Software Design

1. Code structure

- **COMMANDS:** This part specifies the different commands used to transmit the data between the OBC and the Camera. These commands are specified on the datasheet, and are sent using the “Basic” functions. These commands are declared on the camerav2.c file, and a more detailed explanation can be found there.
- **ADDRESSES:** The directions of the camera where the data is stored. It is stored the data itself, the resolution and the compressibility in the memory. Actually and due to the implementation structure, the camerav2.c uses the 0x082000 address, which has been selected by the OBC team.
- **“BASIC'' Functions:** These functions are the ones which directly communicate with the camera and the OBC. These functions are the ones which execute the command directly with the camera, and these functions are used internally inside camerav2.c . One example would be reset, which sends the information directly to the Camera.
- **“COMPOSITE” Functions:** These functions are the main ones, which the OBC will use on the program. These functions are composed of BASIC functions. These functions are InitCam and GetPhoto. These functions, due to its importance, will be explained after.
- **“CORRECTION” Functions:** These functions are used to check if the OBC correctly receives the photo from the camera, and what to do if the error occurs. They are checkACK() and errP(). These functions, due to its importance, will be explained after.
- **“EXTRA” Functions:** These functions are used during the full code, which are used mainly for storeData, either the infoBuffer or the FlashMemory. These functions will be explained later.

Taking into account these summary of the functions, a detailed explanation of each case will be explained.

2. Summary of functions

A summary of the functions developed is presented now:

FUNCTION	TYPE	DESCRIPTION

bool reset(UART_HandleTypeDef huart)	BASIC	Executes reset command.
bool getVersion(UART_HandleTypeDef huart)	BASIC	Executes getVersion command.
bool setResolution(UART_HandleTypeDef huart)	BASIC	Sets the resolution from the user.
bool setCompressibility(UART_HandleTypeDef huart)	BASIC	Sets compression from the user.
bool startCapture(UART_HandleTypeDef huart)	BASIC	Executes startCapture command
bool getDataLength(UART_HandleTypeDef huart)	BASIC	Obtained the length of an image.
bool getData(UART_HandleTypeDef huart)	BASIC	Obtains an image.
bool stopCapture(UART_HandleTypeDef huart)	BASIC	Executes stopCapture command.
bool initCam(UART_HandleTypeDef huart, uint8_t res, uint8_t comp, uint8_t *array)	COMPOSITE	Initializes the camera.
uint16_t getPhoto(UART_HandleTypeDef huart, uint8_t *infoarray)	COMPOSITE	Executes the getPhoto structure, obtaining the photo, length and storing the photo inside Flash Memory.
uint16_t storeDataFlash() *might be removed*	EXTRA	Stores the data inside Flash Memory.
void storeInfo(uint8_t info)	EXTRA	Stores info inside array.
bool checkACK(UART_HandleTypeDef huart, uint8_t c1, uint8_t c2, uint8_t c3, uint8_t c4, uint8_t c5)	CORRECTION	Checks ACK from camera.
bool err(UART_HandleTypeDef huart)	CORRECTION	Executes error protocol.

Table 1: Designed Software functions

Basic functions

As explained before, we have the commands which are sent to the camera, and the commands expected. Anyway, there should be a function which sends that information to the camera using the UART protocol. The STM32 has a native library which implements this (HAL_UART_Transmit, HAL_UART_Receive). In this case, a specific implementation of this function for each command has been done.

A general view of these functions is presented below:

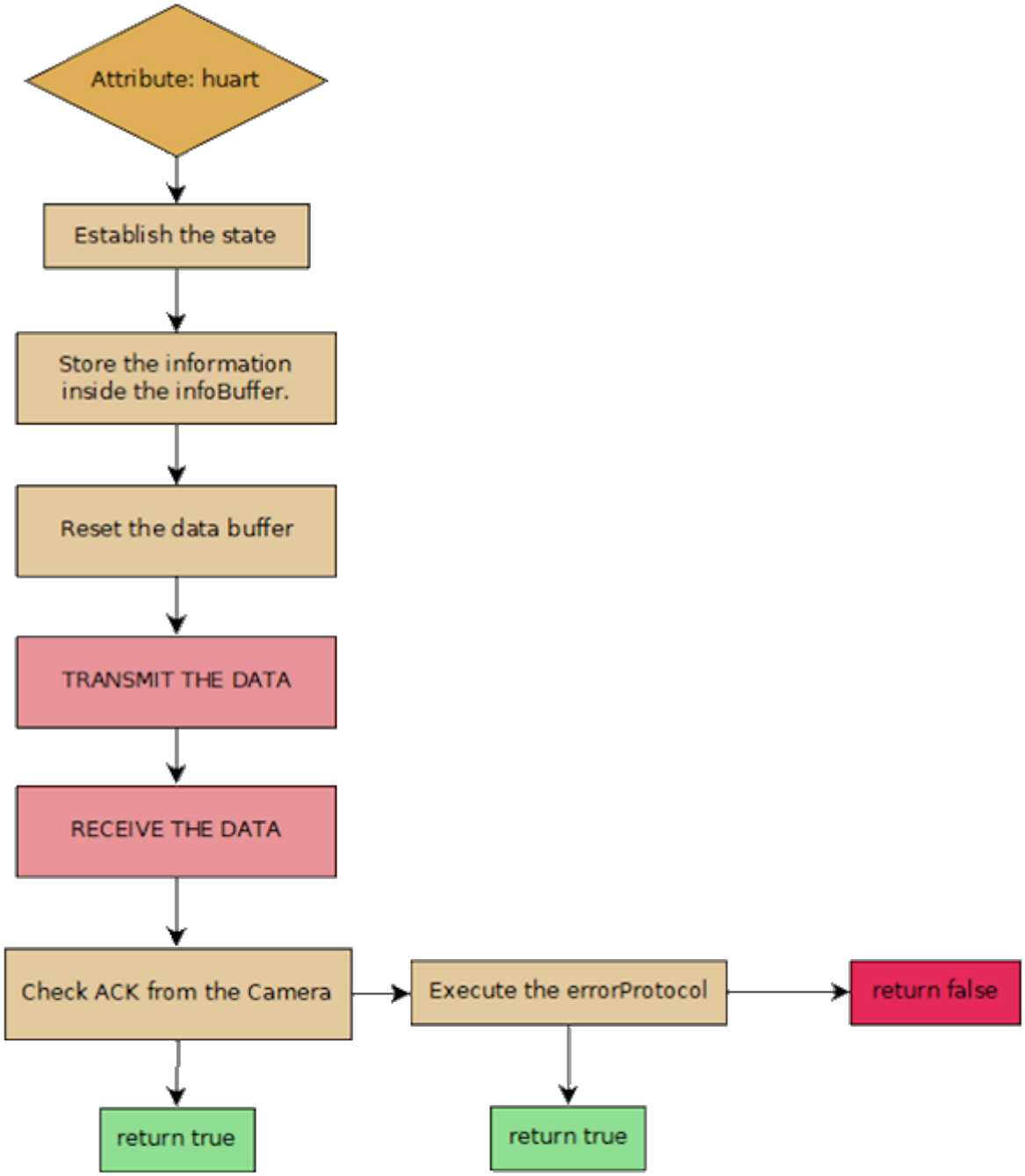

Figure 1: Basic Functions Block Diagram

This scheme can be used for each basic function besides the getData function, at this current point in time., varying the state and the transmit data Buffer and the receive data buffer.
It is known that a general buffer for all the information received can be implemented, but it can also generate numerous problems on the implementation. As it is not necessary to implement and it will complicate the code, different buffers are applied for different data.
The infoBuffer and state variable purpose will be explained in detail in other parts of the

documentation.

getData

The getData function complexity severely differs from other basic functions due to the implementation of circular DMA. In contrast to other functions, and due to the structure of the ones used to store data in the flash memory, two relevant buffers have been created in order to store and manage data before flashing. The use of DMA requires callbacks in order to flash data without losing parts of it. In these callbacks, executed when the first buffer is half full and when it is full, will cause an interrupt (not desired but necessary) where the flashing will occur. Here we will make use of an auxiliary buffer in order to copy the second half of the first buffer onto it in order to not cause overlap or flashing of repeated data.

Note that the callbacks are not called when all the data or half is received, they are called in response to the fullness of the buffer (as long as its length coincides with the amount of data specified in the HAL DMA functions). To solve the problem and know how much data shall be received and DMA stopped we make use of the prior fuction, getDataLenght, which will provide us with the total length of the picture. As callbacks are executed, an index will be increased and through comparisons of it, the current callback and the length it'll be determined when to stop the DMA and the execution of the function.

A comprehensive slightly simplified scheme is provided next:

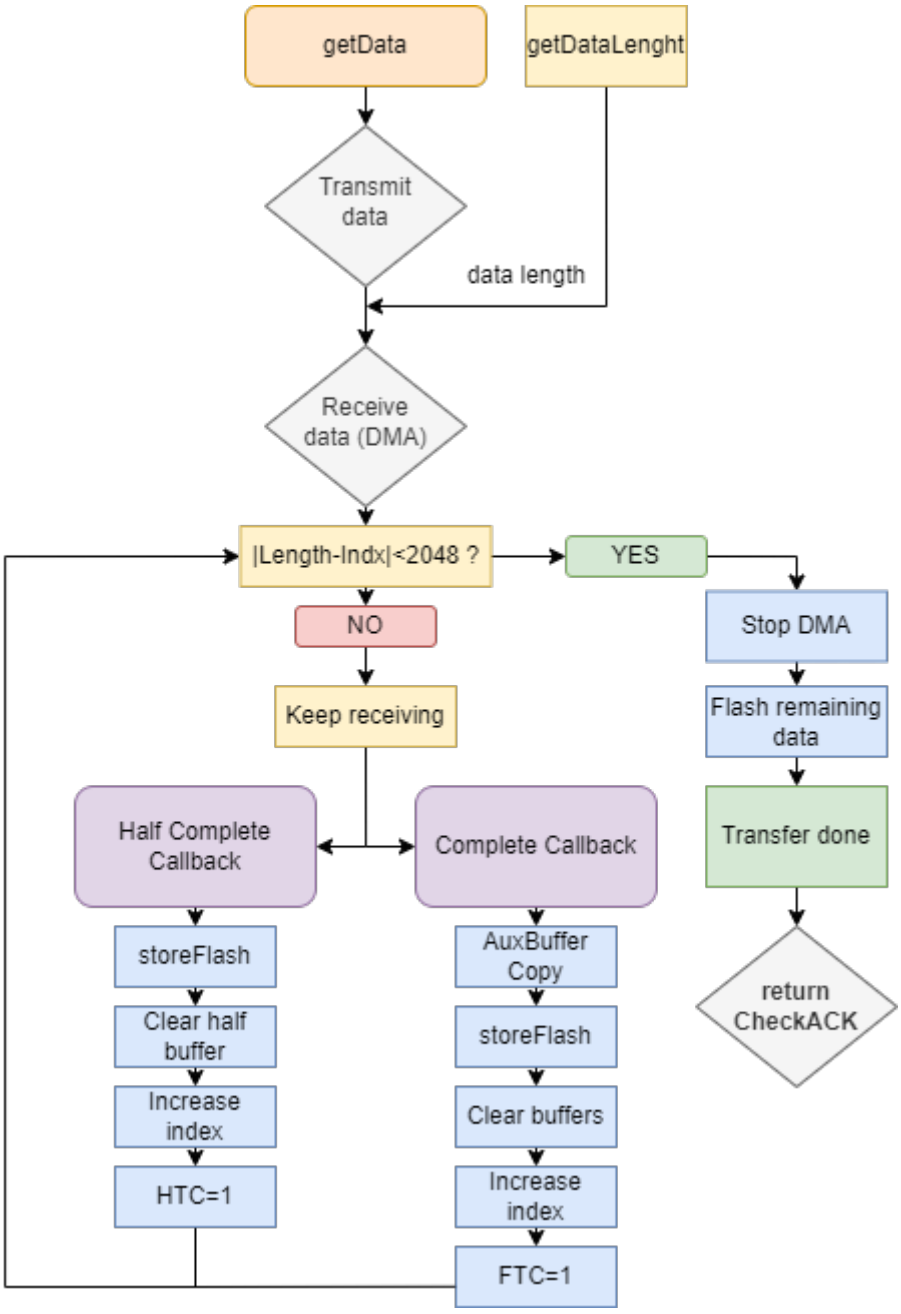

Figure 2: getData Function Block Diagram

Composite functions

These functions used different basic functions in order to follow the protocol from the mentioned datasheet.

initCam

The initCam function initializes the camera with the resolution and the compressibility:
In this case, the initCam is implemented using the huart, the resolution and compressibility needed, and an array in which errors will be stored.
A general scheme of the code is shown below:

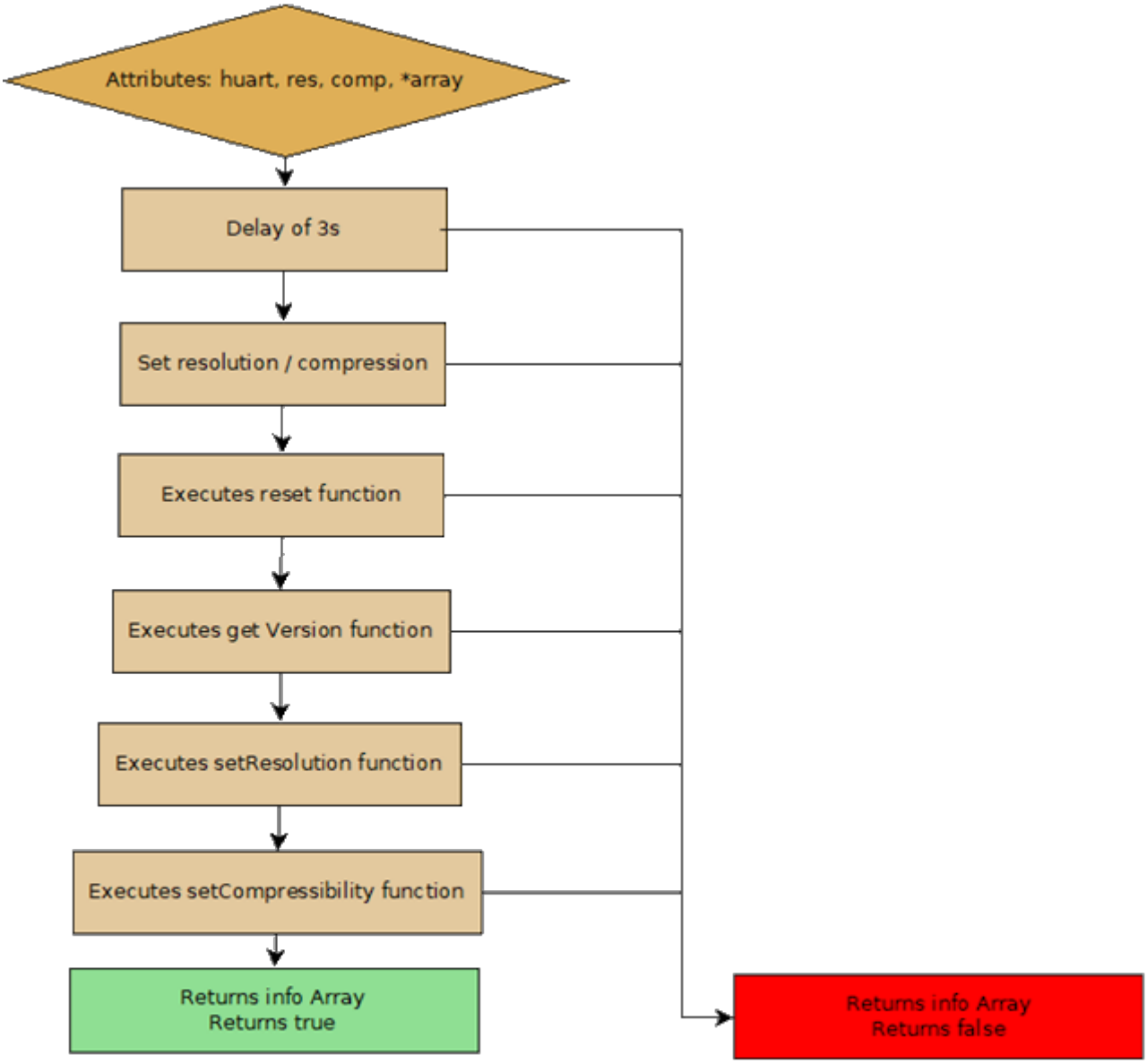

Figure 3: initCam Block Diagram

getPhoto

The getPhoto function is implemented using an UART and an info array attribute. This array is where the buffer of errors will be stored. This function is in charge of executing the order to take the photo, taking it, storing it, and executing the order to stop the camera. A general scheme of the code is shown below:

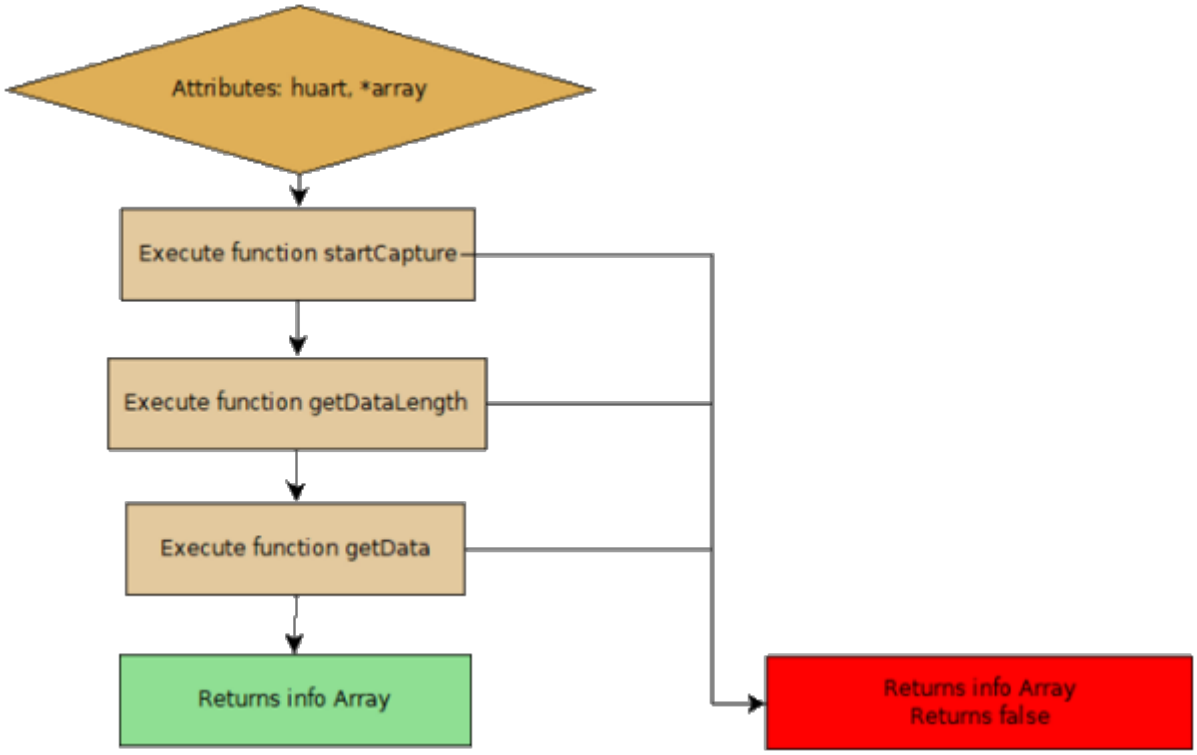

Figure 4: getPhoto Block Diagram

Revision #1
Created 15 November 2024 17:56:07 by roger.almirall
Updated 15 November 2024 17:56:17 by roger.almirall