STM32 Based Real Time Parameters display on Real Time Video

Meghanand A. Bhamare
Department of Electronics and Communication
Amity University Mumbai, Bhatan, Panvel, Maharashtra,India
mabhamare@mum.amity.edu

***Abstract*—This paper discuss about implementation of the real time parameters like speed, temperature, pressure, distance on the real time video. Here the parameter value is embedded with the images coming from the camera and then displayed over LCD. To do this we make changes in the Bitmap of image being received from the camera. This application is developed on STM3240G-EVAL board using onboard peripherals.**

***Keywords* – STM32F4, DCMI, DMA, FSMC,NDTR**

I Introduction

This application is developed on STM3240G-EVAL board using onboard peripherals and interfaces like DMA, FSMC DCMI, SRAM, Camera and LCD. First, image is captured from camera interfaced with STNM32 using DCMI interface of STM32F4[1]. One channel of DMA works as a data link between DCMI (taking data from camera) and external SRAM which is interfaced with the STM32 using FSMC. Images is processed by STM32 as per requirement and and with the help of another channel of DMA it will be displayed on LCD. Camera can be operated in two modes i.e. Snapshot mode or continuous image capturing mode. In former case we need to enable the image capturing command of DCMI (since the capturing stops automatically once one image is captured completely) when current image is processed and displayed on LCD. Whereas in latter case the image is captured continuously and we process and display the image in active period of Vsynch (Vsynch signal is responsible for image capturing). Regarding the display of text over image user can have control over font selection and color. Any data coming from a sensor or any other varying data can also be displayed in real time.

II Hardware used

The STM3240G-EVAL evaluation board is a complete demonstration and development platform for the STM32F4 series and includes an embedded STM32F407IGH6 high-performance ARM®Cortex™-M4F 32-bit microcontroller[2]. The full range of hardware features on the board is provided to help evaluate all peripherals (USB-OTG HS, USB-OTG FS, Ethernet, motor control, CAN, MicroSD Card™, smartcard, USART, Audio DAC, RS-232, IrDA, SRAM, MEMS, EEPROM… etc.) and develop your own applications. Extension headers make it possible to easily connect a daughterboard or wrapping board for your specific application. The in-circuit ST-LINK/V2 tool can be easily used for JTAG and SWD interface debugging and programming.

III Software overview

1) Initializing system:

DMA Setting: - Direct memory access (DMA) is used in order to provide high-speed data transfer between peripherals and memory and between memory and memory. In STM32F4 we have two DMA
controllers which have 16 streams in total (8 for each DMA controller), each dedicated to managing memory access requests from one or more peripherals. Here we are using DMA2 stream1 is used as a data link between Camera and SRAM and DMA 2 stream 4 is used between SRAM and LCD.

Each DMA transfer consists of three operations:

1. Loading from the peripheral data register or a location in memory, addressed through the DMA_SxPAR or DMA_SxM0AR register.
2. Storage of the data loaded to the peripheral data register or a location in memory addressed through the DMA_SxPAR or DMA_SxM0AR register.
3. Post-decrement of the DMA_SxNDTR register, which contains the number of transactions that still have to be performed.

Whenever a complete frame of image is captured we First disable the DMA2_Stream1 (Primary Channel) in order to read NDTR (no of data transfers remaining) Register and avoid any unwanted data transaction from DCMI to SRAM using DMA2 Stream1. Also disable the DMA2 Stream4 in order to configure it.

2) DCMI Setting:

The DCMI is used to receive data from CMOS camera module[4]. The digital camera interface uses two clock domains PIXCLK and HCLK. The signals generated with PIXCLK are sampled on the rising edge of HCLK once they are stable. An enable signal is generated in the HCLK domain, to indicate that data coming from themcamera are stable and can be sampled. The maximum PIXCLK period must be higher than 2.5 HCLK periods. The data flow is synchronized either by hardware using the optional HSYNC (horizontal synchronization) and VSYNC (vertical synchronization) signals.

3) DCMI interrupt handling:-

As mentioned earlier that the DCMI / Camera is operated in one of the two modes i.e. Snapshot and continuous capture mode. If DCMI in set to run in snapshot mode. In this mode, a single frame is captured (CM = '1' in the DCMI_CR register). After the CAPTURE bit is set in DCMI_CR, the interface waits for the detection of a start of frame before sampling the data. The camera interface is automatically disabled (CAPTURE bit cleared in DCMI_CR) after receiving the first complete frame. An interrupt is generated (IT_FRAME) if it is enabled. In case of an overrun, the frame is lost and the CAPTURE bit is cleared. When a frame is completed properly a frame complete interrupt is generated after receiving the interrupt we first clear the interrupt. Length of image captured is calculated using NDTR register of primary As DMA is used in between peripheral and memory hence maximum transfer will depend on completed transfer multipliedby peripheral data width. NDTR contains remaining transfers so if we subtract if from maximum count we will get completed transfers and each transfer is of 4 bytes. So

Data Transferred = ((0xFFFF - NDTR_value) * Perip_Data_Width)

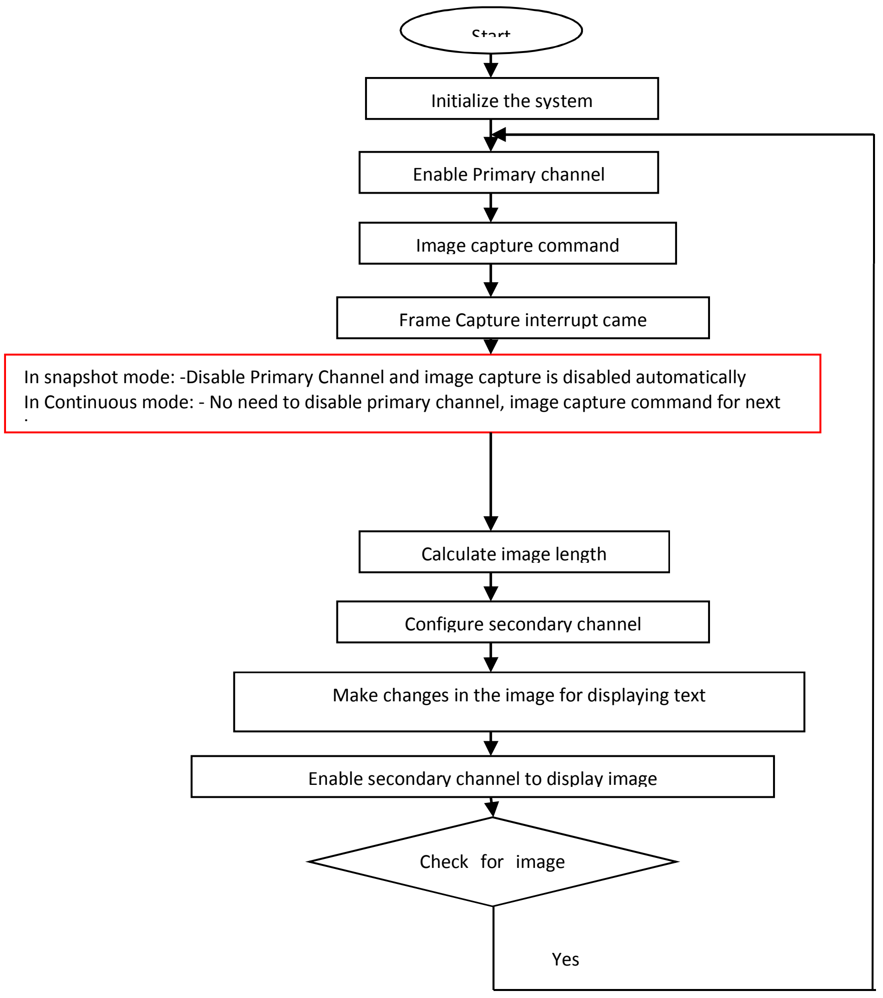
DMA will continue to transfer data flow from Camera to Memory till complete image is not captured. Image capture flag is used to indicate image is captured and stored in to memory. Calculate length of the image captured based on Image captured flag status. Then set DMA2 stream4 peripheral address as intended RAM address. Start making changes in image to display intended change. Enable DMA 2 stream 4 to display image on LCD. We use high interrupt status register HIFCR to check stream transfer complete interrupt flag. When it is set by hardware clear it and disable DMA 2 stream 4 to read NDTR register. Calculate the value of Data Transferred in this

Transaction and increment the counter accordingly by using the formula.

Data Transferred = ((0xFFFF - NDTR_value) * Perip_Data_Width)

In this case it will be 2 bytes as for LCD has 16 bit register. Transfer complete image with maximum transfer permitted is 64K at a time destination address to DMA, which will be starting address of memory added with length of the transferred image. Continue this until length of the captured image is greater than length of image transferred to LCD. Enable DMA 2 stream 4 to update registers. When complete image is transferred then disable DMA2 stream4 and enable DMA2 stream1 so as to capture next image

4) Flowchart


Start
Initialize the system
Enable Primary channel
Image capture command
Frame Capture interrupt came
In snapshot mode: -Disable Primary Channel and image capture is disabled automatically
In Continuous mode: - No need to disable primary channel, image capture command for next
Calculate image length
Configure secondary channel
Make changes in the image for displaying text
Enable secondary channel to display image
Check for image
Yes

5) Image modification:

A function is called to display a string along with location on LCD(X and Y co-ordinates)[3]. Intended font size for the text which is already decided is also passed through another function. Let us say font size is 16x24 where 24 is height of the character and 16 is width. It means if we refer to ASCII table for the same we can see 24 half words in the matrix, each one for one row. Character will be written as row to column approach means first half word in first row then next half word in next row and so on. Next thing will be to decide exact pixel location on LCD out of 153600 places (320*240*2). A function is called to decide the same based on condition that original co-ordinates passes to string location are less than maximum size of the screen. Location is decided based on following formula.

Location= ((X*320 + Y)*2);

For ex. If we want to write a character on location {10, 10} then pointer has to travel maximum width of 320 per X location and when it reaches to respective row, it has to travel respective column which is decided by Y. This will provide location for first pixel of first character.

Then we have to decide exact change at this location of image buffer. For that first we need to access first word of ASCII table for the same character. The respective word which has 16 locations will be masked with 0x0001 for every location and it will be decided whether respective pixel has to be modified or not. For Example if first word is 0xFFFF then it will be masked with 0x1 for every bit position and we come to know that every bit is 1 so all the pixels has to be modified. Modification means fill respective pixel location with color. For ex. if we want to use white color then pixel will be modified to 0xFFFF. Increase location to next value while keeping in mind maximum size of the column

Continue this procedure for all remaining words of the character ASCII table. Means take next word from table and find individual mask value for every bit and if it is '1' then modify respective location for given color. This will modify one character in the image at given location. Then we move to next character keeping in mind that when we write next character it should have sufficient location (minimum 16 columns and 24 rows) to appear completely. If it reaches to end of column we move on to next row and start writing fresh character. If it's end of all rows and columns then writing starts from first line.

IV Results

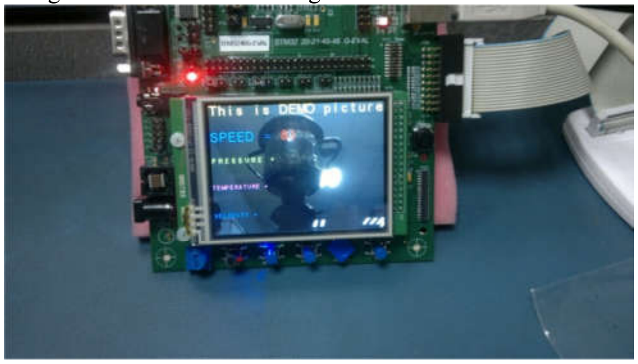
1) Timing Analysis:

Following table summarizes timing calculation for different sections of the program.

Pin No	Computed for	Time
PG8	Text Writing	4.9ms
PC7	DCMI Interrupt	139ms
PI9	Time required to capture one image (Primary Channel enabling to disabling)	121ms
PG6	Time to Display modified image	13ms
Total Time	Complete Loop	139ms
CN2 (6)	PCLK - DCMI	18MHz
CN4(4)	VSYNC - DCMI	64ms(high time) @ 14 Hz
CN2(36)	HYSNC-DCMI	43uS(high time) @ 4Khz
CN19/CN1(3)	WR-FSMC	5.8MHz
C19/CN1(4)	RD-FSMC	5.8MHz

2) Sample result

This picture shows snapshot of a video, where different parameters can be displayed on a video. Parameters value changes in real time and same gets modified on the video.



I References

[1] STM32F40XX Series Datasheet, *STMicroelectronics*, May 2012
[2]RM0090 STM32F40XX Series Reference manual, *STMicroelectronics*, Sep. 2011
[3]AN3241 Application Note, *STMicroelectronics*, July 2010
[4]UM1461 User Manual, *STMicroelectronics*, April 2012

V Conclusion

To display a real time parameter over video is always a challenge for maintaing frame rate. With the proposed algorithm it is possible to achieve frame rate around 8. As per DCMI theory VSYNC signal plays important role in capturing image and it can be noted that a frame is lost in case overrun. It happens as DCMI is set in snapshot mode and it is not enable till complete image is not captured, modified and displayed. So from timing analysis it can be observed that lot of time is wasted during capture of single image as DCMI is idle once image is captured till it is not being modified and displayed. Frame rate can be improved with the help of double buffer management.