RAISONANCE
a KEOLABS brand

EvoPrimer for STM32F MCU

Fun, easy introduction kit for STM32F429ZI microcontrollers

User Manual

Document version
01 October 2013

Contents

1. Introduction

The EvoPrimer for STM32F429 microcontrollers is a fun, cost-effective evaluation and development package based on Raisonance's versatile, innovative Open4 platform and an STM32F429 target board.

> **Note**: The STMicroelectronics product EvoPrimer for STM32F429 microcontrollers (ST order code: STM3242IPRIMER) is a derivative product of the Raisonance Open4. The same EvoPrimer target boards, extension boards, software tools and sample applications are used with both EvoPrimer and Open4.

1.1 Purpose of this manual

This manual provides an overview and installation procedure for your EvoPrimer. Armed with the knowledge in this manual users can quickly understand the target microcontroller's features and create their own applications.

1.2 Scope of this manual

This manual is applicable to all versions of EvoPrimer for STM32F429 microcontrollers. It describes the EvoPrimer's basic use and its hardware and firmware features.

Further information can be found in:

- LIS3LV02DL MEMS Inertial Sensor Datasheet, available from http://www.st.com/.
- STM32F429ZIH6 High-performance AC Line, http://www.st.com/.
- STM32F4x Flash programming manual, http://www.st.com/.
- STM32F4x advanced ARM-based 32-bit MCU reference manual, http://www.st.com/.
- STM32F429xx Datasheet, http://www.st.com/.
- STM32F3xx/F4xxx Cortex™-M4 programming manual, http://www.st.com/.
- Cortex-M4 Technical Reference Manual, see http://www.arm.com/.
- ILI9325 Datasheet” describes the 262K color single-chip TFT controller/driver, see http://www.ilitek.com/.
- LCM-TGG000240YP04-24 Model.pdf” describes the 320x240 display of the EvoPrimer base.
- The GNU Compiler Collection (GCC) manuals are available from the Ride7 environment.
- The CircleOS conception document.

Visit the Circle community web site at www.stm32circle.com for more resources and software tool downloads.

1.3 Additional help or information

If you want additional help or information, if you find any errors or omissions, or if you have suggestions for improving this manual, go to the KEOLABS' site for Raisonance microcontroller development tools www.raisonance.com, or contact the microcontroller support team.

Microcontroller website:	www.raisonance.com
Support extranet site:	support-raisonance.com (software updates, registration, bugs database, etc.)
Support Forum:	forum.raisonance.com/index.php
Support Email:	support@raisonance.com

1.4 Raisonance brand microcontroller application development tools

January 1, 2012, Raisonance became the brand under which the company KEOLABS sells its microcontroller hardware and software application development tools.

All Raisonance branded products regardless of their date of purchase or distribution are licensed to users, supported and maintained by KEOLABS in accordance with the companies' standard licensing maintenance and support agreements for its microcontroller application development tools. For information about these standard agreements, go to:

Support and Maintenance Agreement:	http://www.raisonance.com/warranty.html
End User License Agreement:	http://www.raisonance.com/software-license.html

2. Presentation

Your EvoPrimer is composed of two major hardware elements:

- a base platform,
- a target board featuring an STM32F429ZIH6.

When the target board is connected to the base, the STM32F429 drives the base's hardware features for evaluation purposes. When the base is connected to a PC the target microcontroller can be reprogrammed and application software can be created and debugged using the Ride7 Raisonance software tools. The target board includes additional features that can be implemented in user applications, in addition to the features provided on the base.

EvoPrimer

2.1 Base main features

The base provides the following capabilities:

- USB debug/programming connection (SWD for STM32)
- QVGA (320x240) back-lit color LCD, with touchscreen capability
- Joystick/push button (mechanical)
- 4 push buttons (based on touchscreen capability)
- MicroSD card connector
- Audio circuit with microphone/loudspeaker/jack
- 3D MEMS accelerometer
- Li-Ion battery with charge management circuitry
- Add-on connector (USART, SPI, I2C, ADC,...)

2.2 Target board main features

The STM32F429 target board integrates the following hardware features:

- STM32F429ZIH6 microcontroller (2MB Flash, 192KB SRAM)
- USB OTG device / host, Full Speed (12 MB/s)
- DMA2D (Chrom-ART accelerator)
- SDRAM controller
- 2 MB SDRAM
- STA529A audio codec

2.3 Development software

The development software offered with the EvoPrimer provides everything you need to program an STM32F429 and debug applications, including:

- USB host connection for in-circuit programming and debugging.
- Ride7 integrated development environment for code editing, device programming and application debugging (debug up to 64K of code, with included version. For information about upgrade to an unlimited version of Ride, visit http://www.stm32circle.com/resources).
- GNU C /C++ compiler (unlimited compiling)
- CircleOS operating system and base services. An online community available at http://www.stm32circle.com makes it possible to share your EvoPrimer experience with others on the forums, retrieve useful resources, application notes, firmware, demos (C sources and projects) and applications, all available for free download.

3. EvoPrimer hardware

3.1 Package contents

Your EvoPrimer unit has been carefully packed. Examine the equipment for damage that may have occurred during shipment. If you find any damage, or if any of the items are not included, please contact Raisonance.

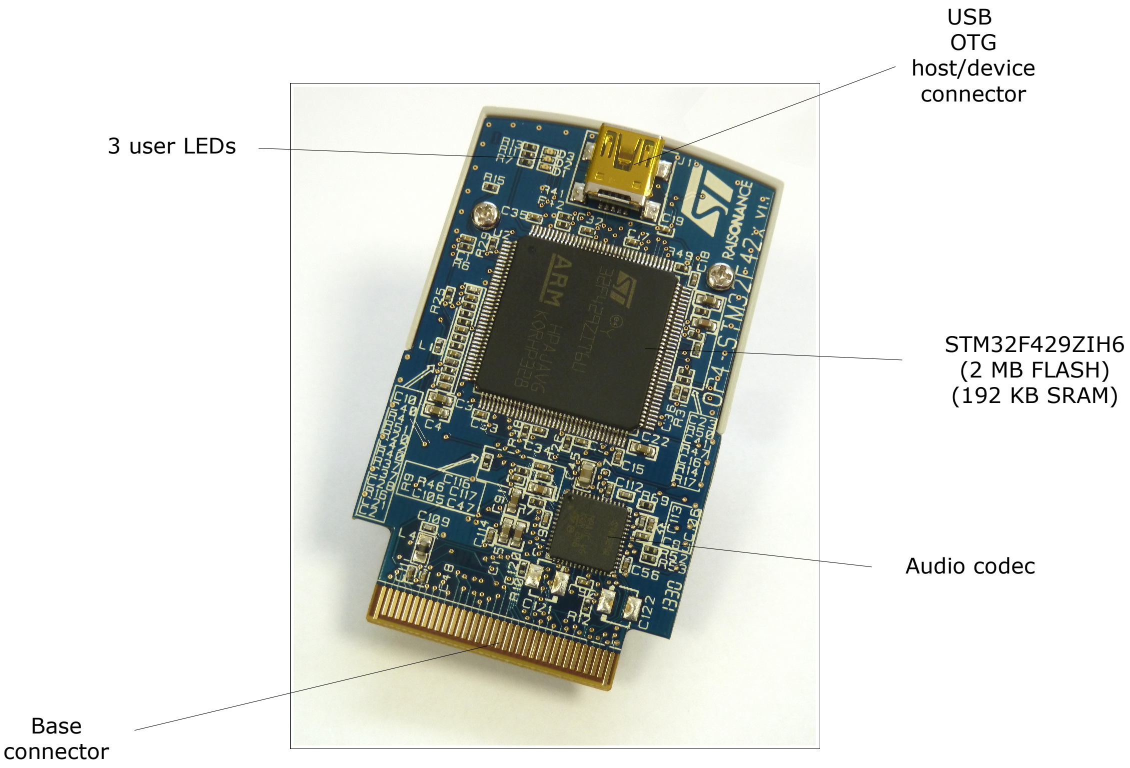
Your EvoPrimer is delivered in two packages:

1. EvoPrimer base with USB cable.
2. STM32F429 target board.

Note: Download software tools after free registration at www.stm32circle.com

3.2 Components overview

Here is a brief overview of the main components of the STM32F429 target board:



3.3 Target board features

3.3.1 STM32F429 microcontroller features

The target board is equipped with an STM32F429ZIH6 from STMicroelectronics' STM32F429 32-bit ARM Cortex™-M4 core-based microcontrollers.

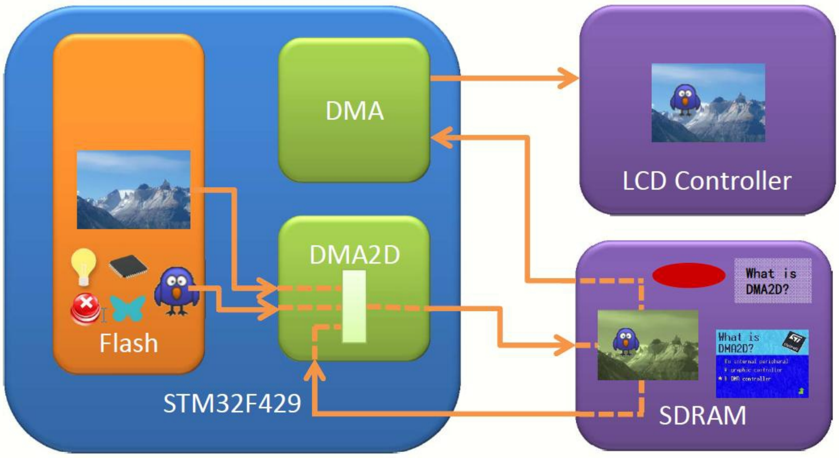
Its main characteristics are:

- ARM 32-bit Cortex™-M4F CPU, 180MHz, 225DMips with 1.25DMips/MHz, Adaptive real-time accelerator (ART Accelerator™) with 0-wait state execution from Flash, DSP instructions
- LCD-TFT controller up to VGA resolution with dedicated Chrom-ART Accelerator™ (DMA2D) for enhanced graphic content creation
- Integrated SDRAM controller
- 2 MB Flash program memory, 192KB SRAM
- Embedded oscillators (for high-speed crystal + RTC)
- SWD debug interface
- Fast input/output: up to 164 I/Os, ADC, DAC
- Embedded communication peripherals: HS, USB 2.0, OTG, dual CAN, UART, USART, SPI, I2C, LIN, IrDA, SDIO, ISO7816 interface
- Up to 17 timers: watchdog, PWM, Systick timer, and more

  The STM32F429ZIH6 has other features (such as a camera interface) that could be used if your application provided the relevant peripheral. See the STM32F429ZIH6 datasheet for full details.

3.3.2 Chrom-ART Accelerator™ (DMA2D)

The TFT LCD controller interface with dual-layer support uses ST's Chrom-ART graphics accelerator to create content twice as fast as the core alone. In addition to raw data copy, the Chrom-ART Accelerator can also perform functions such as image format conversion or image blending (image mixing with some transparency). As a result, the Chrom-ART Accelerator boosts graphics content creation and saves processing bandwidth of the Cortex-M4 core for the rest of the application.



3.3.3 DMA2D interface with CircleOS

The DMA2D feature abstraction API can communicate with the CircleOS API to provide dynamic objects, layering /ordering, transparency, bitmap translation , movement , animation & more…

For clarity and ease of use, the files containing the features called by our example applications from the DMA2D Abstraction API (dma2d_graphic.c / .h) are grouped separately from the CircleOS API files.

Our solution takes data from Flash, processes it with DMA2D using SDRAM then sends the resulting data to DMA which puts it on the LCD.

3.3.4 Audio codec

The STM32F429 target board includes an audio codec component (Ref. STA529A) which supports voice quality audio features of the base (recording and playback).

3.3.5 Mini-USB connector

The STM32F429 target board includes a Mini-USB connector which can be used for USB 2.0 implementation in demonstration and user applications.

3.4 Base features

The base provides the following capabilities:

- USB debug/programming connection (SWD for STM32)
- QVGA (320x240) back-lit color LCD, with touchscreen capability
- Joystick/push button (mechanical)
- 4 push buttons (based on touchscreen capability)
- MicroSD card connector
- Audio circuit with microphone/loudspeaker/jack
- 3D MEMS accelerometer
- Li-Ion battery with charge management circuitry
- Add-on connectors (USART, SPI, I2C, ADC,...)

3.4.1 3D MEMS accelerometer

The base is equipped with a MEMS inertial sensor (LIS3LV02DL from STMicroelectronics) which can be used to select commands in coordination with a graphic pointer. When you start the EvoPrimer for the first time, you will see a small ball moving according to the orientation of the EvoPrimer. The information about the 3D position is provided by the MEMS.

3.4.2 Power supply

The EvoPrimer base features a 400mAh Li-Ion rechargeable battery, equipped with a voltage regulator and a battery charger.

When the USB connector is linked to a PC, the host voltage (supplied by the PC) is used to recharge the battery. When no USB host is connected, the battery supplies the power for the EvoPrimer. When the battery is fully charged, the EvoPrimer can be used for about 6 hours. The duration of the batteries depends on the activity.

3.4.3 Extension connector

The base has a 20-pin HE14 female right-angle extension connector that is accessible when the orange part of the case is removed. The extension board mechanical description is available in the “resources” section of the www.stm32circle.com web site.

Extension connector pins				
Pin	**Printed name**	**STM32 pin**	**STM32 name**	**Description**
1	V2V8	Vcc	VCC	Connected to the output of the U6 regulator (3.1V). This source could provide approximately 100mA to the extension board.
2	GND	GND	GND	Ground
3	SCL	PB.8	CX_I2CSCL	Can be used either as a standard GPIO or considered as an alternate function: - I2C1_SCL (clock)
4	SDA	PB.9	CX_I2CSDA	Can be used either as a standard GPIO or considered as an alternate function: - I2C1_SDA (data)
5	MISO	PA.6	CX_SPI1_MISO	Can be used either as a standard GPIO or considered as an alternate function: - SPI1_MISO - TIM3_CH1 - TIM13_CH1 - ADC12_IN6
6	SD	PA.7	CX_SPI1_MOSI	Can be used either as a standard GPIO or considered as an alternate function: - SPI1_MOSI - TIM1_CH1N - TIM14_CH1 - TIM3_CH2 - TIM1_CH1N - ADC12_IN7
7	SCK	PA.5	CX_SPI1CK	Can be used either as a standard GPIO or considered as an alternate function: - SPI1_SCK - TIM2_CH1 - TIM8_CH1N - ADC12_IN5 - DAC_OUT2
8	WS	PA.4	CX_SPI1_NSS	Can be used either as a standard GPIO or considered as an alternate function: - SPI1_NSS - TIM2_CH1_ETR

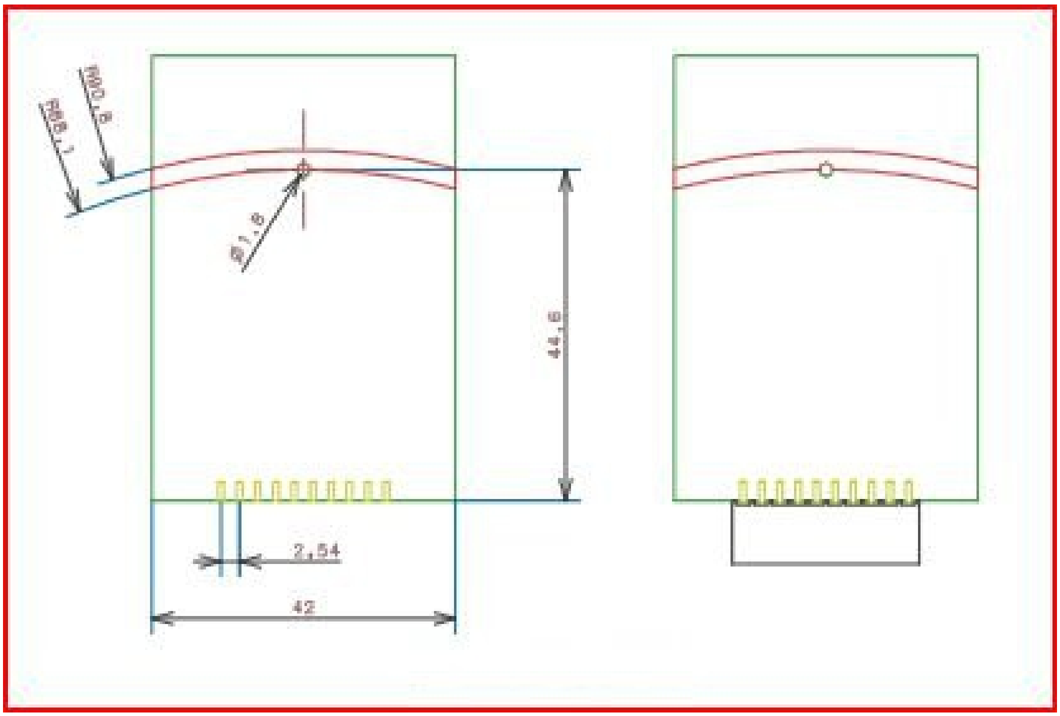
Extension connector pins				
Pin	**Printed name**	**STM32 pin**	**STM32 name**	**Description**
9	CANH	PA.11	CAN_RX	Can be used either as a standard GPIO or considered as an alternate function: - CAN1_RX - OTG_FS_DM Note: there is no CAN transceiver on the board.
10	CANL	PA.12	CAN_TX	Can be used either as a standard GPIO or considered as an alternate function: - CAN1_TX - OTG_FS_DP Note: there is no CAN transceiver on the board.
11	ADC1	PA.3	CX_ADC1	Can be used either as a standard GPIO or considered as an alternate function: - ADC123_IN3 - TIM2_CH4 - TIM5_CH4 - TIM9_CH2 - USART2_RX
12	ADC2	PA.2	CX_ADC2	Can be used either as a standard GPIO or considered as an alternate function: - ADC123_IN2 - TIM5_CH3 - TIM9_CH1 - TIM2_CH3 - USART2_TX
13	A_TIM	PB.7	CX_TIM	Can be used either as a standard GPIO or considered as an alternate function: - TIM4_CH3 - TIM10_CH1
14	CTS	PG.13	CX_USART_CTS	Can be used either as a standard GPIO or considered as an alternate function: - USART6_CTS
15	RTS	PG.12	CX_USART_RTS	Can be used either as a standard GPIO or considered as an alternate function: - USART6_RTS
16	TX	PG.14	CX_USART_TX	Can be used either as a standard GPIO or considered as an alternate function: - USART6_TX
17	CK	N/A	P_BUTTON	Push button input. Can wakeup the extension board.
18	RX	PC.7	CX_USART_RX	Can be used either as a standard GPIO or considered as an alternate function: - USART6_RX - TIM3_CH2 - TIM8_CH2

Extension connector pins				
Pin	**Printed name**	**STM32 pin**	**STM32 name**	**Description**
19	VEXT	N/A	VBAT	Base battery voltage provided
20	GND	VSS	GND	Ground

Notes: Printed names on the extension board may not accurately match the pin names that exist for the STM32F429 target board because the base is a generic design that supports other processor families (namely STM8) which do not share the same feature set as STM32.

3.4.4 Application-specific extension boards

The base has been designed to be extended through application-specific extension boards. Such extension boards connect to the extension connector and are compatible (mechanically and electronically) with the previous STM32-Primer2, so you should be able to reuse your previous designs without modification to the extension board. However, some limitations may apply, for instance if you use specific processor/peripheral capabilities that are not portable between designs. The following picture shows the mechanical constraints that you should respect to ensure that your extension board fits inside the EvoPrimer case.



Extension board mechanical specifications

4. Getting started

This chapter explains how to switch on your EvoPrimer and describes the provided applications.

4.1 Switch on

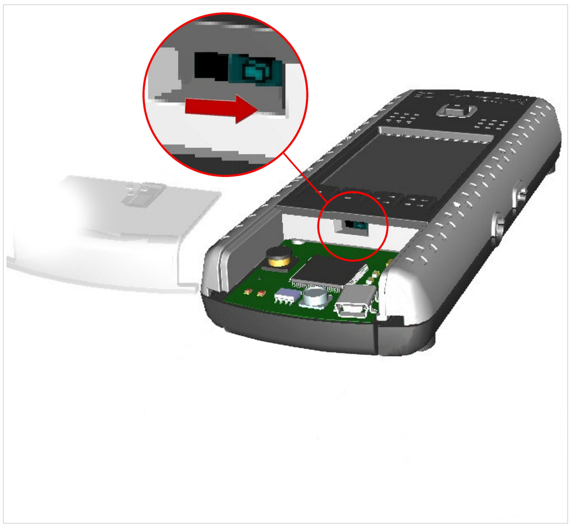
The procedure for getting started is reduced to four simple operations:

1. Insert a target board into the EvoPrimer base (see below).
2. Switch the power switch to provide current to the target board (see below).
3. Slide the transparent cover in place.
4. Press the joystick to power up the EvoPrimer.

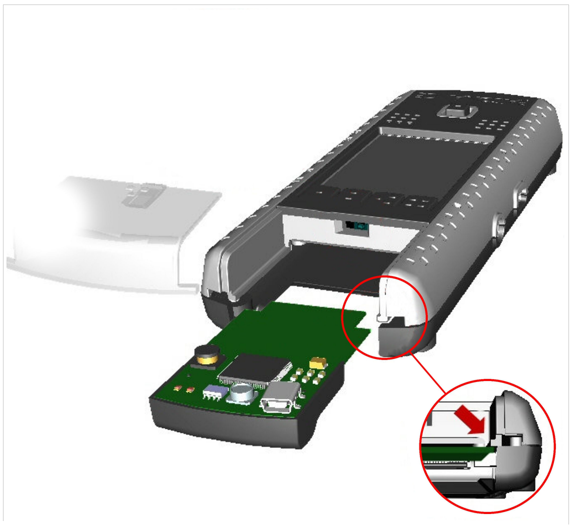
4.1.1 Insert the target board

The first thing to do is to insert the target board:

1. Remove the transparent cover.
2. Switch off the battery (see image below left).
3. Insert the target board into the base, as shown in the image, ensuring that the PCB is against the guides.
4. Push until the top of the target board fit with the side of the base.



Switching OFF the EvoPrimer

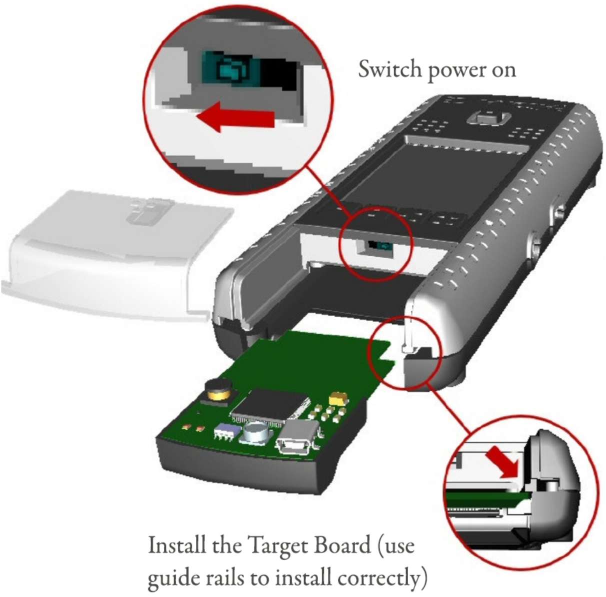


Inserting the target board

4.1.2 Connect battery, charge and power up

Your EvoPrimer is initially shipped with the battery disconnected in order to prevent discharging or harm occurring to the battery. You need to connect the battery before operating your EvoPrimer:

1. Open the EvoPrimer’s case by removing the cover which is on the target board.
2. Switch the button as shown on the image to power on your device.
3. Close the case.
4. Connect the EvoPrimer to the USB port on a PC to charge its battery. Only the red LED (L0) will be lit if the battery is charging. The green LED (L1) is lit when fully charged.
5. After charging the battery, press the push button to activate the EvoPrimer.



Switch power on

Install the Target Board (use guide rails to install correctly)

4.2 Using your EvoPrimer applications

The base provides user interface features such as a color touchscreen, audio circuit, MEMS accelerometer-based navigation, joystick, push buttons, GUI, SD card connector, IrDA and much more.

The target board operates evaluation features when connected to the base, and includes additional features such as audio Codec and a USB connector.

4.2.1 Play

After the opening screen is displayed, press the push button to call the main menu.

Tilt the EvoPrimer backward/forward or use the joystick to navigate in the menu, use the push button to select a menu command. Initial menu items include:

Config Configure parameters for the EvoPrimer including the pointer, backlight for the LCD display and time.

Maze This is an addictive game provided as an example. Full source is available from www.stm32circle.com.

Applic Select an application to run. The application that you choose will then appear in the main menu.

SD card Display information saved on the SD card, or allows to configure your EvoPrimer as a mass storage device, and transfer files from/to your PC from/to the SD card.

About Display information about the CircleOS version and available memory.

Shutdown Turn off the EvoPrimer. To restart the EvoPrimer, simply press the push button.

Quit Quit the menu.

Note: When you receive the EvoPrimer, the MEMS based controls are calibrated to a “zero” position that matches a 30° angle from the horizontal (corresponds to the position of a book when reading). To practice controlling the EvoPrimer, move the blue dot around the main screen. It takes a little practice.

4.2.2 Pre-installed applications for DMA2D (Chrom-ART Accelerator)

There are 2 pre-installed examples that illustrate the use of DMA2D.

- **DMA2D-demo** – A little demo and presentation that shows the graphics capabilities of the DMA2D.
- **SPACE 2D** – A space invaders style game.

4.2.3 Pre-installed applications

The EvoPrimer includes some pre-installed applications:

- **Maze** – A game where the player navigates inside a labyrinth, eating dots to win points while avoiding ghosts.
- **Breakout** – A game where the player uses a paddle to bounce a ball against a wall of blocks, destroying the block to win points. Win the game by destroying all the blocks.
- **CxDemo** – A game that demonstrates the extension ports.
- **FileSystemTest** – A game that demonstrates the file system tests.
- **Gobang** – An Othello like game.
- **Square** – A game where the player must avoid meeting moving squares.
- **Graphics** – Application that shows the drawing capabilities of the EvoPrimer.
- **Kaleidos** – Transforms your EvoPrimer to a kaleidoscope.
- **Cubic-4 –** Last step of the tutorial, that transforms your EvoPrimer to a bubble level system.
- **Fonts –** Application that shows the fonts management by the EvoPrimer.
- **Full screen** – Application that shows how writing on all the screen.
- **COMMTEST** – Tests the USB com port, the joystick, and the MicroSD card connector.
- **EchoIrDA** – IrDA echo application, which makes it easy to connect your EvoPrimer to another IrDA appliance such as a cell phone or another EvoPrimer.

4.3 Configuration menu

Configuration and test applications have been pre-installed to check your EvoPrimer's capabilities. From the main menu of the EvoPrimer, launch the `Config` command. These parameters can be set:

1. **Power**:
   CPU Freq: Chooses the speed of your EvoPrimer.
   Backlight: Tunes the backlight intensity. Note that the backlight is the main source of power consumption (with the LCD monitor itself). Reducing the backlight intensity allow to extend the duration when the EvoPrimer is powered by the battery.
2. **Interface**:
   User input: Selects the user input for commands menu (accelerometer or joystick or accelerometer + joystick or touchscreen)
   Loudspeaker: enables or not the loudspeaker. If not, the sounds are only sent to the jack headphones connector.
   Menu: Selects whether or not to display the menu with a large font.
   Beep: Selects whether or not to send a "beep" when the EvoPrimer is tapped twice.
   TS Calibrate: Launches a calibration procedure of the touchscreen.
3. **Autorun**: The current application runs whenever your EvoPrimer is started. In this mode, there is no need to select the application in order to start it.
4. **Time**: Sets your EvoPrimer time. Note: the RTC clock remains valid even if the EvoPrimer is off.
5. **Test**: Performs a quick factory test of the EvoPrimer.

You can add more applications to your EvoPrimer by downloading them on the web site www.stm32circle.com/projects. Refer to the chapter 5 Managing your CircleOS applications

4.4 Compiling, programming and debugging

4.4.1 Install software

To explore, modify and create new applications on your EvoPrimer you need to install some software:

1. Register and download the CD-ROM-Image file on the Resource page of www.stm32circle.com.
2. Extract the files and install:

- **Ride7**, the Raisonance IDE to write and debug new applications. This includes the RLink driver which is necessary to pilot the programming and debugging of your EvoPrimer from your PC.
- **RKit-ARM**, the Ride7 add-on that contains the complete GNU software toolchain based on the GCC compiler.
- Some utilities to manage your Circle applications (Circle is the OS embedded on your EvoPrimer; refer to Chapter5.1 "CircleOS architecture" for details).

3. Launch Ride7.
4. Connect your PC to the Debug USB port on the base.

Complete tool documentation is provided with the software installation.

Note: Ride7 should be installed *before* connecting to this USB port.

4.4.2 Explore the STM32 Toggle application

Some examples are installed with the RKit-ARM for Ride7. For instance, this Toggle example will help you build and program a very simple EvoPrimer application:

“[RIDE7_INSTALL_DIR]\Examples\ARM\Primer\STM32EvoPrimer\toggle_STM3242x\toggle.rprj”

Follow these steps in order to execute the example on your EvoPrimer :

1. Open Ride7.
2. Select **Open | Project** and navigate to the example.
3. Click on **Project | Make Project**.
4. Once the project is built, connect your EvoPrimer to your PC through USB.
5. Click **Debug | Start**.
6. Once the debugger is ready, you can run the example through **Debug | Run**.

You can then play with the sample application on your EvoPrimer.

5. Managing your CircleOS applications

5.1 CircleOS architecture

The STM32F429 target board is equipped with an STM32F429ZIH6 that contains 2MB of Flash ROM and 256+4KB of SRAM. The EvoPrimer embeds the CircleOS operating system. It provides services that will help you develop your STM32F429 EvoPrimer applications, including:

1. Application management,
2. LCD graphic functions,
3. MEMS functions,
4. LED, Buzzer and Push Button functions,
5. Menu functions,
6. Scheduler task,
7. Audio functions,
8. File system functions,
9. ...

CircleOS can load several independent applications.

Each application is run by CircleOS when selected, has the full availability of the CPU and can use all the RAM that is not being used by CircleOS (i.e. 16KB in the memory address range from 2000000h to 20003FFFh). It is scheduled by the CircleOS with full privileges on the device, until it explicitly quits.

5.2 The CircleOS Scheduler

CircleOS acts in several stages: An initialization stage which occurs upon device reset, a periodic Systick interrupt, and the scheduling of applications.

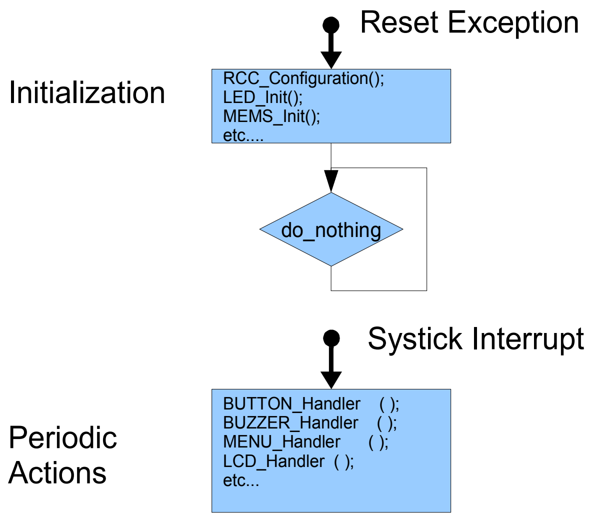
5.2.1 Initialization stage

During the initialization stage, the hardware configuration is performed, and the periodic system timer (Systick) is installed.

The Systick period depends on the RCC settings: it can be modified through the menu **Config | CPU Speed** to the following predefined values:

Level	CPU freq (MHz)	Systick freq (kHz)
1	30	0.75
2	48	1
3	60	1.5
4	72	2
5	120	3

The ratio CPU_freq / SysTick applies for all these values.



5.2.2 Periodic Systick interrupt

The periodic Systick makes a call to the CircleOS Systick interrupt handler which performs a short process on each of the STM32-EvoPrimer components: LEDs, button, buzzer, LCD and so on.

Note that the MEMS handler is also called from the Systick interrupt.

I2S transfers towards the audio Codec chip are handled through the DMA interrupt.

5.2.3 Application scheduler

CircleOS is the base application of the EvoPrimer. It handles the menu selections and reacts to user actions. When an application is run (usually through a menu selection), CircleOS calls an initialization routine for the application, CircleOS then repeatedly calls the application handler at the Systick frequency until it returns a `MENU_LEAVE` value.

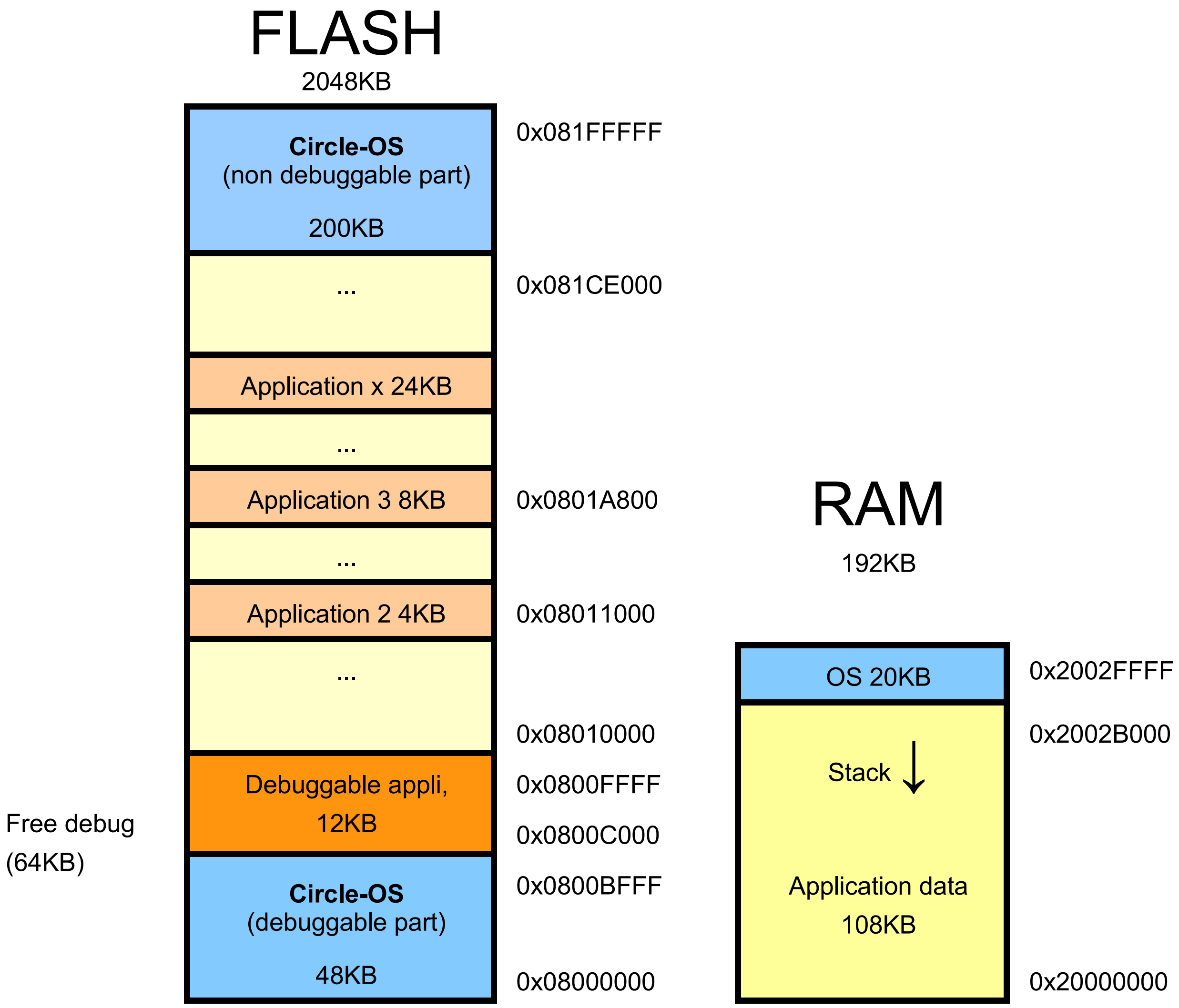
You will find more details about programming CircleOS applications in Chapter 6 “Developing CircleOS applications”.

5.3 CircleOS resource usage

5.3.1 Memory usage

The CircleOS firmware requires 48KB of debug code in Flash, 200KB of constants and non-debuggable code in Flash and 20KB of RAM (including the stack usage for the applications). The remaining 1776KB of Flash (2048 – 248) are available for applications, which can be added or removed at will using a programming tool (see below).

The following shows an example of memory mapping:



5.3.2 Resources used by CircleOS

The following figure (from the ST datasheet) shows the peripherals used by CircleOS:

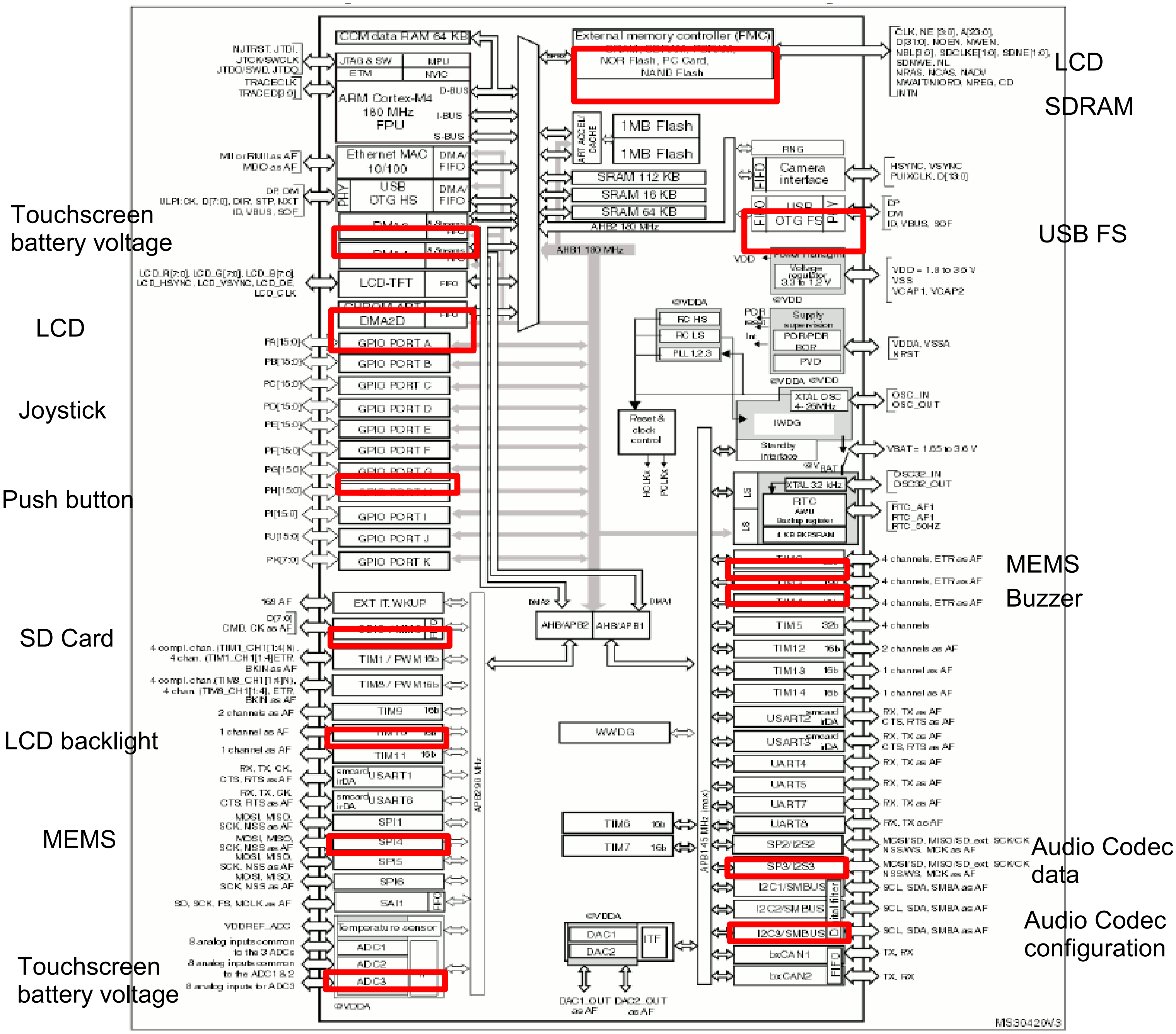


Illustration 5.1: STM32F429ZIH6 peripherals

5.4 DMA2D application programming interface

The DMA2D API provided by CircleOS covers:

- Handling of bit maps (display, overlaying and moving)
- Using transparency (pixel-level, global and blending)
- Dynamic versus Read-Only objects
- Management of objects and touch sensing
- Use of layering (Z-order)
- Use of memory features

Using the API is a simple way to manage the graphics capability of the DMA2D to include use of:

- Two (or more) screens
- Up to 256 objects per screen
- Display and movement of objects anywhere on the 2 dimensional space of the screen.

Refer to the CircleOS conception document for more details.

5.4.1 Handling bitmaps

The DMA2D peripheral allows a bitmap file to be loaded almost directly onto an LCD.

The DMA2D manages most of the bitmap formats including: 8, 16, 24 and 32-bit, direct colors and indexed colors (using a palette).

5.4.1.1 Display of a picture

In the C program, a bitmap must be declared as a byte array initialized by the contents of the binary file (translated into a .h test file).

This example describes the code used to display the 640 by 320 pixel bitmap (photo_x8 with 8-bit indexed color palette) shown here, on a display that is 320 by 240 pixels.

Because the image is larger than the size of the LCD panel, it is automatically cropped by the DMA2D API, as shown here.

The following code declares this background image:

```
const unsigned char photo_x8[] = //A picture from a camera...
    {                            //Indexed colors (8 bit palette)
    0xff,0xff,                   //32-bit alignment required for the palette
    #include "image\background\photo640X320_x8.h"    //The legacy .bmp file
    };
```

To display the picture photo_x8 as the background of a screen, all you have to do is to create an object, create a link between the object and the bitmap, then make the bitmap visible.

```
    BG_Image = DMA2D_ObjectCreate(640,320,1);        //Create an empty object
    DMA2D_ObjectAssignBitmap(BG_Image,photo_x8,0);   //Link bitmap and object
    DMA2D_ObjectSetToScreen(BG_Image, SCREEN_ID); //Attach the object to a screen
    DMA2D_ObjectSetVisible(BG_Image, 1);       //Make the object visible
    DMA2D_ScreenRedraw(SCREEN_ID);             //and repaint the whole screen.
```

The palette for the bitmap is loaded automatically by the DMA2D, based on the header information that is decoded by the application software (in this case the API).

The last command (`DMA2D_ScreenRedraw`) scans the list of objects assigned to a screen and repaints all the visible ones. Note that the size of the background image (640x320 pixels in this example) can be bigger than the size of the screen (320x240 for our demonstration platform).

The bitmap can be positioned based on provided coordinates. Point 0 for the X and Y axes is the lower left hand corner of the LCD.
To fill the entire display area of the LCD, the coordinates X and Y must be negative (or equal to 0). Positive coordinates will leave an area that has not been drawn starting in the lower left corner of the display to the coordinate where the lower left corner of the background image is displayed.

If you want to center the image, you can "move" the background before redrawing:

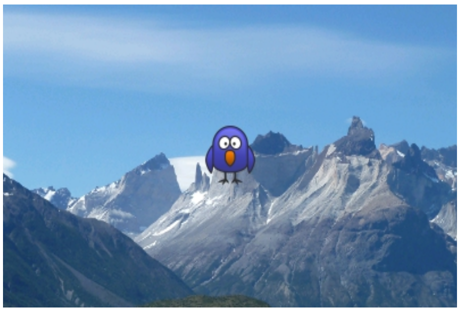
```
    DMA2D_ObjectMove(BG_Image, PosX_BG = -160, PosY_BG = -120 );  //set the position
```

Because the overall object is too big to be displayed in our screen, it is automatically cropped at the borders of the LCD panel.

5.4.1.2 Overlaying bitmaps

The DMA2D allows other objects to be placed on the background.

Overlaying objects on a background or on other objects is particularly advantageous with the DMA2D because only the areas affected by a modification are redrawn.
Without the DMA2D, changing the position of a displayed object requires either pixel-by-pixel redrawing of the full display, or complex application code (and a lot of processor time) to manage this functionality in the same manner as the DMA2D.



With the STM32F429, the DMA2D peripheral writes only the affected pixels. So, to add an object over this background, we simply create it, set the position (relative to the display and not the background) and make it visible:

```
   int x_bird = 100, y_bird = 100;
    FG_Bird = DMA2D_ObjectCreate(48, 48 , 1);
    DMA2D_ObjectAssignBitmap(FG_Bird, bird_A888, 0);
    DMA2D_ObjectSetToScreen(FG_Bird, SCREEN_ID);
    DMA2D_ObjectSetVisible(FG_Bird, 1);
    DMA2D_ObjectMove(FG_Bird, x_bird, y_bird);
```

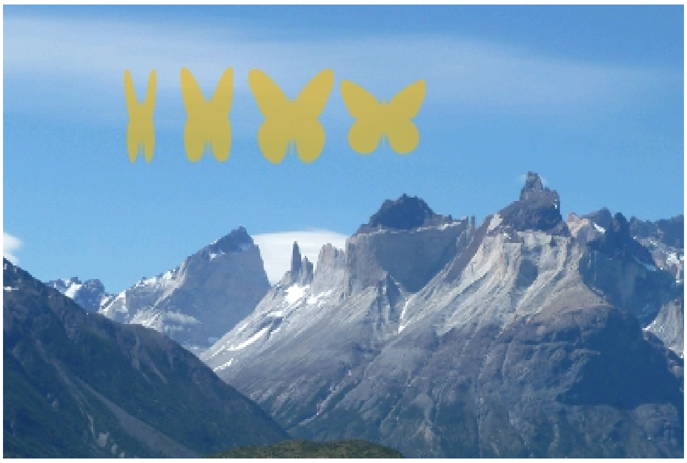
5.4.1.3 Moving objects

To make an object move across the display (for example, to make our bird fly), it is only necessary to periodically recall the last line with a new, different position (x_bird, y_bird):

```
    DMA2D_ObjectMove(FG_Bird, x_bird++, y_bird++); // bird moves up to the right.
  ..
    DMA2D_ScreenRedraw( SCREEN_ID);           //and repaint the whole screen.
```

It is possible to animate the flying bird by managing several bitmaps, each showing the bird in a different state of movement, for which we periodically change the bitmap assignment. This is how the butterfly is handled in the demonstration:

```
    DMA2D_ObjectAssignBitmap(FG_Bird, bird_fly_A888[n], 0);
  ...
    DMA2D_ScreenRedraw(SCREEN_ID);            //and repaint the whole screen.
```



Thanks to the integration of the DMA2D peripheral, the display changes that allow us to animate movement are represented by just a few lines of simple code.

5.4.2 Transparency

The transparency of an object that is overlayed on the background is handled at different levels:

- native to the original bitmap (32-bit bitmap file = 24-bit color plus an 8-bit alpha),
- global alpha constant.

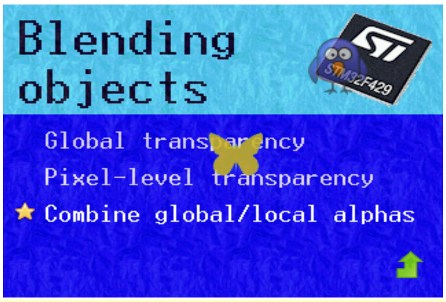
5.4.2.1 Global alpha constant

The global alpha constant defines the global transparency of an object when it is displayed. If an object has both the original bitmap transparency and a global transparency, the resulting alpha value for each pixel will be the combination of the two parameters. The two alpha values are multiplied and the result divided by 256:

$$alpha_{out} = \frac{(alpha_{pixel} * alpha_{global})}{256}$$

This means that a fully transparent (invisible) pixel in the bitmap (alpha = 0 in the bitmap file) remains invisible regardless of the global transparency of the object.

5.4.2.2 Blending

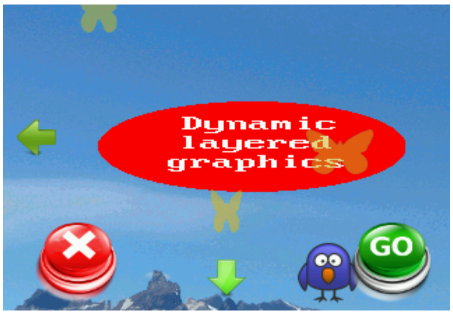


This example shows how alpha factors are combined when objects are overlayed.

The following code shows how to create an editable object, with a fully transparent part and an opaque part, that can be tuned by the global transparency ratio:

```
DMA2D_ObjectSetTransparency(FG_Text,0); //Full transparency for the next command
LCD_FillRect( 0,0,220,80,RGB_RED);      //Fill the whole object as transparent
DMA2D_ObjectSetTransparency(FG_Text,0xff);  //Elliptic surface will be opaque
DRAW_Ellipse( 110,30,110,35,RGB_RED,RGB_RED,1,1);  //Draw a centered opaque ellipse
DMA2D_ObjectSetTransparency(FG_Text,0x80);  //Set the global transparency
```

5.4.3 Editable objects



5.4.3.1 Dynamic vs Read only objects

The API allows objects to be handled in RAM.

These objects can be dynamically edited by the application.

In the image on the right, the text is written directly onto a dynamic object.

The red ellipse itself has been drawn using the standard library functions as shown in the following code:

```
FG_Text = DMA2D_ObjectCreate(220,80 ,0);  // 3rd parameter means "read-only==0"
 DMA2D_ObjectSetToScreen(FG_Text, SCREEN_ID);

 //The following commands will be applied to the object instead of the screen
 DMA2D_ObjectSelect(FG_Text);
 DMA2D_ObjectSetTransparency(FG_Text,0);
 LCD_FillRect( 0, 0, 220, 80, RGB_RED);
 DMA2D_ObjectSetTransparency(FG_Text,0xff);
 DRAW_Ellipse(110,30,110,35,RGB_RED, RGB_RED, 1, 1);
 DRAW_SetBGndColor(RGB_RED);                   // Background color for the text
 DRAW_SetTextColor(RGB_WHITE);
 DRAW_DisplayStringWithMode( 60, 45, "Dynamic", 0, NORMAL_TEXT, LEFT);
 DRAW_DisplayStringWithMode( 57, 30, "layered", 0, NORMAL_TEXT, LEFT);
 DRAW_DisplayStringWithMode( 50, 15, "graphics", 0, NORMAL_TEXT, LEFT);
```

5.4.3.2 Editing a dynamic object

After calling `DMA2D_ObjectSelect()`, these DRAW_xxx commands are assigned to the currently selected object. The coordinates are considered relative to the object and (0,0) match with the lower left corner of the rectangle containing the object.

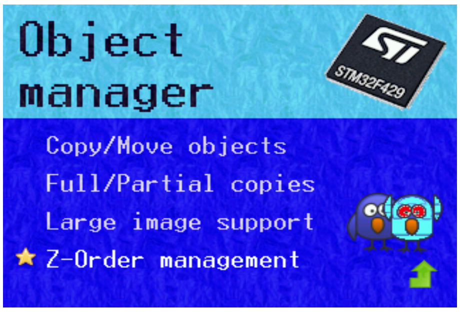
5.4.4 Advanced features

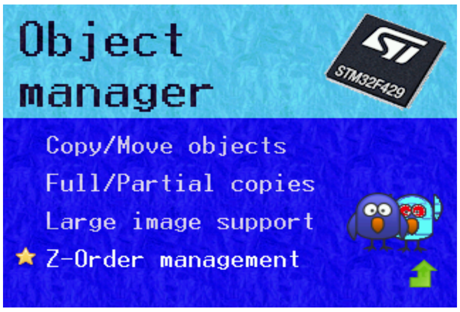
5.4.4.1 Z-order

The objects are sorted to define the order of drawing. Graphically, the Z-order specifies the relative position of the object layer (eg. Its position on the Z-axis).

The API provides functions to increment/decrement the objects Z-order, to place the object as the upper layer, or to place it as the background.

```
DMA2D_ObjectPushZ(Bird2_Image, SCREEN_SLIDE);
```





5.4.4.2 Touchscreen and objects

When implementing objects in a touchscreen interface, the function `DMA2D_ObjectFind` returns a pointer to the object that is the owner of the pixel located at (x,y).

So, when the touchscreen is pressed, it's quite simple to detect which object has been touched:

```
if (  TOUCHSCR_IsPressed() )
       {
       posx = TOUCHSCR_GetPosX();

       posy = TOUCHSCR_GetPosY();

       SelectedObject = DMA2D_ObjectFind(CurrentScreen, posx, posy );

       if ( SelectedObject   == FG_Bird )
           {
           bird_hit = 1;

            ....
```

5.5 Managing applications on your EvoPrimer

The applications can be managed by the **circle_mgr.exe** utility, which is in the directory [RIDE7_INSTALL_DIR]\Bin. This utility can:

- List the currently loaded CircleOS applications.
- Add new CircleOS applications.
- Remove CircleOS applications.
- Check how much Flash memory is available.

Refer to the http://www.stm32circle.com/ web site for more information about this utility.

The following commands are available with the **circle_mgr.exe** utility:

Command	**Syntax**	**Description**

List	L	List the loaded applications.
		The following information will be output:
		circle_mgr.exe L
		`Reading FAT table...`
		`App0: Name=Maze, Addr=0x08006000, Size=8KB`
		`App1: Name=Breakout, Addr=0x08008000, Size=4KB`
		`Largest free block= 92KB`

Add	Afilename	Add a new application (object file).
		circle_mgr.exe Ac:\tmp\level.o
		`Linking file C:\tmp\level.o...`
		`Link of C:\tmp\level.o succeeded...`
		`Hex file generated...`
		`Blank-checking the FLASH area...OK`
		`Programming file _tmp_.ld.hex to flash...OK`
		`Registering application in FAT... OK`

Erase	E*
	Eappname
		`circle_mgr.exe EMaze` `/*remove only 'Maze'*/`
		`circle_mgr.exe E*     /*remove ALL apps */`

Wait	W	When a command list is launched through a batch file, the W command allows you to pause the execution and to check the intermediate results.
Start	S	Start CPU execution

⚠ **Caution: Any** hex file can be programmed to the Flash memory of your EvoPrimer using the **cortex_pgm.exe** utility. However, doing this will destroy your CircleOS firmware, and you will have to reinstall it if you wish to use it later (refer to chapter 5.7 for details).

5.6 Selecting the current application

One application is considered to be the “current application”. The ID of the current application is saved in the backup memory. From the main menu, you can launch it directly.

To change the current application, select the “Application” command from the main menu. Then select the application you wish to specify it as the “current application” and push the button. The new “current application” name will now appear in the main menu.

5.7 Downloading new applications

On the http://www.stm32circle.com/projects Circle web site you will find a database where the members can share their applications with the stm32circle community.

An application can include both the source files and the object files, or just the object files.

An application is generally made of one object file, but may occasionally have several of them. The linking of the application in such a case can be done either using the **circle_mgr.exe** software (available in the “[RIDE7_INSTALL_DIR]\bin” directory) or within the Ride7 environment. When an application is split into several object files, these object files must be placed in a library in order to pass a unique filename as an argument to **circle_mgr.exe**.

5.8 Resetting your EvoPrimer

5.8.1 Hardware reset

A hardware reset will restart your EvoPrimer (restoring it to a clean state), but will not remove its ROM contents.

1. Remove any USB cable.
2. Remove the plastic case protecting the target board.
3. Change the switch to its open state (this disconnects the internal battery) then replace the switch as it was.
4. Put the plastic case back in place.
5. Restart your EvoPrimer by pressing its button.

5.8.2 Software reset - CircleOS

This software reset erases all the Flash memory in your EvoPrimer and all applications. The CircleOS is replaced. This operation takes approximately 30 seconds:

1. Connect your EvoPrimer's debug USB port to your PC.
2. Power-up your EvoPrimer by pressing its button.
3. Open a command prompt from Windows (Start | Programs | Accessories | Command prompt).
4. Reload the initial program, using the command:

`Update_EvoPrimer_STM3242x_Circle_OS.bat` from Ride7 or
`Program_Evo_STM3242x_CircleOS_only` from stm32circle website in the upgrade zip file.

Note: This operation can also be done with the following command:
`Cortex_pgm TSTM32F429xI E PEvo_Circle_STM3242x.hex S`

where `TSTM32F429xI` selects the appropriate device
`E` erases the ROM.
`Evo_Circle_STM3242x.hex` is the full application available on the web site.
`S` restarts the device.

5.8.3 Software reset - Factory configuration

If you have been experimenting with CircleOS applications and have modified your EvoPrimer configuration, you may want to restore the initial (factory) configuration. This operation takes approximately 30 seconds:

1. Connect your EvoPrimer's debug USB port to your PC.
2. Power-up your EvoPrimer by pressing its button.
3. Open a command prompt from Windows (Start | Programs | Accessories | Command prompt).
4. Launch the command **`Restore_EvoPrimer_STM3242x_Circle_Factory.bat`**
   If the command is not recognized, change the current directory to the Ride7 installation directory. This can be done with the following command (adapt it to your actual configuration if you did not install Ride7 in its default location):
   ```
   cd "C:\program files\Raisonance\Ride\lib\ARM\CircleOS"
   ```
5. The batch script erases your EvoPrimer, reprograms it with its factory ROM image which is in the circle hex file, then restarts the device.

Note: This operation can also be done with the following command:
`Cortex_pgm TSTM32F429xI E PEvo_Evo_STM3242x_Circle_Factory S`
where `PEvo_STM3242x_Circle_Factory.hex` programs the device with the proper hex file.

6. Developing CircleOS applications

The full source files of CircleOS are available on the http://www.stm32circle.com/ web site.

Once registered, you will be able to download them, along with many resources for developing your application. The games originally delivered with the EvoPrimer show working application examples.

6.1 Developing your first CircleOS application

Creation of a CircleOS application is done automatically in Ride7:

1. Navigate to **Project | New project**.
2. Set the **Type** selection list to **New application**.
3. In the **Processor** selection list, select the **STM3242x_Evo_CircleOS** device.
4. Select an application name such as **My CircleOS application**.
5. Define the location where your new project will be created.
6. Click the **Finish** button. Your new project will be created, with an application containing a CircleOS application skeleton as well as the **Evo_Circle_STM3242x.elf** and **FAT_OP4.elf** files necessary to connect your application to CircleOS.
7. Open the *Application.c* file.
8. Search for the `Application_Name` variable in the file.
9. Change the `Application_Name` value from `My App` to `HELLO`.
10. In the `Application_Handler` function, create a new string as follows:
    `const char msg[] = "Hello, World!";`
11. Use the `DRAW_DisplayString CircleOS` service to display the `msg` string variable you just created:
    `DRAW_DisplayString( 5, 20, msg, sizeof(msg)); // X, Y, string, length`
12. Build your project using the **Project | Make Project** command.
13. Connect your EvoPrimer to your PC using the USB cable (using the Prrimers debug USB port).
14. From Ride7, go to **Debug | Start**, this programs your application to your EvoPrimer. This may take about 15 seconds.
15. Go to **Debug | Run**.
16. On your EvoPrimer, select your application name on the main menu.

Your application is now on your EvoPrimer. For further information about CircleOS application programming and available OS services, please visit *http://www.stm32circle.com/*.

6.2 Libraries

Some common services are offered to ease your development of CircleOS applications.

1. The STM32 libraries, written by ST, provide access to the embedded peripherals (such as timers, ADC, communication interfaces, thermometer, etc...) of the STM32 microcontroller.
2. The low-level CircleOS functions that provide an easy access to the EvoPrimer's on-board peripherals: 3D accelerometer, LCD monitor, button, buzzer, battery, LEDs.
3. The graphical functions that provide powerful high-level functionality: Menu management, pointers (linked to the 3D accelerometer), character maps, sound.

The source files of these libraries can be found on:

- ST web site for the STM32 libraries (includes other documentation about the STM32 library)
- On the http://www.stm32circle.com/ web site for the CircleOS libraries (registration required).

Library documentation is accessible from Ride7.

6.3 Debugging your application

In order to debug your application:

1. Go to **Project | Properties** in Ride7.
2. In the **Configuration** selection box, select the **Circle_Debug** configuration (which is the default).

Ride7 takes care of all the settings required to switch between debug and release mode through the use of these configurations.

> **Note**: The standard STM32F429 EvoPrimer is limited to debugging in the first 64KB only. A software key can be purchased on *http://www.stm32circle.com/* to allow debugging in the whole 2048KB of memory.

6.4 Sharing your application with the Circle community

Once your application works properly, you can share it with the other members through the *http://www.stm32circle.com/* community.

7. Conformity and recycling

ROHS Compliance (Restriction of Hazardous Substances)

KEOLABS products are certified to comply with the European Union RoHS Directive (2002/95/EC) which restricts the use of six hazardous chemicals in its products for the protection of human health and the environment.

The restricted substances are as follows: lead, mercury, cadmium, hexavalent chromium, polybrominated biphenyls (PBB), and polybrominated diphenyl ethers (PBDE).

CE Compliance (Conformité Européenne)

KEOLABS products are certified to comply with the European Union CE Directive.

In a domestic environment, the user is responsible for taking protective measures from possible radio interference the products may cause.

FCC Compliance (Federal Communications Commission)

KEOLABS products are certified as Class A products in compliance with the American FCC requirements. In a domestic environment, the user is responsible for taking protective measures from possible radio interference the products may cause.

WEEE Compliance (The Waste Electrical & Electronic Equipment Directive)

KEOLABS disposes of its electrical equipment according to the WEEE Directive (2002/96/EC).

Upon request, KEOLABS can recycle customer's redundant products.

For more information on conformity and recycling, please visit the KEOLABS website *www.keolabs.com*

8. Glossary

Term	Description
STM32F429 EvoPrimer	EvoPrimer for STM32F429 MCU, a microcontroller evaluation & development platform
EvoPrimer	Commercial products which are derived from the Raisonance Open4 and distributed by STMicroelectronics for exploring, evaluating, developing and fine tuning applications for a variety of ST 8 and 32-bit microcontrollers. Certain use limitations may apply to these commercial products
Open4	Raisonance versatile hardware platform, which is used in a variety of commercial products that allow users to explore, evaluate, develop and fine tune applications for a range of microcontrollers that are mounted on Open4 compatible target boards. Also refered to as the "base platform." Also distributed under the name EvoPrimer base
Target board	Boards featuring target microcontrollers which, when installed on an Open4-derived base allow execution of applications, programming of the target microcontroller and debugging of the applications running on the target microcontroller. Hardware features of target boards will vary depending on the target MCU and its features. Some features of the base platform may not be available depending on the features and capabilities of the target microcontroller
Extension board	Application-specific extension board, connected to the extension connector.
CircleOS	Embedded OS running on the Primer development kits.

9. Index

10. History

Date	Description
01 Oct 13	Initial version.

KEOLABS

Disclaimer

Information in this document is subject to change without notice and does not represent a commitment on the part of the manufacturer. The software described in this document is provided under license and may only be used or copied in accordance with the terms of the agreement. It is illegal to copy the software onto any medium, except as specifically allowed in the license or nondisclosure agreement.

No part of this manual may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording, or information storage and retrieval systems, for any purpose other than the purchaser's personal use, without prior written permission.

Every effort has been made to ensure the accuracy of this manual and to give appropriate credit to persons, companies and trademarks referenced herein.

This manual exists both in paper and electronic form (pdf).

Please check the printed version against the .pdf installed on the computer in the installation directory of the most recent version of the software, for the most up-to-date version.

The examples of code used in this document are for illustration purposes only and accuracy is not guaranteed. Please check the code before use.