1 Function Descriptions

Table 1: view_dump_thread.c

Function Name	Description
`int GetClockMs()`	Collects the current time
`int LengthHelper()`	Determines the number of digits in the number of milliseconds
`void *SaveImageThreadFcn()`	Saves the captured images
`void *SendImageThreadFcn()`	Sends an image stream over UDP

Table 2: VDMA.c

Function Name	Description
`int InitTcpImageServer()`	Initializes the TCP protocol on the server side for sending images
`int InitTcpImgDataServer()`	Initializes the TCP protocol on the server side for sending image data
`void SendTcpImageFromServer()`	Sends an image from server to client over TCP
`void SendTcpImgDataFromServer()`	Sends image data from server to the client over TCP
`int SendImage()`	Sending an image over UDP
`int SendImage_mem()`	Sends image data over TCP
`int SaveJpgImage()`	Saves as an JPG image and sends it via TCP
`int SaveJpgImageData()`	Sends JPG data via TCP
`int SaveBmpImage()`	Saves an BMP image and sends it via TCP
`int SaveBmpImageData()`	Sends BMP data via TCP

Table 3: ProcessImage.c

Function Name	Description
`BYTE* ConvertBMPToRGBBuffer()`	Converts BGR buffer to RGB format
`int ConvertImage()`	Prepares the image for conversion
`int ConvertImageData()`	Prepares the image data for conversion
`int write_jpeg_file_img()`	converts and compresses a BMP image to JPG
`int write_jpeg_file()`	converts and compresses BMP image data to JPG

2 API Functions

GetClockMs

Description	The function calculates the current millisecond by a call to the function clock_gettime().
Synopsis	int GetClockMs()
Parameters	-
Return value	uint64_t, current millisecond. Upon failure: 0.

LengthHelper

Description	Determines the length of the millisecond number
Synopsis	int LengthHelper(int x)
Parameters	x: the return value from GetClockMs(), the amount of milliseconds
Return value	1, 2 or 3. Upon failure: -1 or 0.

SaveImageThreadFcn

Description	Main function for sending images over TCP. It defines the filenames with milliseconds added to the names and makes a function call to one of the image feed functions. Functions that are used by SendImageThreadFcn are GetClockMs(), LengthHelper(), SaveBmpImageData(), SaveBmpImage(), SaveJpgImageData() and SaveJpgImage().
Synopsis	void *SaveImageThreadFcn(void *arg)
Parameters	
Return value	-

SendImageThreadFcn

Description	Makes a function call to SendImage() when the variable stream is true in the main loop.
Synopsis	void *SendImageThreadFcn(void *arg)
Parameters	
Return value	-

InitTcpImageServer

Description	Initialization of the server for image transfering. Establishes a socket connection with the client through the functions socket(), setsockopt(), bind(), listen() and accept(). setsockopt uses the parameter SO_REUSEADDR for the purpose of reusing the address for the second image in an image pair through the same connection session. Finally InitTcpImageServer makes a function call to SendTcpImageFromServer() to send the images. The function is being used by SaveBmpImage() and SaveJpgImage().
Synopsis	int InitTcpImageServer(char *filename0)
Parameters	filename0: a char pointer of the filename that is passed over to the function SendTcpImageFromServer() which uses the address to send the image.
Return value	0. Upon failure: 1.

InitTcpImgDataServer

Description	Initialization of the server for image data transfering. Establishes a socket connection with the client through the functions socket(), setsockopt(), bind(), listen() and accept(). setsockopt uses the parameter SO_REUSEADDR for the purpose of reusing the address for the second image in an image pair through the same connection session. Finally InitTcpImageServer makes a function call to SendTcpImgDataFromServer() to send the images. The function is being used by SaveBmpImageData() and SaveJpgImageData().
Synopsis	int InitTcpImgDataServer(unsigned char *memory_data, char *name, unsigned long mem_size)
Parameters	memory_data: pointer to a buffer containing the image data to be sent. name: pointer of the filename that will point to the memory address of the image. mem_size: size of the image data to be sent for the client to know how much data to receive.
Return value	0. Upon failure: 1.

SendTcpImageFromServer

Description	Communicates with the client to send the image pair. The function sends image size and filename and receives verifications. Finally the image is divided into smaller packets that are being sent to the client. The function is being called by InitTcpImageServer().
Synopsis	void SendTcpImageFromServer(int socket, char* filename)
Parameters	socket: the return value from accept(). connection has failed if socket ¡ 0. filename: pointer of the filename that will point to the memory address of the image.
Return value	None. Error handling by control checks.

SendTcpImgDataFromServer

Description	Communicates with the client to send the image data pair. The function sends image size and filename and receives verifications. Finally the image data is divided into smaller packets that are being sent to the client. The function is being called by InitTcpImgDataServer().
Synopsis	void SendTcpImgDataFromServer(int socket, unsigned char *memo_data, char *name, unsigned long mem_size)
Parameters	socket: the return value from accept(). connection has failed if socket ¡ 0. memo_data: pointer to a buffer containing the image data to be sent. name: pointer to the filename. mem_size: size of the image data to be sent for the client to know how much data to receive.
Return value	None. Error handling by control checks.

SendImage

Description	Takes the image data from the shared memory and sends the data in a number of sequences through UDP for the live image stream.
Synopsis	int SendImage(uint32_t BaseAddress, uint16_t width, uint16_t height, uint16_t bpp)
Parameters	BaseAddress: The offset from where the first image data array starts in the shared memory. To be used for mapping the image data. width: width of image. height: height of image. bpp: bytes per pixel.
Return value	0. Upon failure: -1

SaveJpgImage

Description	Changes the extension to .jpg, picks up the two image data arrays from the shared memory, calls ConverImage() for format conversion from BMP to JPG and finally calls InitTcpImageServer() for file transfer. SaveJpgImage() is called from within SaveImageThreadFcn() and uses the functions ConvertImage() and InitTcpImageServer(). The function saves the image to file before transfer.
Synopsis	int SaveJpgImage(uint32_t BaseAddress,uint32_t BaseAddress1, char* file_name,char* file_name1, uint16_t width, uint16_t height, uint16_t bpp, uint8_t scale)
Parameters	BaseAddress: The offset from where the first image data array starts in the shared memory. To be used for mapping the image data. BaseAddress1: The offset from where the second image data array starts in the shared memory. To be used for mapping the image data. file_name: Pointer of first filename, gets the extension changed to .jpg. Later used to open the image file for sending. file_name1: Pointer of second filename, gets the extension changed to .jpg. Later used to open the image file for sending. width: width of image. height: height of image. bpp: bytes per pixel. scale: scaling of image, default: 1.
Return value	0. Upon failure: -1

SaveJpgImageData

Description	Changes the extension to .jpg, picks up the two image data arrays from the shared memory, calls ConverImage() for format conversion from BMP to JPG and finally calls InitTcpImageServer() for data transfer. SaveJpgImage-Data() is called from within SaveImageThreadFcn() and uses the functions ConvertImageData() and InitTcpImg-DataServer(). The function sends the image data without saving to file.
Synopsis	int SaveJpgImageData(uint32_t BaseAddress,uint32_t BaseAddress1, char* file_name,char* file_name1, uint16_t width, uint16_t height, uint16_t bpp, uint8_t scale)
Parameters	BaseAddress: The offset from where the first image data array starts in the shared memory. To be used for mapping of the image data. BaseAddress1: The offset from where the second image data array starts in the shared memory. To be used for mapping of the image data. file_name: Pointer of first filename, gets the extension changed to .jpg. file_name1: Pointer of second filename, gets the extension changed to .jpg. width: width of image. height: height of image. bpp: bytes per pixel. scale: scaling of image, default: 1
Return value	0. Upon failure: -1

SaveBmpImage

Description	Picks up the two image data arrays from the shared memory and creates two new files, one for each image in the image pair. The function uses predefined file header and info header that are being saved to the files together with the BMP data. Calls InitTcpImageServer() for data transfer. SaveBmpImage() is called from within SaveImageThreadFcn() and uses the function InitTcpImageServer().
Synopsis	int SaveBmpImage(uint32_t BaseAddress,uint32_t BaseAddress1, char* filename,char* filename1, uint16_t width, uint16_t height, uint16_t bpp, uint8_t scale)
Parameters	BaseAddress: The offset from where the first image data array starts in the shared memory. To be used for mapping of the image data. BaseAddress1: The offset from where the second image data array starts in the shared memory. To be used for mapping of the image data. filename: Pointer of first filename. Later used to open the image file for sending. filename1: Pointer of second filename. Later used to open the image file for sending. width: width of image. height: height of image. bpp: bytes per pixel. scale: scaling of image, default: 1
Return value	0. Upon failure: -1

SaveBmpImageData

Description	Picks up the two image data arrays from the shared memory. The function then calls InitTcpImgDataServer() for data transfer. The predefined file header and info header are used in the client where they are saved as files together with the BMP data after the data transfer. SaveBmpImageData() is called from within SaveImageThreadFcn() and uses the function InitTcpImgDataServer().
Synopsis	int SaveBmpImageData(uint32_t BaseAddress,uint32_t BaseAddress1, char* filename,char* filename1, uint16_t width, uint16_t height, uint16_t bpp, uint8_t scale)
Parameters	BaseAddress: The offset from where the first image data array starts in the shared memory. To be used for mapping of the image data. BaseAddress1: The offset from where the second image data array starts in the shared memory. To be used for mapping of the image data. filename: Pointer of first filename. filename1: Pointer of second filename. width: width of image. height: height of image. bpp: bytes per pixel. scale: scaling of image, default: 1
Return value	0. Upon failure: -1

ConvertBMPToRGBBuffer

Description	Converts the BGR Bitmap buffer to RGB by swapping the R and B bytes. Uses the global variable image_buffer which is assigned in ConvertImage().
Synopsis	BYTE* ConvertBMPToRGBBuffer (BYTE* Buffer, int width, int height)
Parameters	Buffer: the BGR Bitmap buffer to be converted to RGB format. width: the width in pixels. height: the height in pixels.
Return value	An RGB buffer. Upon failure: NULL

ConvertImage

Description	Assigns the image array to a global variable which is used in the function ConvertBMPToRGBBuffer(). Then it makes a function call to write_jpeg_file_img() for conversion.
Synopsis	int ConvertImage(unsigned char *img, unsigned char *img_arr, int image_width, int image_height, int bpp)
Parameters	img: pointer of the filename to create a new file. img_arr: the image data, stored in the global variable image_buffer. image_width: not used. image_height: not used. bpp: not used.
Return value	0.

ConvertImageData

Description	Assigns the image array to a global variable which is used in the function ConvertBMPToRGBBuffer(). Then it makes a function call to write_jpeg_file() for conversion.
Synopsis	int ConvertImageData(unsigned char *img, unsigned char *img_arr, int image_width, int image_height, int bpp, unsigned char * mem, unsigned long *mem_size)
Parameters	img: pointer of the filename. img_arr: the image data, stored in the global variable image_buffer. image_width: not used. image_height: not used. bpp: not used. mem: an array to be passed on to write_jpeg_file(). Will later be filled with the converted image data. mem_size: returned value from write_jpeg_file() to return with the size of the converted data.
Return value	0.

write_jpeg_file_img

Description	Converts the BMP file format to JPG. Uses the returned buffer from ConvertBMPToRGBBuffer(), which contains the global variable image_buffer, to compress and convert the data and then write to file.
Synopsis	int write_jpeg_file_img(unsigned char *filename)
Parameters	filename: pointer of filename to create a file.
Return value	Image data written to file. Upon failure: -1

write_jpeg_file

Description	Converts the BMP file format to JPG. Uses the returned buffer from ConvertBMPToRGBBuffer(), which contains the global variable image_buffer, to compress and convert the data and then return it for sending without writing to file.
Synopsis	int write_jpeg_file(unsigned char *filename, unsigned char *mem_out)
Parameters	filename: not used. mem_out: the converted image data which is sent back as a parameter, first through write_jpeg_file() and then ConvertImageData().
Return value	the size of the converted image data