OV7670_RGB444_ENABLE 0x02

Definition at line 174 of file CameraOV7670.h.

5.11.1.54 #define OV7670_RGB444_RGBX 0x01

Definition at line 177 of file CameraOV7670.h.

5.11.1.55 #define OV7670_TSLB_YLAST 0x04

Definition at line 108 of file CameraOV7670.h.

5.12 CameraOV7670.h

```
00001
00011 #ifndef __ARDUINO_DRIVER_CAMERA_OV7670_H__
00012 #define __ARDUINO_DRIVER_CAMERA_OV7670_H__ 1
00013
00014 // CCIR656 enable
00015 #define OV7670_COM1_CCIR656            0x40
00016
00017 // Soft sleep mode
00018 #define OV7670_COM2_SSLEEP             0x10
00019
```

```
00020 // Byte swap
00021 #define OV7670_COM3_SWAP              0x40
00022
00023 // Enable scaling
00024 #define OV7670_COM3_SCALEEN           0x08
00025
00026 // Enable downsamp/crop/window
00027 #define OV7670_COM3_DCWEN             0x04
00028
00029 // Use external clock directly
00030 #define OV7670_CLKRC_EXT              0x40
00031
00032 // Mask for internal clock scale
00033 #define OV7670_CLKRC_SCALE            0x3f
00034
00035 // Reset
00036 #define OV7670_COM7_RESET             0x80
00037
00038 //
00039 #define OV7670_COM7_FMT_MASK          0x38
00040
00041 //
00042 #define OV7670_COM7_FMT_VGA           0x00
00043
00044 // CIF format
00045 #define OV7670_COM7_FMT_CIF           0x20
00046
00047 // QVGA format
00048 #define OV7670_COM7_FMT_QVGA          0x10
00049
00050 // QCIF format
00051 #define OV7670_COM7_FMT_QCIF          0x08
00052
00053 // bits 0 and 2 - RGB format
00054 #define OV7670_COM7_RGB               0x04
00055
00056 // YUV
00057 #define OV7670_COM7_YUV               0x00
00058
00059 // Bayer format
00060 #define OV7670_COM7_BAYER             0x01
00061
00062 // Processed bayer
00063 #define OV7670_COM7_PBAYER            0x05
00064
00065 // Enable fast AGC/AEC
00066 #define OV7670_COM8_FASTAEC           0x80
00067
00068 // Unlimited AEC step size
00069 #define OV7670_COM8_AECSTEP           0x40
00070
00071 // Band filter enable
00072 #define OV7670_COM8_BFILT             0x20
00073
00074 // Auto gain enable
00075 #define OV7670_COM8_AGC               0x04
00076
00077 // White balance enable
00078 #define OV7670_COM8_AWB               0x02
00079
00080 // Auto exposure enable
00081 #define OV7670_COM8_AEC               0x01
00082
00083 // HSYNC instead of HREF
00084 #define OV7670_COM10_HSYNC            0x40
00085
00086 // Suppress PCLK on horiz blank
00087 #define OV7670_COM10_PCLK_HB          0x20
00088
00089 // Reverse HREF
00090 #define OV7670_COM10_HREF_REV         0x08
00091
00092 // VSYNC on clock leading edge
00093 #define OV7670_COM10_VS_LEAD          0x04
00094
00095 // VSYNC negative
00096 #define OV7670_COM10_VS_NEG           0x02
00097
00098 // HSYNC negative
00099 #define OV7670_COM10_HS_NEG           0x01
00100
00101 // Mirror image
00102 #define OV7670_MVFP_MIRROR            0x20
00103
00104 // Vertical flip
00105 #define OV7670_MVFP_FLIP              0x10
00106
```

```
00107 // UYVY or VYUY - see com13
00108 #define OV7670_TSLB_YLAST             0x04
00109
00110 // NIght mode enable
00111 #define OV7670_COM11_NIGHT            0x80
00112
00113 // Two bit NM frame rate
00114 #define OV7670_COM11_NMFR             0x60
00115
00116 // Auto detect 50/60 Hz
00117 #define OV7670_COM11_HZAUTO           0x10
00118
00119 // Manual 50Hz select
00120 #define OV7670_COM11_50HZ             0x08
00121
00122 // Exp
00123 #define OV7670_COM11_EXP              0x02
00124
00125 // HREF always
00126 #define OV7670_COM12_HREF             0x80
00127
00128 // Gamma enable
00129 #define OV7670_COM13_GAMMA            0x80
00130
00131 // UV saturation auto adjustment
00132 #define OV7670_COM13_UVSAT            0x40
00133
00134 // V before U - w/TSLB
00135 #define OV7670_COM13_UVSWAP           0x01
00136
00137 // DCW/PCLK-scale enable
00138 #define OV7670_COM14_DCWEN            0x10
00139
00140 // Data range 10 to F0
00141 #define OV7670_COM15_R10F0            0x00
00142
00143 // Data range 01 to FE
00144 #define OV7670_COM15_R01FE            0x80
00145
00146 // Data range 00 to FF
00147 #define OV7670_COM15_R00FF            0xc0
00148
00149 // RGB565 output
00150 #define OV7670_COM15_RGB565           0x10
00151
00152 // RGB555 output
00153 #define OV7670_COM15_RGB555           0x30
00154
00155 // AWB gain enable
00156 #define OV7670_COM16_AWBGAIN          0x08
00157
00158 // AEC window - must match COM4
00159 #define OV7670_COM17_AECWIN           0xc0
00160
00161 // DSP Color bar
00162 #define OV7670_COM17_CBAR             0x08
00163
00164 // Length
00165 #define OV7670_CMATRIX_LEN            0x06
00166
00167 // Black pixel correction enable
00168 #define OV7670_R76_BLKPCOR            0x80
00169
00170 // White pixel correction enable
00171 #define OV7670_R76_WHTPCOR            0x40
00172
00173 // Turn on RGB444, overrides 5x5
00174 #define OV7670_RGB444_ENABLE          0x02
00175
00176 // Empty nibble at end
00177 #define OV7670_RGB444_RGBX            0x01
00178
00179 #include <OutputStream.h>
00180 #include <Camera.h>
00181 #include <Wire.h>
00182
00183 class CameraOV7670 : public Camera {
00184 private:
00185
00186     unsigned char (*read)();
00187
00188     unsigned char vsyncPin;
00189
00190     unsigned char hsyncPin;
00191
00192     unsigned char address;
00193 public:
```

```
00194
00195     enum Register {
00196
00197         // Gain lower 8 bits (rest in vref)
00198         REG_GAIN = 0x00,
00199
00200         // blue gain
00201         REG_BLUE = 0x01,
00202
00203         // red gain
00204         REG_RED = 0x02,
00205
00206         // Pieces of GAIN, VSTART, VSTOP
00207         REG_VREF = 0x03,
00208
00209         // Control 1
00210         REG_COM1 = 0x04,
00211
00212         // U/B Average level
00213         REG_BAVE = 0x05,
00214
00215         // Y/Gb Average level
00216         REG_GBAVE = 0x06,
00217
00218         // AEC MS 5 bits
00219         REG_AECHH = 0x07,
00220
00221         // V/R Average level
00222         REG_RAVE = 0x08,
00223
00224         // Control 2
00225         REG_COM2 = 0x09,
00226
00227         // Product ID MSB
00228         REG_PID = 0x0a,
00229
00230         // Product ID LSB
00231         REG_VER = 0x0b,
00232
00233         // Control 3
00234         REG_COM3 = 0x0c,
00235
00236         // Control 4
00237         REG_COM4 = 0x0d,
00238
00239         // All "reserved"
00240         REG_COM5 = 0x0e,
00241
00242         // Control 6
00243         REG_COM6 = 0x0f,
00244
00245         // More bits of AEC value
00246         REG_AECH = 0x10,
00247
00248         // Clocl control
00249         REG_CLKRC = 0x11,
00250
00251         // Control 7
00252         REG_COM7 = 0x12,
00253
00254         // Control 8
00255         REG_COM8 = 0x13,
00256
00257         // Control 9  - gain ceiling
00258         REG_COM9 = 0x14,
00259
00260         // Control 10
00261         REG_COM10 = 0x15,
00262
00263         // Horiz start high bits
00264         REG_HSTART = 0x17,
00265
00266         // Horiz stop high bits
00267         REG_HSTOP = 0x18,
00268
00269         // Vert start high bits
00270         REG_VSTART = 0x19,
00271
00272         // Vert stop high bits
00273         REG_VSTOP = 0x1a,
00274
00275         // Pixel delay after HREF
00276         REG_PSHFT = 0x1b,
00277
00278         // Manuf. ID high
00279         REG_MIDH = 0x1c,
00280
```

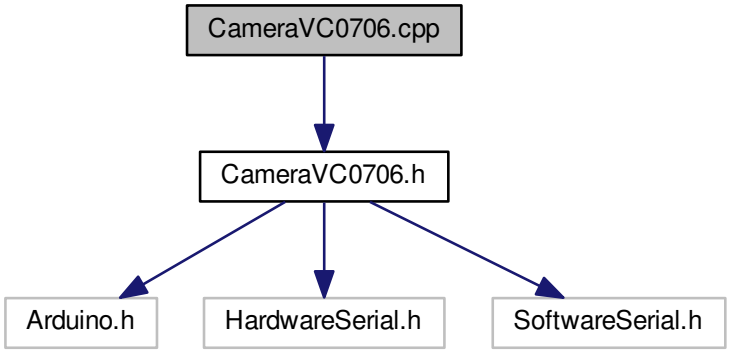
```
00281         // Manuf. ID low
00282         REG_MIDL = 0x1d,
00283
00284         // Mirror / vflip
00285         REG_MVFP = 0x1e,
00286
00287         // AGC upper limit
00288         REG_AEW = 0x24,
00289
00290         // AGC lower limit
00291         REG_AEB = 0x25,
00292
00293         // AGC/AEC fast mode op region
00294         REG_VPT = 0x26,
00295
00296         // HSYNC rising edge delay
00297         REG_HSYST = 0x30,
00298
00299         // HSYNC falling edge delay
00300         REG_HSYEN = 0x31,
00301
00302         // HREF pieces
00303         REG_HREF = 0x32,
00304
00305         // Lots of stuff
00306         REG_TSLB = 0x3a,
00307
00308         // Control 11
00309         REG_COM11 = 0x3b,
00310
00311         // Control 12
00312         REG_COM12 = 0x3c,
00313
00314         // Control 13
00315         REG_COM13 = 0x3d,
00316
00317         // Control 14
00318         REG_COM14 = 0x3e,
00319
00320         // Edge enhancement factor
00321         REG_EDGE = 0x3f,
00322
00323         // Control 15
00324         REG_COM15 = 0x40,
00325
00326         // Control 16
00327         REG_COM16 = 0x41,
00328
00329         // Control 17
00330         REG_COM17 = 0x42,
00331
00332         // CMatrix base
00333         REG_CMATRIX_BASE = 0x4f,
00334
00335         // CMatrix sign
00336         REG_CMATRIX_SIGN = 0x58,
00337
00338         // Brightness
00339         REG_BRIGHT = 0x55,
00340
00341         // Contrast control
00342         REG_CONTRAS = 0x56,
00343
00344         // Fix gain control
00345         REG_GFIX = 0x69,
00346
00347         // OV's name
00348         REG_R76 = 0x76,
00349
00350         // RGB 444 control
00351         REG_RGB444 = 0x8c,
00352
00353         // Hist AEC/AGC control 1
00354         REG_HAECC1 = 0x9f,
00355
00356         // Hist AEC/AGC control 2
00357         REG_HAECC2 = 0xa0,
00358
00359         // 50hz banding step limit
00360         REG_BD50MAX = 0xa5,
00361
00362         // Hist AEC/AGC control 3
00363         REG_HAECC3 = 0xa6,
00364
00365         // Hist AEC/AGC control 4
00366         REG_HAECC4 = 0xa7,
00367
```

```
00368         // Hist AEC/AGC control 5
00369         REG_HAECC5 = 0xa8,
00370
00371         // Hist AEC/AGC control 6
00372         REG_HAECC6 = 0xa9,
00373
00374         // Hist AEC/AGC control 7
00375         REG_HAECC7 = 0xaa,
00376
00377         // 60hz banding step limit
00378         REG_BD60MAX = 0xab
00379     };
00380
00389     CameraOV7670(unsigned char (*read)(), unsigned char vsyncPin,
00390             unsigned char hsyncPin);
00391
00395     void clearBuffers();
00396
00403     int readFrame(OutputStream *out);
00404 };
00405
00406 #endif /* __ARDUINO_DRIVER_CAMERA_OV7670_H__ */
```

5.13 CameraVC0706.cpp File Reference

`#include "CameraVC0706.h"`

Include dependency graph for CameraVC0706.cpp:



5.14 CameraVC0706.cpp

```
00001 #include "CameraVC0706.h"
00002
00003 #if VC0760_DEBUG == 1
00004 CameraVC0706::CameraVC0706(HardwareSerial *serial, Stream *debug) :
00005         serial(serial), debug(debug) {
00006     rxBufferPointer = 0;
00007     serialNumber = 0x00;
00008     framePointer = 0;
00009 }
00010 #else
00011 CameraVC0706::CameraVC0706(HardwareSerial *serial) : serial(serial) {
00012     rxBufferPointer = 0;
00013     serialNumber = 0x00;
00014     framePointer = 0;
00015 }
00016 #endif
00017
00018 bool CameraVC0706::begin(long baud) {
00019     serial->begin(baud);
00020     return true;
00021 }
```

```
00022
00023 bool CameraVC0706::close() {
00024     serial->end();
00025     return true;
00026 }
00027
00028 bool CameraVC0706::capture() {
00029     return executeBufferControl(STOP_CURRENT_FRAME);
00030 }
00031
00032 bool CameraVC0706::resume() {
00033     return executeBufferControl(RESUME_FRAME);
00034 }
00035
00036 bool CameraVC0706::executeBufferControl(unsigned char control) {
00037     unsigned char args[] = { (unsigned char) (control & 0x03) };
00038     return executeCommand(FBUF_CTRL, args, sizeof(args), 5);
00039 }
00040
00041 unsigned int CameraVC0706::readFrame(unsigned char *buf,
00042         unsigned int frameOffset, unsigned int bufferOffset,
00043         unsigned int len) {
00044     unsigned int bytesRead = 0;
00045     unsigned char args[] = { 0x00, 0x0a, 0x00, 0x00,
00046             (unsigned char) ((frameOffset >> 8) & 0xff),
00047             (unsigned char) (frameOffset & 0xff), 0x00, 0x00,
00048             (unsigned char) ((len >> 8) & 0xff), (unsigned char) (len
00049                     & 0xff), (VC0760_CAMERA_DELAY >> 8) & 0xff,
00050             VC0760_CAMERA_DELAY & 0xff };
00051
00052     if (!executeCommand(READ_FBUF, args, sizeof(args), 5)) {
00053         return 0;
00054     }
00055     while (bytesRead < len) {
00056         delay(10);
00057         bytesRead += read(&buf[bufferOffset + bytesRead],
00058                 len - bytesRead);
00059     }
00060     readResponse(5);
00061     return bytesRead;
00062 }
00063
00064 bool CameraVC0706::setDownSize(unsigned char widthDownSize,
00065         unsigned char heightDownSize) {
00066     unsigned char args[] = { (unsigned char) ((widthDownSize & 0x03)
00067             | ((heightDownSize << 2) & 0x0c)) };
00068     return executeCommand(DOWNSIZE_CTRL, args, sizeof(args), 5);
00069 }
00070
00071 unsigned char CameraVC0706::getDownSize() {
00072     unsigned char args[] = { };
00073     bool run = executeCommand(DOWNSIZE_STATUS, args, sizeof(args), 6);
00074     if (run) {
00075         return 0;
00076     }
00077     return rxBuffer[5];
00078 }
00079
00080 unsigned int CameraVC0706::getFrameLength() {
00081     unsigned int frameLength = 0;
00082     unsigned char args[] = { 0x00 };
00083     if (!executeCommand(GET_FBUF_LEN, args, sizeof(args), 9)
00084             && rxBuffer[4] == 0x04) {
00085         return 0;
00086     }
00087     frameLength |= rxBuffer[7];
00088     frameLength <<= 8;
00089     frameLength |= rxBuffer[8];
00090     return frameLength;
00091 }
00092
00093 bool CameraVC0706::setHorizontalMirror(unsigned char by,
00094         unsigned char mirrorMode) {
00095     unsigned char args[] = { (unsigned char) (by & 0x01),
00096             (unsigned char) (mirrorMode & 0x01) };
00097     return executeCommand(MIRROR_CTRL, args, sizeof(args), 5);
00098 }
00099
00100 unsigned char CameraVC0706::getHorizontalMirrorStatus() {
00101     unsigned char args[] = { };
00102     bool run = executeCommand(MIRROR_STATUS, args, sizeof(args), 7);
00103     unsigned char status = 0;
00104     if (run) {
00105         status = (rxBuffer[6] & 0x01) | ((rxBuffer[5] << 1) & 0x02);
00106     }
00107     return status;
00108 }
```

```
00109
00110 bool CameraVC0706::setColorControl(unsigned char by,
00111         unsigned char colorControlMode) {
00112     unsigned char args[] = { (unsigned char) (by & 0x01),
00113             (unsigned char) (colorControlMode & 0x01) };
00114     return executeCommand(COLOR_CTRL, args, sizeof(args), 5);
00115 }
00116
00117 unsigned char CameraVC0706::getColorControlStatus() {
00118     unsigned char args[] = { };
00119     bool run = executeCommand(COLOR_STATUS, args, sizeof(args), 8);
00120     unsigned char status = 0;
00121     if (run) {
00122         status = (rxBuffer[5] & 0x01) | ((rxBuffer[6] << 2) & 0x06);
00123     }
00124     return status;
00125 }
00126
00127 bool CameraVC0706::setOutputResolution(unsigned char resolution) {
00128     unsigned char args[] = { 0x04, 0x01, 0x00, 0x19, resolution };
00129     return executeCommand(WRITE_DATA, args, sizeof(args), 5);
00130 }
00131
00132 bool CameraVC0706::setMotionMonitoring(bool monitor) {
00133     unsigned char args[] = { (unsigned char) monitor };
00134     return executeCommand(COMM_MOTION_CTRL, args, sizeof(args), 5);
00135 }
00136
00137 bool CameraVC0706::getMotionMonitoringStatus() {
00138     unsigned char args[] = { };
00139     return (executeCommand(COMM_MOTION_STATUS, args, sizeof(args), 6))
00140             && rxBuffer[5];
00141 }
00142
00143 bool CameraVC0706::pollMotionMonitoring(unsigned int timeout,
00144         void (*callback)(void *)) {
00145     long start = 0;
00146     bool detected = false;
00147     start = millis();
00148     do {
00149         if (readResponse(5) > 0) {
00150             detected = verifyResponse(COMM_MOTION_DETECTED);
00151         }
00152         if (detected && callback != 0) {
00153             callback(this);
00154         }
00155     } while (!detected && ((millis() - start) < timeout));
00156     return detected;
00157 }
00158
00159 bool CameraVC0706::setMotionControl(unsigned char motionControl,
00160         unsigned char param0, unsigned char param1) {
00161     unsigned char args[] = { motionControl, param0, param1 };
00162     return executeCommand(MOTION_CTRL, args, sizeof(args), 5);
00163 }
00164
00165 unsigned int CameraVC0706::write(unsigned char *buf,
00166         unsigned int size) {
00167     unsigned int txLength = 0;
00168
00169 #if VC0760_DEBUG == 1
00170     debug->print("About to write: ");
00171     debug->print(size);
00172     debug->println(" bytes.");
00173 #endif
00174
00175     txLength = serial->write(&buf[0], size);
00176
00177 #if VC0760_DEBUG == 1
00178     if (txLength < 0) {
00179         debug->println("UART TX error.");
00180     } else if (txLength != size) {
00181         debug->print("Sent bytes ");
00182         debug->print(txLength);
00183         debug->print(" differs from the size to be send ");
00184         debug->println(size);
00185     }
00186 #endif
00187
00188     return txLength;
00189 }
00190
00191 unsigned int CameraVC0706::read(unsigned char *buf, unsigned int size) {
00192     unsigned char c = 0;
00193     unsigned int rxLength = 0;
00194     unsigned char count = size;
00195     while (count-- > 0 && serial->available()
```

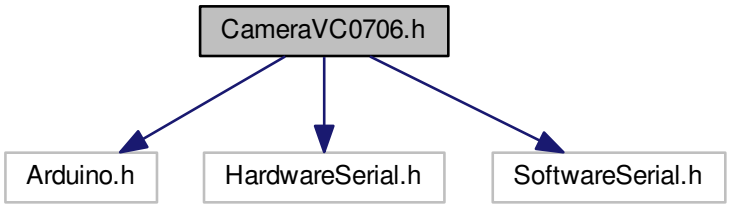
```
00196             && (c = serial->read()) != -1) {
00197         buf[rxLength++] = c;
00198     }
00199
00200 #if VC0760_DEBUG == 1
00201     if (c < 0) {
00202         debug->println("Error on read.");
00203     } else if (rxLength == 0) {
00204         debug->println("No data received on read.");
00205     } else if (rxLength != size) {
00206         debug->print("Read bytes: ");
00207         debug->print(rxLength);
00208         debug->print(" differs from the size to be read: ");
00209         debug->println(size);
00210     } else {
00211         debug->print("It matches! ");
00212         debug->print(rxLength);
00213         debug->print(" bytes read when expecting: ");
00214         debug->println(size);
00215     }
00216 #endif
00217
00218     return rxLength;
00219 }
00220
00221 bool CameraVC0706::executeCommand(unsigned char cmd,
00222         unsigned char *args, unsigned char argc,
00223         unsigned int responseLength) {
00224     if (!sendCommand(cmd, args, argc)) {
00225         return false;
00226     }
00227     delay(50);
00228     if (!readResponse(responseLength)) {
00229         return false;
00230     }
00231     if (!verifyResponse(cmd)) {
00232         return false;
00233     }
00234     return true;
00235 }
00236
00237 unsigned int CameraVC0706::sendCommand(unsigned char cmd,
00238         unsigned char *args, unsigned int argc) {
00239     unsigned int sentBytes = 0;
00240     unsigned int bufSize = 4 + argc;
00241     unsigned char buf[bufSize];
00242     buf[0] = VC0760_PROTOCOL_SIGN_TX;
00243     buf[1] = serialNumber;
00244     buf[2] = cmd;
00245     buf[3] = argc;
00246     memcpy(&buf[4], args, argc);
00247     printBuff(buf, bufSize);
00248     sentBytes = write(buf, bufSize);
00249
00250 #if VC0760_DEBUG == 1
00251     debug->print(sentBytes);
00252     debug->println(" bytes written.");
00253 #endif
00254
00255     if (sentBytes != bufSize) {
00256
00257 #if VC0760_DEBUG == 1
00258         debug->print("Sent different amount than expected: ");
00259         debug->println(bufSize);
00260 #endif
00261
00262         return 0;
00263     }
00264     return sentBytes;
00265 }
00266
00267 bool CameraVC0706::verifyResponse(unsigned char cmd) {
00268     if ((rxBuffer[0] != VC0760_PROTOCOL_SIGN_RX)
00269             || (rxBuffer[1] != serialNumber) || (rxBuffer[2] != cmd)
00270             || (rxBuffer[3] != 0x00)) {
00271         return false;
00272     }
00273     return true;
00274 }
00275
00276 unsigned int CameraVC0706::readResponse(unsigned int length) {
00277     rxBufferPointer = read(rxBuffer, length);
00278 #if VC0760_DEBUG == 1
00279     printBuff(rxBuffer, rxBufferPointer);
00280 #endif
00281     return rxBufferPointer;
00282 }
```

```
00283
00284 #if VC0760_DEBUG == 1
00285 void CameraVC0706::printBuff(unsigned char *buf, unsigned int c) {
00286     debug->println("Printing buffer:");
00287     for (unsigned int i = 0; i < c; i++) {
00288         debug->print(i);
00289         debug->print(" : ");
00290         debug->println(buf[i], HEX);
00291     }
00292 }
00293 #endif
00294
00295 bool CameraVC0706::reset() {
00296     unsigned char args[] = { };
00297     bool run = executeCommand(SYSTEM_RESET, args, sizeof(args), 5);
00298
00299 #if VC0760_DEBUG == 1
00300     if (run) {
00301         debug->println("Waiting the system to reset.");
00302         delay(10);
00303     }
00304 #endif
00305
00306     return run;
00307 }
00308
00309 float CameraVC0706::getVersion() {
00310     unsigned int i = 0;
00311     float version = 0.0;
00312     unsigned char args[] = { };
00313     if (!executeCommand(GEN_VERSION, args, sizeof(args), 16)) {
00314         return version;
00315     }
00316     while (rxBuffer[i++] != ' ')
00317         ;
00318     version += rxBuffer[i] - '0';
00319     version += 0.1 * (rxBuffer[i + 2] - '0');
00320     return version;
00321 }
00322
00323 bool CameraVC0706::setOsdCharacters(unsigned char x, unsigned char y,
00324         unsigned char *str, unsigned char len) {
00325     if (len > 14) {
00326         len = 14;
00327     }
00328     unsigned char args[2 + len];
00329     args[0] = len;
00330     args[1] = ((x << 6) & 0x60) | (y & 0x1f);
00331     memcpy(&args[2], str, len);
00332     return executeCommand(OSD_ADD_CHAR, args, sizeof(args), 5);
00333 }
00334
00335 bool CameraVC0706::setCompression(unsigned char compression) {
00336     unsigned char args[] = { 0x01, 0x01, 0x12, 0x04, compression };
00337     return executeCommand(WRITE_DATA, args, sizeof(args), 5);
00338 }
00339
00340 unsigned char CameraVC0706::getCompression() {
00341     unsigned char args[] = { 0x01, 0x01, 0x12, 0x04 };
00342     bool run = executeCommand(READ_DATA, args, sizeof(args), 6);
00343     unsigned char compression = 0;
00344     if (run) {
00345         compression = rxBuffer[5];
00346     }
00347     return compression;
00348 }
00349
00350 bool CameraVC0706::setTVOutput(unsigned char onOff) {
00351     unsigned char args[] = { (unsigned char) (onOff & 0x01) };
00352     return executeCommand(TV_OUT_CTRL, args, sizeof(args), 5);
00353 }
00354
00355 bool CameraVC0706::setBoudRate(long baudRate) {
00356     this->baudRate = baudRate;
00357     unsigned char args[] = { 0x01, (unsigned char) ((baudRate >> 8)
00358             & 0xff), (unsigned char) (baudRate & 0xff) };
00359     return executeCommand(SET_PORT, args, sizeof(args), 5);
00360 }
```

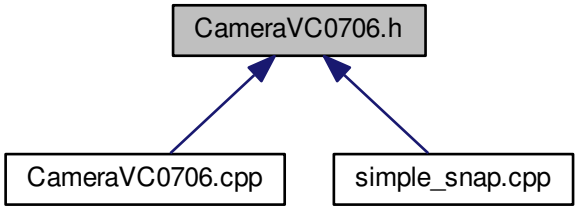
5.15 CameraVC0706.h File Reference

```
#include <Arduino.h>
#include <HardwareSerial.h>
#include <SoftwareSerial.h>
```

Include dependency graph for CameraVC0706.h:



This graph shows which files directly or indirectly include this file:



Classes

- class CameraVC0706

Macros

- #define VC0760_DEBUG 1
- #define VC0760_PROTOCOL_SIGN_TX 0x56
- #define VC0760_PROTOCOL_SIGN_RX 0x76
- #define VC0760_RX_BUFFER_SIZE 0x0f
- #define VC0760_CAMERA_DELAY 0x0100

5.15.1 Macro Definition Documentation

5.15.1.1 #define VC0760_CAMERA_DELAY 0x0100

Definition at line 23 of file CameraVC0706.h.

5.15.1.2 #define VC0760_DEBUG 1

Raspberry - CameraVC0706 implementation.

CameraVC0706.h

The class CameraVC0706.

Author

Dalmir da Silva dalmirdasilva@gmail.com

Definition at line 18 of file CameraVC0706.h.

5.15.1.3 #define VC0760_PROTOCOL_SIGN_RX 0x76

Definition at line 20 of file CameraVC0706.h.

5.15.1.4 #define VC0760_PROTOCOL_SIGN_TX 0x56

Definition at line 19 of file CameraVC0706.h.

5.15.1.5 #define VC0760_RX_BUFFER_SIZE 0x0f

Definition at line 22 of file CameraVC0706.h.

5.16 CameraVC0706.h

```
00001
00011 #ifndef __RASPBERRY_DRIVER_CAMERA_VC0706_H__
00012 #define __RASPBERRY_DRIVER_CAMERA_VC0706_H__ 1
00013
00014 #include <Arduino.h>
00015 #include <HardwareSerial.h>
00016 #include <SoftwareSerial.h>
00017
00018 #define VC0760_DEBUG                1
00019 #define VC0760_PROTOCOL_SIGN_TX     0x56
00020 #define VC0760_PROTOCOL_SIGN_RX     0x76
00021
00022 #define VC0760_RX_BUFFER_SIZE       0x0f
00023 #define VC0760_CAMERA_DELAY         0x0100
00024
00025 class CameraVC0706 {
00026
00027     unsigned char rxBuffer[VC0760_RX_BUFFER_SIZE];
00028
00029     unsigned int rxBufferPointer;
00030
00031     unsigned char serialNumber;
00032
00033     unsigned int framePointer;
00034
00035     unsigned int baudRate;
00036
00037     HardwareSerial *serial;
00038
00039 #if VC0760_DEBUG == 1
00040     Stream *debug;
00041 #endif
00042
00043 public:
00044
00045     enum DownSize {
00046         NO_ZOON = 0x00, HALF_SIZE = 0x01, QUARTER_SIZE = 0x02
00047     };
00048
00049     enum ControlBy {
00050         GPIO = 0x00, UART = 0x01,
00051     };
00052
00053     enum MotionControl {
00054
00055         // Motion control and enabling control
00056         MOTION_CONTROL = 0,
00057
```

```
00058         // Alarm-output attribute
00059         ALARM_ATTRIBUTE = 1,
00060
00061         // Alarm-output enabling control
00062         ALARM_ENABLING = 2,
00063
00064         // Alarm-output control
00065         ALARM_CONTROL = 3
00066     };
00067
00068     enum ColorControlMode {
00069
00070         // Automatically step black-white and color.
00071         AUTO_STEP_BLACK_WHITE = 0,
00072
00073         // Manually step color, select color.
00074         MANUAL_STEP_SELECT_COLOR = 1,
00075
00076         // Manually step color, select black-white.
00077         MANUAL_STEP_SELECT_BLACK_WHITE = 2
00078     };
00079
00080     enum Command {
00081
00082         // Get Firmware version information
00083         GEN_VERSION = 0x11,
00084
00085         // Set serial number
00086         SET_SERIAL_NUMBER = 0x21,
00087
00088         // Set port
00089         SET_PORT = 0x24,
00090
00091         // System reset
00092         SYSTEM_RESET = 0x26,
00093
00094         // Read data register
00095         READ_DATA = 0x30,
00096
00097         // Write data register
00098         WRITE_DATA = 0x31,
00099
00100         // Read buffer register
00101         READ_FBUF = 0x32,
00102
00103         // Write buffer register
00104         WRITE_FBUF = 0x33,
00105
00106         // Get image lengths in frame buffer
00107         GET_FBUF_LEN = 0x34,
00108
00109         // Set image lengths in frame buffer
00110         SET_FBUF_LEN = 0x35,
00111
00112         // Control frame buffer register
00113         FBUF_CTRL = 0x36,
00114
00115         // Motion detect on or off in communication interface
00116         COMM_MOTION_CTRL = 0x37,
00117
00118         // Get motion monitoring status in communication interface
00119         COMM_MOTION_STATUS = 0x38,
00120
00121         // Motion has been detected by communication interface
00122         COMM_MOTION_DETECTED = 0x39,
00123
00124         // Mirror control
00125         MIRROR_CTRL = 0x3A,
00126
00127         // Mirror status
00128         MIRROR_STATUS = 0x3B,
00129
00130         // Control color
00131         COLOR_CTRL = 0x3C,
00132
00133         // Color status
00134         COLOR_STATUS = 0x3D,
00135
00136         // Power mode control
00137         POWER_SAVE_CTRL = 0x3E,
00138
00139         // Power save mode or not
00140         POWER_SAVE_STATUS = 0x3F,
00141
00142         // Control AE
00143         AE_CTRL = 0x40,
00144
```

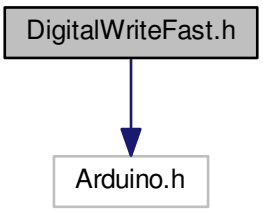
```
00145         // AE status
00146         AE_STATUS = 0x41,
00147
00148         // Motion control
00149         MOTION_CTRL = 0x42,
00150
00151         // Get motion status
00152         MOTION_STATUS = 0x43,
00153
00154         // TV output on or off control
00155         TV_OUT_CTRL = 0x44,
00156
00157         // Add characters to OSD channels (unsupported by the VC0706 firmware)
00158         OSD_ADD_CHAR = 0x45,
00159
00160         // Downsize Control
00161         DOWNSIZE_CTRL = 0x54,
00162
00163         // Downsize status
00164         DOWNSIZE_STATUS = 0x55,
00165
00166         // Get SPI flash size
00167         GET_FLASH_SIZE = 0x60,
00168
00169         // Erase one block of the flash
00170         ERASE_FLASH_SECTOR = 0x61,
00171
00172         // Erase the whole flash
00173         ERASE_FLASH_ALL = 0x62,
00174
00175         // Read and show logo
00176         READ_LOGO = 0x70,
00177
00178         // Bitmap operation
00179         SET_BITMAP = 0x71,
00180
00181         // Write mass data at a time
00182         BATCH_WRITE = 0x80
00183     };
00184
00185     enum OutputResolution {
00186         RES_640X480 = 0x00, RES_320X240 = 0x11,
      RES_160X120 = 0x22
00187     };
00188
00189     enum BufferControl {
00190
00191         // Stop current frame
00192         STOP_CURRENT_FRAME = 0x00,
00193
00194         // Stop next frame
00195         STOP_NEXT_FRAME = 0x01,
00196
00197         // Resume frame
00198         RESUME_FRAME = 0x03,
00199
00200         // Step frame
00201         STEP_FRAME = 0x03
00202     };
00203
00204     enum BaudRate {
00205         B_9600 = 0xaec8,
00206         B_19200 = 0x56e4,
00207         B_38400 = 0x2af2,
00208         B_57600 = 0x1c4c,
00209         B_115200 = 0x0da6
00210     };
00211
00212 #if VC0760_DEBUG == 1
00213
00219     CameraVC0706(HardwareSerial *serial, Stream *debug);
00220 #else
00221
00226     CameraVC0706(HardwareSerial *serial);
00227 #endif
00228
00232     bool begin(long baud);
00233
00237     bool close();
00238
00242     bool capture();
00243
00247     bool resume();
00248
00278     bool setDownSize(unsigned char widthDownSize,
00279             unsigned char heightDownSize);
00280
```

```
00302     unsigned char getDownSize();
00303
00321     unsigned int getFrameLength();
00322
00351     unsigned int readFrame(unsigned char *buf, unsigned int frameOffset,
00352             unsigned int bufferOffset, unsigned int len);
00353
00377     bool setHorizontalMirror(unsigned char by,
00378             unsigned char mirrorMode);
00379
00400     unsigned char getHorizontalMirrorStatus();
00401
00426     bool setColorControl(unsigned char by,
00427             unsigned char colorControlMode);
00428
00455     unsigned char getColorControlStatus();
00456
00462     bool setOutputResolution(unsigned char resolution);
00463
00479     bool setMotionMonitoring(bool monitor);
00480
00500     bool getMotionMonitoringStatus();
00501
00563     bool setMotionControl(unsigned char motionControl,
00564             unsigned char param0, unsigned char param1);
00565
00583     bool pollMotionMonitoring(unsigned int timeout,
00584             void (*callback)(void *));
00585
00608     bool setOsdCharacters(unsigned char x, unsigned char y,
00609             unsigned char *str, unsigned char len);
00610
00616     bool setCompression(unsigned char compression);
00617
00623     unsigned char getCompression();
00624
00634     float getVersion();
00635
00639     bool reset();
00640
00657     bool executeBufferControl(unsigned char control);
00658
00664     bool setTVOutput(unsigned char onOff);
00665
00683     bool setBoudRate(long baudRate);
00684
00693     bool executeCommand(unsigned char cmd, unsigned char *args,
00694             unsigned char argc, unsigned int responseLength);
00695
00696 private:
00697
00704 #if VC0760_DEBUG == 1
00705     void printBuff(unsigned char *buf, unsigned int c);
00706 #endif
00707
00714     unsigned int write(unsigned char *buf, unsigned int size);
00715
00722     unsigned int read(unsigned char *buf, unsigned int size);
00723
00733     unsigned int sendCommand(unsigned char cmd, unsigned char *args,
00734             unsigned int argc);
00735
00742     bool verifyResponse(unsigned char cmd);
00743
00752     unsigned int readResponse(unsigned int length);
00753 };
00754
00755 #endif /* __RASPBERRY_DRIVER_CAMERA_VC0706_H__ */
```

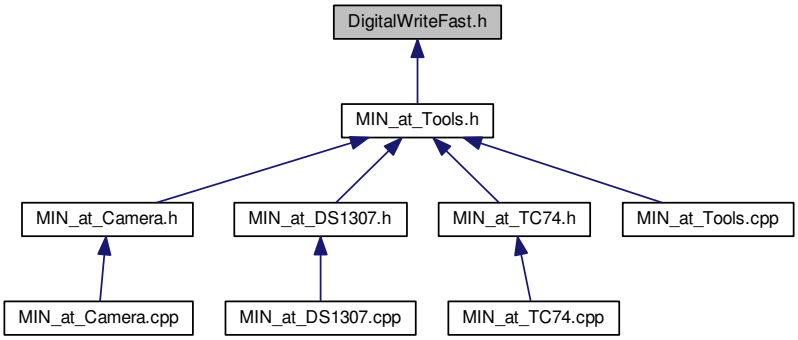
5.17 DigitalWriteFast.h File Reference

```
#include <Arduino.h>
```

Include dependency graph for DigitalWriteFast.h:



This graph shows which files directly or indirectly include this file:



Macros

- #define DIGITALWRITEFAST
- #define BIT_READ(value, bit) (((value) >> (bit)) & 0x01)
- #define BIT_SET(value, bit) ((value) |= (1UL << (bit)))
- #define BIT_CLEAR(value, bit) ((value) &= ~(1UL << (bit)))
- #define BIT_WRITE(value, bit, bitvalue) (bitvalue ? BIT_SET(value, bit) : BIT_CLEAR(value, bit))
- #define digitalPinToPortReg(P) (((P) >= 0 && (P) <= 7) ? &PORTD : (((P) >= 8 && (P) <= 13) ? &PORTB : &PORTC))
- #define digitalPinToDDRReg(P) (((P) >= 0 && (P) <= 7) ? &DDRD : (((P) >= 8 && (P) <= 13) ? &DDRB : &DDRC))
- #define digitalPinToPINReg(P) (((P) >= 0 && (P) <= 7) ? &PIND : (((P) >= 8 && (P) <= 13) ? &PINB : &PINC))
- #define __digitalPinToBit(P) (((P) >= 0 && (P) <= 7) ? (P) : (((P) >= 8 && (P) <= 13) ? (P) - 8 : (P) - 14))
- #define __digitalPinToTimer(P)
- #define __digitalPinToTimerBit(P)
- #define __atomicWrite__(A, P, V)

- #define digitalWriteFast(P, V)
- #define pinModeFast(P, V)
- #define noAnalogWrite(P)
- #define digitalReadFast(P) ((int) _digitalReadFast_((P)))
- #define _digitalReadFast_(P)

5.17.1 Macro Definition Documentation

5.17.1.1 #define __atomicWrite__(A, P, V)

Value:

```
if ( (int)(A) < 0x40) { bitWrite(*(A), __digitalPinToBit(P), (V) );}  \
else {                                                                  \
uint8_t register saveSreg = SREG;                                  \
cli();                                                             \
bitWrite(*(A), __digitalPinToBit(P), (V) );                          \
SREG=saveSreg;                                                     \
}
```

Definition at line 121 of file DigitalWriteFast.h.

5.17.1.2 #define __digitalPinToBit(P) (((P) >= 0 && (P) <= 7) ? (P) : (((P) >= 8 && (P) <= 13) ? (P) - 8 : (P) - 14))

Definition at line 92 of file DigitalWriteFast.h.

5.17.1.3 #define __digitalPinToTimer(P)

Value:

```
(((P) ==  6 || (P) ==  5) ? &TCCR0A : \
(((P) ==  9 || (P) == 10) ? &TCCR1A : \
(((P) == 11 || (P) ==  3) ? &TCCR2A : 0)))
```

Definition at line 104 of file DigitalWriteFast.h.

5.17.1.4 #define __digitalPinToTimerBit(P)

Value:

```
(((P) ==  6) ? COM0A1 : (((P) ==  5) ? COM0B1 : \
(((P) ==  9) ? COM1A1 : (((P) == 10) ? COM1B1 : \
(((P) == 11) ? COM2A1 : COM2B1)))))
```

Definition at line 108 of file DigitalWriteFast.h.

5.17.1.5 #define _digitalReadFast_(P)

Value:

```
(__builtin_constant_p(P) ) ? ( \
    ( BIT_READ(*digitalPinToPINReg(P),
      __digitalPinToBit(P))) ) : \
    digitalRead((P))
```

Definition at line 157 of file DigitalWriteFast.h.

5.17.1.6 #define BIT_CLEAR(value, bit) ((value) &= ~(1UL << (bit)))

Definition at line 8 of file DigitalWriteFast.h.

5.17.1.7 #define BIT_READ(value, bit) (((value) >> (bit)) & 0x01)

Definition at line 6 of file DigitalWriteFast.h.

5.17.1.8 #define BIT_SET(*value, bit*) ((value) |= (1UL << (bit)))

Definition at line 7 of file DigitalWriteFast.h.

5.17.1.9 #define BIT_WRITE(*value, bit, bitvalue*) (bitvalue ? BIT_SET(value, bit) : BIT_CLEAR(value, bit))

Definition at line 9 of file DigitalWriteFast.h.

5.17.1.10 #define digitalPinToDDRReg(*P*) (((P) >= 0 && (P) <= 7) ? &DDRD : (((P) >= 8 && (P) <= 13) ? &DDRB : &DDRC))

Definition at line 88 of file DigitalWriteFast.h.

5.17.1.11 #define digitalPinToPINReg(*P*) (((P) >= 0 && (P) <= 7) ? &PIND : (((P) >= 8 && (P) <= 13) ? &PINB : &PINC))

Definition at line 90 of file DigitalWriteFast.h.

5.17.1.12 #define digitalPinToPortReg(*P*) (((P) >= 0 && (P) <= 7) ? &PORTD : (((P) >= 8 && (P) <= 13) ? &PORTB : &PORTC))

Definition at line 86 of file DigitalWriteFast.h.

5.17.1.13 #define digitalReadFast(*P*) ((int) _digitalReadFast_((P)))

Definition at line 156 of file DigitalWriteFast.h.

5.17.1.14 #define DIGITALWRITEFAST

Definition at line 2 of file DigitalWriteFast.h.

5.17.1.15 #define digitalWriteFast(*P, V*)

Value:

```
do {                        \
if (__builtin_constant_p(P) && __builtin_constant_p(V))   __atomicWrite__((uint8_t*)
      digitalPinToPortReg(P),P,V) \
else  digitalWrite((P), (V));         \
}while (0)
```

Definition at line 132 of file DigitalWriteFast.h.

5.17.1.16 #define noAnalogWrite(*P*)

Value:

```
do {if (__builtin_constant_p(P) )  __atomicWrite((uint8_t*) __digitalPinToTimer(P),P,0)
      \
        else turnOffPWM((P));   \
} while (0)
```

Definition at line 148 of file DigitalWriteFast.h.

5.17.1.17 #define pinModeFast(*P, V*)

Value:

```
do {if (__builtin_constant_p(P) && __builtin_constant_p(V)) __atomicWrite__((uint8_t*)
      digitalPinToDDRReg(P),P,V) \
else pinMode((P), (V)); \
} while (0)
```

Definition at line 140 of file DigitalWriteFast.h.

5.18 DigitalWriteFast.h

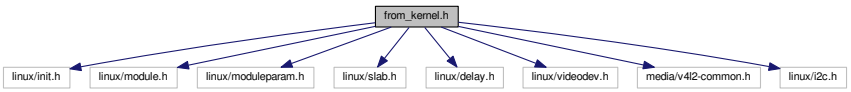
```
00001 #ifndef DIGITALWRITEFAST_H
00002 #define DIGITALWRITEFAST
00003
00004 #include <Arduino.h>
00005
00006 #define BIT_READ(value, bit) (((value) >> (bit)) & 0x01)
00007 #define BIT_SET(value, bit) ((value) |= (1UL << (bit)))
00008 #define BIT_CLEAR(value, bit) ((value) &= ~(1UL << (bit)))
00009 #define BIT_WRITE(value, bit, bitvalue) (bitvalue ? BIT_SET(value, bit) : BIT_CLEAR(value, bit))
00010
00011 #if !defined(digitalPinToPortReg)
00012 #if defined(__AVR_ATmega1280__) || defined(__AVR_ATmega2560__)
00013 // Arduino Mega Pins
00014 #define digitalPinToPortReg(P) \
00015 (((P) >= 22 && (P) <= 29) ? &PORTA : \
00016 ((((P) >= 10 && (P) <= 13) || ((P) >= 50 && (P) <= 53)) ? &PORTB : \
00017 (((P) >= 30 && (P) <= 37) ? &PORTC : \
00018 ((((P) >= 18 && (P) <= 21) || (P) == 38) ? &PORTD : \
00019 ((((P) >= 0 && (P) <= 3) || (P) == 5) ? &PORTE : \
00020 (((P) >= 54 && (P) <= 61) ? &PORTF : \
00021 ((((P) >= 39 && (P) <= 41) || (P) == 4) ? &PORTG : \
00022 ((((P) >= 6 && (P) <= 9) || (P) == 16 || (P) == 17) ? &PORTH : \
00023 (((P) == 14 || (P) == 15) ? &PORTJ : \
00024 (((P) >= 62 && (P) <= 69) ? &PORTK : &PORTL)))))))))
00025
00026 #define digitalPinToDDRReg(P) \
00027 (((P) >= 22 && (P) <= 29) ? &DDRA : \
00028 ((((P) >= 10 && (P) <= 13) || ((P) >= 50 && (P) <= 53)) ? &DDRB : \
00029 (((P) >= 30 && (P) <= 37) ? &DDRC : \
00030 ((((P) >= 18 && (P) <= 21) || (P) == 38) ? &DDRD : \
00031 ((((P) >= 0 && (P) <= 3) || (P) == 5) ? &DDRE : \
00032 (((P) >= 54 && (P) <= 61) ? &DDRF : \
00033 ((((P) >= 39 && (P) <= 41) || (P) == 4) ? &DDRG : \
00034 ((((P) >= 6 && (P) <= 9) || (P) == 16 || (P) == 17) ? &DDRH : \
00035 (((P) == 14 || (P) == 15) ? &DDRJ : \
00036 (((P) >= 62 && (P) <= 69) ? &DDRK : &DDRL)))))))))
00037
00038 #define digitalPinToPINReg(P) \
00039 (((P) >= 22 && (P) <= 29) ? &PINA : \
00040 ((((P) >= 10 && (P) <= 13) || ((P) >= 50 && (P) <= 53)) ? &PINB : \
00041 (((P) >= 30 && (P) <= 37) ? &PINC : \
00042 ((((P) >= 18 && (P) <= 21) || (P) == 38) ? &PIND : \
00043 ((((P) >= 0 && (P) <= 3) || (P) == 5) ? &PINE : \
00044 (((P) >= 54 && (P) <= 61) ? &PINF : \
00045 ((((P) >= 39 && (P) <= 41) || (P) == 4) ? &PING : \
00046 ((((P) >= 6 && (P) <= 9) || (P) == 16 || (P) == 17) ? &PINH : \
00047 (((P) == 14 || (P) == 15) ? &PINJ : \
00048 (((P) >= 62 && (P) <= 69) ? &PINK : &PINL)))))))))
00049
00050 #define __digitalPinToBit(P) \
00051 (((P) >=  7 && (P) <=  9) ? (P) - 3 : \
00052 (((P) >= 10 && (P) <= 13) ? (P) - 6 : \
00053 (((P) >= 22 && (P) <= 29) ? (P) - 22 : \
00054 (((P) >= 30 && (P) <= 37) ? 37 - (P) : \
00055 (((P) >= 39 && (P) <= 41) ? 41 - (P) : \
00056 (((P) >= 42 && (P) <= 49) ? 49 - (P) : \
00057 (((P) >= 50 && (P) <= 53) ? 53 - (P) : \
00058 (((P) >= 54 && (P) <= 61) ? (P) - 54 : \
00059 (((P) >= 62 && (P) <= 69) ? (P) - 62 : \
00060 (((P) == 0 || (P) == 15 || (P) == 17 || (P) == 21) ? 0 : \
00061 (((P) == 1 || (P) == 14 || (P) == 16 || (P) == 20) ? 1 : \
00062 (((P) == 19) ? 2 : \
00063 (((P) == 5 || (P) == 6 || (P) == 18) ? 3 : \
00064 (((P) == 2) ? 4 : \
00065 (((P) == 3 || (P) == 4) ? 5 : 7)))))))))))))))
00066
00067 // 15 PWM
00068 #define __digitalPinToTimer(P) \
00069 (((P) == 13 || (P) ==  4) ? &TCCR0A : \
00070 (((P) == 11 || (P) == 12) ? &TCCR1A : \
00071 (((P) == 10 || (P) ==  9) ? &TCCR2A : \
00072 (((P) ==  5 || (P) ==  2 || (P) ==  3) ? &TCCR3A : \
00073 (((P) ==  6 || (P) ==  7 || (P) ==  8) ? &TCCR4A : \
00074 (((P) == 46 || (P) == 45 || (P) == 44) ? &TCCR5A : 0))))))
00075 #define __digitalPinToTimerBit(P) \
00076 (((P) == 13) ? COM0A1 : (((P) ==  4) ? COM0B1 : \
00077 (((P) == 11) ? COM1A1 : (((P) == 12) ? COM1B1 : \
00078 (((P) == 10) ? COM2A1 : (((P) ==  9) ? COM2B1 : \
00079 (((P) ==  5) ? COM3A1 : (((P) ==  2) ? COM3B1 : (((P) ==  3) ? COM3C1 : \
00080 (((P) ==  6) ? COM4A1 : (((P) ==  7) ? COM4B1 : (((P) ==  8) ? COM4C1 : \
00081 (((P) == 46) ? COM5A1 : (((P) == 45) ? COM5B1 : COM5C1)))))))))))))))
00082
00083 #else
00084
00085 // Standard Arduino Pins
```

```
00086 #define digitalPinToPortReg(P) \
00087 (((P) >= 0 && (P) <= 7) ? &PORTD : (((P) >= 8 && (P) <= 13) ? &PORTB : &PORTC))
00088 #define digitalPinToDDRReg(P) \
00089 (((P) >= 0 && (P) <= 7) ? &DDRD : (((P) >= 8 && (P) <= 13) ? &DDRB : &DDRC))
00090 #define digitalPinToPINReg(P) \
00091 (((P) >= 0 && (P) <= 7) ? &PIND : (((P) >= 8 && (P) <= 13) ? &PINB : &PINC))
00092 #define __digitalPinToBit(P) \
00093 (((P) >= 0 && (P) <= 7) ? (P) : (((P) >= 8 && (P) <= 13) ? (P) - 8 : (P) - 14))
00094
00095 #if defined(__AVR_ATmega8__)
00096 // 3 PWM
00097 #define __digitalPinToTimer(P) \
00098 (((P) ==  9 || (P) == 10) ? &TCCR1A : (((P) == 11) ? &TCCR2 : 0))
00099 #define __digitalPinToTimerBit(P) \
00100 (((P) ==  9) ? COM1A1 : (((P) == 10) ? COM1B1 : COM21))
00101 #else  //168,328
00102
00103 // 6 PWM
00104 #define __digitalPinToTimer(P) \
00105 (((P) ==  6 || (P) ==  5) ? &TCCR0A : \
00106 (((P) ==  9 || (P) == 10) ? &TCCR1A : \
00107 (((P) == 11 || (P) ==  3) ? &TCCR2A : 0)))
00108 #define __digitalPinToTimerBit(P) \
00109 (((P) ==  6) ? COM0A1 : (((P) ==  5) ? COM0B1 : \
00110 (((P) ==  9) ? COM1A1 : (((P) == 10) ? COM1B1 : \
00111 (((P) == 11) ? COM2A1 : COM2B1)))))
00112 #endif  //defined(__AVR_ATmega8__)
00113
00114
00115 #endif  //mega
00116 #endif  //#if !defined(digitalPinToPortReg)
00117
00118
00119
00120
00121 #define __atomicWrite__(A,P,V) \
00122 if ( (int)(A) < 0x40) { bitWrite(*(A), __digitalPinToBit(P), (V) );}  \
00123 else {                                                                 \
00124 uint8_t register saveSreg = SREG;                                     \
00125 cli();                                                                \
00126 bitWrite(*(A), __digitalPinToBit(P), (V) );                           \
00127 SREG=saveSreg;                                                        \
00128 }
00129
00130
00131 #ifndef digitalWriteFast
00132 #define digitalWriteFast(P, V) \
00133 do {                          \
00134 if (__builtin_constant_p(P) && __builtin_constant_p(V))   __atomicWrite__((uint8_t*)
       digitalPinToPortReg(P),P,V) \
00135 else  digitalWrite((P), (V));         \
00136 }while (0)
00137 #endif  //#ifndef digitalWriteFast2
00138
00139 #if !defined(pinModeFast)
00140 #define pinModeFast(P, V) \
00141 do {if (__builtin_constant_p(P) && __builtin_constant_p(V)) __atomicWrite__((uint8_t*)
       digitalPinToDDRReg(P),P,V) \
00142 else pinMode((P), (V)); \
00143 } while (0)
00144 #endif
00145
00146
00147 #ifndef noAnalogWrite
00148 #define noAnalogWrite(P) \
00149     do {if (__builtin_constant_p(P) )  __atomicWrite((uint8_t*) __digitalPinToTimer(P),P,0) \
00150         else turnOffPWM((P));   \
00151 } while (0)
00152 #endif
00153
00154
00155 #ifndef digitalReadFast
00156     #define digitalReadFast(P) ( (int) _digitalReadFast_((P)) )
00157     #define _digitalReadFast_(P ) \
00158     (__builtin_constant_p(P) ) ? ( \
00159     ( BIT_READ(*digitalPinToPINReg(P), __digitalPinToBit(P))) ) : \
00160     digitalRead((P))
00161 #endif
00162
00163 #endif
```

5.19 from_kernel.h File Reference

```
#include <linux/init.h>
#include <linux/module.h>
#include <linux/moduleparam.h>
#include <linux/slab.h>
#include <linux/delay.h>
#include <linux/videodev.h>
#include <media/v4l2-common.h>
#include <linux/i2c.h>
```

Include dependency graph for from_kernel.h:



Classes

- struct ov7670_info
- struct regval_list
- struct ov7670_format_struct
- struct ov7670_win_size
- struct ov7670_control

Macros

- #define VGA_WIDTH 640
- #define VGA_HEIGHT 480
- #define QVGA_WIDTH 320
- #define QVGA_HEIGHT 240
- #define CIF_WIDTH 352
- #define CIF_HEIGHT 288
- #define QCIF_WIDTH 176
- #define QCIF_HEIGHT 144
- #define OV7670_FRAME_RATE 30
- #define OV7670_I2C_ADDR 0x42
- #define REG_GAIN 0x00 /∗ Gain lower 8 bits (rest in vref) ∗/
- #define REG_BLUE 0x01 /∗ blue gain ∗/
- #define REG_RED 0x02 /∗ red gain ∗/
- #define REG_VREF 0x03 /∗ Pieces of GAIN, VSTART, VSTOP ∗/
- #define REG_COM1 0x04 /∗ Control 1 ∗/
- #define COM1_CCIR656 0x40 /∗ CCIR656 enable ∗/
- #define REG_BAVE 0x05 /∗ U/B Average level ∗/
- #define REG_GbAVE 0x06 /∗ Y/Gb Average level ∗/
- #define REG_AECHH 0x07 /∗ AEC MS 5 bits ∗/
- #define REG_RAVE 0x08 /∗ V/R Average level ∗/
- #define REG_COM2 0x09 /∗ Control 2 ∗/
- #define COM2_SSLEEP 0x10 /∗ Soft sleep mode ∗/
- #define REG_PID 0x0a /∗ Product ID MSB ∗/
- #define REG_VER 0x0b /∗ Product ID LSB ∗/
- #define REG_COM3 0x0c /∗ Control 3 ∗/

- #define COM3_SWAP 0x40 /* Byte swap */
- #define COM3_SCALEEN 0x08 /* Enable scaling */
- #define COM3_DCWEN 0x04 /* Enable downsamp/crop/window */
- #define REG_COM4 0x0d /* Control 4 */
- #define REG_COM5 0x0e /* All "reserved" */
- #define REG_COM6 0x0f /* Control 6 */
- #define REG_AECH 0x10 /* More bits of AEC value */
- #define REG_CLKRC 0x11 /* Clocl control */
- #define CLK_EXT 0x40 /* Use external clock directly */
- #define CLK_SCALE 0x3f /* Mask for internal clock scale */
- #define REG_COM7 0x12 /* Control 7 */
- #define COM7_RESET 0x80 /* Register reset */
- #define COM7_FMT_MASK 0x38
- #define COM7_FMT_VGA 0x00
- #define COM7_FMT_CIF 0x20 /* CIF format */
- #define COM7_FMT_QVGA 0x10 /* QVGA format */
- #define COM7_FMT_QCIF 0x08 /* QCIF format */
- #define COM7_RGB 0x04 /* bits 0 and 2 - RGB format */
- #define COM7_YUV 0x00 /* YUV */
- #define COM7_BAYER 0x01 /* Bayer format */
- #define COM7_PBAYER 0x05 /* "Processed bayer" */
- #define REG_COM8 0x13 /* Control 8 */
- #define COM8_FASTAEC 0x80 /* Enable fast AGC/AEC */
- #define COM8_AECSTEP 0x40 /* Unlimited AEC step size */
- #define COM8_BFILT 0x20 /* Band filter enable */
- #define COM8_AGC 0x04 /* Auto gain enable */
- #define COM8_AWB 0x02 /* White balance enable */
- #define COM8_AEC 0x01 /* Auto exposure enable */
- #define REG_COM9 0x14 /* Control 9 - gain ceiling */
- #define REG_COM10 0x15 /* Control 10 */
- #define COM10_HSYNC 0x40 /* HSYNC instead of HREF */
- #define COM10_PCLK_HB 0x20 /* Suppress PCLK on horiz blank */
- #define COM10_HREF_REV 0x08 /* Reverse HREF */
- #define COM10_VS_LEAD 0x04 /* VSYNC on clock leading edge */
- #define COM10_VS_NEG 0x02 /* VSYNC negative */
- #define COM10_HS_NEG 0x01 /* HSYNC negative */
- #define REG_HSTART 0x17 /* Horiz start high bits */
- #define REG_HSTOP 0x18 /* Horiz stop high bits */
- #define REG_VSTART 0x19 /* Vert start high bits */
- #define REG_VSTOP 0x1a /* Vert stop high bits */
- #define REG_PSHFT 0x1b /* Pixel delay after HREF */
- #define REG_MIDH 0x1c /* Manuf. ID high */
- #define REG_MIDL 0x1d /* Manuf. ID low */
- #define REG_MVFP 0x1e /* Mirror / vflip */
- #define MVFP_MIRROR 0x20 /* Mirror image */
- #define MVFP_FLIP 0x10 /* Vertical flip */
- #define REG_AEW 0x24 /* AGC upper limit */
- #define REG_AEB 0x25 /* AGC lower limit */
- #define REG_VPT 0x26 /* AGC/AEC fast mode op region */
- #define REG_HSYST 0x30 /* HSYNC rising edge delay */
- #define REG_HSYEN 0x31 /* HSYNC falling edge delay */
- #define REG_HREF 0x32 /* HREF pieces */
- #define REG_TSLB 0x3a /* lots of stuff */
- #define TSLB_YLAST 0x04 /* UYVY or VYUY - see com13 */
- #define REG_COM11 0x3b /* Control 11 */

- #define COM11_NIGHT 0x80 /* NIght mode enable */
- #define COM11_NMFR 0x60 /* Two bit NM frame rate */
- #define COM11_HZAUTO 0x10 /* Auto detect 50/60 Hz */
- #define COM11_50HZ 0x08 /* Manual 50Hz select */
- #define COM11_EXP 0x02
- #define REG_COM12 0x3c /* Control 12 */
- #define COM12_HREF 0x80 /* HREF always */
- #define REG_COM13 0x3d /* Control 13 */
- #define COM13_GAMMA 0x80 /* Gamma enable */
- #define COM13_UVSAT 0x40 /* UV saturation auto adjustment */
- #define COM13_UVSWAP 0x01 /* V before U - w/TSLB */
- #define REG_COM14 0x3e /* Control 14 */
- #define COM14_DCWEN 0x10 /* DCW/PCLK-scale enable */
- #define REG_EDGE 0x3f /* Edge enhancement factor */
- #define REG_COM15 0x40 /* Control 15 */
- #define COM15_R10F0 0x00 /* Data range 10 to F0 */
- #define COM15_R01FE 0x80 /* 01 to FE */
- #define COM15_R00FF 0xc0 /* 00 to FF */
- #define COM15_RGB565 0x10 /* RGB565 output */
- #define COM15_RGB555 0x30 /* RGB555 output */
- #define REG_COM16 0x41 /* Control 16 */
- #define COM16_AWBGAIN 0x08 /* AWB gain enable */
- #define REG_COM17 0x42 /* Control 17 */
- #define COM17_AECWIN 0xc0 /* AEC window - must match COM4 */
- #define COM17_CBAR 0x08 /* DSP Color bar */
- #define REG_CMATRIX_BASE 0x4f
- #define CMATRIX_LEN 6
- #define REG_CMATRIX_SIGN 0x58
- #define REG_BRIGHT 0x55 /* Brightness */
- #define REG_CONTRAS 0x56 /* Contrast control */
- #define REG_GFIX 0x69 /* Fix gain control */
- #define REG_RGB444 0x8c /* RGB 444 control */
- #define R444_ENABLE 0x02 /* Turn on RGB444, overrides 5x5 */
- #define R444_RGBX 0x01 /* Empty nibble at end */
- #define REG_HAECC1 0x9f /* Hist AEC/AGC control 1 */
- #define REG_HAECC2 0xa0 /* Hist AEC/AGC control 2 */
- #define REG_BD50MAX 0xa5 /* 50hz banding step limit */
- #define REG_HAECC3 0xa6 /* Hist AEC/AGC control 3 */
- #define REG_HAECC4 0xa7 /* Hist AEC/AGC control 4 */
- #define REG_HAECC5 0xa8 /* Hist AEC/AGC control 5 */
- #define REG_HAECC6 0xa9 /* Hist AEC/AGC control 6 */
- #define REG_HAECC7 0xaa /* Hist AEC/AGC control 7 */
- #define REG_BD60MAX 0xab /* 60hz banding step limit */
- #define N_OV7670_FMTS (sizeof(ov7670_formats)/sizeof(ov7670_formats[0]))
- #define BYTES_PER_PIXEL 2
- #define N_WIN_SIZES (sizeof(ov7670_win_sizes)/sizeof(ov7670_win_sizes[0]))
- #define SIN_STEP 5
- #define N_CONTROLS (sizeof(ov7670_controls)/sizeof(ov7670_controls[0]))

Functions

- MODULE_AUTHOR ("Jonathan Corbet <corbet@lwn.net>")
- MODULE_DESCRIPTION ("A low-level driver for OmniVision ov7670 sensors")
- MODULE_LICENSE ("GPL")
- static int ov7670_read (struct i2c_client *c, unsigned char reg, unsigned char *value)
- static int ov7670_write (struct i2c_client *c, unsigned char reg, unsigned char value)
- static int ov7670_write_array (struct i2c_client *c, struct regval_list *vals)
- static void ov7670_reset (struct i2c_client *client)
- static int ov7670_init (struct i2c_client *client)
- static int ov7670_detect (struct i2c_client *client)
- static int ov7670_set_hw (struct i2c_client *client, int hstart, int hstop, int vstart, int vstop)
- static int ov7670_enum_fmt (struct i2c_client *c, struct v4l2_fmtdesc *fmt)
- static int ov7670_try_fmt (struct i2c_client *c, struct v4l2_format *fmt, struct ov7670_format_struct **ret_fmt, struct ov7670_win_size **ret_wsize)
- static int ov7670_s_fmt (struct i2c_client *c, struct v4l2_format *fmt)
- static int ov7670_g_parm (struct i2c_client *c, struct v4l2_streamparm *parms)
- static int ov7670_s_parm (struct i2c_client *c, struct v4l2_streamparm *parms)
- static int ov7670_store_cmatrix (struct i2c_client *client, int matrix[CMATRIX_LEN])
- static int ov7670_sine (int theta)
- static int ov7670_cosine (int theta)
- static void ov7670_calc_cmatrix (struct ov7670_info *info, int matrix[CMATRIX_LEN])
- static int ov7670_t_sat (struct i2c_client *client, int value)
- static int ov7670_q_sat (struct i2c_client *client, __s32 *value)
- static int ov7670_t_hue (struct i2c_client *client, int value)
- static int ov7670_q_hue (struct i2c_client *client, __s32 *value)
- static unsigned char ov7670_sm_to_abs (unsigned char v)
- static unsigned char ov7670_abs_to_sm (unsigned char v)
- static int ov7670_t_brightness (struct i2c_client *client, int value)
- static int ov7670_q_brightness (struct i2c_client *client, __s32 *value)
- static int ov7670_t_contrast (struct i2c_client *client, int value)
- static int ov7670_q_contrast (struct i2c_client *client, __s32 *value)
- static int ov7670_q_hflip (struct i2c_client *client, __s32 *value)
- static int ov7670_t_hflip (struct i2c_client *client, int value)
- static int ov7670_q_vflip (struct i2c_client *client, __s32 *value)
- static int ov7670_t_vflip (struct i2c_client *client, int value)
- static struct ov7670_control * ov7670_find_control (__u32 id)
- static int ov7670_queryctrl (struct i2c_client *client, struct v4l2_queryctrl *qc)
- static int ov7670_g_ctrl (struct i2c_client *client, struct v4l2_control *ctrl)
- static int ov7670_s_ctrl (struct i2c_client *client, struct v4l2_control *ctrl)
- static int ov7670_attach (struct i2c_adapter *adapter)
- static int ov7670_detach (struct i2c_client *client)
- static int ov7670_command (struct i2c_client *client, unsigned int cmd, void *arg)
- static int __init ov7670_mod_init (void)
- static void __exit ov7670_mod_exit (void)
- module_init (ov7670_mod_init)
- module_exit (ov7670_mod_exit)

Variables

- static struct regval_list ov7670_default_regs [ ]
- static struct regval_list ov7670_fmt_yuv422 [ ]
- static struct regval_list ov7670_fmt_rgb565 [ ]
- static struct regval_list ov7670_fmt_rgb444 [ ]
- static struct ov7670_format_struct ov7670_formats [ ]
- static struct regval_list ov7670_qcif_regs [ ]
- static struct ov7670_win_size ov7670_win_sizes [ ]
- static const int ov7670_sin_table [ ]
- static struct ov7670_control ov7670_controls [ ]
- static struct i2c_driver ov7670_driver

5.19.1 Macro Definition Documentation

5.19.1.1 #define BYTES_PER_PIXEL 2

Definition at line 511 of file from_kernel.h.

5.19.1.2 #define CIF_HEIGHT 288

Definition at line 34 of file from_kernel.h.

5.19.1.3 #define CIF_WIDTH 352

Definition at line 33 of file from_kernel.h.

5.19.1.4 #define CLK_EXT 0x40 /* Use external clock directly */

Definition at line 72 of file from_kernel.h.

5.19.1.5 #define CLK_SCALE 0x3f /* Mask for internal clock scale */

Definition at line 73 of file from_kernel.h.

5.19.1.6 #define CMATRIX_LEN 6

Definition at line 156 of file from_kernel.h.

5.19.1.7 #define COM10_HREF_REV 0x08 /* Reverse HREF */

Definition at line 96 of file from_kernel.h.

5.19.1.8 #define COM10_HS_NEG 0x01 /* HSYNC negative */

Definition at line 99 of file from_kernel.h.

5.19.1.9 #define COM10_HSYNC 0x40 /* HSYNC instead of HREF */

Definition at line 94 of file from_kernel.h.

5.19.1.10 #define COM10_PCLK_HB 0x20 /* Suppress PCLK on horiz blank */

Definition at line 95 of file from_kernel.h.

5.19.1.11 #define COM10_VS_LEAD 0x04 /* VSYNC on clock leading edge */

Definition at line 97 of file from_kernel.h.

5.19.1.12 #define COM10_VS_NEG 0x02 /* VSYNC negative */

Definition at line 98 of file from_kernel.h.

5.19.1.13 #define COM11_50HZ 0x08 /* Manual 50Hz select */

Definition at line 123 of file from_kernel.h.

5.19.1.14 #define COM11_EXP 0x02

Definition at line 124 of file from_kernel.h.

5.19.1.15 #define COM11_HZAUTO 0x10 /* Auto detect 50/60 Hz */

Definition at line 122 of file from_kernel.h.

5.19.1.16 #define COM11_NIGHT 0x80 /* NIght mode enable */

Definition at line 120 of file from_kernel.h.

5.19.1.17 #define COM11_NMFR 0x60 /* Two bit NM frame rate */

Definition at line 121 of file from_kernel.h.

5.19.1.18 #define COM12_HREF 0x80 /* HREF always */

Definition at line 126 of file from_kernel.h.

5.19.1.19 #define COM13_GAMMA 0x80 /* Gamma enable */

Definition at line 128 of file from_kernel.h.

5.19.1.20 #define COM13_UVSAT 0x40 /* UV saturation auto adjustment */

Definition at line 129 of file from_kernel.h.

5.19.1.21 #define COM13_UVSWAP 0x01 /* V before U - w/TSLB */

Definition at line 130 of file from_kernel.h.

5.19.1.22 #define COM14_DCWEN 0x10 /* DCW/PCLK-scale enable */

Definition at line 132 of file from_kernel.h.

5.19.1.23 #define COM15_R00FF 0xc0 /* 00 to FF */

Definition at line 137 of file from_kernel.h.

5.19.1.24 #define COM15_R01FE 0x80 /* 01 to FE */

Definition at line 136 of file from_kernel.h.

5.19.1.25 #define COM15_R10F0 0x00 /* Data range 10 to F0 */

Definition at line 135 of file from_kernel.h.

5.19.1.26 #define COM15_RGB555 0x30 /* RGB555 output */

Definition at line 139 of file from_kernel.h.

5.19.1.27 #define COM15_RGB565 0x10 /* RGB565 output */

Definition at line 138 of file from_kernel.h.

5.19.1.28 #define COM16_AWBGAIN 0x08 /* AWB gain enable */

Definition at line 141 of file from_kernel.h.

5.19.1.29 #define COM17_AECWIN 0xc0 /* AEC window - must match COM4 */

Definition at line 143 of file from_kernel.h.

5.19.1.30 #define COM17_CBAR 0x08 /* DSP Color bar */

Definition at line 144 of file from_kernel.h.

5.19.1.31 #define COM1_CCIR656 0x40 /* CCIR656 enable */

Definition at line 54 of file from_kernel.h.

5.19.1.32 #define COM2_SSLEEP 0x10 /* Soft sleep mode */

Definition at line 60 of file from_kernel.h.

5.19.1.33 #define COM3_DCWEN 0x04 /* Enable downsamp/crop/window */

Definition at line 66 of file from_kernel.h.

5.19.1.34 #define COM3_SCALEEN 0x08 /* Enable scaling */

Definition at line 65 of file from_kernel.h.

5.19.1.35 #define COM3_SWAP 0x40 /* Byte swap */

Definition at line 64 of file from_kernel.h.

5.19.1.36 #define COM7_BAYER 0x01 /* Bayer format */

Definition at line 83 of file from_kernel.h.

5.19.1.37 #define COM7_FMT_CIF 0x20 /* CIF format */

Definition at line 78 of file from_kernel.h.

5.19.1.38 #define COM7_FMT_MASK 0x38

Definition at line 76 of file from_kernel.h.

5.19.1.39 #define COM7_FMT_QCIF 0x08 /* QCIF format */

Definition at line 80 of file from_kernel.h.

5.19.1.40 #define COM7_FMT_QVGA 0x10 /* QVGA format */

Definition at line 79 of file from_kernel.h.

5.19.1.41 #define COM7_FMT_VGA 0x00

Definition at line 77 of file from_kernel.h.

5.19.1.42 #define COM7_PBAYER 0x05 /* "Processed bayer" */

Definition at line 84 of file from_kernel.h.

5.19.1.43 #define COM7_RESET 0x80 /* Register reset */

Definition at line 75 of file from_kernel.h.

5.19.1.44 #define COM7_RGB 0x04 /* bits 0 and 2 - RGB format */

Definition at line 81 of file from_kernel.h.

5.19.1.45 #define COM7_YUV 0x00 /* YUV */

Definition at line 82 of file from_kernel.h.

5.19.1.46 #define COM8_AEC 0x01 /* Auto exposure enable */

Definition at line 91 of file from_kernel.h.

5.19.1.47 #define COM8_AECSTEP 0x40 /* Unlimited AEC step size */

Definition at line 87 of file from_kernel.h.

5.19.1.48 #define COM8_AGC 0x04 /* Auto gain enable */

Definition at line 89 of file from_kernel.h.

5.19.1.49 #define COM8_AWB 0x02 /* White balance enable */

Definition at line 90 of file from_kernel.h.

5.19.1.50 #define COM8_BFILT 0x20 /* Band filter enable */

Definition at line 88 of file from_kernel.h.

5.19.1.51 #define COM8_FASTAEC 0x80 /* Enable fast AGC/AEC */

Definition at line 86 of file from_kernel.h.

5.19.1.52 #define MVFP_FLIP 0x10 /* Vertical flip */

Definition at line 109 of file from_kernel.h.

5.19.1.53 #define MVFP_MIRROR 0x20 /* Mirror image */

Definition at line 108 of file from_kernel.h.

5.19.1.54 #define N_CONTROLS (sizeof(ov7670_controls)/sizeof(ov7670_controls[0]))

Definition at line 1152 of file from_kernel.h.

5.19.1.55 #define N_OV7670_FMTS (sizeof(ov7670_formats)/sizeof(ov7670_formats[0]))

Definition at line 506 of file from_kernel.h.

5.19.1.56 #define N_WIN_SIZES (sizeof(ov7670_win_sizes)/sizeof(ov7670_win_sizes[0]))

Definition at line 599 of file from_kernel.h.

5.19.1.57 #define OV7670_FRAME_RATE 30

Definition at line 41 of file from_kernel.h.

5.19.1.58 #define OV7670_I2C_ADDR 0x42

Definition at line 46 of file from_kernel.h.

5.19.1.59 #define QCIF_HEIGHT 144

Definition at line 36 of file from_kernel.h.

5.19.1.60 #define QCIF_WIDTH 176

Definition at line 35 of file from_kernel.h.

5.19.1.61 #define QVGA_HEIGHT 240

Definition at line 32 of file from_kernel.h.

5.19.1.62 #define QVGA_WIDTH 320

Definition at line 31 of file from_kernel.h.

5.19.1.63 #define R444_ENABLE 0x02 /* Turn on RGB444, overrides 5x5 */

Definition at line 166 of file from_kernel.h.

5.19.1.64 #define R444_RGBX 0x01 /* Empty nibble at end */

Definition at line 167 of file from_kernel.h.

5.19.1.65 #define REG_AEB 0x25 /* AGC lower limit */

Definition at line 112 of file from_kernel.h.

5.19.1.66 #define REG_AECH 0x10 /* More bits of AEC value */

Definition at line 70 of file from_kernel.h.

5.19.1.67 #define REG_AECHH 0x07 /* AEC MS 5 bits */

Definition at line 57 of file from_kernel.h.

5.19.1.68 #define REG_AEW 0x24 /* AGC upper limit */

Definition at line 111 of file from_kernel.h.

5.19.1.69 #define REG_BAVE 0x05 /* U/B Average level */

Definition at line 55 of file from_kernel.h.

5.19.1.70 #define REG_BD50MAX 0xa5 /* 50hz banding step limit */

Definition at line 172 of file from_kernel.h.

5.19.1.71 #define REG_BD60MAX 0xab /* 60hz banding step limit */

Definition at line 178 of file from_kernel.h.

5.19.1.72 #define REG_BLUE 0x01 /* blue gain */

Definition at line 50 of file from_kernel.h.

5.19.1.73 #define REG_BRIGHT 0x55 /* Brightness */

Definition at line 160 of file from_kernel.h.

5.19.1.74 #define REG_CLKRC 0x11 /* Clocl control */

Definition at line 71 of file from_kernel.h.

5.19.1.75 #define REG_CMATRIX_BASE 0x4f

Definition at line 155 of file from_kernel.h.

5.19.1.76 #define REG_CMATRIX_SIGN 0x58

Definition at line 157 of file from_kernel.h.

5.19.1.77 #define REG_COM1 0x04 /* Control 1 */

Definition at line 53 of file from_kernel.h.

5.19.1.78 #define REG_COM10 0x15 /* Control 10 */

Definition at line 93 of file from_kernel.h.

5.19.1.79 #define REG_COM11 0x3b /* Control 11 */

Definition at line 119 of file from_kernel.h.

5.19.1.80 #define REG_COM12 0x3c /* Control 12 */

Definition at line 125 of file from_kernel.h.

5.19.1.81 #define REG_COM13 0x3d /* Control 13 */

Definition at line 127 of file from_kernel.h.

5.19.1.82 #define REG_COM14 0x3e /* Control 14 */

Definition at line 131 of file from_kernel.h.

5.19.1.83 #define REG_COM15 0x40 /* Control 15 */

Definition at line 134 of file from_kernel.h.

5.19.1.84 #define REG_COM16 0x41 /* Control 16 */

Definition at line 140 of file from_kernel.h.

5.19.1.85 #define REG_COM17 0x42 /* Control 17 */

Definition at line 142 of file from_kernel.h.

5.19.1.86 #define REG_COM2 0x09 /* Control 2 */

Definition at line 59 of file from_kernel.h.

5.19.1.87 #define REG_COM3 0x0c /* Control 3 */

Definition at line 63 of file from_kernel.h.

5.19.1.88 #define REG_COM4 0x0d /* Control 4 */

Definition at line 67 of file from_kernel.h.

5.19.1.89 #define REG_COM5 0x0e /* All "reserved" */

Definition at line 68 of file from_kernel.h.

5.19.1.90 #define REG_COM6 0x0f /* Control 6 */

Definition at line 69 of file from_kernel.h.

5.19.1.91 #define REG_COM7 0x12 /* Control 7 */

Definition at line 74 of file from_kernel.h.

5.19.1.92 #define REG_COM8 0x13 /* Control 8 */

Definition at line 85 of file from_kernel.h.

5.19.1.93 #define REG_COM9 0x14 /* Control 9 - gain ceiling */

Definition at line 92 of file from_kernel.h.

5.19.1.94 #define REG_CONTRAS 0x56 /* Contrast control */

Definition at line 161 of file from_kernel.h.

5.19.1.95 #define REG_EDGE 0x3f /* Edge enhancement factor */

Definition at line 133 of file from_kernel.h.

5.19.1.96 #define REG_GAIN 0x00 /* Gain lower 8 bits (rest in vref) */

Definition at line 49 of file from_kernel.h.

5.19.1.97 #define REG_GbAVE 0x06 /* Y/Gb Average level */

Definition at line 56 of file from_kernel.h.

5.19.1.98 #define REG_GFIX 0x69 /* Fix gain control */

Definition at line 163 of file from_kernel.h.

5.19.1.99 #define REG_HAECC1 0x9f /* Hist AEC/AGC control 1 */

Definition at line 169 of file from_kernel.h.

5.19.1.100 #define REG_HAECC2 0xa0 /* Hist AEC/AGC control 2 */

Definition at line 170 of file from_kernel.h.

5.19.1.101 #define REG_HAECC3 0xa6 /* Hist AEC/AGC control 3 */

Definition at line 173 of file from_kernel.h.

5.19.1.102 #define REG_HAECC4 0xa7 /* Hist AEC/AGC control 4 */

Definition at line 174 of file from_kernel.h.

5.19.1.103 #define REG_HAECC5 0xa8 /* Hist AEC/AGC control 5 */

Definition at line 175 of file from_kernel.h.

5.19.1.104 #define REG_HAECC6 0xa9 /* Hist AEC/AGC control 6 */

Definition at line 176 of file from_kernel.h.

5.19.1.105 #define REG_HAECC7 0xaa /* Hist AEC/AGC control 7 */

Definition at line 177 of file from_kernel.h.

5.19.1.106 #define REG_HREF 0x32 /* HREF pieces */

Definition at line 116 of file from_kernel.h.

5.19.1.107 #define REG_HSTART 0x17 /* Horiz start high bits */

Definition at line 100 of file from_kernel.h.

5.19.1.108 #define REG_HSTOP 0x18 /* Horiz stop high bits */

Definition at line 101 of file from_kernel.h.

5.19.1.109 #define REG_HSYEN 0x31 /* HSYNC falling edge delay */

Definition at line 115 of file from_kernel.h.

5.19.1.110 #define REG_HSYST 0x30 /* HSYNC rising edge delay */

Definition at line 114 of file from_kernel.h.

5.19.1.111 #define REG_MIDH 0x1c /* Manuf. ID high */

Definition at line 105 of file from_kernel.h.

5.19.1.112 #define REG_MIDL 0x1d /* Manuf. ID low */

Definition at line 106 of file from_kernel.h.

5.19.1.113 #define REG_MVFP 0x1e /* Mirror / vflip */

Definition at line 107 of file from_kernel.h.

5.19.1.114 #define REG_PID 0x0a /* Product ID MSB */

Definition at line 61 of file from_kernel.h.

5.19.1.115 #define REG_PSHFT 0x1b /* Pixel delay after HREF */

Definition at line 104 of file from_kernel.h.

5.19.1.116 #define REG_RAVE 0x08 /* V/R Average level */

Definition at line 58 of file from_kernel.h.

5.19.1.117 #define REG_RED 0x02 /* red gain */

Definition at line 51 of file from_kernel.h.

5.19.1.118 #define REG_RGB444 0x8c /* RGB 444 control */

Definition at line 165 of file from_kernel.h.

5.19.1.119 #define REG_TSLB 0x3a /* lots of stuff */

Definition at line 117 of file from_kernel.h.

5.19.1.120 #define REG_VER 0x0b /* Product ID LSB */

Definition at line 62 of file from_kernel.h.

5.19.1.121 #define REG_VPT 0x26 /* AGC/AEC fast mode op region */

Definition at line 113 of file from_kernel.h.

5.19.1.122 #define REG_VREF 0x03 /* Pieces of GAIN, VSTART, VSTOP */

Definition at line 52 of file from_kernel.h.

5.19.1.123 #define REG_VSTART 0x19 /* Vert start high bits */

Definition at line 102 of file from_kernel.h.

5.19.1.124 #define REG_VSTOP 0x1a /* Vert stop high bits */

Definition at line 103 of file from_kernel.h.

5.19.1.125 #define SIN_STEP 5

Definition at line 839 of file from_kernel.h.

5.19.1.126 #define TSLB_YLAST 0x04 /* UYVY or VYUY - see com13 */

Definition at line 118 of file from_kernel.h.

5.19.1.127 #define VGA_HEIGHT 480

Definition at line 30 of file from_kernel.h.

5.19.1.128 #define VGA_WIDTH 640

Definition at line 29 of file from_kernel.h.

5.19.2 Function Documentation

5.19.2.1 MODULE_AUTHOR ("Jonathan Corbet <corbet@lwn.net>")

5.19.2.2 MODULE_DESCRIPTION ("A low-level driver for OmniVision ov7670 sensors")

5.19.2.3 module_exit (ov7670_mod_exit)

5.19.2.4 module_init (ov7670_mod_init)

5.19.2.5 MODULE_LICENSE ("GPL")

5.19.2.6 static unsigned char ov7670_abs_to_sm (unsigned char *v*) [static]

Definition at line 964 of file from_kernel.h.

5.19.2.7 static int ov7670_attach (struct i2c_adapter * *adapter*) [static]

Definition at line 1212 of file from_kernel.h.

5.19.2.8 static void ov7670_calc_cmatrix (struct ov7670_info * *info*, int *matrix[CMATRIX_LEN]*) [static]

Definition at line 878 of file from_kernel.h.

5.19.2.9 static int ov7670_command (struct i2c_client * *client*, unsigned int *cmd*, void * *arg*) [static]

Definition at line 1269 of file from_kernel.h.

5.19.2.10 static int ov7670_cosine (int *theta*) [static]

Definition at line 865 of file from_kernel.h.

5.19.2.11 static int ov7670_detach (struct i2c_client * *client*) [static]

Definition at line 1260 of file from_kernel.h.

5.19.2.12 static int ov7670_detect (struct i2c_client * *client*) [static]

Definition at line 441 of file from_kernel.h.

5.19.2.13 static int ov7670_enum_fmt (struct i2c_client * *c*, struct v4l2_fmtdesc * *fmt*) [static]

Definition at line 634 of file from_kernel.h.

5.19.2.14 static struct ov7670_control* ov7670_find_control (__u32 *id*) [static]

Definition at line 1154 of file from_kernel.h.

5.19.2.15 static int ov7670_g_ctrl (struct i2c_client * *client*, struct v4l2_control * *ctrl*) [static]

Definition at line 1176 of file from_kernel.h.

5.19.2.16 static int ov7670_g_parm (struct i2c_client * *c*, struct v4l2_streamparm * *parms*) [static]

Definition at line 733 of file from_kernel.h.

5.19.2.17 static int ov7670_init (struct i2c_client * *client*) [static]

Definition at line 434 of file from_kernel.h.

5.19.2.18 static void __exit ov7670_mod_exit (void) [static]

Definition at line 1327 of file from_kernel.h.

5.19.2.19 static int __init ov7670_mod_init (void) [static]

Definition at line 1321 of file from_kernel.h.

5.19.2.20 static int ov7670_q_brightness (struct i2c_client * *client*, __s32 * *value*) [static]

Definition at line 985 of file from_kernel.h.

5.19.2.21 static int ov7670_q_contrast (struct i2c_client * *client*, __s32 * *value*) [static]

Definition at line 999 of file from_kernel.h.

5.19.2.22 static int ov7670_q_hflip (struct i2c_client * *client*, __s32 * *value*) [static]

Definition at line 1008 of file from_kernel.h.

5.19.2.23 static int ov7670_q_hue (struct i2c_client * *client*, __s32 * *value*) [static]

Definition at line 943 of file from_kernel.h.

5.19.2.24 static int ov7670_q_sat (struct i2c_client * *client*, __s32 * *value*) [static]

Definition at line 920 of file from_kernel.h.

5.19.2.25 static int ov7670_q_vflip (struct i2c_client * *client*, __s32 * *value*) [static]

Definition at line 1036 of file from_kernel.h.

5.19.2.26 static int ov7670_queryctrl (struct i2c_client * *client*, struct v4l2_queryctrl * *qc*) [static]

Definition at line 1165 of file from_kernel.h.

5.19.2.27 static int ov7670_read (struct i2c_client * *c*, unsigned char *reg*, unsigned char * *value*) [static]

Definition at line 390 of file from_kernel.h.

5.19.2.28 static void ov7670_reset (struct i2c_client * *client*) [static]

Definition at line 427 of file from_kernel.h.

5.19.2.29 static int ov7670_s_ctrl (struct i2c_client * *client*, struct v4l2_control * *ctrl*) [static]

Definition at line 1189 of file from_kernel.h.

5.19.2.30 static int ov7670_s_fmt (struct i2c_client * *c*, struct v4l2_format * *fmt*) [static]

Definition at line 696 of file from_kernel.h.

5.19.2.31 static int ov7670_s_parm (struct i2c_client * *c*, struct v4l2_streamparm * *parms*) [static]

Definition at line 753 of file from_kernel.h.

5.19.2.32 static int ov7670_set_hw (struct i2c_client * *client*, int *hstart*, int *hstop*, int *vstart*, int *vstop*) [static]

Definition at line 605 of file from_kernel.h.

5.19.2.33 static int ov7670_sine (int *theta*) [static]

Definition at line 847 of file from_kernel.h.

5.19.2.34 static unsigned char ov7670_sm_to_abs (unsigned char *v*) [static]

Definition at line 955 of file from_kernel.h.

5.19.2.35 static int ov7670_store_cmatrix (struct i2c_client * *client*, int *matrix[CMATRIX_LEN]*) [static]

Definition at line 794 of file from_kernel.h.

5.19.2.36 static int ov7670_t_brightness (struct i2c_client * *client*, int *value*) [static]

Definition at line 972 of file from_kernel.h.

5.19.2.37 static int ov7670_t_contrast (struct i2c_client * *client*, int *value*) [static]

Definition at line 994 of file from_kernel.h.

5.19.2.38 static int ov7670_t_hflip (struct i2c_client * *client*, int *value*) [static]

Definition at line 1019 of file from_kernel.h.

5.19.2.39 static int ov7670_t_hue (struct i2c_client * *client*, int *value*) [static]

Definition at line 928 of file from_kernel.h.

5.19.2.40 static int ov7670_t_sat (struct i2c_client * *client*, int *value*) [static]

Definition at line 908 of file from_kernel.h.

5.19.2.41 static int ov7670_t_vflip (struct i2c_client * *client*, int *value*) [static]

Definition at line 1047 of file from_kernel.h.

5.19.2.42 static int ov7670_try_fmt (struct i2c_client * *c*, struct v4l2_format * *fmt*, struct ov7670_format_struct ** *ret_fmt*, struct ov7670_win_size ** *ret_wsize*) [static]

Definition at line 649 of file from_kernel.h.

5.19.2.43 static int ov7670_write (struct i2c_client * *c*, unsigned char *reg*, unsigned char *value*) [static]

Definition at line 402 of file from_kernel.h.

5.19.2.44 static int ov7670_write_array (struct i2c_client * *c*, struct regval_list * *vals*) [static]

Definition at line 412 of file from_kernel.h.

5.19.3 Variable Documentation

5.19.3.1 struct ov7670_control ov7670_controls[ ] [static]

5.19.3.2 struct regval_list ov7670_default_regs[ ] [static]

Definition at line 207 of file from_kernel.h.

5.19.3.3 static struct i2c_driver ov7670_driver [static]

Initial value:

```
= {
    .driver = {
        .name = "ov7670",
    },
    .id         = I2C_DRIVERID_OV7670,
    .class      = I2C_CLASS_CAM_DIGITAL,
    .attach_adapter = ov7670_attach,
    .detach_client  = ov7670_detach,
    .command    = ov7670_command,
}
```

Definition at line 1210 of file from_kernel.h.

5.19.3.4 struct regval_list ov7670_fmt_rgb444[ ] [static]

Initial value:

```
= {
    { REG_COM7, COM7_RGB },
    { REG_RGB444, R444_ENABLE },
    { REG_COM1, 0x40 },
    { REG_COM15, COM15_R01FE|COM15_RGB565 },
    { REG_COM9, 0x38 },
    { 0x4f, 0xb3 },
    { 0x50, 0xb3 },
    { 0x51, 0    },
    { 0x52, 0x3d },
    { 0x53, 0xa7 },
    { 0x54, 0xe4 },
    { REG_COM13, COM13_GAMMA|COM13_UVSAT|0x2 },
    { 0xff, 0xff },
}
```

Definition at line 367 of file from_kernel.h.

5.19.3.5 struct regval_list ov7670_fmt_rgb565[ ] [static]

Initial value:

```
= {
    { REG_COM7, COM7_RGB },
    { REG_RGB444, 0 },
    { REG_COM1, 0x0 },
    { REG_COM15, COM15_RGB565 },
    { REG_COM9, 0x38 },
    { 0x4f, 0xb3 },
    { 0x50, 0xb3 },
    { 0x51, 0    },
    { 0x52, 0x3d },
    { 0x53, 0xa7 },
    { 0x54, 0xe4 },
    { REG_COM13, COM13_GAMMA|COM13_UVSAT },
    { 0xff, 0xff },
}
```

Definition at line 351 of file from_kernel.h.

5.19.3.6 struct regval_list ov7670_fmt_yuv422[ ] [static]

Initial value:

```
= {
    { REG_COM7, 0x0 },
    { REG_RGB444, 0 },
    { REG_COM1, 0 },
    { REG_COM15, COM15_R00FF },
    { REG_COM9, 0x18 },
    { 0x4f, 0x80 },
    { 0x50, 0x80 },
    { 0x51, 0    },
    { 0x52, 0x22 },
    { 0x53, 0x5e },
    { 0x54, 0x80 },
    { REG_COM13, COM13_GAMMA|COM13_UVSAT },
    { 0xff, 0xff },
}
```

Definition at line 335 of file from_kernel.h.

5.19.3.7 struct ov7670_format_struct ov7670_formats[ ] [static]

Initial value:

```
= {
    {
        .desc       = "YUYV 4:2:2",
        .pixelformat    = V4L2_PIX_FMT_YUYV,
```

```
        .regs       = ov7670_fmt_yuv422,
        .cmatrix    = { 128, -128, 0, -34, -94, 128 },
    },
    {
        .desc       = "RGB 444",
        .pixelformat    = V4L2_PIX_FMT_RGB444,
        .regs       = ov7670_fmt_rgb444,
        .cmatrix    = { 179, -179, 0, -61, -176, 228 },
    },
    {
        .desc       = "RGB 565",
        .pixelformat    = V4L2_PIX_FMT_RGB565,
        .regs       = ov7670_fmt_rgb565,
        .cmatrix    = { 179, -179, 0, -61, -176, 228 },
    },
}
```

5.19.3.8 struct regval_list ov7670_qcif_regs[ ] [static]

Initial value:

```
= {
    { REG_COM3, COM3_SCALEEN|COM3_DCWEN },
    { REG_COM3, COM3_DCWEN },
    { REG_COM14, COM14_DCWEN | 0x01},
    { 0x73, 0xf1 },
    { 0xa2, 0x52 },
    { 0x7b, 0x1c },
    { 0x7c, 0x28 },
    { 0x7d, 0x3c },
    { 0x7f, 0x69 },
    { REG_COM9, 0x38 },
    { 0xa1, 0x0b },
    { 0x74, 0x19 },
    { 0x9a, 0x80 },
    { 0x43, 0x14 },
    { REG_COM13, 0xc0 },
    { 0xff, 0xff },
}
```

Definition at line 523 of file from_kernel.h.

5.19.3.9 const int ov7670_sin_table[ ] [static]

Initial value:

```
= {
       0,    87,   173,   258,   342,   422,
     499,   573,   642,   707,   766,   819,
     866,   906,   939,   965,   984,   996,
    1000
}
```

Definition at line 840 of file from_kernel.h.

5.19.3.10 struct ov7670_win_size ov7670_win_sizes[ ] [static]

5.20 from_kernel.h

```
00001 /*
00002  * A V4L2 driver for OmniVision OV7670 cameras.
00003  *
00004  * Copyright 2006 One Laptop Per Child Association, Inc.  Written
00005  * by Jonathan Corbet with substantial inspiration from Mark
00006  * McClelland's ovcamchip code.
00007  *
00008  * This file may be distributed under the terms of the GNU General
00009  * Public License, version 2.
00010  */
00011 #include <linux/init.h>
00012 #include <linux/module.h>
00013 #include <linux/moduleparam.h>
00014 #include <linux/slab.h>
00015 #include <linux/delay.h>
00016 #include <linux/videodev.h>
00017 #include <media/v4l2-common.h>
```

```
00018 #include <linux/i2c.h>
00019
00020
00021 MODULE_AUTHOR("Jonathan Corbet <corbet@lwn.net>");
00022 MODULE_DESCRIPTION("A low-level driver for OmniVision ov7670 sensors");
00023 MODULE_LICENSE("GPL");
00024
00025 /*
00026  * Basic window sizes.  These probably belong somewhere more globally
00027  * useful.
00028  */
00029 #define VGA_WIDTH   640
00030 #define VGA_HEIGHT  480
00031 #define QVGA_WIDTH  320
00032 #define QVGA_HEIGHT 240
00033 #define CIF_WIDTH   352
00034 #define CIF_HEIGHT  288
00035 #define QCIF_WIDTH  176
00036 #define QCIF_HEIGHT 144
00037
00038 /*
00039  * Our nominal (default) frame rate.
00040  */
00041 #define OV7670_FRAME_RATE 30
00042
00043 /*
00044  * The 7670 sits on i2c with ID 0x42
00045  */
00046 #define OV7670_I2C_ADDR 0x42
00047
00048 /* Registers */
00049 #define REG_GAIN    0x00    /* Gain lower 8 bits (rest in vref) */
00050 #define REG_BLUE    0x01    /* blue gain */
00051 #define REG_RED     0x02    /* red gain */
00052 #define REG_VREF    0x03    /* Pieces of GAIN, VSTART, VSTOP */
00053 #define REG_COM1    0x04    /* Control 1 */
00054 #define  COM1_CCIR656     0x40  /* CCIR656 enable */
00055 #define REG_BAVE    0x05    /* U/B Average level */
00056 #define REG_GbAVE   0x06    /* Y/Gb Average level */
00057 #define REG_AECHH   0x07    /* AEC MS 5 bits */
00058 #define REG_RAVE    0x08    /* V/R Average level */
00059 #define REG_COM2    0x09    /* Control 2 */
00060 #define  COM2_SSLEEP      0x10  /* Soft sleep mode */
00061 #define REG_PID     0x0a    /* Product ID MSB */
00062 #define REG_VER     0x0b    /* Product ID LSB */
00063 #define REG_COM3    0x0c    /* Control 3 */
00064 #define  COM3_SWAP    0x40    /* Byte swap */
00065 #define  COM3_SCALEEN     0x08    /* Enable scaling */
00066 #define  COM3_DCWEN   0x04    /* Enable downsamp/crop/window */
00067 #define REG_COM4    0x0d    /* Control 4 */
00068 #define REG_COM5    0x0e    /* All "reserved" */
00069 #define REG_COM6    0x0f    /* Control 6 */
00070 #define REG_AECH    0x10    /* More bits of AEC value */
00071 #define REG_CLKRC   0x11    /* Clocl control */
00072 #define   CLK_EXT     0x40    /* Use external clock directly */
00073 #define   CLK_SCALE   0x3f    /* Mask for internal clock scale */
00074 #define REG_COM7    0x12    /* Control 7 */
00075 #define   COM7_RESET      0x80    /* Register reset */
00076 #define   COM7_FMT_MASK   0x38
00077 #define   COM7_FMT_VGA    0x00
00078 #define   COM7_FMT_CIF    0x20    /* CIF format */
00079 #define   COM7_FMT_QVGA   0x10    /* QVGA format */
00080 #define   COM7_FMT_QCIF   0x08    /* QCIF format */
00081 #define   COM7_RGB    0x04    /* bits 0 and 2 - RGB format */
00082 #define   COM7_YUV    0x00    /* YUV */
00083 #define   COM7_BAYER      0x01    /* Bayer format */
00084 #define   COM7_PBAYER     0x05    /* "Processed bayer" */
00085 #define REG_COM8    0x13    /* Control 8 */
00086 #define   COM8_FASTAEC    0x80    /* Enable fast AGC/AEC */
00087 #define   COM8_AECSTEP    0x40    /* Unlimited AEC step size */
00088 #define   COM8_BFILT      0x20    /* Band filter enable */
00089 #define   COM8_AGC    0x04    /* Auto gain enable */
00090 #define   COM8_AWB    0x02    /* White balance enable */
00091 #define   COM8_AEC    0x01    /* Auto exposure enable */
00092 #define REG_COM9    0x14    /* Control 9  - gain ceiling */
00093 #define REG_COM10   0x15    /* Control 10 */
00094 #define   COM10_HSYNC     0x40    /* HSYNC instead of HREF */
00095 #define   COM10_PCLK_HB   0x20    /* Suppress PCLK on horiz blank */
00096 #define   COM10_HREF_REV  0x08    /* Reverse HREF */
00097 #define   COM10_VS_LEAD   0x04    /* VSYNC on clock leading edge */
00098 #define   COM10_VS_NEG    0x02    /* VSYNC negative */
00099 #define   COM10_HS_NEG    0x01    /* HSYNC negative */
00100 #define REG_HSTART  0x17    /* Horiz start high bits */
00101 #define REG_HSTOP   0x18    /* Horiz stop high bits */
00102 #define REG_VSTART  0x19    /* Vert start high bits */
00103 #define REG_VSTOP   0x1a    /* Vert stop high bits */
00104 #define REG_PSHFT   0x1b    /* Pixel delay after HREF */
```

```
00105 #define REG_MIDH    0x1c    /* Manuf. ID high */
00106 #define REG_MIDL    0x1d    /* Manuf. ID low */
00107 #define REG_MVFP    0x1e    /* Mirror / vflip */
00108 #define   MVFP_MIRROR     0x20    /* Mirror image */
00109 #define   MVFP_FLIP   0x10    /* Vertical flip */
00110
00111 #define REG_AEW     0x24    /* AGC upper limit */
00112 #define REG_AEB     0x25    /* AGC lower limit */
00113 #define REG_VPT     0x26    /* AGC/AEC fast mode op region */
00114 #define REG_HSYST   0x30    /* HSYNC rising edge delay */
00115 #define REG_HSYEN   0x31    /* HSYNC falling edge delay */
00116 #define REG_HREF    0x32    /* HREF pieces */
00117 #define REG_TSLB    0x3a    /* lots of stuff */
00118 #define   TSLB_YLAST      0x04    /* UYVY or VYUY - see com13 */
00119 #define REG_COM11   0x3b    /* Control 11 */
00120 #define   COM11_NIGHT     0x80    /* NIght mode enable */
00121 #define   COM11_NMFR      0x60    /* Two bit NM frame rate */
00122 #define   COM11_HZAUTO    0x10    /* Auto detect 50/60 Hz */
00123 #define   COM11_50HZ      0x08    /* Manual 50Hz select */
00124 #define   COM11_EXP   0x02
00125 #define REG_COM12   0x3c    /* Control 12 */
00126 #define   COM12_HREF      0x80    /* HREF always */
00127 #define REG_COM13   0x3d    /* Control 13 */
00128 #define   COM13_GAMMA     0x80    /* Gamma enable */
00129 #define   COM13_UVSAT     0x40    /* UV saturation auto adjustment */
00130 #define   COM13_UVSWAP    0x01    /* V before U - w/TSLB */
00131 #define REG_COM14   0x3e    /* Control 14 */
00132 #define   COM14_DCWEN     0x10    /* DCW/PCLK-scale enable */
00133 #define REG_EDGE    0x3f    /* Edge enhancement factor */
00134 #define REG_COM15   0x40    /* Control 15 */
00135 #define   COM15_R10F0     0x00    /* Data range 10 to F0 */
00136 #define   COM15_R01FE     0x80    /*            01 to FE */
00137 #define   COM15_R00FF     0xc0    /*            00 to FF */
00138 #define   COM15_RGB565    0x10    /* RGB565 output */
00139 #define   COM15_RGB555    0x30    /* RGB555 output */
00140 #define REG_COM16   0x41    /* Control 16 */
00141 #define   COM16_AWBGAIN   0x08    /* AWB gain enable */
00142 #define REG_COM17   0x42    /* Control 17 */
00143 #define   COM17_AECWIN    0xc0    /* AEC window - must match COM4 */
00144 #define   COM17_CBAR      0x08    /* DSP Color bar */
00145
00146 /*
00147  * This matrix defines how the colors are generated, must be
00148  * tweaked to adjust hue and saturation.
00149  *
00150  * Order: v-red, v-green, v-blue, u-red, u-green, u-blue
00151  *
00152  * They are nine-bit signed quantities, with the sign bit
00153  * stored in 0x58.  Sign for v-red is bit 0, and up from there.
00154  */
00155 #define REG_CMATRIX_BASE 0x4f
00156 #define   CMATRIX_LEN 6
00157 #define REG_CMATRIX_SIGN 0x58
00158
00159
00160 #define REG_BRIGHT  0x55    /* Brightness */
00161 #define REG_CONTRAS 0x56    /* Contrast control */
00162
00163 #define REG_GFIX    0x69    /* Fix gain control */
00164
00165 #define REG_RGB444  0x8c    /* RGB 444 control */
00166 #define   R444_ENABLE     0x02    /* Turn on RGB444, overrides 5x5 */
00167 #define   R444_RGBX   0x01    /* Empty nibble at end */
00168
00169 #define REG_HAECC1  0x9f    /* Hist AEC/AGC control 1 */
00170 #define REG_HAECC2  0xa0    /* Hist AEC/AGC control 2 */
00171
00172 #define REG_BD50MAX 0xa5    /* 50hz banding step limit */
00173 #define REG_HAECC3  0xa6    /* Hist AEC/AGC control 3 */
00174 #define REG_HAECC4  0xa7    /* Hist AEC/AGC control 4 */
00175 #define REG_HAECC5  0xa8    /* Hist AEC/AGC control 5 */
00176 #define REG_HAECC6  0xa9    /* Hist AEC/AGC control 6 */
00177 #define REG_HAECC7  0xaa    /* Hist AEC/AGC control 7 */
00178 #define REG_BD60MAX 0xab    /* 60hz banding step limit */
00179
00180
00181 /*
00182  * Information we maintain about a known sensor.
00183  */
00184 struct ov7670_format_struct;  /* coming later */
00185 struct ov7670_info {
00186     struct ov7670_format_struct *fmt;  /* Current format */
00187     unsigned char sat;       /* Saturation value */
00188     int hue;         /* Hue value */
00189 };
00190
00191
```

```
00192
00193
00194 /*
00195  * The default register settings, as obtained from OmniVision.  There
00196  * is really no making sense of most of these - lots of "reserved" values
00197  * and such.
00198  *
00199  * These settings give VGA YUYV.
00200  */
00201
00202 struct regval_list {
00203     unsigned char reg_num;
00204     unsigned char value;
00205 };
00206
00207 static struct regval_list ov7670_default_regs[] = {
00208     { REG_COM7, COM7_RESET },
00209 /*
00210  * Clock scale: 3 = 15fps
00211  *              2 = 20fps
00212  *              1 = 30fps
00213  */
00214     { REG_CLKRC, 0x1 },     /* OV: clock scale (30 fps) */
00215     { REG_TSLB,  0x04 },     /* OV */
00216     { REG_COM7, 0 },    /* VGA */
00217     /*
00218      * Set the hardware window.  These values from OV don't entirely
00219      * make sense - hstop is less than hstart.  But they work...
00220      */
00221     { REG_HSTART, 0x13 }, { REG_HSTOP, 0x01 },
00222     { REG_HREF, 0xb6 }, { REG_VSTART, 0x02 },
00223     { REG_VSTOP, 0x7a },    { REG_VREF, 0x0a },
00224
00225     { REG_COM3, 0 },    { REG_COM14, 0 },
00226     /* Mystery scaling numbers */
00227     { 0x70, 0x3a },     { 0x71, 0x35 },
00228     { 0x72, 0x11 },     { 0x73, 0xf0 },
00229     { 0xa2, 0x02 },     { REG_COM10, 0x0 },
00230
00231     /* Gamma curve values */
00232     { 0x7a, 0x20 },     { 0x7b, 0x10 },
00233     { 0x7c, 0x1e },     { 0x7d, 0x35 },
00234     { 0x7e, 0x5a },     { 0x7f, 0x69 },
00235     { 0x80, 0x76 },     { 0x81, 0x80 },
00236     { 0x82, 0x88 },     { 0x83, 0x8f },
00237     { 0x84, 0x96 },     { 0x85, 0xa3 },
00238     { 0x86, 0xaf },     { 0x87, 0xc4 },
00239     { 0x88, 0xd7 },     { 0x89, 0xe8 },
00240
00241     /* AGC and AEC parameters.  Note we start by disabling those features,
00242        then turn them only after tweaking the values. */
00243     { REG_COM8, COM8_FASTAEC | COM8_AECSTEP |
    COM8_BFILT },
00244     { REG_GAIN, 0 },    { REG_AECH, 0 },
00245     { REG_COM4, 0x40 }, /* magic reserved bit */
00246     { REG_COM9, 0x18 }, /* 4x gain + magic rsvd bit */
00247     { REG_BD50MAX, 0x05 },    { REG_BD60MAX, 0x07 },
00248     { REG_AEW, 0x95 },    { REG_AEB, 0x33 },
00249     { REG_VPT, 0xe3 },    { REG_HAECC1, 0x78 },
00250     { REG_HAECC2, 0x68 }, { 0xa1, 0x03 }, /* magic */
00251     { REG_HAECC3, 0xd8 }, { REG_HAECC4, 0xd8 },
00252     { REG_HAECC5, 0xf0 }, { REG_HAECC6, 0x90 },
00253     { REG_HAECC7, 0x94 },
00254     { REG_COM8, COM8_FASTAEC|COM8_AECSTEP|
    COM8_BFILT|COM8_AGC|COM8_AEC },
00255
00256     /* Almost all of these are magic "reserved" values.  */
00257     { REG_COM5, 0x61 }, { REG_COM6, 0x4b },
00258     { 0x16, 0x02 },     { REG_MVFP, 0x07|MVFP_MIRROR },
00259     { 0x21, 0x02 },     { 0x22, 0x91 },
00260     { 0x29, 0x07 },     { 0x33, 0x0b },
00261     { 0x35, 0x0b },     { 0x37, 0x1d },
00262     { 0x38, 0x71 },     { 0x39, 0x2a },
00263     { REG_COM12, 0x78 },    { 0x4d, 0x40 },
00264     { 0x4e, 0x20 },     { REG_GFIX, 0 },
00265     { 0x6b, 0x4a },     { 0x74, 0x10 },
00266     { 0x8d, 0x4f },     { 0x8e, 0 },
00267     { 0x8f, 0 },        { 0x90, 0 },
00268     { 0x91, 0 },        { 0x96, 0 },
00269     { 0x9a, 0 },        { 0xb0, 0x84 },
00270     { 0xb1, 0x0c },     { 0xb2, 0x0e },
00271     { 0xb3, 0x82 },     { 0xb8, 0x0a },
00272
00273     /* More reserved magic, some of which tweaks white balance */
00274     { 0x43, 0x0a },     { 0x44, 0xf0 },
00275     { 0x45, 0x34 },     { 0x46, 0x58 },
00276     { 0x47, 0x28 },     { 0x48, 0x3a },
```

```
00277     { 0x59, 0x88 },      { 0x5a, 0x88 },
00278     { 0x5b, 0x44 },      { 0x5c, 0x67 },
00279     { 0x5d, 0x49 },      { 0x5e, 0x0e },
00280     { 0x6c, 0x0a },      { 0x6d, 0x55 },
00281     { 0x6e, 0x11 },      { 0x6f, 0x9f }, /* "9e for advance AWB" */
00282     { 0x6a, 0x40 },      { REG_BLUE, 0x40 },
00283     { REG_RED, 0x60 },
00284     { REG_COM8, COM8_FASTAEC|COM8_AECSTEP|
      COM8_BFILT|COM8_AGC|COM8_AEC|COM8_AWB },
00285
00286     /* Matrix coefficients */
00287     { 0x4f, 0x80 },      { 0x50, 0x80 },
00288     { 0x51, 0 },         { 0x52, 0x22 },
00289     { 0x53, 0x5e },      { 0x54, 0x80 },
00290     { 0x58, 0x9e },
00291
00292     { REG_COM16, COM16_AWBGAIN }, { REG_EDGE, 0 },
00293     { 0x75, 0x05 },      { 0x76, 0xe1 },
00294     { 0x4c, 0 },         { 0x77, 0x01 },
00295     { REG_COM13, 0xc3 },    { 0x4b, 0x09 },
00296     { 0xc9, 0x60 },      { REG_COM16, 0x38 },
00297     { 0x56, 0x40 },
00298
00299     { 0x34, 0x11 },      { REG_COM11, COM11_EXP|COM11_HZAUTO },
00300     { 0xa4, 0x88 },      { 0x96, 0 },
00301     { 0x97, 0x30 },      { 0x98, 0x20 },
00302     { 0x99, 0x30 },      { 0x9a, 0x84 },
00303     { 0x9b, 0x29 },      { 0x9c, 0x03 },
00304     { 0x9d, 0x4c },      { 0x9e, 0x3f },
00305     { 0x78, 0x04 },
00306
00307     /* Extra-weird stuff.  Some sort of multiplexor register */
00308     { 0x79, 0x01 },      { 0xc8, 0xf0 },
00309     { 0x79, 0x0f },      { 0xc8, 0x00 },
00310     { 0x79, 0x10 },      { 0xc8, 0x7e },
00311     { 0x79, 0x0a },      { 0xc8, 0x80 },
00312     { 0x79, 0x0b },      { 0xc8, 0x01 },
00313     { 0x79, 0x0c },      { 0xc8, 0x0f },
00314     { 0x79, 0x0d },      { 0xc8, 0x20 },
00315     { 0x79, 0x09 },      { 0xc8, 0x80 },
00316     { 0x79, 0x02 },      { 0xc8, 0xc0 },
00317     { 0x79, 0x03 },      { 0xc8, 0x40 },
00318     { 0x79, 0x05 },      { 0xc8, 0x30 },
00319     { 0x79, 0x26 },
00320
00321     { 0xff, 0xff }, /* END MARKER */
00322 };
00323
00324
00325 /*
00326  * Here we'll try to encapsulate the changes for just the output
00327  * video format.
00328  *
00329  * RGB656 and YUV422 come from OV; RGB444 is homebrewed.
00330  *
00331  * IMPORTANT RULE: the first entry must be for COM7, see ov7670_s_fmt for why.
00332  */
00333
00334
00335 static struct regval_list ov7670_fmt_yuv422[] = {
00336     { REG_COM7, 0x0 },  /* Selects YUV mode */
00337     { REG_RGB444, 0 },     /* No RGB444 please */
00338     { REG_COM1, 0 },
00339     { REG_COM15, COM15_R00FF },
00340     { REG_COM9, 0x18 }, /* 4x gain ceiling; 0x8 is reserved bit */
00341     { 0x4f, 0x80 },      /* "matrix coefficient 1" */
00342     { 0x50, 0x80 },      /* "matrix coefficient 2" */
00343     { 0x51, 0    },      /* vb */
00344     { 0x52, 0x22 },      /* "matrix coefficient 4" */
00345     { 0x53, 0x5e },      /* "matrix coefficient 5" */
00346     { 0x54, 0x80 },      /* "matrix coefficient 6" */
00347     { REG_COM13, COM13_GAMMA|COM13_UVSAT },
00348     { 0xff, 0xff },
00349 };
00350
00351 static struct regval_list ov7670_fmt_rgb565[] = {
00352     { REG_COM7, COM7_RGB }, /* Selects RGB mode */
00353     { REG_RGB444, 0 },     /* No RGB444 please */
00354     { REG_COM1, 0x0 },
00355     { REG_COM15, COM15_RGB565 },
00356     { REG_COM9, 0x38 },      /* 16x gain ceiling; 0x8 is reserved bit */
00357     { 0x4f, 0xb3 },      /* "matrix coefficient 1" */
00358     { 0x50, 0xb3 },      /* "matrix coefficient 2" */
00359     { 0x51, 0    },      /* vb */
00360     { 0x52, 0x3d },      /* "matrix coefficient 4" */
00361     { 0x53, 0xa7 },      /* "matrix coefficient 5" */
00362     { 0x54, 0xe4 },      /* "matrix coefficient 6" */
```

```
00363     { REG_COM13, COM13_GAMMA|COM13_UVSAT },
00364     { 0xff, 0xff },
00365 };
00366
00367 static struct regval_list ov7670_fmt_rgb444[] = {
00368     { REG_COM7, COM7_RGB }, /* Selects RGB mode */
00369     { REG_RGB444, R444_ENABLE },    /* Enable xxxxrrrr ggggbbbb */
00370     { REG_COM1, 0x40 }, /* Magic reserved bit */
00371     { REG_COM15, COM15_R01FE|COM15_RGB565 }, /* Data range needed? */
00372     { REG_COM9, 0x38 },      /* 16x gain ceiling; 0x8 is reserved bit */
00373     { 0x4f, 0xb3 },      /* "matrix coefficient 1" */
00374     { 0x50, 0xb3 },      /* "matrix coefficient 2" */
00375     { 0x51, 0    },      /* vb */
00376     { 0x52, 0x3d },      /* "matrix coefficient 4" */
00377     { 0x53, 0xa7 },      /* "matrix coefficient 5" */
00378     { 0x54, 0xe4 },      /* "matrix coefficient 6" */
00379     { REG_COM13, COM13_GAMMA|COM13_UVSAT|0x2 },  /* Magic rsvd bit */
00380     { 0xff, 0xff },
00381 };
00382
00383
00384
00385
00386 /*
00387  * Low-level register I/O.
00388  */
00389
00390 static int ov7670_read(struct i2c_client *c, unsigned char reg,
00391         unsigned char *value)
00392 {
00393     int ret;
00394
00395     ret = i2c_smbus_read_byte_data(c, reg);
00396     if (ret >= 0)
00397         *value = (unsigned char) ret;
00398     return ret;
00399 }
00400
00401
00402 static int ov7670_write(struct i2c_client *c, unsigned char reg,
00403         unsigned char value)
00404 {
00405     return i2c_smbus_write_byte_data(c, reg, value);
00406 }
00407
00408
00409 /*
00410  * Write a list of register settings; ff/ff stops the process.
00411  */
00412 static int ov7670_write_array(struct i2c_client *c, struct
    regval_list *vals)
00413 {
00414     while (vals->reg_num != 0xff || vals->value != 0xff) {
00415         int ret = ov7670_write(c, vals->reg_num, vals->value);
00416         if (ret < 0)
00417             return ret;
00418         vals++;
00419     }
00420     return 0;
00421 }
00422
00423
00424 /*
00425  * Stuff that knows about the sensor.
00426  */
00427 static void ov7670_reset(struct i2c_client *client)
00428 {
00429     ov7670_write(client, REG_COM7, COM7_RESET);
00430     msleep(1);
00431 }
00432
00433
00434 static int ov7670_init(struct i2c_client *client)
00435 {
00436     return ov7670_write_array(client, ov7670_default_regs);
00437 }
00438
00439
00440
00441 static int ov7670_detect(struct i2c_client *client)
00442 {
00443     unsigned char v;
00444     int ret;
00445
00446     ret = ov7670_init(client);
00447     if (ret < 0)
00448         return ret;
```

```
00449     ret = ov7670_read(client, REG_MIDH, &v);
00450     if (ret < 0)
00451         return ret;
00452     if (v != 0x7f) /* OV manuf. id. */
00453         return -ENODEV;
00454     ret = ov7670_read(client, REG_MIDL, &v);
00455     if (ret < 0)
00456         return ret;
00457     if (v != 0xa2)
00458         return -ENODEV;
00459     /*
00460      * OK, we know we have an OmniVision chip...but which one?
00461      */
00462     ret = ov7670_read(client, REG_PID, &v);
00463     if (ret < 0)
00464         return ret;
00465     if (v != 0x76)  /* PID + VER = 0x76 / 0x73 */
00466         return -ENODEV;
00467     ret = ov7670_read(client, REG_VER, &v);
00468     if (ret < 0)
00469         return ret;
00470     if (v != 0x73)  /* PID + VER = 0x76 / 0x73 */
00471         return -ENODEV;
00472     return 0;
00473 }
00474
00475
00476 /*
00477  * Store information about the video data format.  The color matrix
00478  * is deeply tied into the format, so keep the relevant values here.
00479  * The magic matrix nubmers come from OmniVision.
00480  */
00481 static struct ov7670_format_struct {
00482     __u8 *desc;
00483     __u32 pixelformat;
00484     struct regval_list *regs;
00485     int cmatrix[CMATRIX_LEN];
00486 } ov7670_formats[] = {
00487     {
00488         .desc       = "YUYV 4:2:2",
00489         .pixelformat    = V4L2_PIX_FMT_YUYV,
00490         .regs       = ov7670_fmt_yuv422,
00491         .cmatrix    = { 128, -128, 0, -34, -94, 128 },
00492     },
00493     {
00494         .desc       = "RGB 444",
00495         .pixelformat    = V4L2_PIX_FMT_RGB444,
00496         .regs       = ov7670_fmt_rgb444,
00497         .cmatrix    = { 179, -179, 0, -61, -176, 228 },
00498     },
00499     {
00500         .desc       = "RGB 565",
00501         .pixelformat    = V4L2_PIX_FMT_RGB565,
00502         .regs       = ov7670_fmt_rgb565,
00503         .cmatrix    = { 179, -179, 0, -61, -176, 228 },
00504     },
00505 };
00506 #define N_OV7670_FMTS (sizeof(ov7670_formats)/sizeof(ov7670_formats[0]))
00507
00508 /*
00509  * All formats we support are 2 bytes/pixel.
00510  */
00511 #define BYTES_PER_PIXEL 2
00512
00513 /*
00514  * Then there is the issue of window sizes.  Try to capture the info here.
00515  */
00516
00517 /*
00518  * QCIF mode is done (by OV) in a very strange way - it actually looks like
00519  * VGA with weird scaling options - they do *not* use the canned QCIF mode
00520  * which is allegedly provided by the sensor.  So here's the weird register
00521  * settings.
00522  */
00523 static struct regval_list ov7670_qcif_regs[] = {
00524     { REG_COM3, COM3_SCALEEN|COM3_DCWEN },
00525     { REG_COM3, COM3_DCWEN },
00526     { REG_COM14, COM14_DCWEN | 0x01},
00527     { 0x73, 0xf1 },
00528     { 0xa2, 0x52 },
00529     { 0x7b, 0x1c },
00530     { 0x7c, 0x28 },
00531     { 0x7d, 0x3c },
00532     { 0x7f, 0x69 },
00533     { REG_COM9, 0x38 },
00534     { 0xa1, 0x0b },
00535     { 0x74, 0x19 },
```

```
00536     { 0x9a, 0x80 },
00537     { 0x43, 0x14 },
00538     { REG_COM13, 0xc0 },
00539     { 0xff, 0xff },
00540 };
00541
00542 static struct ov7670_win_size {
00543     int width;
00544     int height;
00545     unsigned char com7_bit;
00546     int hstart;       /* Start/stop values for the camera.  Note */
00547     int hstop;     /* that they do not always make complete */
00548     int vstart;       /* sense to humans, but evidently the sensor */
00549     int vstop;     /* will do the right thing... */
00550     struct regval_list *regs; /* Regs to tweak */
00551 /* h/vref stuff */
00552 } ov7670_win_sizes[] = {
00553     /* VGA */
00554     {
00555         .width     = VGA_WIDTH,
00556         .height     = VGA_HEIGHT,
00557         .com7_bit   = COM7_FMT_VGA,
00558         .hstart     = 158,      /* These values from */
00559         .hstop      =  14,      /* Omnivision */
00560         .vstart     =  10,
00561         .vstop      = 490,
00562         .regs       = NULL,
00563     },
00564     /* CIF */
00565     {
00566         .width      = CIF_WIDTH,
00567         .height     = CIF_HEIGHT,
00568         .com7_bit   = COM7_FMT_CIF,
00569         .hstart     = 170,      /* Empirically determined */
00570         .hstop      =  90,
00571         .vstart     =  14,
00572         .vstop      = 494,
00573         .regs       = NULL,
00574     },
00575     /* QVGA */
00576     {
00577         .width      = QVGA_WIDTH,
00578         .height     = QVGA_HEIGHT,
00579         .com7_bit   = COM7_FMT_QVGA,
00580         .hstart     = 164,      /* Empirically determined */
00581         .hstop      =  20,
00582         .vstart     =  14,
00583         .vstop      = 494,
00584         .regs       = NULL,
00585     },
00586     /* QCIF */
00587     {
00588         .width      = QCIF_WIDTH,
00589         .height     = QCIF_HEIGHT,
00590         .com7_bit   = COM7_FMT_VGA, /* see comment above */
00591         .hstart     = 456,      /* Empirically determined */
00592         .hstop      =  24,
00593         .vstart     =  14,
00594         .vstop      = 494,
00595         .regs       = ov7670_qcif_regs,
00596     },
00597 };
00598
00599 #define N_WIN_SIZES (sizeof(ov7670_win_sizes)/sizeof(ov7670_win_sizes[0]))
00600
00601
00602 /*
00603  * Store a set of start/stop values into the camera.
00604  */
00605 static int ov7670_set_hw(struct i2c_client *client, int hstart, int hstop,
00606         int vstart, int vstop)
00607 {
00608     int ret;
00609     unsigned char v;
00610 /*
00611  * Horizontal: 11 bits, top 8 live in hstart and hstop.  Bottom 3 of
00612  * hstart are in href[2:0], bottom 3 of hstop in href[5:3].  There is
00613  * a mystery "edge offset" value in the top two bits of href.
00614  */
00615     ret =  ov7670_write(client, REG_HSTART, (hstart >> 3) & 0xff);
00616     ret += ov7670_write(client, REG_HSTOP, (hstop >> 3) & 0xff);
00617     ret += ov7670_read(client, REG_HREF, &v);
00618     v = (v & 0xc0) | ((hstop & 0x7) << 3) | (hstart & 0x7);
00619     msleep(10);
00620     ret += ov7670_write(client, REG_HREF, v);
00621 /*
00622  * Vertical: similar arrangement, but only 10 bits.
```

```
00623  */
00624     ret += ov7670_write(client, REG_VSTART, (vstart >> 2) & 0xff);
00625     ret += ov7670_write(client, REG_VSTOP, (vstop >> 2) & 0xff);
00626     ret += ov7670_read(client, REG_VREF, &v);
00627     v = (v & 0xf0) | ((vstop & 0x3) << 2) | (vstart & 0x3);
00628     msleep(10);
00629     ret += ov7670_write(client, REG_VREF, v);
00630     return ret;
00631 }
00632
00633
00634 static int ov7670_enum_fmt(struct i2c_client *c, struct v4l2_fmtdesc *fmt)
00635 {
00636     struct ov7670_format_struct *ofmt;
00637
00638     if (fmt->index >= N_OV7670_FMTS)
00639         return -EINVAL;
00640
00641     ofmt = ov7670_formats + fmt->index;
00642     fmt->flags = 0;
00643     strcpy(fmt->description, ofmt->desc);
00644     fmt->pixelformat = ofmt->pixelformat;
00645     return 0;
00646 }
00647
00648
00649 static int ov7670_try_fmt(struct i2c_client *c, struct v4l2_format *fmt,
00650         struct ov7670_format_struct **ret_fmt,
00651         struct ov7670_win_size **ret_wsize)
00652 {
00653     int index;
00654     struct ov7670_win_size *wsize;
00655     struct v4l2_pix_format *pix = &fmt->fmt.pix;
00656
00657     for (index = 0; index < N_OV7670_FMTS; index++)
00658         if (ov7670_formats[index].pixelformat == pix->pixelformat)
00659             break;
00660     if (index >= N_OV7670_FMTS)
00661         return -EINVAL;
00662     if (ret_fmt != NULL)
00663         *ret_fmt = ov7670_formats + index;
00664     /*
00665      * Fields: the OV devices claim to be progressive.
00666      */
00667     if (pix->field == V4L2_FIELD_ANY)
00668         pix->field = V4L2_FIELD_NONE;
00669     else if (pix->field != V4L2_FIELD_NONE)
00670         return -EINVAL;
00671     /*
00672      * Round requested image size down to the nearest
00673      * we support, but not below the smallest.
00674      */
00675     for (wsize = ov7670_win_sizes; wsize < ov7670_win_sizes +
      N_WIN_SIZES;
00676          wsize++)
00677         if (pix->width >= wsize->width && pix->height >= wsize->height)
00678             break;
00679     if (wsize >= ov7670_win_sizes + N_WIN_SIZES)
00680         wsize--;   /* Take the smallest one */
00681     if (ret_wsize != NULL)
00682         *ret_wsize = wsize;
00683     /*
00684      * Note the size we'll actually handle.
00685      */
00686     pix->width = wsize->width;
00687     pix->height = wsize->height;
00688     pix->bytesperline = pix->width*BYTES_PER_PIXEL;
00689     pix->sizeimage = pix->height*pix->bytesperline;
00690     return 0;
00691 }
00692
00693 /*
00694  * Set a format.
00695  */
00696 static int ov7670_s_fmt(struct i2c_client *c, struct v4l2_format *fmt)
00697 {
00698     int ret;
00699     struct ov7670_format_struct *ovfmt;
00700     struct ov7670_win_size *wsize;
00701     struct ov7670_info *info = i2c_get_clientdata(c);
00702     unsigned char com7;
00703
00704     ret = ov7670_try_fmt(c, fmt, &ovfmt, &wsize);
00705     if (ret)
00706         return ret;
00707     /*
00708      * COM7 is a pain in the ass, it doesn't like to be read then
```

```
00709      * quickly written afterward.  But we have everything we need
00710      * to set it absolutely here, as long as the format-specific
00711      * register sets list it first.
00712      */
00713     com7 = ovfmt->regs[0].value;
00714     com7 |= wsize->com7_bit;
00715     ov7670_write(c, REG_COM7, com7);
00716     /*
00717      * Now write the rest of the array.  Also store start/stops
00718      */
00719     ov7670_write_array(c, ovfmt->regs + 1);
00720     ov7670_set_hw(c, wsize->hstart, wsize->hstop, wsize->
      vstart,
00721               wsize->vstop);
00722     ret = 0;
00723     if (wsize->regs)
00724         ret = ov7670_write_array(c, wsize->regs);
00725     info->fmt = ovfmt;
00726     return 0;
00727 }
00728
00729 /*
00730  * Implement G/S_PARM.  There is a "high quality" mode we could try
00731  * to do someday; for now, we just do the frame rate tweak.
00732  */
00733 static int ov7670_g_parm(struct i2c_client *c, struct v4l2_streamparm *parms)
00734 {
00735     struct v4l2_captureparm *cp = &parms->parm.capture;
00736     unsigned char clkrc;
00737     int ret;
00738
00739     if (parms->type != V4L2_BUF_TYPE_VIDEO_CAPTURE)
00740         return -EINVAL;
00741     ret = ov7670_read(c, REG_CLKRC, &clkrc);
00742     if (ret < 0)
00743         return ret;
00744     memset(cp, 0, sizeof(struct v4l2_captureparm));
00745     cp->capability = V4L2_CAP_TIMEPERFRAME;
00746     cp->timeperframe.numerator = 1;
00747     cp->timeperframe.denominator = OV7670_FRAME_RATE;
00748     if ((clkrc & CLK_EXT) == 0 && (clkrc & CLK_SCALE) > 1)
00749         cp->timeperframe.denominator /= (clkrc & CLK_SCALE);
00750     return 0;
00751 }
00752
00753 static int ov7670_s_parm(struct i2c_client *c, struct v4l2_streamparm *parms)
00754 {
00755     struct v4l2_captureparm *cp = &parms->parm.capture;
00756     struct v4l2_fract *tpf = &cp->timeperframe;
00757     unsigned char clkrc;
00758     int ret, div;
00759
00760     if (parms->type != V4L2_BUF_TYPE_VIDEO_CAPTURE)
00761         return -EINVAL;
00762     if (cp->extendedmode != 0)
00763         return -EINVAL;
00764     /*
00765      * CLKRC has a reserved bit, so let's preserve it.
00766      */
00767     ret = ov7670_read(c, REG_CLKRC, &clkrc);
00768     if (ret < 0)
00769         return ret;
00770     if (tpf->numerator == 0 || tpf->denominator == 0)
00771         div = 1;  /* Reset to full rate */
00772     else
00773         div = (tpf->numerator*OV7670_FRAME_RATE)/tpf->denominator;
00774     if (div == 0)
00775         div = 1;
00776     else if (div > CLK_SCALE)
00777         div = CLK_SCALE;
00778     clkrc = (clkrc & 0x80) | div;
00779     tpf->numerator = 1;
00780     tpf->denominator = OV7670_FRAME_RATE/div;
00781     return ov7670_write(c, REG_CLKRC, clkrc);
00782 }
00783
00784
00785
00786 /*
00787  * Code for dealing with controls.
00788  */
00789
00790
00791
00792
00793
00794 static int ov7670_store_cmatrix(struct i2c_client *client,
```

```
00795         int matrix[CMATRIX_LEN])
00796 {
00797     int i, ret;
00798     unsigned char signbits;
00799
00800     /*
00801      * Weird crap seems to exist in the upper part of
00802      * the sign bits register, so let's preserve it.
00803      */
00804     ret = ov7670_read(client, REG_CMATRIX_SIGN, &signbits);
00805     signbits &= 0xc0;
00806
00807     for (i = 0; i < CMATRIX_LEN; i++) {
00808         unsigned char raw;
00809
00810         if (matrix[i] < 0) {
00811             signbits |= (1 << i);
00812             if (matrix[i] < -255)
00813                 raw = 0xff;
00814             else
00815                 raw = (-1 * matrix[i]) & 0xff;
00816         }
00817         else {
00818             if (matrix[i] > 255)
00819                 raw = 0xff;
00820             else
00821                 raw = matrix[i] & 0xff;
00822         }
00823         ret += ov7670_write(client, REG_CMATRIX_BASE + i, raw);
00824     }
00825     ret += ov7670_write(client, REG_CMATRIX_SIGN, signbits);
00826     return ret;
00827 }
00828
00829
00830 /*
00831  * Hue also requires messing with the color matrix.  It also requires
00832  * trig functions, which tend not to be well supported in the kernel.
00833  * So here is a simple table of sine values, 0-90 degrees, in steps
00834  * of five degrees.  Values are multiplied by 1000.
00835  *
00836  * The following naive approximate trig functions require an argument
00837  * carefully limited to -180 <= theta <= 180.
00838  */
00839 #define SIN_STEP 5
00840 static const int ov7670_sin_table[] = {
00841       0,    87,   173,   258,   342,   422,
00842     499,   573,   642,   707,   766,   819,
00843     866,   906,   939,   965,   984,   996,
00844     1000
00845 };
00846
00847 static int ov7670_sine(int theta)
00848 {
00849     int chs = 1;
00850     int sine;
00851
00852     if (theta < 0) {
00853         theta = -theta;
00854         chs = -1;
00855     }
00856     if (theta <= 90)
00857         sine = ov7670_sin_table[theta/SIN_STEP];
00858     else {
00859         theta -= 90;
00860         sine = 1000 - ov7670_sin_table[theta/SIN_STEP];
00861     }
00862     return sine*chs;
00863 }
00864
00865 static int ov7670_cosine(int theta)
00866 {
00867     theta = 90 - theta;
00868     if (theta > 180)
00869         theta -= 360;
00870     else if (theta < -180)
00871         theta += 360;
00872     return ov7670_sine(theta);
00873 }
00874
00875
00876
00877
00878 static void ov7670_calc_cmatrix(struct ov7670_info *info,
00879         int matrix[CMATRIX_LEN])
00880 {
00881     int i;
```

```
00882     /*
00883      * Apply the current saturation setting first.
00884      */
00885     for (i = 0; i < CMATRIX_LEN; i++)
00886         matrix[i] = (info->fmt->cmatrix[i]*info->sat) >> 7;
00887     /*
00888      * Then, if need be, rotate the hue value.
00889      */
00890     if (info->hue != 0) {
00891         int sinth, costh, tmpmatrix[CMATRIX_LEN];
00892
00893         memcpy(tmpmatrix, matrix, CMATRIX_LEN*sizeof(int));
00894         sinth = ov7670_sine(info->hue);
00895         costh = ov7670_cosine(info->hue);
00896
00897         matrix[0] = (matrix[3]*sinth + matrix[0]*costh)/1000;
00898         matrix[1] = (matrix[4]*sinth + matrix[1]*costh)/1000;
00899         matrix[2] = (matrix[5]*sinth + matrix[2]*costh)/1000;
00900         matrix[3] = (matrix[3]*costh - matrix[0]*sinth)/1000;
00901         matrix[4] = (matrix[4]*costh - matrix[1]*sinth)/1000;
00902         matrix[5] = (matrix[5]*costh - matrix[2]*sinth)/1000;
00903     }
00904 }
00905
00906
00907
00908 static int ov7670_t_sat(struct i2c_client *client, int value)
00909 {
00910     struct ov7670_info *info = i2c_get_clientdata(client);
00911     int matrix[CMATRIX_LEN];
00912     int ret;
00913
00914     info->sat = value;
00915     ov7670_calc_cmatrix(info, matrix);
00916     ret = ov7670_store_cmatrix(client, matrix);
00917     return ret;
00918 }
00919
00920 static int ov7670_q_sat(struct i2c_client *client, __s32 *value)
00921 {
00922     struct ov7670_info *info = i2c_get_clientdata(client);
00923
00924     *value = info->sat;
00925     return 0;
00926 }
00927
00928 static int ov7670_t_hue(struct i2c_client *client, int value)
00929 {
00930     struct ov7670_info *info = i2c_get_clientdata(client);
00931     int matrix[CMATRIX_LEN];
00932     int ret;
00933
00934     if (value < -180 || value > 180)
00935         return -EINVAL;
00936     info->hue = value;
00937     ov7670_calc_cmatrix(info, matrix);
00938     ret = ov7670_store_cmatrix(client, matrix);
00939     return ret;
00940 }
00941
00942
00943 static int ov7670_q_hue(struct i2c_client *client, __s32 *value)
00944 {
00945     struct ov7670_info *info = i2c_get_clientdata(client);
00946
00947     *value = info->hue;
00948     return 0;
00949 }
00950
00951
00952 /*
00953  * Some weird registers seem to store values in a sign/magnitude format!
00954  */
00955 static unsigned char ov7670_sm_to_abs(unsigned char v)
00956 {
00957     if ((v & 0x80) == 0)
00958         return v + 128;
00959     else
00960         return 128 - (v & 0x7f);
00961 }
00962
00963
00964 static unsigned char ov7670_abs_to_sm(unsigned char v)
00965 {
00966     if (v > 127)
00967         return v & 0x7f;
00968     else
```

```
00969         return (128 - v) | 0x80;
00970 }
00971
00972 static int ov7670_t_brightness(struct i2c_client *client, int value)
00973 {
00974     unsigned char com8, v;
00975     int ret;
00976
00977     ov7670_read(client, REG_COM8, &com8);
00978     com8 &= ~COM8_AEC;
00979     ov7670_write(client, REG_COM8, com8);
00980     v = ov7670_abs_to_sm(value);
00981     ret = ov7670_write(client, REG_BRIGHT, v);
00982     return ret;
00983 }
00984
00985 static int ov7670_q_brightness(struct i2c_client *client, __s32 *value)
00986 {
00987     unsigned char v;
00988     int ret = ov7670_read(client, REG_BRIGHT, &v);
00989
00990     *value = ov7670_sm_to_abs(v);
00991     return ret;
00992 }
00993
00994 static int ov7670_t_contrast(struct i2c_client *client, int value)
00995 {
00996     return ov7670_write(client, REG_CONTRAS, (unsigned char) value);
00997 }
00998
00999 static int ov7670_q_contrast(struct i2c_client *client, __s32 *value)
01000 {
01001     unsigned char v;
01002     int ret = ov7670_read(client, REG_CONTRAS, &v);
01003
01004     *value = v;
01005     return ret;
01006 }
01007
01008 static int ov7670_q_hflip(struct i2c_client *client, __s32 *value)
01009 {
01010     int ret;
01011     unsigned char v;
01012
01013     ret = ov7670_read(client, REG_MVFP, &v);
01014     *value = (v & MVFP_MIRROR) == MVFP_MIRROR;
01015     return ret;
01016 }
01017
01018
01019 static int ov7670_t_hflip(struct i2c_client *client, int value)
01020 {
01021     unsigned char v;
01022     int ret;
01023
01024     ret = ov7670_read(client, REG_MVFP, &v);
01025     if (value)
01026         v |= MVFP_MIRROR;
01027     else
01028         v &= ~MVFP_MIRROR;
01029     msleep(10);  /* FIXME */
01030     ret += ov7670_write(client, REG_MVFP, v);
01031     return ret;
01032 }
01033
01034
01035
01036 static int ov7670_q_vflip(struct i2c_client *client, __s32 *value)
01037 {
01038     int ret;
01039     unsigned char v;
01040
01041     ret = ov7670_read(client, REG_MVFP, &v);
01042     *value = (v & MVFP_FLIP) == MVFP_FLIP;
01043     return ret;
01044 }
01045
01046
01047 static int ov7670_t_vflip(struct i2c_client *client, int value)
01048 {
01049     unsigned char v;
01050     int ret;
01051
01052     ret = ov7670_read(client, REG_MVFP, &v);
01053     if (value)
01054         v |= MVFP_FLIP;
01055     else
```

```
01056         v &= ~MVFP_FLIP;
01057     msleep(10);  /* FIXME */
01058     ret += ov7670_write(client, REG_MVFP, v);
01059     return ret;
01060 }
01061
01062
01063 static struct ov7670_control {
01064     struct v4l2_queryctrl qc;
01065     int (*query)(struct i2c_client *c, __s32 *value);
01066     int (*tweak)(struct i2c_client *c, int value);
01067 } ov7670_controls[] =
01068 {
01069     {
01070         .qc = {
01071             .id = V4L2_CID_BRIGHTNESS,
01072             .type = V4L2_CTRL_TYPE_INTEGER,
01073             .name = "Brightness",
01074             .minimum = 0,
01075             .maximum = 255,
01076             .step = 1,
01077             .default_value = 0x80,
01078             .flags = V4L2_CTRL_FLAG_SLIDER
01079         },
01080         .tweak = ov7670_t_brightness,
01081         .query = ov7670_q_brightness,
01082     },
01083     {
01084         .qc = {
01085             .id = V4L2_CID_CONTRAST,
01086             .type = V4L2_CTRL_TYPE_INTEGER,
01087             .name = "Contrast",
01088             .minimum = 0,
01089             .maximum = 127,
01090             .step = 1,
01091             .default_value = 0x40,   /* XXX ov7670 spec */
01092             .flags = V4L2_CTRL_FLAG_SLIDER
01093         },
01094         .tweak = ov7670_t_contrast,
01095         .query = ov7670_q_contrast,
01096     },
01097     {
01098         .qc = {
01099             .id = V4L2_CID_SATURATION,
01100             .type = V4L2_CTRL_TYPE_INTEGER,
01101             .name = "Saturation",
01102             .minimum = 0,
01103             .maximum = 256,
01104             .step = 1,
01105             .default_value = 0x80,
01106             .flags = V4L2_CTRL_FLAG_SLIDER
01107         },
01108         .tweak = ov7670_t_sat,
01109         .query = ov7670_q_sat,
01110     },
01111     {
01112         .qc = {
01113             .id = V4L2_CID_HUE,
01114             .type = V4L2_CTRL_TYPE_INTEGER,
01115             .name = "HUE",
01116             .minimum = -180,
01117             .maximum = 180,
01118             .step = 5,
01119             .default_value = 0,
01120             .flags = V4L2_CTRL_FLAG_SLIDER
01121         },
01122         .tweak = ov7670_t_hue,
01123         .query = ov7670_q_hue,
01124     },
01125     {
01126         .qc = {
01127             .id = V4L2_CID_VFLIP,
01128             .type = V4L2_CTRL_TYPE_BOOLEAN,
01129             .name = "Vertical flip",
01130             .minimum = 0,
01131             .maximum = 1,
01132             .step = 1,
01133             .default_value = 0,
01134         },
01135         .tweak = ov7670_t_vflip,
01136         .query = ov7670_q_vflip,
01137     },
01138     {
01139         .qc = {
01140             .id = V4L2_CID_HFLIP,
01141             .type = V4L2_CTRL_TYPE_BOOLEAN,
01142             .name = "Horizontal mirror",
```

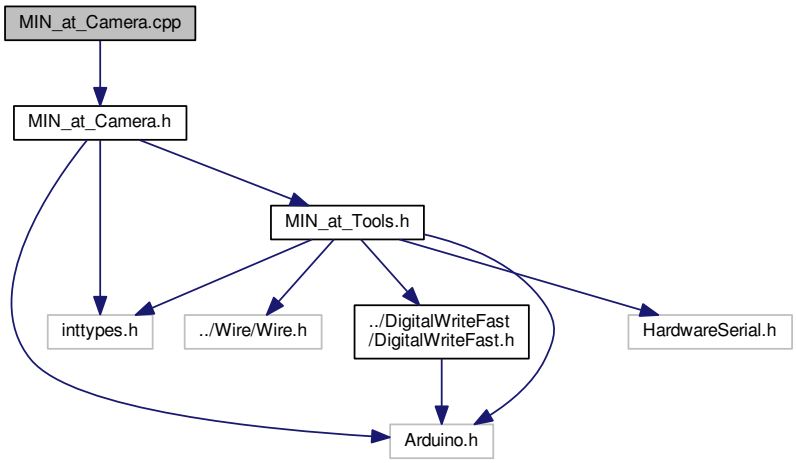
```
01143             .minimum = 0,
01144             .maximum = 1,
01145             .step = 1,
01146             .default_value = 0,
01147         },
01148         .tweak = ov7670_t_hflip,
01149         .query = ov7670_q_hflip,
01150     },
01151 };
01152 #define N_CONTROLS (sizeof(ov7670_controls)/sizeof(ov7670_controls[0]))
01153
01154 static struct ov7670_control *ov7670_find_control(__u32 id)
01155 {
01156     int i;
01157
01158     for (i = 0; i < N_CONTROLS; i++)
01159         if (ov7670_controls[i].qc.id == id)
01160             return ov7670_controls + i;
01161     return NULL;
01162 }
01163
01164
01165 static int ov7670_queryctrl(struct i2c_client *client,
01166         struct v4l2_queryctrl *qc)
01167 {
01168     struct ov7670_control *ctrl = ov7670_find_control(qc->id);
01169
01170     if (ctrl == NULL)
01171         return -EINVAL;
01172     *qc = ctrl->qc;
01173     return 0;
01174 }
01175
01176 static int ov7670_g_ctrl(struct i2c_client *client, struct v4l2_control *ctrl)
01177 {
01178     struct ov7670_control *octrl = ov7670_find_control(ctrl->id);
01179     int ret;
01180
01181     if (octrl == NULL)
01182         return -EINVAL;
01183     ret = octrl->query(client, &ctrl->value);
01184     if (ret >= 0)
01185         return 0;
01186     return ret;
01187 }
01188
01189 static int ov7670_s_ctrl(struct i2c_client *client, struct v4l2_control *ctrl)
01190 {
01191     struct ov7670_control *octrl = ov7670_find_control(ctrl->id);
01192     int ret;
01193
01194     if (octrl == NULL)
01195         return -EINVAL;
01196     ret =  octrl->tweak(client, ctrl->value);
01197     if (ret >= 0)
01198         return 0;
01199     return ret;
01200 }
01201
01202
01203
01204
01205
01206
01207 /*
01208  * Basic i2c stuff.
01209  */
01210 static struct i2c_driver ov7670_driver;
01211
01212 static int ov7670_attach(struct i2c_adapter *adapter)
01213 {
01214     int ret;
01215     struct i2c_client *client;
01216     struct ov7670_info *info;
01217
01218     /*
01219      * For now: only deal with adapters we recognize.
01220      */
01221     if (adapter->id != I2C_HW_SMBUS_CAFE)
01222         return -ENODEV;
01223
01224     client = kzalloc(sizeof (struct i2c_client), GFP_KERNEL);
01225     if (! client)
01226         return -ENOMEM;
01227     client->adapter = adapter;
01228     client->addr = OV7670_I2C_ADDR;
01229     client->driver = &ov7670_driver,
```

```
01230     strcpy(client->name, "OV7670");
01231     /*
01232      * Set up our info structure.
01233      */
01234     info = kzalloc(sizeof (struct ov7670_info), GFP_KERNEL);
01235     if (! info) {
01236         ret = -ENOMEM;
01237         goto out_free;
01238     }
01239     info->fmt = &ov7670_formats[0];
01240     info->sat = 128; /* Review this */
01241     i2c_set_clientdata(client, info);
01242
01243     /*
01244      * Make sure it's an ov7670
01245      */
01246     ret = ov7670_detect(client);
01247     if (ret)
01248         goto out_free_info;
01249     i2c_attach_client(client);
01250     return 0;
01251
01252   out_free_info:
01253     kfree(info);
01254   out_free:
01255     kfree(client);
01256     return ret;
01257 }
01258
01259
01260 static int ov7670_detach(struct i2c_client *client)
01261 {
01262     i2c_detach_client(client);
01263     kfree(i2c_get_clientdata(client));
01264     kfree(client);
01265     return 0;
01266 }
01267
01268
01269 static int ov7670_command(struct i2c_client *client, unsigned int cmd,
01270         void *arg)
01271 {
01272     switch (cmd) {
01273     case VIDIOC_INT_G_CHIP_IDENT:
01274         * (enum v4l2_chip_ident *) arg = V4L2_IDENT_OV7670;
01275         return 0;
01276
01277     case VIDIOC_INT_RESET:
01278         ov7670_reset(client);
01279         return 0;
01280
01281     case VIDIOC_INT_INIT:
01282         return ov7670_init(client);
01283
01284     case VIDIOC_ENUM_FMT:
01285         return ov7670_enum_fmt(client, (struct v4l2_fmtdesc *) arg);
01286     case VIDIOC_TRY_FMT:
01287         return ov7670_try_fmt(client, (struct v4l2_format *) arg, NULL, NULL);
01288     case VIDIOC_S_FMT:
01289         return ov7670_s_fmt(client, (struct v4l2_format *) arg);
01290     case VIDIOC_QUERYCTRL:
01291         return ov7670_queryctrl(client, (struct v4l2_queryctrl *) arg);
01292     case VIDIOC_S_CTRL:
01293         return ov7670_s_ctrl(client, (struct v4l2_control *) arg);
01294     case VIDIOC_G_CTRL:
01295         return ov7670_g_ctrl(client, (struct v4l2_control *) arg);
01296     case VIDIOC_S_PARM:
01297         return ov7670_s_parm(client, (struct v4l2_streamparm *) arg);
01298     case VIDIOC_G_PARM:
01299         return ov7670_g_parm(client, (struct v4l2_streamparm *) arg);
01300     }
01301     return -EINVAL;
01302 }
01303
01304
01305
01306 static struct i2c_driver ov7670_driver = {
01307     .driver = {
01308         .name = "ov7670",
01309     },
01310     .id         = I2C_DRIVERID_OV7670,
01311     .class      = I2C_CLASS_CAM_DIGITAL,
01312     .attach_adapter = ov7670_attach,
01313     .detach_client  = ov7670_detach,
01314     .command    = ov7670_command,
01315 };
01316
```

```
01317
01318 /*
01319  * Module initialization
01320  */
01321 static int __init ov7670_mod_init(void)
01322 {
01323     printk(KERN_NOTICE "OmniVision ov7670 sensor driver, at your service\n");
01324     return i2c_add_driver(&ov7670_driver);
01325 }
01326
01327 static void __exit ov7670_mod_exit(void)
01328 {
01329     i2c_del_driver(&ov7670_driver);
01330 }
01331
01332 module_init(ov7670_mod_init);
01333 module_exit(ov7670_mod_exit);
```

5.21 MIN_at_Camera.cpp File Reference

`#include "MIN_at_Camera.h"`
Include dependency graph for MIN_at_Camera.cpp:



Variables

- Camera Cam = Camera()

5.21.1 Variable Documentation

5.21.1.1 Camera Cam = Camera()

Definition at line 358 of file MIN_at_Camera.cpp.

5.22 MIN_at_Camera.cpp

```
00001 // ArduinoCam Version 1.0 (2012-09-01) Firmware (Duemilanove Atmega168)
00002 // Copyright 2012 richard.prinz@min.at
00003 // See http://www.min.at/prinz/oe1rib/ArduinoCam/ for more infos
00004 //
00005 // This file is part of ArduinoCam
00006 //
00007 // ArduinoCam is free software and hardware design:
```

```
00008 // you can redistribute the software and the hardware design and/or modify it under
00009 // the terms of the GNU General Public License as published by the Free Software Foundation,
00010 // either version 3 of the License, or (at your option) any later version.
00011 //
00012 // ArduinoCam is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY;
00013 // without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
00014 // See the GNU General Public License for more details.
00015 //
00016 // You should have received a copy of the GNU General Public License along with ArduinoCam.
00017 // If not, see http://www.gnu.org/licenses/.
00018
00019 #include "MIN_at_Camera.h"
00020
00021
00022
00023 // OV7620 Arduino camera shield
00024
00025
00026
00027 // Constructors ////////////////////////////////////////////////////////////////
00028
00029 Camera::Camera()
00030 {
00031 }
00032
00033
00034
00035 // Public Methods //////////////////////////////////////////////////////////////
00036
00037 void Camera::Begin()
00038 {
00039     Begin(OV_7620_ADDR);
00040 }
00041
00042 void Camera::Begin(byte address)
00043 {
00044     Begin((int)address);
00045 }
00046
00047 void Camera::Begin(int address)
00048 {
00049     _addr = address;
00050
00051     DDRD = B00000000;
00052     DDRC = B00000000;
00053
00054     pinModeFast(BUS_RRST, OUTPUT);
00055     pinModeFast(BUS_RCK, OUTPUT);
00056
00057     digitalWriteFast(BUS_RRST, HIGH);
00058     digitalWriteFast(BUS_RCK, LOW);
00059 }
00060
00061 bool Camera::Reset()
00062 {
00063     byte b = 0;
00064
00065     // (0x12) initiate camera reset
00066     Tools::I2C_WriteValue(_addr, OV_COMMON_A,
      I2C_SHORT_ADDR, 0x80, 300);
00067
00068     // try to read manufacturer id 0x7FA2 from camera
00069     Tools::I2C_Write(_addr, OV_ID_H, I2C_SHORT_ADDR);
00070     Wire.requestFrom(_addr, 2);
00071
00072     if(Wire.available())
00073     {
00074         b = Wire.read();
00075         if(b != 0x7F)
00076             return false;
00077     }
00078     else
00079         return false;
00080
00081     if(Wire.available())
00082     {
00083         b = Wire.read();
00084         if(b != 0xA2)
00085             return false;
00086     }
00087     else
00088         return false;
00089
00090     return true;
00091 }
00092
00093 void Camera::Init()
```

```
00094 {
00095     ResetVideoPointer();
00096 
00097     // (0x11) HSYNC-neg, CHSYNC-neg, VSYNC-pos
00098     //Tools::I2C_WriteValue(_addr, OV_CLOCK_RATE, I2C_SHORT_ADDR, 0x00, 10);
00099 
00100     // (0x12) Mirror image, auto white balance, AGC
00101     //Tools::I2C_WriteValue(_addr, OV_COMMON_A, I2C_SHORT_ADDR, 0x64, 10);
00102 
00103     // (0x13) set 8bit mode, enable bus, auto adjust
00104     Tools::I2C_WriteValue(_addr, OV_COMMON_B,
      I2C_SHORT_ADDR, 0x21, 10);
00105 
00106 
00107     // (0x14) set 640 x 480 VGA mode
00108     Tools::I2C_WriteValue(_addr, OV_COMMON_C,
      I2C_SHORT_ADDR, 0x04, 10);
00109     // (0x14) set 320 x 240 QVGA mode
00110     //Tools::I2C_WriteValue(_addr, OV_COMMON_C, I2C_SHORT_ADDR, 0x24, 10);
00111 
00112 
00113     // (0x15) set UYVY for 8 bit output
00114     Tools::I2C_WriteValue(_addr, OV_COMMON_D,
      I2C_SHORT_ADDR, 0x01, 10);
00115 
00116     // (0x16) field mode selection off
00117     Tools::I2C_WriteValue(_addr, OV_FRAME_DROP,
      I2C_SHORT_ADDR, 0x00, 10);
00118 
00119     // (0x27) set SRAM control
00120     Tools::I2C_WriteValue(_addr, OV_COMMON_G,
      I2C_SHORT_ADDR, 0xE3, 10);
00121 
00122 
00123     // (0x28) set interlaced scan mode
00124     Tools::I2C_WriteValue(_addr, OV_COMMON_H,
      I2C_SHORT_ADDR, 0x00, 10);
00125     // (0x28) set progressive scan mode
00126     //Tools::I2C_WriteValue(_addr, OV_COMMON_H, I2C_SHORT_ADDR, 0x20, 10);
00127 
00128 
00129     // (0x67) set color space
00130     // 0x1A = YUV, 0x5A = Analog YUV, 0x9A = CCIR 601 YCrCb
00131     // 0xDA = PAL YUV
00132     //Tools::I2C_WriteValue(_addr, OV_COLOR_SPACE, I2C_SHORT_ADDR, 0x1A, 10);
00133 }
00134 
00135 void Camera::ColorBar(bool Value)
00136 {
00137     Tools::I2C_SetBitAt(_addr, OV_COMMON_A,
      I2C_SHORT_ADDR, 1, Value, 0);
00138 }
00139 
00140 void Camera::Power(bool Value)
00141 {
00142     Tools::I2C_SetBitAt(_addr, OV_COMMON_O,
      I2C_SHORT_ADDR, 5, Value, 0);
00143 }
00144 
00145 void Camera::Mirror(bool Value)
00146 {
00147     Tools::I2C_SetBitAt(_addr, OV_COMMON_A,
      I2C_SHORT_ADDR, 6, Value, 0);
00148 }
00149 
00150 bool Camera::Capture()
00151 {
00152     byte b = 0;
00153     bool status = false;
00154     int i = 0;
00155 
00156     // (0x14) set 320 x 240 QVGA mode
00157     Tools::I2C_WriteValue(_addr, OV_COMMON_C,
      I2C_SHORT_ADDR, 0x24, 10);
00158 
00159     // (0x28) set progressive scan mode
00160     Tools::I2C_WriteValue(_addr, OV_COMMON_H,
      I2C_SHORT_ADDR, 0x20, 10);
00161 
00162     // start single frame transfer
00163     Tools::I2C_WriteValue(_addr, OV_COMMON_B,
      I2C_SHORT_ADDR, 0x23, 100);
00164 
00165 again:
00166     i++;
00167     delay(5);
00168 
```

```
00169     b = Cam.ReadConfigByte(OV_COMMON_B);
00170     if( (b & 0x02) == 0x00 )
00171     {
00172         status = true;
00173         i = 11;
00174     }
00175
00176     if(i < 10)
00177         goto again;
00178
00179
00180     // (0x14) set 640 x 480 VGA mode
00181     Tools::I2C_WriteValue(_addr, OV_COMMON_C,
      I2C_SHORT_ADDR, 0x04, 10);
00182
00183     // (0x28) set interlaced scan mode
00184     Tools::I2C_WriteValue(_addr, OV_COMMON_H,
      I2C_SHORT_ADDR, 0x00, 10);
00185
00186     return status;
00187 }
00188
00189 void Camera::Dump(bool Hex)
00190 {
00191     byte cntr = 0;
00192     byte b = 0;
00193
00194     ResetVideoPointer();
00195
00196     for(int lc = 0; lc < 240; lc++)
00197     {
00198         for(int pc = 0; pc < 640; pc++)
00199         {
00200             // read U/V, Y
00201             b = ReadNextVideoByte();
00202
00203             if(Hex)
00204                 DumpVideoByte(b, &(++cntr));
00205             else
00206                 Serial.write(b);
00207         }
00208     }
00209
00210     digitalWriteFast(BUS_RCK, LOW);
00211     Serial.println();
00212 }
00213
00214 void Camera::DumpConfig()
00215 {
00216     byte b = 0;
00217
00218     // (0x11) HSYNC-neg, CHSYNC-neg, VSYNC-pos
00219     Serial.print("OV_CLOCK_RATE (0x11: 0x00): ");
00220     b = Cam.ReadConfigByte(OV_CLOCK_RATE);
00221     Camera::DebugPrintValue(b);
00222
00223     // (0x12) mirror image, testpattern, reset
00224     Serial.print("OV_COMMON_A   (0x12: 0x64): ");
00225     b = Cam.ReadConfigByte(OV_COMMON_A);
00226     Camera::DebugPrintValue(b);
00227
00228     // (0x13) set 8bit mode, enable bus, auto adjust
00229     Serial.print("OV_COMMON_B   (0x13: 0x21): ");
00230     b = Cam.ReadConfigByte(OV_COMMON_B);
00231     Camera::DebugPrintValue(b);
00232
00233     // (0x14) set 320 x 240 QVGA mode
00234     Serial.print("OV_COMMON_C   (0x14: 0x24): ");
00235     b = Cam.ReadConfigByte(OV_COMMON_C);
00236     Camera::DebugPrintValue(b);
00237
00238     // (0x16) field mode selection off
00239     Serial.print("OV_FRAME_DROP (0x16: 0x00): ");
00240     b = Cam.ReadConfigByte(OV_FRAME_DROP);
00241     Camera::DebugPrintValue(b);
00242
00243     // (0x27) set SRAM control
00244     Serial.print("OV_COMMON_G   (0x27: 0xE3): ");
00245     b = Cam.ReadConfigByte(OV_COMMON_G);
00246     Camera::DebugPrintValue(b);
00247
00248     // (0x28) set progressive scan mode
00249     Serial.print("OV_COMMON_H   (0x28: 0x20): ");
00250     b = Cam.ReadConfigByte(OV_COMMON_H);
00251     Camera::DebugPrintValue(b);
00252
00253     // (0x67) color space selection
```

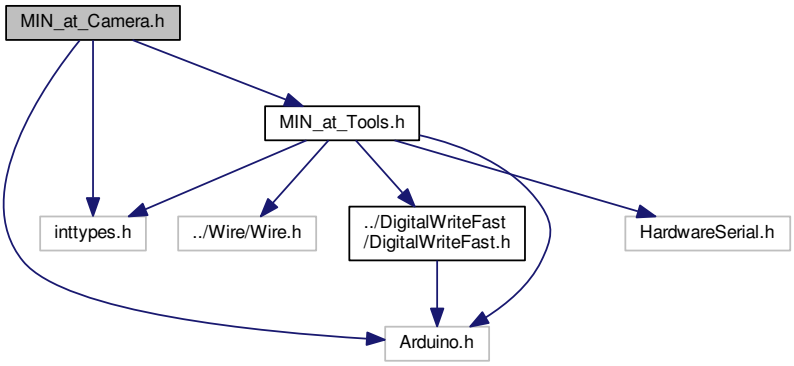
```
00254     Serial.print("OV_COLOR_SPACE(0x67: 0x1A): ");
00255     b = Cam.ReadConfigByte(OV_COLOR_SPACE);
00256     Camera::DebugPrintValue(b);
00257 
00258     // (0x71) set progressive scan mode
00259     Serial.print("OV_COMMON_L   (0x71: 0x00): ");
00260     b = Cam.ReadConfigByte(OV_COMMON_L);
00261     Camera::DebugPrintValue(b);
00262 }
00263 
00264 byte Camera::ReadConfigByte(byte MemAddr)
00265 {
00266     byte b = 0;
00267 
00268     Tools::I2C_ReadByteDefault(_addr, MemAddr,
      I2C_SHORT_ADDR, &b, 0);
00269 
00270     return b;
00271 }
00272 
00273 uint8_t Camera::ReadNextVideoByte()
00274 {
00275     // read clock
00276 
00277     digitalWriteFast(BUS_RCK, LOW);
00278     delayMicroseconds(20);
00279 
00280     digitalWriteFast(BUS_RCK, HIGH);
00281     delayMicroseconds(20);
00282 
00283     return (PIND & B11111100) | (PINC & B00000011);
00284 }
00285 
00286 void Camera::DumpVideoByte(byte VideoByte, byte *count)
00287 {
00288     Serial.print(Tools::FormatHEX(VideoByte, 0));
00289 
00290     if(*count == 16)
00291     {
00292         *count = 0;
00293         Serial.println();
00294     }
00295     else
00296         Serial.print(" ");
00297 }
00298 
00299 void Camera::ResetVideoPointer()
00300 {
00301     // reset sram read/write pointer
00302     digitalWriteFast(BUS_RRST, LOW);
00303     delay(1);
00304 
00305     // read clock
00306     digitalWriteFast(BUS_RCK, LOW);
00307     delay(1);
00308     digitalWriteFast(BUS_RCK, HIGH);
00309     delay(1);
00310     digitalWriteFast(BUS_RCK, LOW);
00311     delay(1);
00312     digitalWriteFast(BUS_RCK, HIGH);
00313     delay(1);
00314 
00315     digitalWriteFast(BUS_RRST, HIGH);
00316 }
00317 
00318 void Camera::DebugPrintValue(byte Value)
00319 {
00320   Serial.print(Tools::FormatHEX(Value, 1));
00321   Serial.print(" ");
00322   Serial.print(Tools::FormatBIN(Value));
00323   Serial.println();
00324 }
00325 
00326 void Camera::UYV2RGB(byte U, byte Y, byte V, byte *R, byte *G, byte *B)
00327 {
00328   float Yx = 1.164 * (Y - 16.0);
00329   float Ux = U - 128.0;
00330   float Vx = V - 128.0;
00331 
00332   *R = Clip(Yx + 1.596 * Ux);
00333   *G = Clip(Yx - 0.813 * Ux - 0.392 * Vx);
00334   *B = Clip(Yx + 2.017 * Vx);
00335 }
00336 
00337 
00338 
00339 // Private Methods /////////////////////////////////////////////////////////////
```

```
00340
00341 uint8_t Camera::Clip(float Value)
00342 {
00343   float v = round(Value);
00344
00345   if(v < 0)
00346     return 0;
00347
00348   if(v > 255)
00349     return 255;
00350
00351   return (byte)Value;
00352 }
00353
00354
00355
00356 // Preinstantiate Objects //////////////////////////////////////////////////
00357
00358 Camera Cam = Camera();
00359
```

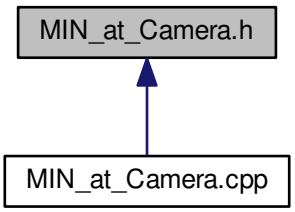
5.23 MIN_at_Camera.h File Reference

```
#include <inttypes.h>
#include <Arduino.h>
#include "MIN_at_Tools.h"
```

Include dependency graph for MIN_at_Camera.h:



This graph shows which files directly or indirectly include this file:



Classes

- class Camera

Macros

- #define OV_7620_ADDR 0x21
- #define OV_AGC 0x00
- #define OV_BLUE_GAIN 0x01
- #define OV_RED_GAIN 0x02
- #define OV_SATURATION 0x03
- #define OV_BRIGHTNESS 0x06
- #define OV_ANALOG_SHARPNESS 0x07
- #define OV_WBAL_BLUE 0x0C
- #define OV_WBAL_RED 0x0D
- #define OV_AUTO_EXPOSURE 0x10
- #define OV_CLOCK_RATE 0x11
- #define OV_COMMON_A 0x12
- #define OV_COMMON_B 0x13
- #define OV_COMMON_C 0x14
- #define OV_COMMON_D 0x15
- #define OV_FRAME_DROP 0x16
- #define OV_HWIN_START 0x17
- #define OV_HWIN_END 0x18
- #define OV_VWIN_START 0x19
- #define OV_VWIN_END 0x1A
- #define OV_PIXEL_SHIFT 0x1B
- #define OV_ID_H 0x1C
- #define OV_ID_L 0x1D
- #define OV_COMMON_E 0x20
- #define OV_YCHAN_OFFSET 0x21
- #define OV_UCHAN_OFFSET 0x22
- #define OV_CRYSTAL_CURRENT 0x23
- #define OV_AEW_PIXEL_RATIO 0x24
- #define OV_AEB_PIXEL_RATIO 0x25
- #define OV_COMMON_F 0x26
- #define OV_COMMON_G 0x27
- #define OV_COMMON_H 0x28
- #define OV_COMMON_I 0x29
- #define OV_FRAME_RATE_1 0x2A
- #define OV_FRAME_RATE_2 0x2B
- #define OV_BLACK_EXPAND 0x2C
- #define OV_COMMON_J 0x2D
- #define OV_VCHAN_OFFSET 0x2E
- #define OV_SIGNAL_A 0x60
- #define OV_SIGNAL_B 0x61
- #define OV_RGB_GAMMA 0x62
- #define OV_Y_GAMMA 0x64
- #define OV_SIGNAL_C 0x65
- #define OV_AWB_CONTROL 0x66
- #define OV_COLOR_SPACE 0x67
- #define OV_SIGNAL_D 0x68
- #define OV_HEDGE_ENH 0x69
- #define OV_VEDGE_ENH 0x6A

- #define OV_E_O_NOISE 0x6F
- #define OV_COMMON_K 0x70
- #define OV_COMMON_L 0x71
- #define OV_HSYNC_EDGE_1 0x72
- #define OV_HSYNC_EDGE_2 0x73
- #define OV_COMMON_M 0x74
- #define OV_COMMON_N 0x75
- #define OV_COMMON_O 0x76
- #define OV_FIELD_AVG 0x7C
- #define BUS_RRST 0x08
- #define BUS_RCK 0x09

Variables

- Camera Cam

5.23.1 Macro Definition Documentation

5.23.1.1 #define BUS_RCK 0x09

Definition at line 106 of file MIN_at_Camera.h.

5.23.1.2 #define BUS_RRST 0x08

Definition at line 105 of file MIN_at_Camera.h.

5.23.1.3 #define OV_7620_ADDR 0x21

Definition at line 27 of file MIN_at_Camera.h.

5.23.1.4 #define OV_AEB_PIXEL_RATIO 0x25

Definition at line 60 of file MIN_at_Camera.h.

5.23.1.5 #define OV_AEW_PIXEL_RATIO 0x24

Definition at line 59 of file MIN_at_Camera.h.

5.23.1.6 #define OV_AGC 0x00

Definition at line 29 of file MIN_at_Camera.h.

5.23.1.7 #define OV_ANALOG_SHARPNESS 0x07

Definition at line 35 of file MIN_at_Camera.h.

5.23.1.8 #define OV_AUTO_EXPOSURE 0x10

Definition at line 40 of file MIN_at_Camera.h.

5.23.1.9 #define OV_AWB_CONTROL 0x66

Definition at line 77 of file MIN_at_Camera.h.

5.23.1.10 #define OV_BLACK_EXPAND 0x2C

Definition at line 67 of file MIN_at_Camera.h.

5.23.1.11 #define OV_BLUE_GAIN 0x01

Definition at line 30 of file MIN_at_Camera.h.

5.23.1.12 #define OV_BRIGHTNESS 0x06

Definition at line 34 of file MIN_at_Camera.h.

5.23.1.13 #define OV_CLOCK_RATE 0x11

Definition at line 41 of file MIN_at_Camera.h.

5.23.1.14 #define OV_COLOR_SPACE 0x67

Definition at line 78 of file MIN_at_Camera.h.

5.23.1.15 #define OV_COMMON_A 0x12

Definition at line 42 of file MIN_at_Camera.h.

5.23.1.16 #define OV_COMMON_B 0x13

Definition at line 43 of file MIN_at_Camera.h.

5.23.1.17 #define OV_COMMON_C 0x14

Definition at line 44 of file MIN_at_Camera.h.

5.23.1.18 #define OV_COMMON_D 0x15

Definition at line 45 of file MIN_at_Camera.h.

5.23.1.19 #define OV_COMMON_E 0x20

Definition at line 55 of file MIN_at_Camera.h.

5.23.1.20 #define OV_COMMON_F 0x26

Definition at line 61 of file MIN_at_Camera.h.

5.23.1.21 #define OV_COMMON_G 0x27

Definition at line 62 of file MIN_at_Camera.h.

5.23.1.22 #define OV_COMMON_H 0x28

Definition at line 63 of file MIN_at_Camera.h.

5.23.1.23 #define OV_COMMON_I 0x29

Definition at line 64 of file MIN_at_Camera.h.

5.23.1.24 #define OV_COMMON_J 0x2D

Definition at line 68 of file MIN_at_Camera.h.

5.23.1.25 #define OV_COMMON_K 0x70

Definition at line 84 of file MIN_at_Camera.h.

5.23.1.26 #define OV_COMMON_L 0x71

Definition at line 85 of file MIN_at_Camera.h.

5.23.1.27 #define OV_COMMON_M 0x74

Definition at line 88 of file MIN_at_Camera.h.

5.23.1.28 #define OV_COMMON_N 0x75

Definition at line 89 of file MIN_at_Camera.h.

5.23.1.29 #define OV_COMMON_O 0x76

Definition at line 90 of file MIN_at_Camera.h.

5.23.1.30 #define OV_CRYSTAL_CURRENT 0x23

Definition at line 58 of file MIN_at_Camera.h.

5.23.1.31 #define OV_E_O_NOISE 0x6F

Definition at line 83 of file MIN_at_Camera.h.

5.23.1.32 #define OV_FIELD_AVG 0x7C

Definition at line 92 of file MIN_at_Camera.h.

5.23.1.33 #define OV_FRAME_DROP 0x16

Definition at line 46 of file MIN_at_Camera.h.

5.23.1.34 #define OV_FRAME_RATE_1 0x2A

Definition at line 65 of file MIN_at_Camera.h.

5.23.1.35 #define OV_FRAME_RATE_2 0x2B

Definition at line 66 of file MIN_at_Camera.h.

5.23.1.36 #define OV_HEDGE_ENH 0x69

Definition at line 80 of file MIN_at_Camera.h.

5.23.1.37 #define OV_HSYNC_EDGE_1 0x72

Definition at line 86 of file MIN_at_Camera.h.

5.23.1.38 #define OV_HSYNC_EDGE_2 0x73

Definition at line 87 of file MIN_at_Camera.h.

5.23.1.39 #define OV_HWIN_END 0x18

Definition at line 48 of file MIN_at_Camera.h.

5.23.1.40 #define OV_HWIN_START 0x17

Definition at line 47 of file MIN_at_Camera.h.

5.23.1.41 #define OV_ID_H 0x1C

Definition at line 52 of file MIN_at_Camera.h.

5.23.1.42 #define OV_ID_L 0x1D

Definition at line 53 of file MIN_at_Camera.h.

5.23.1.43 #define OV_PIXEL_SHIFT 0x1B

Definition at line 51 of file MIN_at_Camera.h.

5.23.1.44 #define OV_RED_GAIN 0x02

Definition at line 31 of file MIN_at_Camera.h.

5.23.1.45 #define OV_RGB_GAMMA 0x62

Definition at line 73 of file MIN_at_Camera.h.

5.23.1.46 #define OV_SATURATION 0x03

Definition at line 32 of file MIN_at_Camera.h.

5.23.1.47 #define OV_SIGNAL_A 0x60

Definition at line 71 of file MIN_at_Camera.h.

5.23.1.48 #define OV_SIGNAL_B 0x61

Definition at line 72 of file MIN_at_Camera.h.

5.23.1.49 #define OV_SIGNAL_C 0x65

Definition at line 76 of file MIN_at_Camera.h.

5.23.1.50 #define OV_SIGNAL_D 0x68

Definition at line 79 of file MIN_at_Camera.h.

5.23.1.51 #define OV_UCHAN_OFFSET 0x22

Definition at line 57 of file MIN_at_Camera.h.

5.23.1.52 #define OV_VCHAN_OFFSET 0x2E

Definition at line 69 of file MIN_at_Camera.h.

5.23.1.53 #define OV_VEDGE_ENH 0x6A

Definition at line 81 of file MIN_at_Camera.h.

5.23.1.54 #define OV_VWIN_END 0x1A

Definition at line 50 of file MIN_at_Camera.h.

5.23.1.55 #define OV_VWIN_START 0x19

Definition at line 49 of file MIN_at_Camera.h.

5.23.1.56 #define OV_WBAL_BLUE 0x0C

Definition at line 37 of file MIN_at_Camera.h.

5.23.1.57 #define OV_WBAL_RED 0x0D

Definition at line 38 of file MIN_at_Camera.h.

5.23.1.58 #define OV_Y_GAMMA 0x64

Definition at line 75 of file MIN_at_Camera.h.

5.23.1.59 #define OV_YCHAN_OFFSET 0x21

Definition at line 56 of file MIN_at_Camera.h.

5.23.2 Variable Documentation

5.23.2.1 Camera Cam

Definition at line 358 of file MIN_at_Camera.cpp.

5.24 MIN_at_Camera.h

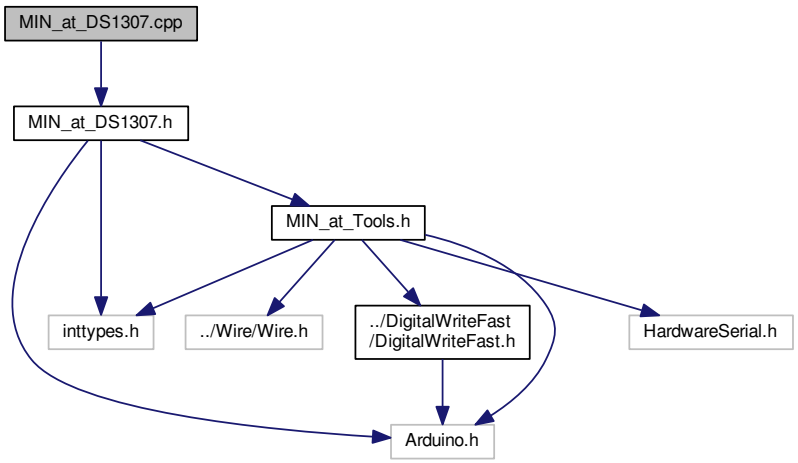
```
00001 // ArduinoCam Version 1.0 (2012-09-01) Firmware (Duemilanove Atmega168)
00002 // Copyright 2012 richard.prinz@min.at
00003 // See http://www.min.at/prinz/oe1rib/ArduinoCam/ for more infos
00004 //
00005 // This file is part of ArduinoCam
00006 //
00007 // ArduinoCam is free software and hardware design:
00008 // you can redistribute the software and the hardware design and/or modify it under
00009 // the terms of the GNU General Public License as published by the Free Software Foundation,
00010 // either version 3 of the License, or (at your option) any later version.
00011 //
00012 // ArduinoCam is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY;
00013 // without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
00014 // See the GNU General Public License for more details.
00015 //
00016 // You should have received a copy of the GNU General Public License along with ArduinoCam.
00017 // If not, see http://www.gnu.org/licenses/.
00018
00019 #ifndef MIN_at_Camera_h
00020 #define MIN_at_Camera_h
00021
00022 #include <inttypes.h>
00023 #include <Arduino.h>
00024 #include "MIN_at_Tools.h"
00025
00026
00027 #define  OV_7620_ADDR            0x21
00028
00029 #define  OV_AGC                  0x00
00030 #define  OV_BLUE_GAIN            0x01
00031 #define  OV_RED_GAIN             0x02
00032 #define  OV_SATURATION           0x03
00033
00034 #define  OV_BRIGHTNESS           0x06
00035 #define  OV_ANALOG_SHARPNESS     0x07
00036
00037 #define  OV_WBAL_BLUE            0x0C
00038 #define  OV_WBAL_RED             0x0D
00039
00040 #define  OV_AUTO_EXPOSURE        0x10
00041 #define  OV_CLOCK_RATE           0x11
00042 #define  OV_COMMON_A             0x12
00043 #define  OV_COMMON_B             0x13
00044 #define  OV_COMMON_C             0x14
00045 #define  OV_COMMON_D             0x15
00046 #define  OV_FRAME_DROP           0x16
00047 #define  OV_HWIN_START           0x17
00048 #define  OV_HWIN_END             0x18
00049 #define  OV_VWIN_START           0x19
00050 #define  OV_VWIN_END             0x1A
00051 #define  OV_PIXEL_SHIFT          0x1B
```

```
00052 #define  OV_ID_H                0x1C
00053 #define  OV_ID_L                0x1D
00054
00055 #define  OV_COMMON_E            0x20
00056 #define  OV_YCHAN_OFFSET        0x21
00057 #define  OV_UCHAN_OFFSET        0x22
00058 #define  OV_CRYSTAL_CURRENT     0x23
00059 #define  OV_AEW_PIXEL_RATIO     0x24
00060 #define  OV_AEB_PIXEL_RATIO     0x25
00061 #define  OV_COMMON_F            0x26
00062 #define  OV_COMMON_G            0x27
00063 #define  OV_COMMON_H            0x28
00064 #define  OV_COMMON_I            0x29
00065 #define  OV_FRAME_RATE_1        0x2A
00066 #define  OV_FRAME_RATE_2        0x2B
00067 #define  OV_BLACK_EXPAND        0x2C
00068 #define  OV_COMMON_J            0x2D
00069 #define  OV_VCHAN_OFFSET        0x2E
00070
00071 #define  OV_SIGNAL_A            0x60
00072 #define  OV_SIGNAL_B            0x61
00073 #define  OV_RGB_GAMMA           0x62
00074
00075 #define  OV_Y_GAMMA             0x64
00076 #define  OV_SIGNAL_C            0x65
00077 #define  OV_AWB_CONTROL         0x66
00078 #define  OV_COLOR_SPACE         0x67
00079 #define  OV_SIGNAL_D            0x68
00080 #define  OV_HEDGE_ENH           0x69
00081 #define  OV_VEDGE_ENH           0x6A
00082
00083 #define  OV_E_O_NOISE           0x6F
00084 #define  OV_COMMON_K            0x70
00085 #define  OV_COMMON_L            0x71
00086 #define  OV_HSYNC_EDGE_1        0x72
00087 #define  OV_HSYNC_EDGE_2        0x73
00088 #define  OV_COMMON_M            0x74
00089 #define  OV_COMMON_N            0x75
00090 #define  OV_COMMON_O            0x76
00091
00092 #define  OV_FIELD_AVG           0x7C
00093
00094 /*
00095 #define  BUS_0                  0x02
00096 #define  BUS_1                  0x03
00097 #define  BUS_2                  0x02
00098 #define  BUS_3                  0x03
00099 #define  BUS_4                  0x04
00100 #define  BUS_5                  0x05
00101 #define  BUS_6                  0x06
00102 #define  BUS_7                  0x07
00103 */
00104
00105 #define  BUS_RRST               0x08
00106 #define  BUS_RCK                0x09
00107
00108 class Camera
00109 {
00110   private:
00111     int _addr;
00112
00113     static uint8_t Clip(float Value);
00114
00115
00116   public:
00117     Camera();
00118
00119     void Begin();
00120     void Begin(uint8_t address);
00121     void Begin(int address);
00122
00123     bool Reset();
00124
00125     void Init();
00126     void ColorBar(bool On);
00127     void Power(bool On);
00128     void Mirror(bool On);
00129     bool Capture();
00130     void Dump(bool Hex);
00131     void DumpConfig();
00132
00133     uint8_t ReadConfigByte(uint8_t MemAddr);
00134     uint8_t ReadNextVideoByte();
00135     void DumpVideoByte(uint8_t pixel, uint8_t *count);
00136     void ResetVideoPointer();
00137
00138     static void DebugPrintValue(uint8_t Value);
```

```
00139     static void UYV2RGB(uint8_t U, uint8_t Y, uint8_t V, uint8_t *R, uint8_t *G, uint8_t *B);
00140 };
00141
00142 extern Camera Cam;
00143
00144 #endif
00145
```

5.25 MIN_at_DS1307.cpp File Reference

`#include "MIN_at_DS1307.h"`
Include dependency graph for MIN_at_DS1307.cpp:



Variables

- DS1307 DS1307x = DS1307()

5.25.1 Variable Documentation

5.25.1.1 DS1307 DS1307x = DS1307()

Definition at line 148 of file MIN_at_DS1307.cpp.

5.26 MIN_at_DS1307.cpp

```
00001 // ArduinoCam Version 1.0 (2012-09-01) Firmware (Duemilanove Atmega168)
00002 // Copyright 2012 richard.prinz@min.at
00003 // See http://www.min.at/prinz/oe1rib/ArduinoCam/ for more infos
00004 //
00005 // This file is part of ArduinoCam
00006 //
00007 // ArduinoCam is free software and hardware design:
00008 // you can redistribute the software and the hardware design and/or modify it under
00009 // the terms of the GNU General Public License as published by the Free Software Foundation,
00010 // either version 3 of the License, or (at your option) any later version.
00011 //
00012 // ArduinoCam is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY;
00013 // without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
00014 // See the GNU General Public License for more details.
00015 //
00016 // You should have received a copy of the GNU General Public License along with ArduinoCam.
00017 // If not, see http://www.gnu.org/licenses/.
```

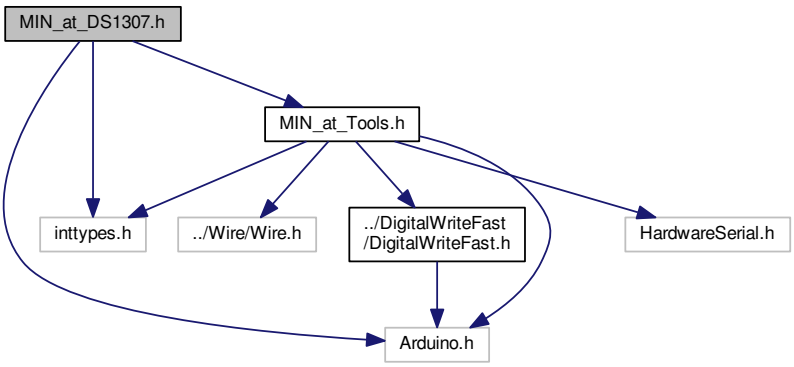
```
00018
00019 #include "MIN_at_DS1307.h"
00020
00021
00022
00023 // Microchip DS1307 RTC
00024
00025
00026
00027 // Constructors ////////////////////////////////////////////////////////////
00028
00029 DS1307::DS1307()
00030 {
00031 }
00032
00033
00034
00035 // Public Methods //////////////////////////////////////////////////////////
00036
00037 void DS1307::Begin()
00038 {
00039     Begin(DS1307_ADDR);
00040 }
00041
00042 void DS1307::Begin(byte address)
00043 {
00044     Begin((int)address);
00045 }
00046
00047 void DS1307::Begin(int address)
00048 {
00049     _addr = address;
00050 }
00051
00052 void DS1307::Reset()
00053 {
00054     Wire.beginTransmission(_addr);
00055     for(int i = 0; i < 8; i++)
00056         Wire.write((uint8_t)0x00);
00057     Wire.endTransmission();
00058 }
00059
00060 uint8_t DS1307::ReadConfigByte()
00061 {
00062     uint8_t temp;
00063     Tools::I2C_ReadByteDefault(_addr,
      DS1307_CONFIG, I2C_SHORT_ADDR, &temp, 0x00);
00064     return temp;
00065 }
00066
00067 void DS1307::WriteConfigByte(uint8_t Value)
00068 {
00069     Tools::I2C_WriteValue(_addr, DS1307_CONFIG,
      I2C_SHORT_ADDR, Value, 10);
00070 }
00071
00072 bool DS1307::ReadTime()
00073 {
00074     uint8_t i = 0;
00075
00076     Tools::I2C_Write(_addr, 0x00, I2C_SHORT_ADDR);
00077     Wire.requestFrom(_addr, 7);
00078
00079     if(Wire.available())
00080     {
00081         _rtc_sec = Tools::bcdToDec(Wire.read() & 0x7f);
00082         i++;
00083     }
00084
00085     if(Wire.available())
00086     {
00087         _rtc_min = Tools::bcdToDec(Wire.read());
00088         i++;
00089     }
00090
00091     if(Wire.available())
00092     {
00093         _rtc_hour = Tools::bcdToDec(Wire.read() & 0x3f);
00094         i++;
00095     }
00096
00097     if(Wire.available())
00098     {
00099         _rtc_wday = Tools::bcdToDec(Wire.read());
00100         i++;
00101     }
00102
```

```
00103     if(Wire.available())
00104     {
00105         _rtc_day = Tools::bcdToDec(Wire.read());
00106         i++;
00107     }
00108
00109     if(Wire.available())
00110     {
00111         _rtc_mon = Tools::bcdToDec(Wire.read());
00112         i++;
00113     }
00114
00115     if(Wire.available())
00116     {
00117         _rtc_year = Tools::bcdToDec(Wire.read());
00118         i++;
00119     }
00120
00121     return i == 7;
00122 }
00123
00124 void DS1307::WriteTime()
00125 {
00126     uint8_t a[] = {_rtc_sec, _rtc_min, _rtc_hour,
00127                    _rtc_wday, _rtc_day, _rtc_mon,
    _rtc_year};
00128     WriteTimeArray(a);
00129 }
00130
00131 void DS1307::WriteTimeArray(uint8_t Array[])
00132 {
00133     Wire.beginTransmission(_addr);
00134     Wire.write((uint8_t)0x00);
00135     for(int i = 0; i < 7; i++)
00136         Wire.write(Array[i]);
00137     Wire.endTransmission();
00138 }
00139
00140
00141
00142 // Private Methods /////////////////////////////////////////////////////////////
00143
00144
00145
00146 // Preinstantiate Objects //////////////////////////////////////////////////////
00147
00148 DS1307 DS1307x = DS1307();
00149
```

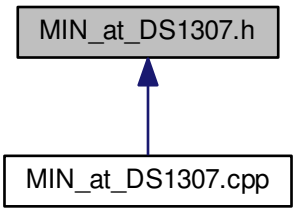
5.27 MIN_at_DS1307.h File Reference

```
#include <inttypes.h>
#include <Arduino.h>
#include "MIN_at_Tools.h"
```

Include dependency graph for MIN_at_DS1307.h:



This graph shows which files directly or indirectly include this file:



Classes

- class DS1307

Macros

- #define DS1307_ADDR 0x68
- #define DS1307_SECONDS 0x00
- #define DS1307_MINUTES 0x01
- #define DS1307_HOURS 0x02
- #define DS1307_DAY 0x03
- #define DS1307_DATE 0x04
- #define DS1307_MONTH 0x05
- #define DS1307_YEAR 0x06
- #define DS1307_CONFIG 0x07

Variables

- DS1307 DS1307x

5.27.1 Macro Definition Documentation

5.27.1.1 #define DS1307_ADDR 0x68

Definition at line 27 of file MIN_at_DS1307.h.

5.27.1.2 #define DS1307_CONFIG 0x07

Definition at line 36 of file MIN_at_DS1307.h.

5.27.1.3 #define DS1307_DATE 0x04

Definition at line 33 of file MIN_at_DS1307.h.

5.27.1.4 #define DS1307_DAY 0x03

Definition at line 32 of file MIN_at_DS1307.h.

5.27.1.5 #define DS1307_HOURS 0x02

Definition at line 31 of file MIN_at_DS1307.h.

5.27.1.6 #define DS1307_MINUTES 0x01

Definition at line 30 of file MIN_at_DS1307.h.

5.27.1.7 #define DS1307_MONTH 0x05

Definition at line 34 of file MIN_at_DS1307.h.

5.27.1.8 #define DS1307_SECONDS 0x00

Definition at line 29 of file MIN_at_DS1307.h.

5.27.1.9 #define DS1307_YEAR 0x06

Definition at line 35 of file MIN_at_DS1307.h.

5.27.2 Variable Documentation

5.27.2.1 DS1307 DS1307x

Definition at line 148 of file MIN_at_DS1307.cpp.

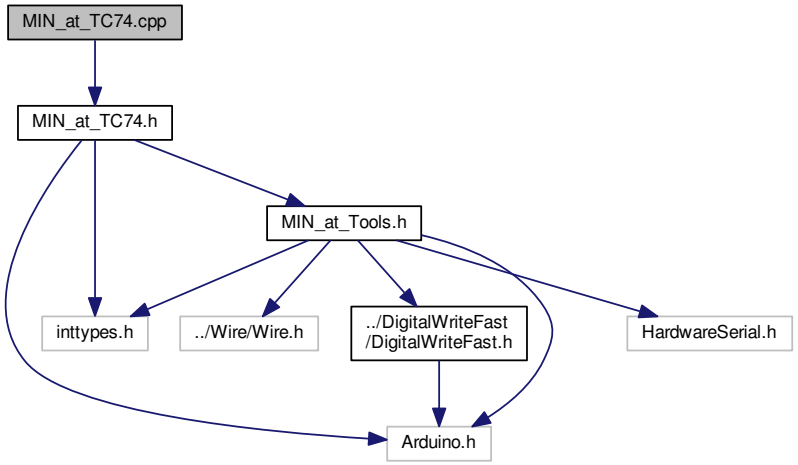
5.28 MIN_at_DS1307.h

```
00001 // ArduinoCam Version 1.0 (2012-09-01) Firmware (Duemilanove Atmega168)
00002 // Copyright 2012 richard.prinz@min.at
00003 // See http://www.min.at/prinz/oe1rib/ArduinoCam/ for more infos
00004 //
00005 // This file is part of ArduinoCam
00006 //
00007 // ArduinoCam is free software and hardware design:
00008 // you can redistribute the software and the hardware design and/or modify it under
00009 // the terms of the GNU General Public License as published by the Free Software Foundation,
00010 // either version 3 of the License, or (at your option) any later version.
00011 //
00012 // ArduinoCam is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY;
00013 // without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
00014 // See the GNU General Public License for more details.
00015 //
00016 // You should have received a copy of the GNU General Public License along with ArduinoCam.
00017 // If not, see http://www.gnu.org/licenses/.
00018
00019 #ifndef MIN_at_DS1307_h
00020 #define MIN_at_DS1307_h
00021
00022 #include <inttypes.h>
00023 #include <Arduino.h>
00024 #include "MIN_at_Tools.h"
00025
00026
00027 #define DS1307_ADDR             0x68
00028
00029 #define DS1307_SECONDS          0x00
00030 #define DS1307_MINUTES          0x01
00031 #define DS1307_HOURS            0x02
00032 #define DS1307_DAY              0x03
00033 #define DS1307_DATE             0x04
00034 #define DS1307_MONTH            0x05
00035 #define DS1307_YEAR             0x06
00036 #define DS1307_CONFIG           0x07
00037
00038 class DS1307
00039 {
00040   private:
00041     int _addr;
00042
00043
00044   public:
00045     uint8_t _rtc_sec;
```

```
00046     uint8_t _rtc_min;
00047     uint8_t _rtc_hour;
00048     uint8_t _rtc_wday;
00049     uint8_t _rtc_day;
00050     uint8_t _rtc_mon;
00051     uint8_t _rtc_year;
00052
00053     DS1307();
00054
00055     void Begin();
00056     void Begin(uint8_t address);
00057     void Begin(int address);
00058
00059     void Reset();
00060     uint8_t ReadConfigByte();
00061     void WriteConfigByte(uint8_t value);
00062     bool ReadTime();
00063     void WriteTime();
00064     void WriteTimeArray(uint8_t Array[]);
00065 };
00066
00067 extern DS1307 DS1307x;
00068
00069 #endif
```

5.29 MIN_at_TC74.cpp File Reference

`#include "MIN_at_TC74.h"`
Include dependency graph for MIN_at_TC74.cpp:



5.30 MIN_at_TC74.cpp

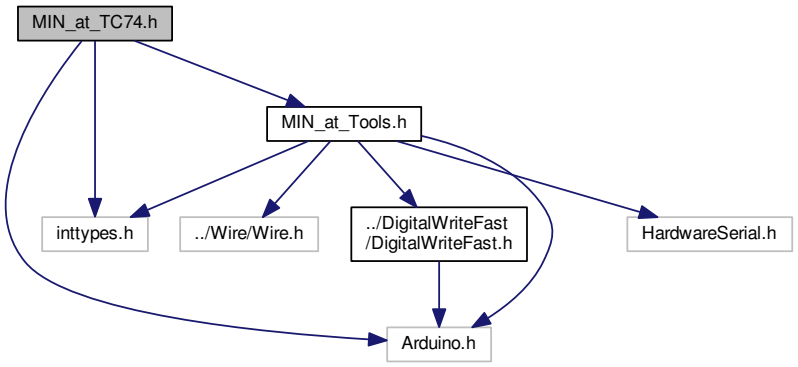
```
00001 // ArduinoCam Version 1.0 (2012-09-01) Firmware (Duemilanove Atmega168)
00002 // Copyright 2012 richard.prinz@min.at
00003 // See http://www.min.at/prinz/oe1rib/ArduinoCam/ for more infos
00004 //
00005 // This file is part of ArduinoCam
00006 //
00007 // ArduinoCam is free software and hardware design:
00008 // you can redistribute the software and the hardware design and/or modify it under
00009 // the terms of the GNU General Public License as published by the Free Software Foundation,
00010 // either version 3 of the License, or (at your option) any later version.
00011 //
00012 // ArduinoCam is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY;
00013 // without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
00014 // See the GNU General Public License for more details.
00015 //
00016 // You should have received a copy of the GNU General Public License along with ArduinoCam.
00017 // If not, see http://www.gnu.org/licenses/.
```

```
00018
00019 #include "MIN_at_TC74.h"
00020
00021
00022
00023 // Microchip TC74 I2C Sensor library
00024
00025
00026
00027 // Constructors ////////////////////////////////////////////////////////////////
00028
00029 TC74::TC74()
00030 {
00031 }
00032
00033
00034
00035 // Public Methods //////////////////////////////////////////////////////////////
00036
00037 void TC74::Begin()
00038 {
00039     Begin(TC74A0_ADDR);
00040 }
00041
00042 void TC74::Begin(byte address)
00043 {
00044     Begin((int)address);
00045 }
00046
00047 void TC74::Begin(int address)
00048 {
00049     _addr = address;
00050 }
00051
00052 void TC74::Standby(bool Value)
00053 {
00054     uint8_t w = (Value ? 0x80 : 0x00);
00055     WriteConfigByte(w);
00056 }
00057
00058 uint8_t TC74::ReadConfigByte()
00059 {
00060     uint8_t temp;
00061     Tools::I2C_ReadByteDefault(_addr, TC74_CONFIG,
      I2C_SHORT_ADDR, &temp, 0x00);
00062     return temp;
00063 }
00064
00065 void TC74::WriteConfigByte(uint8_t Value)
00066 {
00067     Tools::I2C_WriteValue(_addr, TC74_CONFIG,
      I2C_SHORT_ADDR, Value, 0);
00068 }
00069
00070 int8_t TC74::ReadTemperature()
00071 {
00072     uint8_t temp;
00073     Tools::I2C_ReadByteDefault(_addr, TC74_TEMP,
      I2C_SHORT_ADDR, &temp, TC74_VALUE_ERROR);
00074     return (int8_t)temp;
00075 }
00076
00077
00078
00079 // Private Methods /////////////////////////////////////////////////////////////
00080
00081
00082
00083 // Preinstantiate Objects //////////////////////////////////////////////////////
00084
```

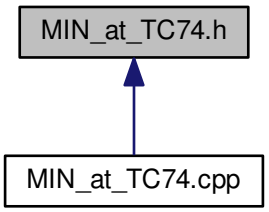
5.31 MIN_at_TC74.h File Reference

```
#include <inttypes.h>
#include <Arduino.h>
#include "MIN_at_Tools.h"
```

Include dependency graph for MIN_at_TC74.h:



This graph shows which files directly or indirectly include this file:



Classes

- class TC74

Macros

- #define TC74A0_ADDR 0x48
- #define TC74A1_ADDR 0x49
- #define TC74A2_ADDR 0x4A
- #define TC74A3_ADDR 0x4B
- #define TC74A4_ADDR 0x4C
- #define TC74A5_ADDR 0x4D
- #define TC74A6_ADDR 0x4E
- #define TC74A7_ADDR 0x4F
- #define TC74_TEMP 0x00
- #define TC74_CONFIG 0x01
- #define TC74_VALUE_ERROR 0x80

5.31.1 Macro Definition Documentation

5.31.1.1 #define TC74_CONFIG 0x01

Definition at line 37 of file MIN_at_TC74.h.

5.31.1.2 #define TC74_TEMP 0x00

Definition at line 36 of file MIN_at_TC74.h.

5.31.1.3 #define TC74_VALUE_ERROR 0x80

Definition at line 38 of file MIN_at_TC74.h.

5.31.1.4 #define TC74A0_ADDR 0x48

Definition at line 27 of file MIN_at_TC74.h.

5.31.1.5 #define TC74A1_ADDR 0x49

Definition at line 28 of file MIN_at_TC74.h.

5.31.1.6 #define TC74A2_ADDR 0x4A

Definition at line 29 of file MIN_at_TC74.h.

5.31.1.7 #define TC74A3_ADDR 0x4B

Definition at line 30 of file MIN_at_TC74.h.

5.31.1.8 #define TC74A4_ADDR 0x4C

Definition at line 31 of file MIN_at_TC74.h.

5.31.1.9 #define TC74A5_ADDR 0x4D

Definition at line 32 of file MIN_at_TC74.h.

5.31.1.10 #define TC74A6_ADDR 0x4E

Definition at line 33 of file MIN_at_TC74.h.

5.31.1.11 #define TC74A7_ADDR 0x4F

Definition at line 34 of file MIN_at_TC74.h.

5.32 MIN_at_TC74.h

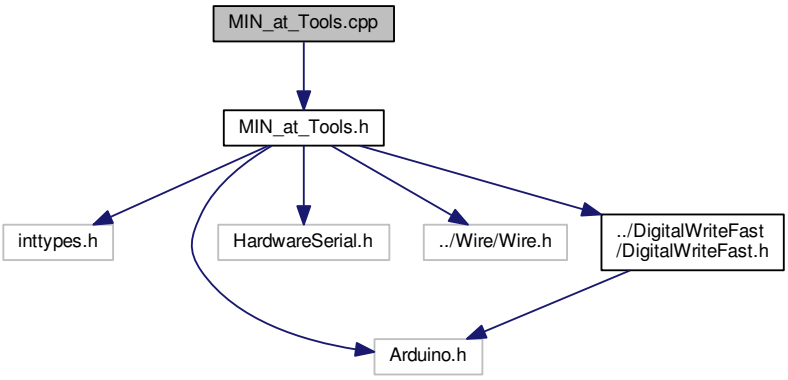
```
00001 // ArduinoCam Version 1.0 (2012-09-01) Firmware (Duemilanove Atmega168)
00002 // Copyright 2012 richard.prinz@min.at
00003 // See http://www.min.at/prinz/oe1rib/ArduinoCam/ for more infos
00004 //
00005 // This file is part of ArduinoCam
00006 //
00007 // ArduinoCam is free software and hardware design:
00008 // you can redistribute the software and the hardware design and/or modify it under
00009 // the terms of the GNU General Public License as published by the Free Software Foundation,
00010 // either version 3 of the License, or (at your option) any later version.
00011 //
00012 // ArduinoCam is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY;
00013 // without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
00014 // See the GNU General Public License for more details.
00015 //
00016 // You should have received a copy of the GNU General Public License along with ArduinoCam.
00017 // If not, see http://www.gnu.org/licenses/.
00018
```

```
00019 #ifndef MIN_at_TC74_h
00020 #define MIN_at_TC74_h
00021
00022 #include <inttypes.h>
00023 #include <Arduino.h>
00024 #include "MIN_at_Tools.h"
00025
00026
00027 #define TC74A0_ADDR             0x48
00028 #define TC74A1_ADDR             0x49
00029 #define TC74A2_ADDR             0x4A
00030 #define TC74A3_ADDR             0x4B
00031 #define TC74A4_ADDR             0x4C
00032 #define TC74A5_ADDR             0x4D
00033 #define TC74A6_ADDR             0x4E
00034 #define TC74A7_ADDR             0x4F
00035
00036 #define TC74_TEMP               0x00
00037 #define TC74_CONFIG             0x01
00038 #define TC74_VALUE_ERROR        0x80
00039
00040
00041 class TC74
00042 {
00043   private:
00044     int _addr;
00045
00046
00047   public:
00048     TC74();
00049
00050     void Begin();
00051     void Begin(uint8_t address);
00052     void Begin(int address);
00053
00054     void Standby(bool Value);
00055
00056     uint8_t ReadConfigByte();
00057     void WriteConfigByte(uint8_t value);
00058     int8_t ReadTemperature();
00059 };
00060
00061 #endif
```

5.33 MIN_at_Tools.cpp File Reference

`#include "MIN_at_Tools.h"`

Include dependency graph for MIN_at_Tools.cpp:



5.34 MIN_at_Tools.cpp

```
00001 // ArduinoCam Version 1.0 (2012-09-01) Firmware (Duemilanove Atmega168)
00002 // Copyright 2012 richard.prinz@min.at
```

```
00003 // See http://www.min.at/prinz/oe1rib/ArduinoCam/ for more infos
00004 //
00005 // This file is part of ArduinoCam
00006 //
00007 // ArduinoCam is free software and hardware design:
00008 // you can redistribute the software and the hardware design and/or modify it under
00009 // the terms of the GNU General Public License as published by the Free Software Foundation,
00010 // either version 3 of the License, or (at your option) any later version.
00011 //
00012 // ArduinoCam is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY;
00013 // without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
00014 // See the GNU General Public License for more details.
00015 //
00016 // You should have received a copy of the GNU General Public License along with ArduinoCam.
00017 // If not, see http://www.gnu.org/licenses/.
00018
00019 #include "MIN_at_Tools.h"
00020
00021
00022
00023 // Constructors ////////////////////////////////////////////////////////////////
00024
00025 Tools::Tools()
00026 {
00027 }
00028
00029
00030
00031 // Public Methods //////////////////////////////////////////////////////////////
00032
00033 int Tools::ReadDec(uint8_t MaxLen, uint8_t Flags, bool *Valid)
00034 {
00035     int v = 0;
00036     uint8_t i = 0;
00037     uint8_t inByte;
00038
00039     *Valid = true;
00040
00041     while(true)
00042     {
00043         if(Serial.available() > 0)
00044         {
00045             inByte = Serial.read();
00046
00047             // ESC = cancel
00048             if(inByte == 0x1B)
00049             {
00050                 *Valid = false;
00051                 break;
00052             }
00053
00054             // handle backspace
00055             if(inByte == CHAR_BACKSPACE)
00056             {
00057                 if(i <= 0)
00058                     Serial.write(CHAR_BELL);
00059                 else
00060                 {
00061                     v = v / 10;
00062                     i--;
00063                     if( (Flags & DI_ECHO) == DI_ECHO )
00064                         Serial.print("\010 \010");
00065                 }
00066                 continue;
00067             }
00068
00069             // CR & LF submits input
00070             if( (inByte == CHAR_CR || inByte == CHAR_LF) && (Flags &
    DI_ALLOW_CR) == DI_ALLOW_CR )
00071                 break;
00072
00073             // if maximum allowed input length reached
00074             if(i >= MaxLen)
00075             {
00076                 Serial.write(CHAR_BELL);
00077                 continue;
00078             }
00079
00080             // only digits 0 - 9 allowed
00081             if(inByte >= '0' && inByte <= '9')
00082             {
00083                 if( (Flags & DI_ECHO) == DI_ECHO )
00084                     Serial.write(inByte);
00085
00086                 v = (v * 10) + (inByte - '0');
00087                 i++;
00088                 if(i >= MaxLen && (Flags & DI_AUTO_SKIP) ==
```

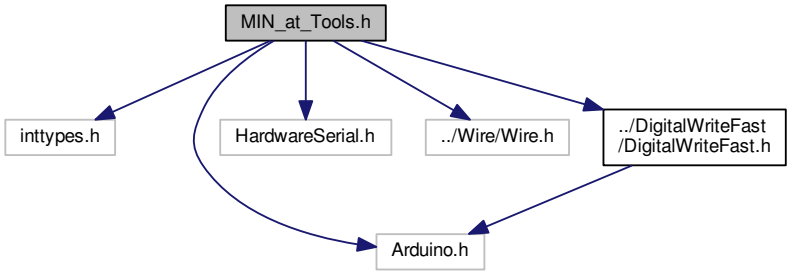
```
    DI_AUTO_SKIP)
00089                     break;
00090             }
00091             else
00092                 Serial.write(CHAR_BELL);
00093         }
00094     }
00095
00096     return v;
00097 }
00098
00099 char *Tools::FormatHEX(uint8_t Value, uint8_t Prefix)
00100 {
00101     static char strOut[5];
00102     snprintf(strOut, sizeof(strOut), "%s%02X", (Prefix > 0 ? "0x" : ""), Value);
00103     return strOut;
00104 }
00105
00106 char *Tools::FormatHEX16(int Value, uint8_t Prefix)
00107 {
00108     static char strOut[7];
00109     snprintf(strOut, sizeof(strOut), "%s%04X", (Prefix > 0 ? "0x" : ""), Value);
00110     return strOut;
00111 }
00112
00113 char *Tools::FormatBIN(uint8_t Value)
00114 {
00115     static char buffer[9];
00116     for (int i = 0; i < 8; i++)
00117         buffer[7-i] = '0' + ((Value & (1 << i)) > 0);
00118     buffer[8] ='\0';
00119     return buffer;
00120 }
00121
00122
00123
00124
00125
00126 void Tools::I2C_Write(uint8_t I2cAddr, uint16_t MemAddr, uint8_t UseLongAddr)
00127 {
00128     Wire.beginTransmission(I2cAddr);
00129     if(UseLongAddr > 0)
00130     {
00131         Wire.write(MemAddr >> 8); // MSB
00132         Wire.write(MemAddr & 0xFF); // LSB
00133     }
00134     else
00135         Wire.write(MemAddr & 0x00FF);
00136     Wire.endTransmission();
00137 }
00138
00139 void Tools::I2C_WriteValue(uint8_t I2cAddr, uint16_t MemAddr, uint8_t UseLongAddr,
00140                            uint8_t Value, int Delay)
00141 {
00142     Wire.beginTransmission(I2cAddr);
00143     if(UseLongAddr > 0)
00144     {
00145         Wire.write(MemAddr >> 8); // MSB
00146         Wire.write(MemAddr & 0xFF); // LSB
00147     }
00148     else
00149         Wire.write(MemAddr & 0x00FF);
00150     Wire.write(Value);
00151     Wire.endTransmission();
00152     if(Delay > 0)
00153         delay(Delay);
00154 }
00155
00156 bool Tools::I2C_ReadByte(uint8_t I2cAddr, uint16_t MemAddr, uint8_t UseLongAddr,
00157                          uint8_t *Value)
00158 {
00159     I2C_Write(I2cAddr, MemAddr, UseLongAddr);
00160     Wire.requestFrom((int)I2cAddr, 1);
00161
00162     if(Wire.available())
00163     {
00164         *Value = Wire.read();
00165         return true;
00166     }
00167     else
00168         return false;
00169 }
00170
00171 bool Tools::I2C_ReadByteDefault(uint8_t I2cAddr, uint16_t MemAddr, uint8_t
    UseLongAddr,
00172                                 uint8_t *Value, uint8_t DefaultValue)
00173 {
```

```
00174     if(!Tools::I2C_ReadByte(I2cAddr, MemAddr, UseLongAddr, Value))
00175     {
00176         *Value = DefaultValue;
00177         return true;
00178     }
00179
00180     return false;
00181 }
00182
00183 void Tools::I2C_SetBitAt(uint8_t I2cAddr, uint16_t MemAddr, uint8_t UseLongAddr,
00184                          uint8_t BitNum, bool Value, int Delay)
00185 {
00186     uint8_t m = 1 << BitNum;
00187     uint8_t b;
00188     if(!I2C_ReadByte(I2cAddr, MemAddr, UseLongAddr, &b))
00189         return;
00190
00191     if(!Value)
00192     {
00193         m = ~m;
00194         b = b & m;
00195     }
00196     else
00197         b = b | m;
00198
00199     I2C_WriteValue(I2cAddr, MemAddr, UseLongAddr, b, Delay);
00200 }
00201
00202 // WARNING: address is a page address, 6-bit end will wrap around
00203 // also, data can be maximum of about 30 bytes, because the Wire
00204 // library has a buffer of 32 bytes
00205 void Tools::I2C_EEWriteBuffer(uint8_t I2cAddr, uint16_t MemAddr,
00206                         byte *Data, byte Length)
00207 {
00208   Wire.beginTransmission(I2cAddr);
00209   Wire.write(MemAddr >> 8); // MSB
00210   Wire.write(MemAddr & 0x00FF); // LSB
00211   byte c;
00212   for (c = 0; c < Length; c++)
00213     Wire.write(Data[c]);
00214   Wire.endTransmission();
00215 }
00216
00217 // maybe let's not read more than 30 or 32 bytes at a time!
00218 void Tools::I2C_EEReadBuffer(uint8_t I2cAddr, uint16_t MemAddr, byte *Data, int
      Length)
00219 {
00220   Wire.beginTransmission(I2cAddr);
00221   Wire.write(MemAddr >> 8); // MSB
00222   Wire.write(MemAddr & 0xFF); // LSB
00223   Wire.endTransmission();
00224   Wire.requestFrom((int)I2cAddr, Length);
00225   int c = 0;
00226   for (c = 0; c < Length; c++)
00227     if (Wire.available())
00228         Data[c] = Wire.read();
00229 }
00230
00231 // Convert binary coded decimal to normal decimal numbers
00232 uint8_t Tools::bcdToDec(uint8_t Value)
00233 {
00234     return ( (Value / 16 * 10) + (Value % 16) );
00235 }
00236
00237 // Convert Decimal to Binary Coded Decimal (BCD)
00238 uint8_t Tools::dec2bcd(uint8_t num)
00239 {
00240     return ( (num / 10 * 16) + (num % 10) );
00241 }
00242
00243
00244
00245 // Private Methods /////////////////////////////////////////////////////////////
00246
00247
00248
00249 // Preinstantiate Objects //////////////////////////////////////////////////////
00250
```

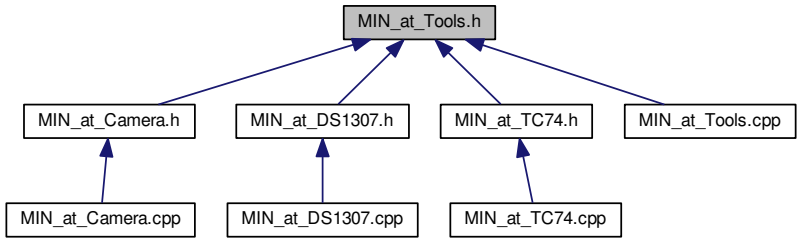
5.35 MIN_at_Tools.h File Reference

```
#include <inttypes.h>
#include <Arduino.h>
#include <HardwareSerial.h>
#include "../Wire/Wire.h"
#include "../DigitalWriteFast/DigitalWriteFast.h"
```

Include dependency graph for MIN_at_Tools.h:



This graph shows which files directly or indirectly include this file:



Classes

- class Tools

Macros

- #define CHAR_BACKSPACE 0x08
- #define CHAR_BELL 0x07
- #define CHAR_CR 0x0D
- #define CHAR_LF 0x0A
- #define I2C_SHORT_ADDR 0
- #define I2C_LONG_ADDR 1
- #define DI_ALLOW_CR 1
- #define DI_ECHO 2
- #define DI_AUTO_SKIP 4

5.35.1 Macro Definition Documentation

5.35.1.1 #define CHAR_BACKSPACE 0x08

Definition at line 28 of file MIN_at_Tools.h.

5.35.1.2 #define CHAR_BELL 0x07

Definition at line 29 of file MIN_at_Tools.h.

5.35.1.3 #define CHAR_CR 0x0D

Definition at line 30 of file MIN_at_Tools.h.

5.35.1.4 #define CHAR_LF 0x0A

Definition at line 31 of file MIN_at_Tools.h.

5.35.1.5 #define DI_ALLOW_CR 1

Definition at line 36 of file MIN_at_Tools.h.

5.35.1.6 #define DI_AUTO_SKIP 4

Definition at line 38 of file MIN_at_Tools.h.

5.35.1.7 #define DI_ECHO 2

Definition at line 37 of file MIN_at_Tools.h.

5.35.1.8 #define I2C_LONG_ADDR 1

Definition at line 34 of file MIN_at_Tools.h.

5.35.1.9 #define I2C_SHORT_ADDR 0

Definition at line 33 of file MIN_at_Tools.h.

5.36 MIN_at_Tools.h

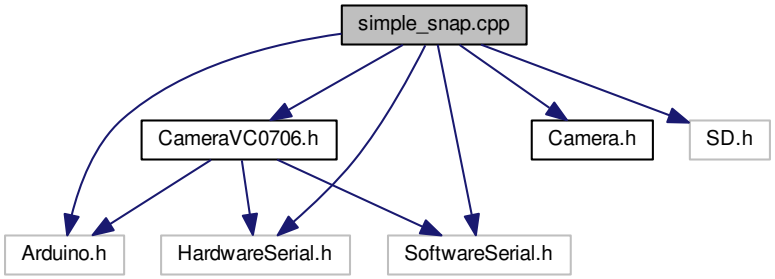
```
00001 // ArduinoCam Version 1.0 (2012-09-01) Firmware (Duemilanove Atmega168)
00002 // Copyright 2012 richard.prinz@min.at
00003 // See http://www.min.at/prinz/oe1rib/ArduinoCam/ for more infos
00004 //
00005 // This file is part of ArduinoCam
00006 //
00007 // ArduinoCam is free software and hardware design:
00008 // you can redistribute the software and the hardware design and/or modify it under
00009 // the terms of the GNU General Public License as published by the Free Software Foundation,
00010 // either version 3 of the License, or (at your option) any later version.
00011 //
00012 // ArduinoCam is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY;
00013 // without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
00014 // See the GNU General Public License for more details.
00015 //
00016 // You should have received a copy of the GNU General Public License along with ArduinoCam.
00017 // If not, see http://www.gnu.org/licenses/.
00018
00019 #ifndef Tools_h
00020 #define Tools_h
00021
00022 #include <inttypes.h>
00023 #include <Arduino.h>
00024 #include <HardwareSerial.h>
00025 #include "../Wire/Wire.h"
00026 #include "../DigitalWriteFast/DigitalWriteFast.h"
00027
00028 #define CHAR_BACKSPACE          0x08
00029 #define CHAR_BELL               0x07
00030 #define CHAR_CR                 0x0D
```

```
00031 #define CHAR_LF                 0x0A
00032
00033 #define I2C_SHORT_ADDR          0
00034 #define I2C_LONG_ADDR           1
00035
00036 #define DI_ALLOW_CR             1
00037 #define DI_ECHO                 2
00038 #define DI_AUTO_SKIP            4
00039
00040
00041 class Tools
00042 {
00043   public:
00044     Tools();
00045
00046     static int ReadDec(uint8_t MaxLen, uint8_t Flags, bool *Valid);
00047
00048     static char *FormatHEX(uint8_t Value, uint8_t Prefix);
00049     static char *FormatHEX16(int Value, uint8_t Prefix);
00050     static char *FormatBIN(uint8_t Value);
00051
00052     static void I2C_Write(uint8_t I2cAddr, uint16_t MemAddr, uint8_t UseLongAddr);
00053     static void I2C_WriteValue(uint8_t I2cAddr, uint16_t MemAddr, uint8_t UseLongAddr,
00054                                uint8_t Value, int Delay);
00055
00056     static bool I2C_ReadByte(uint8_t I2cAddr, uint16_t MemAddr, uint8_t UseLongAddr,
00057                              uint8_t *Value);
00058     static bool I2C_ReadByteDefault(uint8_t I2cAddr, uint16_t MemAddr, uint8_t
      UseLongAddr,
00059                                     uint8_t *Value, uint8_t DefaultValue);
00060
00061     static void I2C_SetBitAt(uint8_t I2cAddr, uint16_t MemAddr, uint8_t UseLongAddr,
00062                              uint8_t BitNum, bool Value, int Delay);
00063
00064     static void I2C_EEWriteBuffer(uint8_t I2cAddr, uint16_t MemAddr,
00065                                   byte *Data, byte Length);
00066     static void I2C_EEReadBuffer(uint8_t I2cAddr, uint16_t MemAddr,
00067                                  byte *Data, int Length);
00068
00069
00070     static uint8_t bcdToDec(uint8_t Value);
00071     static uint8_t dec2bcd(uint8_t num);
00072 };
00073
00074 #endif
00075
```

5.37 simple_snap.cpp File Reference

```
#include <Arduino.h>
#include <Camera.h>
#include <CameraVC0706.h>
#include <SoftwareSerial.h>
#include <HardwareSerial.h>
#include <SD.h>
```

Include dependency graph for simple_snap.cpp:



Functions

- void showUsage (SoftwareSerial *out, unsigned char section)
- unsigned char readParam (SoftwareSerial *in)
- void setup ()
- void loop ()

5.37.1 Function Documentation

5.37.1.1 void loop ( )

Definition at line 157 of file simple_snap.cpp.

5.37.1.2 unsigned char readParam (SoftwareSerial * *in*)

Definition at line 53 of file simple_snap.cpp.

5.37.1.3 void setup ( )

Definition at line 63 of file simple_snap.cpp.

5.37.1.4 void showUsage (SoftwareSerial * *out*, unsigned char *section*)

Definition at line 8 of file simple_snap.cpp.

5.38 simple_snap.cpp

```
00001 #include <Arduino.h>
00002 #include <Camera.h>
00003 #include <CameraVC0706.h>
00004 #include <SoftwareSerial.h>
00005 #include <HardwareSerial.h>
00006 #include <SD.h>
00007
00008 void showUsage(SoftwareSerial *out, unsigned char section) {
00009     switch(section) {
00010         case 0:
00011             out->println("'T' > TV off/on.");
00012             out->println("'o' > Output Resolution.");
00013             out->println("'t' > Take a Picture.");
00014             out->println("'M' > Set Motion Monitoring.");
00015             out->println("'m' > Get Motion Monitoring.");
00016             out->println("'d' > Detect Monitoring.");
00017             out->println("'H' > Set Horizontal Mirror Status.");
00018             out->println("'h' > Get Horizontal Mirror Status.");
00019             out->println("'r' > Reset.");
00020             out->println("'I' > Set Image Compression.");
00021             out->println("'i' > Get Image Compression.");
00022             out->println("'v' > Version.");
00023             break;
00024         case 1:
00025             out->println("TV:");
00026             out->println("'0' > On.");
00027             out->println("'1' > Off.");
00028             break;
00029         case 2:
00030             out->println("Output Output Resolution:");
00031             out->println("'0' > 160x120.");
00032             out->println("'1' > 320x240.");
00033             out->println("'2' > 640x480.");
00034             break;
00035         case 3:
00036             out->println("Set Motion Monitoring:");
00037             out->println("'0' > On.");
00038             out->println("'1' > Off.");
00039             break;
00040         case 4:
00041             out->println("Set Horizontal Mirror Status:");
00042             out->println("'0' > On.");
00043             out->println("'1' > Off.");
00044             break;
00045         case 5:
00046             out->println("Set Image Compression:");
```

```
00047             out->println("'1..9' > n * 10.");
00048             break;
00049     }
00050
00051 }
00052
00053 unsigned char readParam(SoftwareSerial *in) {
00054     unsigned char op = 0;
00055     while(!in->available());
00056     op = in->read();
00057     while(in->available()) {
00058         in->read();
00059     }
00060     return op;
00061 }
00062
00063 void setup() {
00064     SoftwareSerial serial(2, 3);
00065     CameraVC0706 cam(&Serial, &serial);
00066     serial.begin(115200);
00067     cam.begin(115200);
00068     serial.println("Camera initialized.");
00069     unsigned char c, arg0, arg1;
00070     while (true) {
00071         showUsage(&serial, 0);
00072         arg0 = readParam(&serial);
00073
00074         switch (arg0) {
00075
00076             case 'T':
00077                 showUsage(&serial, 1);
00078                 arg1 = readParam(&serial);
00079                 c = (arg1 == '1') ? 1 : 0;
00080                 serial.println(cam.setTVOutput(c) ? "OK" : "Error");
00081                 break;
00082
00083             case 'o':
00084                 showUsage(&serial, 2);
00085                 arg1 = readParam(&serial);
00086                 switch(arg1) {
00087                     case '0':
00088                         c = CameraVC0706::RES_160X120;
00089                         break;
00090                     case '1':
00091                         c = CameraVC0706::RES_320X240;
00092                         break;
00093                     case '2':
00094                         c = CameraVC0706::RES_640X480;
00095                         break;
00096                     default:
00097                         c = CameraVC0706::RES_160X120;
00098                 }
00099                 serial.println(cam.setOutputResolution(c) ? "OK" : "Error");
00100                 break;
00101
00102             case 't':
00103                 serial.println("Soon!");
00104                 break;
00105
00106             case 'M':
00107                 showUsage(&serial, 3);
00108                 arg1 = readParam(&serial);
00109                 c = (arg1 == '1') ? 1 : 0;
00110                 serial.println(cam.setMotionMonitoring(c) ? "OK" : "Error");
00111                 break;
00112
00113             case 'm':
00114                 serial.println(cam.getMotionMonitoringStatus());
00115                 break;
00116
00117             case 'd':
00118                 serial.println("Soon!");
00119                 break;
00120
00121             case 'H':
00122                 showUsage(&serial, 4);
00123                 arg1 = readParam(&serial);
00124                 c = (arg1 == '1') ? 1 : 0;
00125                 serial.println(cam.setHorizontalMirror(
      CameraVC0706::UART, c) ? "OK" : "Error");
00126                 break;
00127
00128             case 'h':
00129                 serial.println(cam.getHorizontalMirrorStatus());
00130                 break;
00131
00132             case 'r':
```

```
00133                 serial.println(cam.reset() ? "OK" : "Error");
00134                 break;
00135
00136             case 'I':
00137                 showUsage(&serial, 5);
00138                 arg1 = readParam(&serial);
00139                 c = arg1 - '0';
00140                 serial.println(cam.setCompression(c*10) ? "OK" : "Error");
00141                 break;
00142
00143             case 'i':
00144                 serial.println(cam.getCompression());
00145                 break;
00146
00147             case 'v':
00148                 serial.println(cam.getVersion());
00149                 break;
00150
00151             default:
00152                 serial.println("Not understood.");
00153         }
00154     }
00155 }
00156
00157 void loop() {
00158 }
```

Index