Remote Terminal for Python 1300 Camera Module

Lukáš Kohout
Kohoutl@utia.cas.cz

Revision history

Rev.	Date	Author	Description
0	28.06.2017	L. Kohout	Document creation
1	10.07.2017	L. Kohout	Serial terminal settings
2			

Contents

Acknowledgement

This work has been supported from project PANACHE, project number ENIAC JU 621217 and MSMT 7H14006.

1 Introduction

A remote terminal for Python 1300 camera module demonstrator is a simple application of STM32756G-EVAL2 board [1]. It uses its touch screen providing a GUI (Graphics User Interface) to control remote device; 1300 camera module in this case. The application is based on the STM32CubeF7 package [2] compiled for STM32756G-EVAL2 board with the IAR EWARM compiler [3]. The remote device is controlled via RS232 interface.

2 Description

The demonstrator uses the STM32756G-EVAL2 LCD with the touch layer. It shows how to create GUI on STM32756G-EVAL2 board enabling control Python 1300 camera module (remote device based on different platform) in a short time. An example of running application is shown in Figure 1. The GUI provides two vertical sliders to control analog and digital gain of the camera and two buttons. The button called *Preview* invokes reading of current image from the camera module. The received image is displayed on the screen. The image has smaller resolution than the original camera resolution is. The camera resolution is 1280x1024 whereas the transferring and displaying image has resolution 213x170 (approximately six-times smaller). Pushing *Reinit* button reinitializes the camera module. All events on touchable controls are converted to commands using RS232 interface. The application is based on the Hello World example of the STemWin middleware of the STM32CubeF7 package.

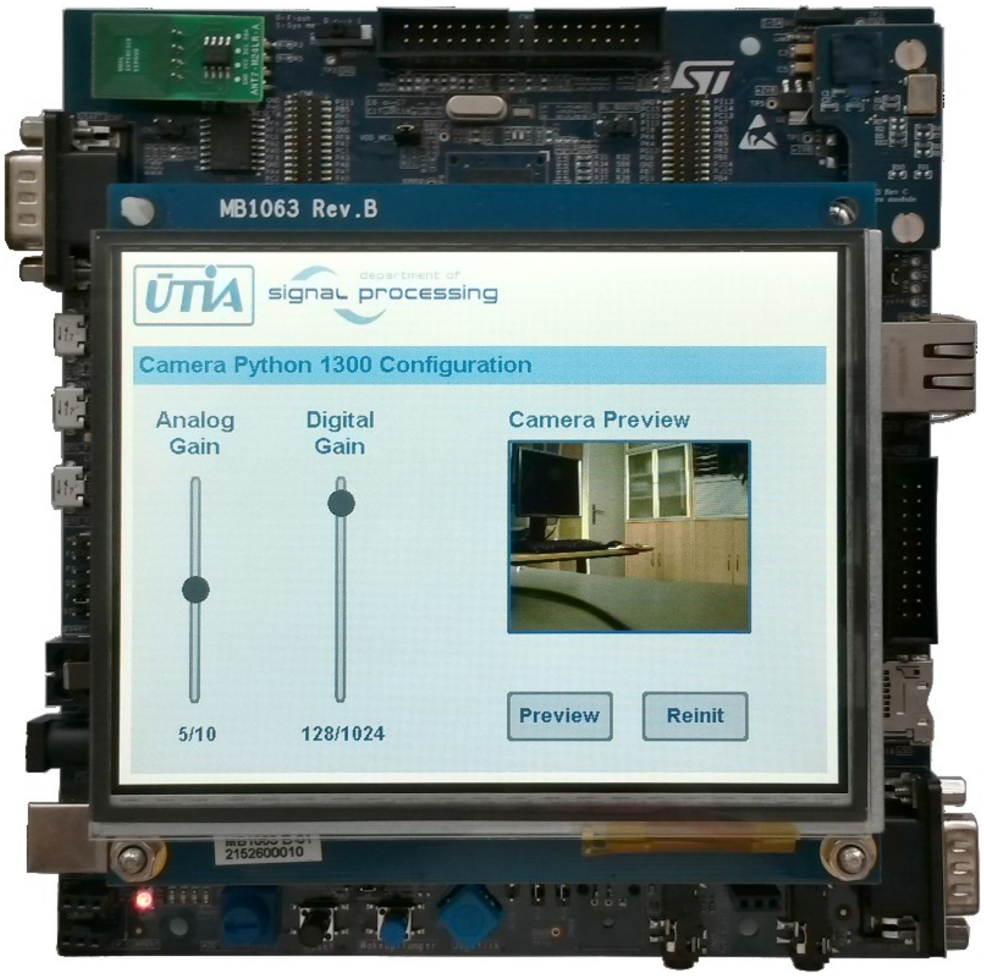


Figure 1: Remote terminal for Python 1300 camera module

2.1 Commands and Messages

To control a remote device, the terminal uses RS232 interface. The unit transforms the screen touches to the commands for the remote device and it reads messages form the remote device. Command always starts with **'!'** character and ends with **'*'** character (**!command*** for instance). All used commands are summarized in Table 1. Messages are remote device responds on the commands. Message starts with **'#'** character and ends with **'*'** character (for example **#message***). Table 2 presents implemented messages. Both, commands and messages, can be optionally delimited by **\CR\LF** sequence.

Table 1: Commands

Command	Description
!SAGxx*	Set Analog Gain, *xx* is decadic value of the analog gain in range 0 to 10
!SDGxxxx*	Set Digital Gain, *xxxx* is decadic value of the digital gain in range 0 to 1024
!RST*	Reinitialize camera module
!GIMG*	Get current Image of the camera

Table 2: Messages

Message	Description
#IMGdata…*	Response on !GIMG* command, where "data…" means bytes of the image. Concretely it is (213 * 170 * 3) Bytes (RGB – 3 Bytes per pixel)

3 Used Tools and Resources

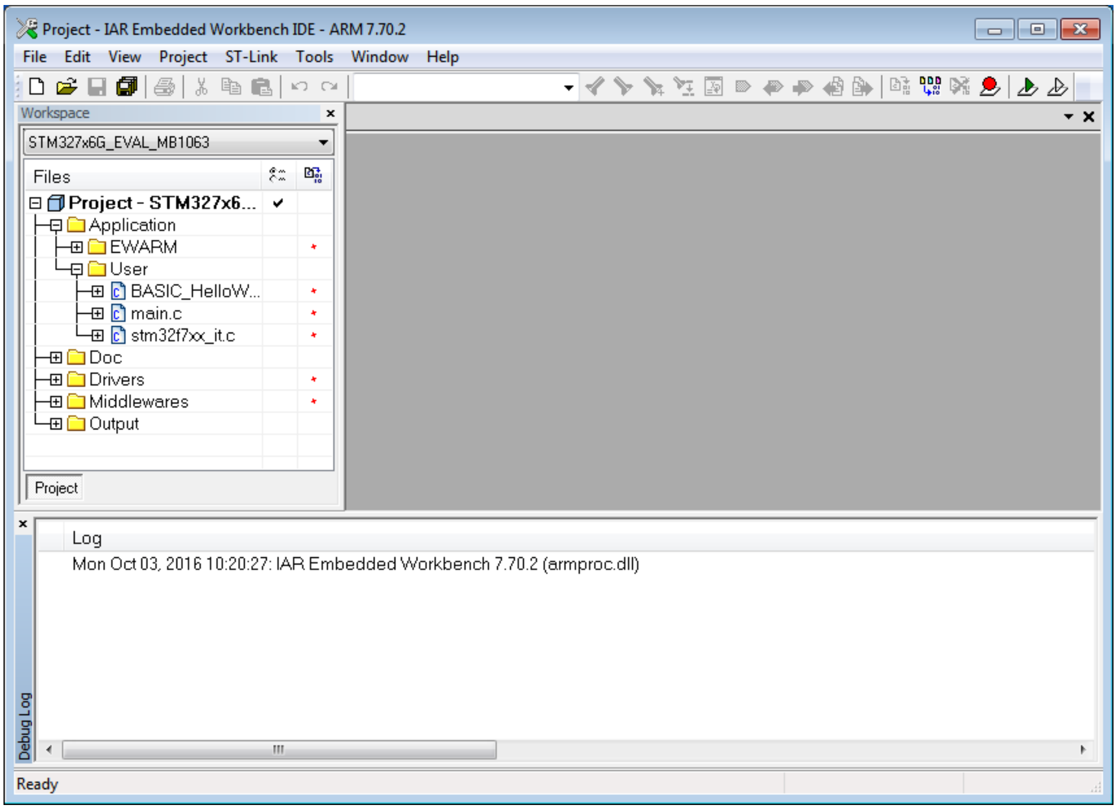
- STM32756G-EVAL2 board [1].
- STM32CubeF7 package is a firmware package for STM32F7 Series. It is downloadable from the STMicroelectronics web page [2]. The package gathers together all the generic embedded software components required to develop an application on STM32F7 microcontrollers.
- Commercial edition of the IAR Embedded Workbench, alternatively a free edition with restrictions to the 30-day time-limited evaluation without code size limitation [3].
- STM32756G-EVAL2_Cam_GUI package provided with this application note, the package description is in Section 5.

4 Implementation

To implement the remote terminal demo, follow the steps bellow.

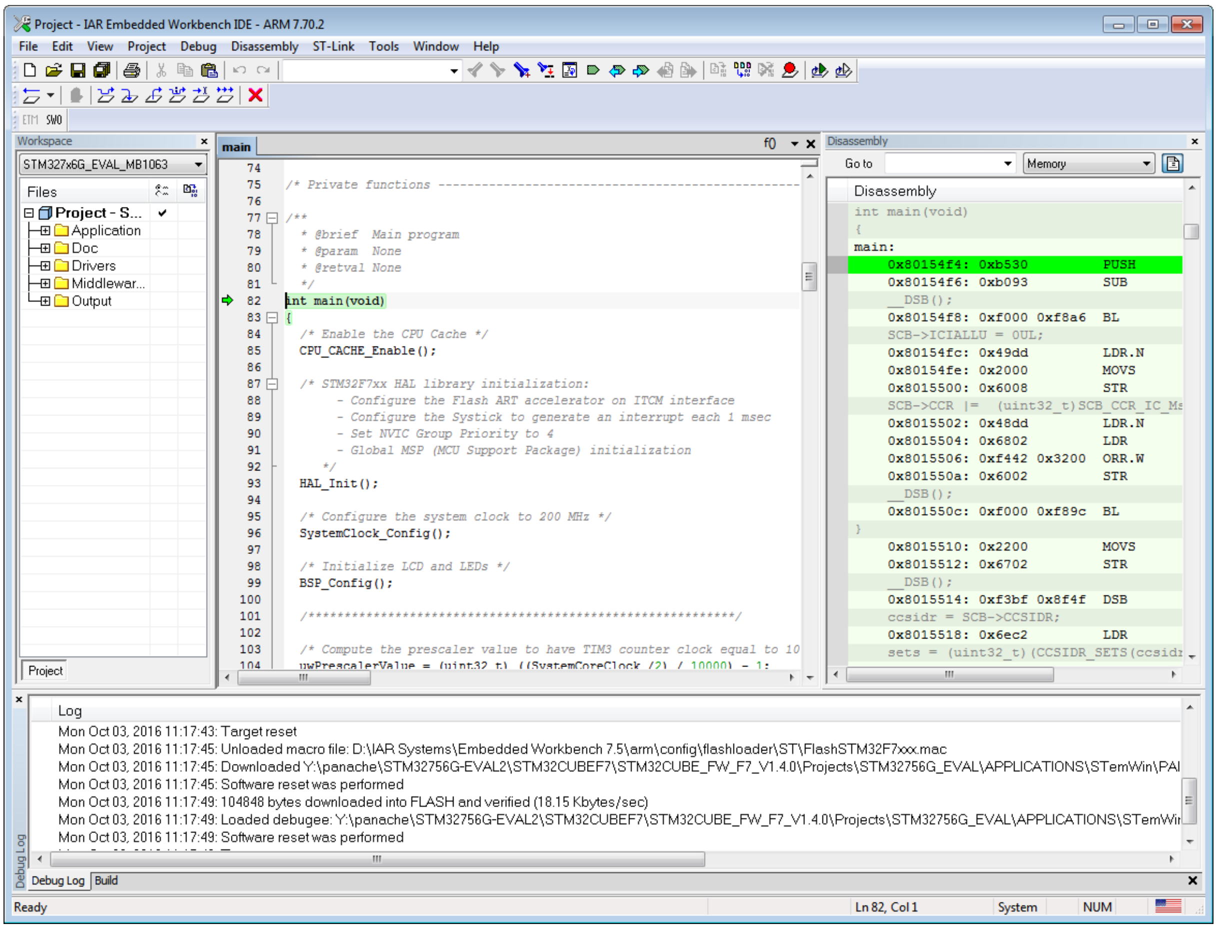
1. Download STM32CubeF7 package from its web page [2]. Used version of the package is 1.4.0. Decompress the package zip file. The package top folder is called *STM32Cube_FW_F7_V1.4.0*. In the text, the package path will be referenced as *$STM32Cube_path$*.
2. Go to the *$STM32Cube_path$\Projects\STM32756G_EVAL\Applications\STemWin* folder and make a copy of the *STemWin_HelloWorld* project folder. The name of the copy will be *cam_gui*.
3. Copy the content of the STM32756G-EVAL2_Cam_gui\sw\src folder to the *$STM32Cube_path$\Projects\STM32756G_EVAL\Applications\STemWin\cam_gui\Src* folder. Files *main.c* and *BASIC_HelloWorld.c* have already existed in the destination folder, rewrite them.

4. Copy the content of the STM32756G-EVAL2_Cam_gui\sw\inc folder to the *$STM32Cube_path$\Projects\STM32756G_EVAL\Applications\STemWin\cam_gui\Inc* folder. Rewrite all already existing files.
5. Start the IAR Embedded Workbench. In the text, this tool will be referenced as EWARM.
6. In the EWARM tool chain, open the *cam_gui* workspace, go to the menu *File→Open→Workspace…* Select the path to the *cam_gui* EWARM workspace description file *$STM32Cube_path$\Projects\STM32756G_EVAL\Applications\STemWin\cam_gui\EWARM\Project.eww*.
7. In the cam_*gui* EWARM workspace, there are missing files *btn.c*, *sldr.c*, *logo.c* and *stm32f7xx_hal_msp.c*. Inspect the workspace file tree, see Figure 2. Add the file to the workspace. In the workspace file tree, select folder *User*, right click on it and from the context menu select *Add→Add Files…* and choose the missing files located in *$STM32Cube_path$\Projects\STM32756G_EVAL\Applications\STemWin\cam_gui\Src* folder.
8. Build the project, menu *Project→Make*.
9. Connect the ST-LINK/V2-1 programming and debugging tool on the STM32756G-EVAL2 board. Plug the USB cable to the CN21 ST-LINK/V2-1 USB connector of the board.
10. Let the STM32756G-EVAL2 board configuration jumpers in their default settings, check it according to board manual [1].
11. Power the board up.



```
Mon Oct 03, 2016 10:20:27: IAR Embedded Workbench 7.70.2 (armproc.dll)
```

Figure 2: Project workspace files



```
/* Private functions ---------------------------------------------------------

/**
  * @brief  Main program
  * @param  None
  * @retval None
  */
int main(void)
{
  /* Enable the CPU Cache */
  CPU_CACHE_Enable();

  /* STM32F7xx HAL library initialization:
       - Configure the Flash ART accelerator on ITCM interface
       - Configure the Systick to generate an interrupt each 1 msec
       - Set NVIC Group Priority to 4
       - Global MSP (MCU Support Package) initialization
     */
  HAL_Init();

  /* Configure the system clock to 200 MHz */
  SystemClock_Config();

  /* Initialize LCD and LEDs */
  BSP_Config();

  /***************************************************************/

  /* Compute the prescaler value to have TIM3 counter clock equal to 10
  uwPrescalerValue = (uint32_t) ((SystemCoreClock /2) / 10000) - 1;
```

```
int main(void)
{
main:
    0x80154f4: 0xb530         PUSH
    0x80154f6: 0xb093         SUB
    __DSB();
    0x80154f8: 0xf000 0xf8a6  BL
    SCB->ICIALLU = 0UL;
    0x80154fc: 0x49dd         LDR.N
    0x80154fe: 0x2000         MOVS
    0x8015500: 0x6008         STR
    SCB->CCR |=  (uint32_t)SCB_CCR_IC_Ms
    0x8015502: 0x48dd         LDR.N
    0x8015504: 0x6802         LDR
    0x8015506: 0xf442 0x3200  ORR.W
    0x801550a: 0x6002         STR
    __DSB();
    0x801550c: 0xf000 0xf89c  BL
}
    0x8015510: 0x2200         MOVS
    0x8015512: 0x6702         STR
    __DSB();
    0x8015514: 0xf3bf 0x8f4f  DSB
    ccsidr = SCB->CCSIDR;
    0x8015518: 0x6ec2         LDR
    sets = (uint32_t)(CCSIDR_SETS(ccsidr
```

```
Log
Mon Oct 03, 2016 11:17:43: Target reset
Mon Oct 03, 2016 11:17:45: Unloaded macro file: D:\IAR Systems\Embedded Workbench 7.5\arm\config\flashloader\ST\FlashSTM32F7xxx.mac
Mon Oct 03, 2016 11:17:45: Downloaded Y:\panache\STM32756G-EVAL2\STM32CUBEF7\STM32CUBE_FW_F7_V1.4.0\Projects\STM32756G_EVAL\APPLICATIONS\STemWin\PAI
Mon Oct 03, 2016 11:17:45: Software reset was performed
Mon Oct 03, 2016 11:17:49: 104848 bytes downloaded into FLASH and verified (18.15 Kbytes/sec)
Mon Oct 03, 2016 11:17:49: Loaded debugee: Y:\panache\STM32756G-EVAL2\STM32CUBEF7\STM32CUBE_FW_F7_V1.4.0\Projects\STM32756G_EVAL\APPLICATIONS\STemWi
Mon Oct 03, 2016 11:17:49: Software reset was performed
```

Figure 3: EWARM debugger.

12. Download and debug the application. Menu *Project→Download and Debug*. This action writes the program code to the eFLASH of the STM32F756NGH6 microcontroller on the STM32756G-EVAL2 board and starts the debugger.

13. Debug the application, examine menu *Debug*. Snapshot of running debugger can be seen in Figure 3.

    NOTE: The program remains in the eFLASH after the debug session finished. If the board is powered up, the latest application will start automatically.

The application can be tested without the Python 1300 camera module, commands can be observed via serial terminal (putty for instance). Settings of the terminal are presented in Table 3

Table 3: Serial terminal settings

Parameter	Value
Baud rate	115200
Data bits	8
Stop bits	1
Parity	None
Flow control	None

5 Package contents

```
.
|- doc/
|    `- cam_gui_appnote.pdf
`- sw/
    |- src
    |    |- BASIC_HelloWorld.c
    |    |- btn.c
    |    |- logo.c
    |    |- main.c
    |    |- sldr.c
    |    `- stm32f7xx_hal_msp.c
    `- inc
         |- btn.h
         |- colors.h
         `- sldr.h
```

6 References

[1] STMicroelectronics, „ STM32756G-EVAL2 - UM1903U User manual Evaluation board with STM32F756NG MCU," 03 2016. [Online]. Available: http://www.st.com/content/ccc/resource/technical/document/user_manual/3d/1c/e0/8e/2f/96/4e/59/DM00188617.pdf/files/DM00188617.pdf/jcr:content/translations/en.DM00188617.pdf

[2] STMicroelectronics, „STM32CubeF7 - Embedded software for STM32F7 series (HAL low level drivers, USB, TCP/IP, File system, RTOS, Graphic - coming with examples running on ST boards: STM32 Nucleo, Discovery kits and Evaluation boards)," [Online]. Available: http://www.st.com/content/st_com/en/products/embedded-software/mcus-embedded-software/stm32-embedded-software/stm32cube-embedded-software/stm32cubef7.html

[3] IAR Systems, „IAR Embedded Workbench," [Online]. Available: https://www.iar.com/iar-embedded-workbench/

7 Disclaimer

This disclaimer is not a license and does not grant any rights to the materials distributed herewith. Except as otherwise provided in a valid license issued to you by UTIA AV CR v.v.i., and to the maximum extent permitted by applicable law:

(1) THIS APPLICATION NOTE AND RELATED MATERIALS LISTED IN THIS PACKAGE CONTENT ARE MADE AVAILABLE "AS IS" AND WITH ALL FAULTS, AND UTIA AV CR V.V.I. HEREBY DISCLAIMS ALL WARRANTIES AND CONDITIONS, EXPRESS, IMPLIED, OR STATUTORY, INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, OR FITNESS FOR ANY PARTICULAR PURPOSE; and

(2) UTIA AV CR v.v.i. shall not be liable (whether in contract or tort, including negligence, or under any other theory of liability) for any loss or damage of any kind or nature related to, arising under or in connection with these materials, including for any direct, or any indirect, special, incidental, or consequential loss or damage (including loss of data, profits, goodwill, or any type of loss or damage suffered as a result of any action brought by a third party) even if such damage or loss was reasonably foreseeable or UTIA AV CR v.v.i. had been advised of the possibility of the same.

Critical Applications:
UTIA AV CR v.v.i. products are not designed or intended to be fail-safe, or for use in any application requiring fail-safe performance, such as life-support or safety devices or systems, Class III medical devices, nuclear facilities, applications related to the deployment of airbags, or any other applications that could lead to death, personal injury, or severe property or environmental damage (individually and collectively, "Critical Applications"). Customer assumes the sole risk and liability of any use of UTIA AV CR v.v.i. products in Critical Applications, subject only to applicable laws and regulations governing limitations on product liability.