**System Workbench. Connect the BSP library**

Today we will continue to work with the free programming environment **System Workbench for STM32**.

In the last lesson, we hooked up and were able to successfully use the FreeRTOS real-time operating system.

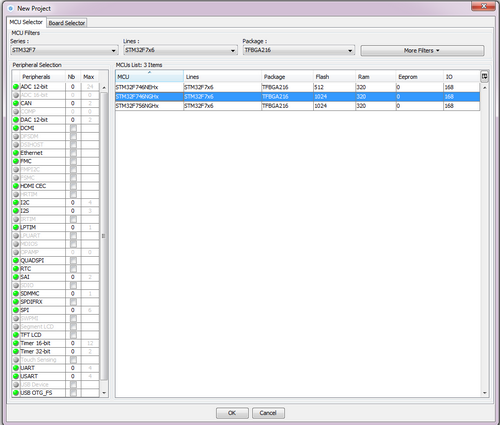
Today we will continue the business started and work with another payment.

For example, take the board **STM32F746G-DISCO** . With this board, we are also familiar from the review.

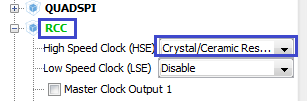
The purpose of this lesson is to study the possibility of connecting certain modules of the BSP library, and also use this library to work with the display installed on the board using FMC and DMA2D technologies.

Well, let's begin!

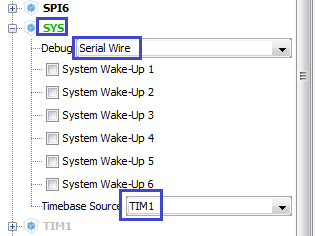
Run MS Cube, create a project by selecting our microcontroller (click on the image to increase the size)

[](http://narodstream.ru/wp-content/uploads/2016/11/image00_0968.png)

In RCC choose this option

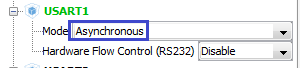


And in SYS there are such



TIM1 is selected due to the fact that the code generator will swear on Systick if we use FreeRTOS.

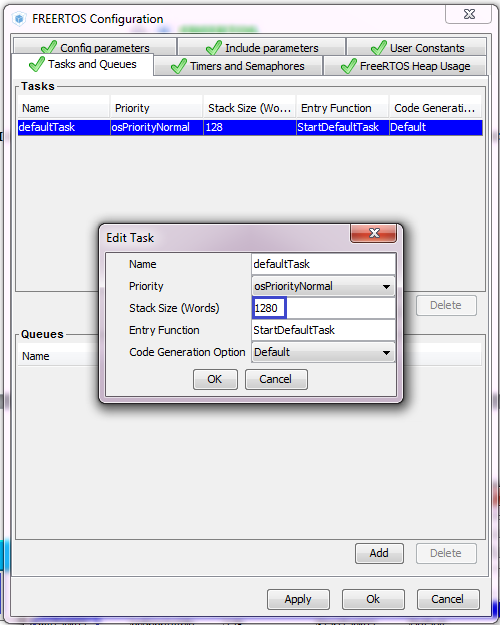
Include also USART, since the BSP library constantly refers to its functions

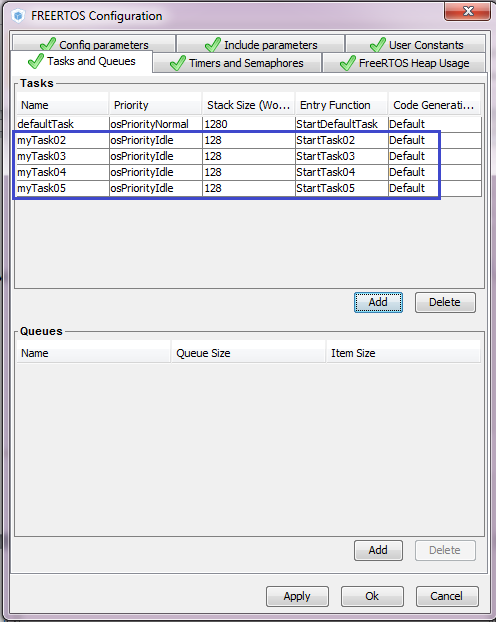


Turn on FreeRTOS

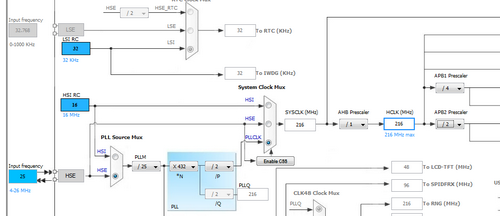
image03

Let's go into Configuration, go into the FreeRTOS properties, add a stack for the main task, and include 4 more tasks besides the main one with the standard stack

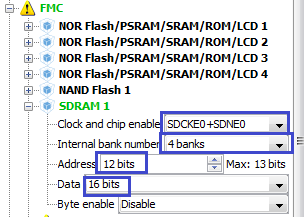




Let's go to Clock Configuration and adjust all dividers there, including the maximum frequency and adding a quartz resonator (click on the image to enlarge the size)

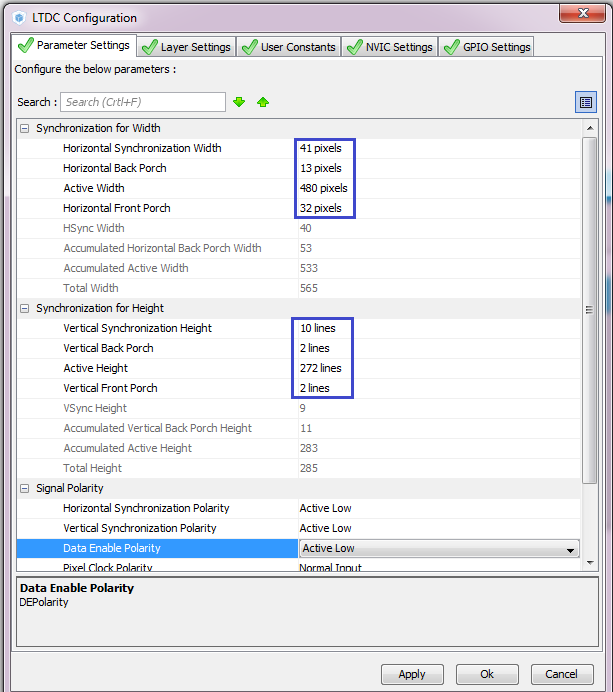
[](http://narodstream.ru/wp-content/uploads/2016/11/image09_0960.png)

We will also include LTDC, FMC and DMA2D. Since we will use BSP, where the initialization happens anew, the accuracy of all settings is not very important, but it is still desirable to make them as specified here

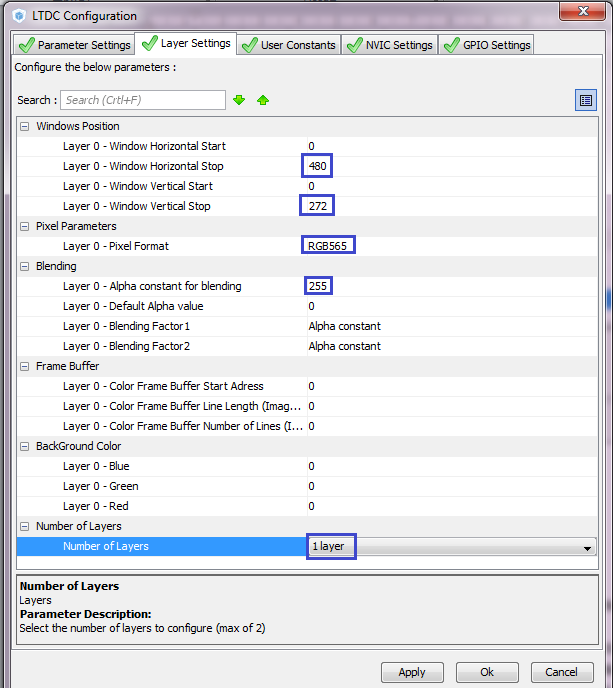
image07image12

Go to Configuration.

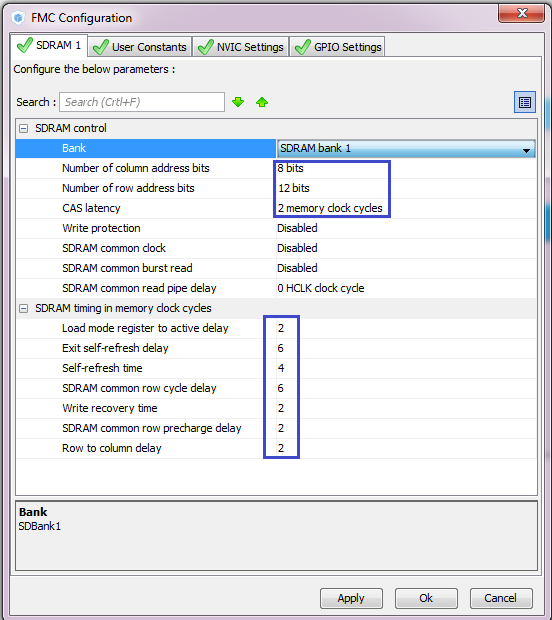
First, configure the LTDC



Next tab LTDC - Layer Settings



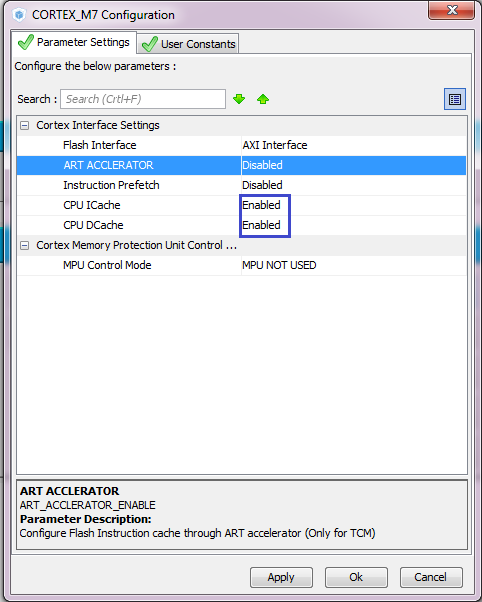
Apply settings, go to FMC and make the following settings



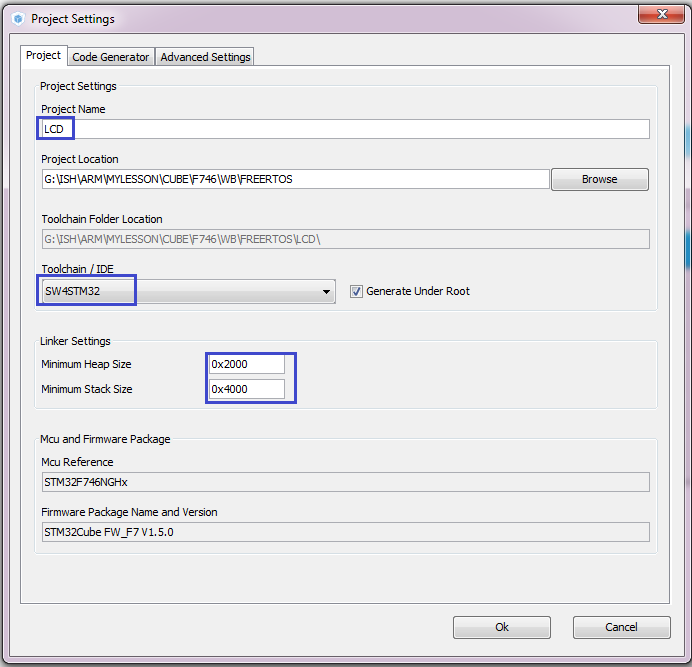
In other bookmarks do not touch anything and apply the settings.

In DMA2D do not touch anything.

Turn on the cache by going to CORTEX\_M7 ARM



Let's go to Project -> Settings and configure our future generated project, adding a little stack and heap value



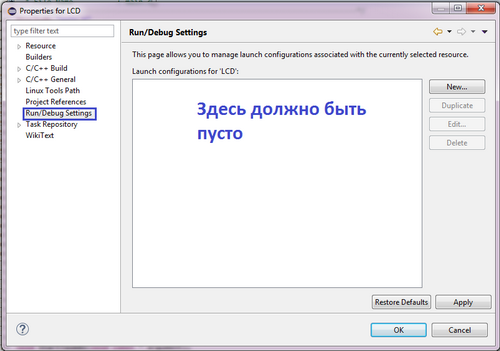
You will, accordingly, have a different path. Click "OK".

Generate the project and close the dialog.

Run the System Workbench and add our generated project in the same way as in previous sessions, removing all projects from the project tree before they get confused.

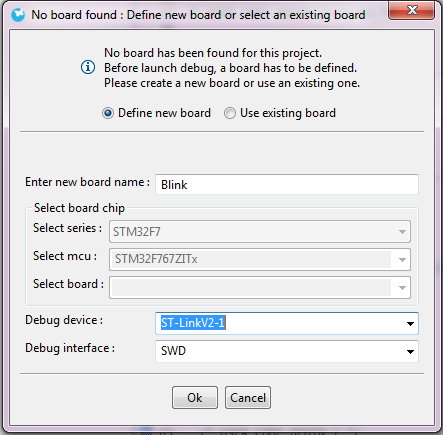
Open the project tree on the left, find the main.c file there and put the input focus there (click the mouse to set the cursor). This helps the compiler if there are projects in the project tree in the project tree, in a simple way, to understand that the given project is current for him and that he works with it.

Before you compile the project and start debugging, we will remove all debuggers in the configuration project properties to avoid unnecessary errors (click on the image to increase the size)

[](http://narodstream.ru/wp-content/uploads/2016/11/image17_0803.png)

Let's try to compile the project. For this, as usual, we select in the menu Project -> Build Project.

If everything is fine without errors, now we will try to debug. If a dialog appears with the choice of the programmer, you will need to select it



If you normally went into debugging, run it, then stop, we try to run the usual Run. If all is well, then we will now deal directly with the purpose of the lesson - actually connecting the BSP library files.

We go to the folder "Drivers" in the folder with the project and create the folder "BSP", go into this folder and create there 2 folders - "STM32746G-Discovery" and "Components". Let's go to the Components folder and copy it from the folder "rk043fn48h" and "ft5336" from the folder "Drive: \ Keil\_v5 \ ARM \ PACK \ Keil \ STM32F7xx\_DFP \ 2.7.0 \ Drivers \ BSP \ Components".

Also in this folder we will create the folder "Common", we will go into it and from the same folder of the above path copy the file ts.h.

We return to the 2 levels above and go to another folder - "STM32746G-Discovery". Copy the following files from the folder "Drive: \ Keil\_v5 \ ARM \ PACK \ Keil \ STM32F7xx\_DFP \ 2.7.0 \ Drivers \ BSP \ STM32746G-Discovery" to the folder:

stm32746g\_discovery.c

stm32746g\_discovery.h

stm32746g\_discovery\_lcd.c

stm32746g\_discovery\_lcd.h

stm32746g\_discovery\_sdram.c

stm32746g\_discovery\_sdram.h

stm32746g\_discovery\_ts.c

stm32746g\_discovery\_ts.h

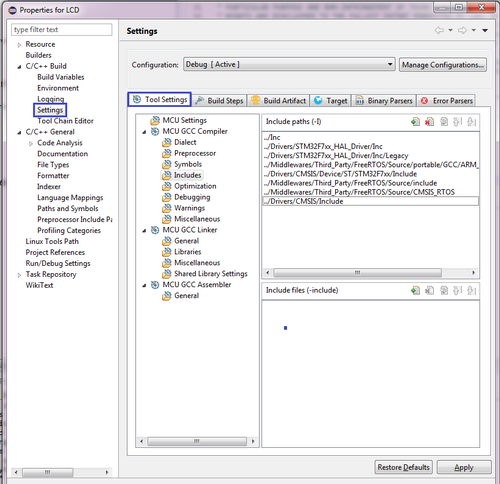
Exit this folder to the topmost level of the project folder and create the "Utilities" folder there. We'll go into this folder and copy the folders "CPU", "Fonts" and "Log" from the folder "Drive: \ Keil\_v5 \ ARM \ PACK \ Keil \ STM32F7xx\_DFP \ 2.7.0 \ Utilities".

Perhaps, at the time of reading the article, the folder name "2.7.0" will be different, as the versions are constantly updated.

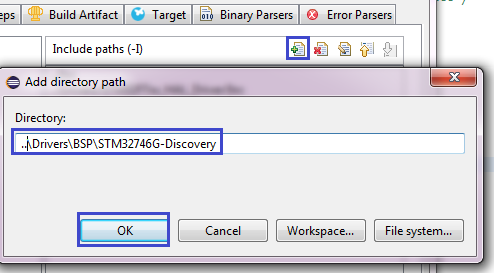
One more condition: the files in the folders added to the project should not contain the attribute "Read only".

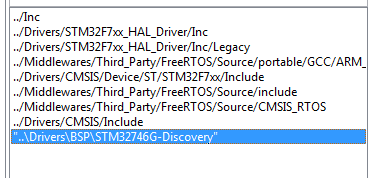
Let's return to WB, update the tree with the project (Refresh (F5)), and rebuild our project.

Now we need to connect the created folders to the project. To do this, go to the C / C ++ Build -> Settings -> Tool Settings item in the project properties, click the plus button and add the desired folder  (click on the image to enlarge the image)

[](http://narodstream.ru/wp-content/uploads/2016/11/image19_0821.png)

We add there the folder "STM32746G-Discovery" as follows, in the process it is desirable to correct the path to the relative one, in order to be similar to other files:





The path should preferably be relative, so that we can then rewrite our project.

The rest of the folders, in principle, do not need to be added, since relative paths are written to them in the library code. Let's try to assemble the project.

If everything is going well, then we'll try to initialize the display and paint it in some color.

In the [**next part of the**](http://narodstream.ru/stm-urok-56-system-workbench-podkljuchaem-biblioteku-bsp-chast-2/) lesson, we will continue to work with our project and test our code in practice by seeing a test on the display.

**Lesson 56**

**Part 2**

## System Workbench. Connect the BSP library

In the [**previous part of the**](http://narodstream.ru/stm-urok-56-system-workbench-podkljuchaem-biblioteku-bsp-chast-1/) lesson we briefly learned about the BSP library, created and configured a new project in Cube MX, connected the BSP library files to the project and tried to assemble the project with the connected BSP library.

In the main.c file, we connect the necessary files of the BSP library

/ \* USER CODE BEGIN Includes \* /

**#include "stm32746g\_discovery.h"**

**#include "stm32746g\_discovery\_lcd.h"**

/ \* USER CODE END Includes \* /

Declare the buffer in main.c

/ \* USER CODE BEGIN PV \* /

/ \* Private variables ------------------- \* /

**#define LCD\_FRAME\_BUFFER SDRAM\_DEVICE\_ADDR**

Call the display initialization

/ \* USER CODE BEGIN 2 \* /

**BSP\_LCD\_Init ();**

/ \* USER CODE END 2 \* /

We will collect the code and let's say the controller. The display should go out.

We now write down the code

        BSP\_LCD\_Init ();

**BSP\_LCD\_LayerDefaultInit (LTDC\_ACTIVE\_LAYER, LCD\_FRAME\_BUFFER);**

**BSP\_LCD\_SelectLayer (LTDC\_ACTIVE\_LAYER);**

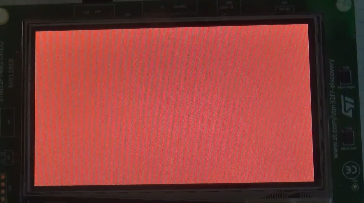
**BSP\_LCD\_DisplayOn ();**

**BSP\_LCD\_Clear (LCD\_COLOR\_RED);**

>

/ \* USER CODE END 2 \* /

Again, we'll collect the code and we'll sew the controller, the display should turn red



Now let's try to blink colored squares of random color with different frequency in different tasks. First, we'll fix the coloring to black

BSP\_LCD\_Clear ( **LCD\_COLOR\_BLACK**);

Add to main.c a call to the library

/ \* USER CODE BEGIN Includes \* /

**#include <stdlib.h>**

#include "stm32746g\_discovery.h"

In the first task, we add the code

void StartDefaultTask (void const \* argument)

{

  / \* USER CODE BEGIN 5 \* /

  / \* Infinite loop \* /

  for (;;)

  {

**BSP\_LCD\_SetTextColor (LCD\_COLOR\_BLACK);**

**BSP\_LCD\_FillRect (0,136,96,136);**

**BSP\_LCD\_SetTextColor ((uint32\_t) (LCD\_COLOR\_TRANSPARENT | ((rand ()% 256) << 16) | ((rand ()% 256) << 8) | (rand ()% 256)));**

**BSP\_LCD\_FillRect (0,0,96,136);**

**osDelay (200);**

**BSP\_LCD\_SetTextColor (LCD\_COLOR\_BLACK);**

**BSP\_LCD\_FillRect (0,0,96,136);**

**BSP\_LCD\_SetTextColor ((uint32\_t) (LCD\_COLOR\_TRANSPARENT | ((rand ()% 256) << 16) | ((rand ()% 256) << 8) | (rand ()% 256)));**

**BSP\_LCD\_FillRect (0,136,96,136);**

**osDelay (200);**  }

  / \* USER CODE END 5 \* /

}

In other tasks, we copy the same, only a few with other coordinates and with other delays

/ \* StartTask02 function \* /

void StartTask02 (void const \* argument)

{

  / \* USER CODE BEGIN StartTask02 \* /

  / \* Infinite loop \* /

  for (;;)

  {

        BSP\_LCD\_SetTextColor (LCD\_COLOR\_BLACK);

        BSP\_LCD\_FillRect ( **96**, 136,96,136);

        BSP\_LCD\_SetTextColor ((uint32\_t) (LCD\_COLOR\_TRANSPARENT | ((rand ()% 256) << 16) | ((rand ()% 256) << 8) | (rand ()% 256)));

        BSP\_LCD\_FillRect ( **96**, 0.96.136);

            osDelay ( **190**);

        BSP\_LCD\_SetTextColor (LCD\_COLOR\_BLACK);

        BSP\_LCD\_FillRect ( **96**, 0.96.136);

        BSP\_LCD\_SetTextColor ((uint32\_t) (LCD\_COLOR\_TRANSPARENT | ((rand ()% 256) << 16) | ((rand ()% 256) << 8) | (rand ()% 256)));

        BSP\_LCD\_FillRect ( **96**, 136,96,136);

        osDelay ( **190**);

  }

  / \* USER CODE END StartTask02 \* /

}

/ \* StartTask03 function \* /

void StartTask03 (void const \* argument)

{

  / \* USER CODE BEGIN StartTask03 \* /

  / \* Infinite loop \* /

  for (;;)

  {

        BSP\_LCD\_SetTextColor (LCD\_COLOR\_BLACK);

        BSP\_LCD\_FillRect ( **192**, 136.96, 136);

        BSP\_LCD\_SetTextColor ((uint32\_t) (LCD\_COLOR\_TRANSPARENT | ((rand ()% 256) << 16) | ((rand ()% 256) << 8) | (rand ()% 256)));

        BSP\_LCD\_FillRect ( **192**, 0.96.136);

        osDelay ( **180**);

        BSP\_LCD\_SetTextColor (LCD\_COLOR\_BLACK);

        BSP\_LCD\_FillRect ( **192**, 0.96.136);

        BSP\_LCD\_SetTextColor ((uint32\_t) (LCD\_COLOR\_TRANSPARENT | ((rand ()% 256) << 16) | ((rand ()% 256) << 8) | (rand ()% 256)));

        BSP\_LCD\_FillRect ( **192**, 136.96, 136);

        osDelay ( **180**);

  }

  / \* USER CODE END StartTask03 \* /

}

/ \* StartTask04 function \* /

void StartTask04 (void const \* argument)

{

  / \* USER CODE BEGIN StartTask04 \* /

  / \* Infinite loop \* /

  for (;;)

  {

        BSP\_LCD\_SetTextColor (LCD\_COLOR\_BLACK);

        BSP\_LCD\_FillRect ( **288**, 136.96, 136);

        BSP\_LCD\_SetTextColor ((uint32\_t) (LCD\_COLOR\_TRANSPARENT | ((rand ()% 256) << 16) | ((rand ()% 256) << 8) | (rand ()% 256)));

        BSP\_LCD\_FillRect ( **288**, 0.96, 136);

            osDelay ( **170**);

        BSP\_LCD\_SetTextColor (LCD\_COLOR\_BLACK);

        BSP\_LCD\_FillRect ( **288**, 0.96, 136);

        BSP\_LCD\_SetTextColor ((uint32\_t) (LCD\_COLOR\_TRANSPARENT | ((rand ()% 256) << 16) | ((rand ()% 256) << 8) | (rand ()% 256)));

        BSP\_LCD\_FillRect ( **288**, 136.96, 136);

        osDelay ( **170**);

  }

  / \* USER CODE END StartTask04 \* /

}

/ \* StartTask05 function \* /

void StartTask05 (void const \* argument)

{

  / \* USER CODE BEGIN StartTask05 \* /

  / \* Infinite loop \* /

  for (;;)

  {

        BSP\_LCD\_SetTextColor (LCD\_COLOR\_BLACK);

        BSP\_LCD\_FillRect ( **384**, 136.96, 136);

        BSP\_LCD\_SetTextColor ((uint32\_t) (LCD\_COLOR\_TRANSPARENT | ((rand ()% 256) << 16) | ((rand ()% 256) << 8) | (rand ()% 256)));

        BSP\_LCD\_FillRect ( **384**, 0.96, 136);

        osDelay ( **160**);

        BSP\_LCD\_SetTextColor (LCD\_COLOR\_BLACK);

        BSP\_LCD\_FillRect ( **384**, 0.95.135);

        BSP\_LCD\_SetTextColor ((uint32\_t) (LCD\_COLOR\_TRANSPARENT | ((rand ()% 256) << 16) | ((rand ()% 256) << 8) | (rand ()% 256)));

        BSP\_LCD\_FillRect ( **384**, 136.95.271);

        osDelay ( **160**);

  }

  / \* USER CODE END StartTask05 \* /

}

After assembly and firmware, you should get an interesting wink. If you watch the video tutorial, I think you will appreciate it.