
STM32 & SCILAB

Readme

Rev 1.1

Table of Contents

2

Title	Page
Objectives	<u>3</u>
Systems Check	<u>4</u>
Hardware setup	<u>8</u>
Software setup	<u>12</u>
Tools usage	<u>14</u>
Installation	<u>15</u>
Xcos application example	<u>26</u>
Build Application flow	<u>48</u>
Run project	<u>50</u>
Diagram code variation	<u>57</u>



Must be read



Recommended reading



Detailed

Objectives

3

- Hands-on workshop to show you the steps needed to quickly develop STM32 graphical applications using SCILAB XCos environment.
- Know tools installations and settings to be able to start development.
- Know « C » Code Generation possibility
- Know how to develop application from scratch
- Know where to obtain additional technical support

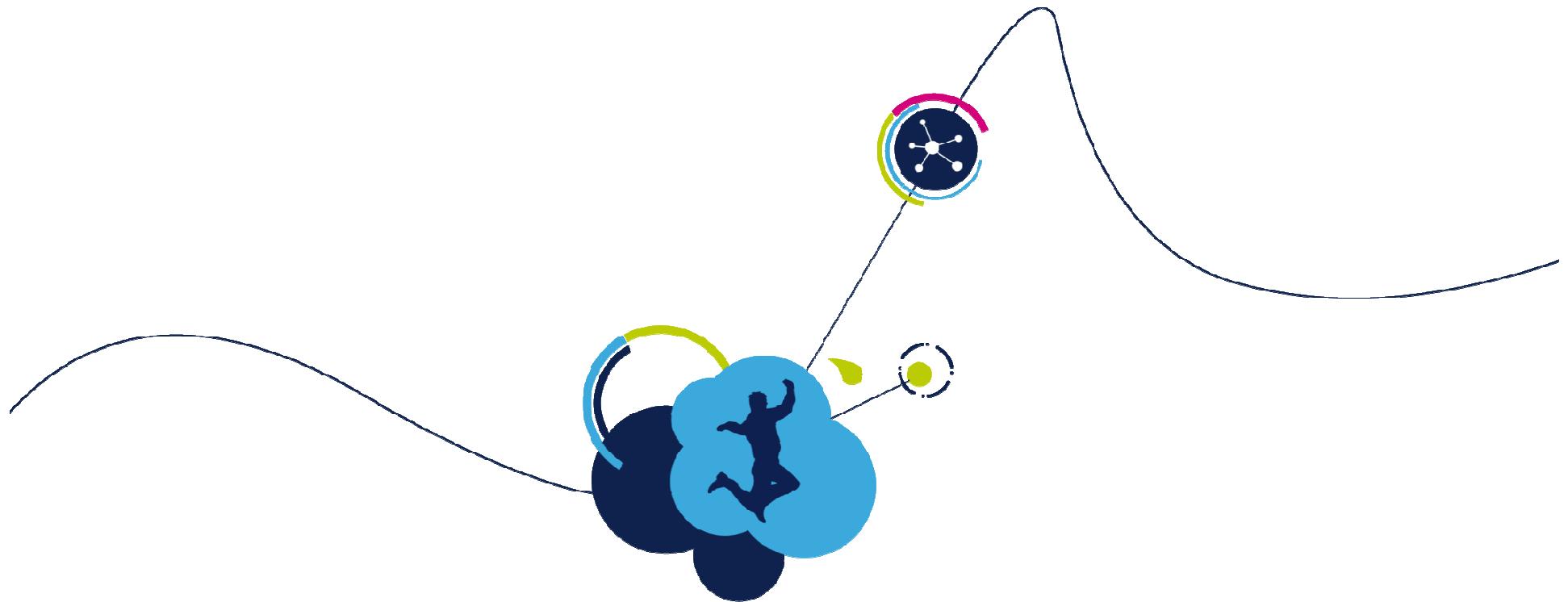


- Mandatory Software

- From SCILAB
 - SCILAB Xcos version 5.5.2
- From STMicroelectronics
 - STM32CubeMX
- One of following Toolchain
 - EWARM from IAR
 - MDK-ARM from Keil
 - TrueSTUDIO from Atollic
 - SW4STM32 from STMicroelectronics
- STM32-SCILAB toolkit to develop STM32 applications

- Hardware

- Any electronic application board with STM32 and SWD/JTAG connection.
- STLink or 3rd parties dongle if not integrated to STM32 application board.



Pre-requisites

Pre-requisites

6



- STM32CubeMX

- Have a look to STM32CubeMX videos to know how using this powerfull tool.

FEATURED VIDEOS

STM32CubeMX in 5 points 

Lasts 9 minutes and 12 seconds

1. STM32CubeMX software installation
2. MCU selection
3. Configuration

- Selection of modes and
- Set up clocks
- Set up peripherals

4. Software

- C code skeleton generation
- And user code with USART communication example

5. Power consumption evaluation with wizard

Insert your user code for USART communications  

Collection of embedded software blocks
abstracting the used STM32 and running
on specific targets

[See All](#)

STM32Cube – Overview 

Lasts 6 minutes and 44 seconds

STM32Cube, a 100% free solution to ease your life, that combines:

- A PC software configuration tool
- STM32 embedded software blocks

C code generation for initialization, according to user choices

LLAPs available from STM32L4 LS F0
LLAPs for STM32 series Q1-2017

Pre-requisites

7



- **Toolchain**

- You must be comfortable with one of following toolchain.

- **Ewarm from IAR**

Embedded Workbench for Arm (Ewarm)



- **μVision from Keil**

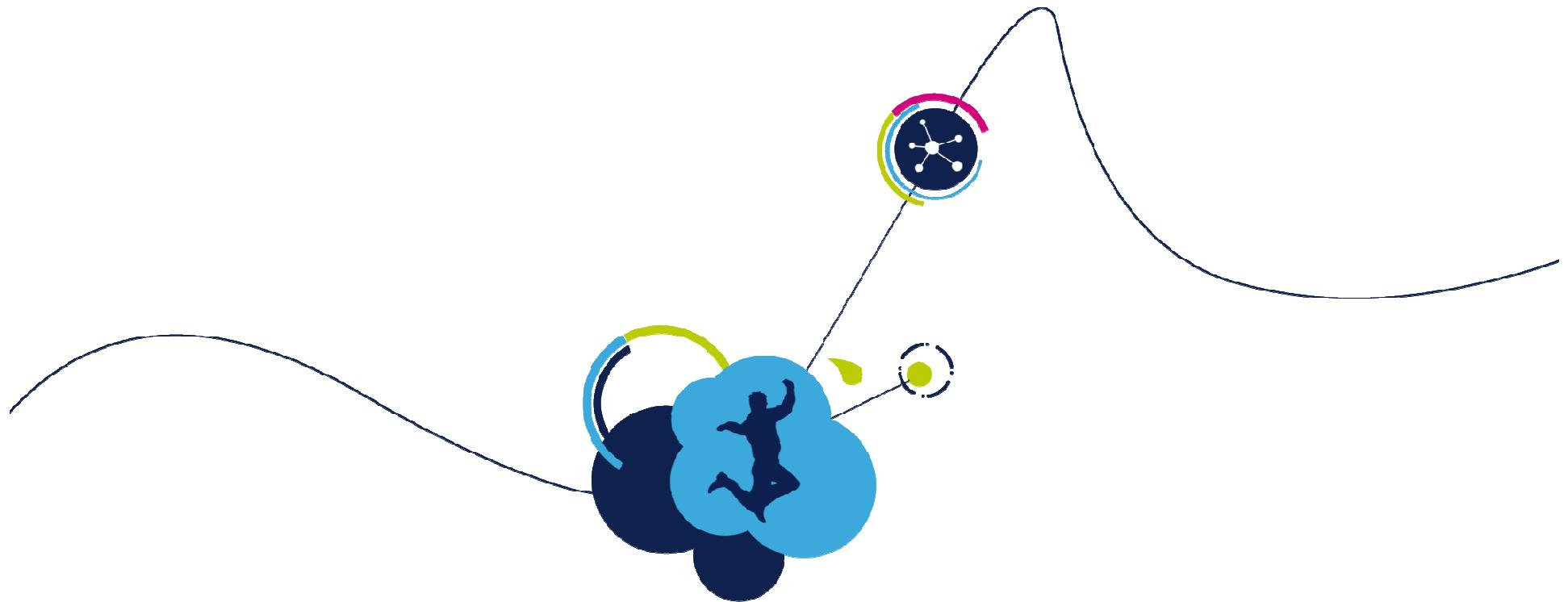


- **TrueSTUDIO from Atollic**



- **SW4STM32 from ST**





Hardware setup

Step #1 – Hardware selection

9



- Use one of STM32 boards including STLink
 - Nucleo, Discovery, EvaluationBoard etc...
 - STM32F3348-DISCO and STM32F429i-DISCO will be used during examples.



Nucleo Board



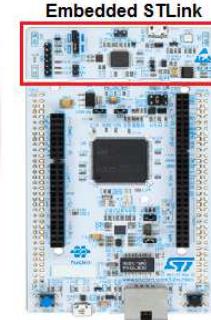
Discovery Board



Evaluation Board



STLink



Embedded STLink

- Or STM32 application board connected to SWD (Single Wire Debug)/JTAG dongle.
 - STLink, ULink2, JLink etc..



STLink



ULink2



JLink

Step #2 – Hardware connection

10



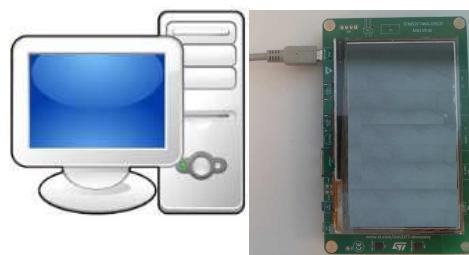
- Connect USB dongle port to PC USB port
 - And connect STM32 HE10 20 pins dongle connector to STM32 target board

Exemple: Connect STLink
dongle USB to PC on one side.



Exemple:
Connect HE10 20 pins STLink dongle connector
to HE10 20 pins connector of
STM32 Evaluation Board on the other side.

- Or connect PC USB port to embedded STLink



Exemple: Connect USB PC port to STLink USB port embedded in
STM32 board.



Usually, all ST recent boards embedd
STLink tool.

Step #3 – Hardware connection

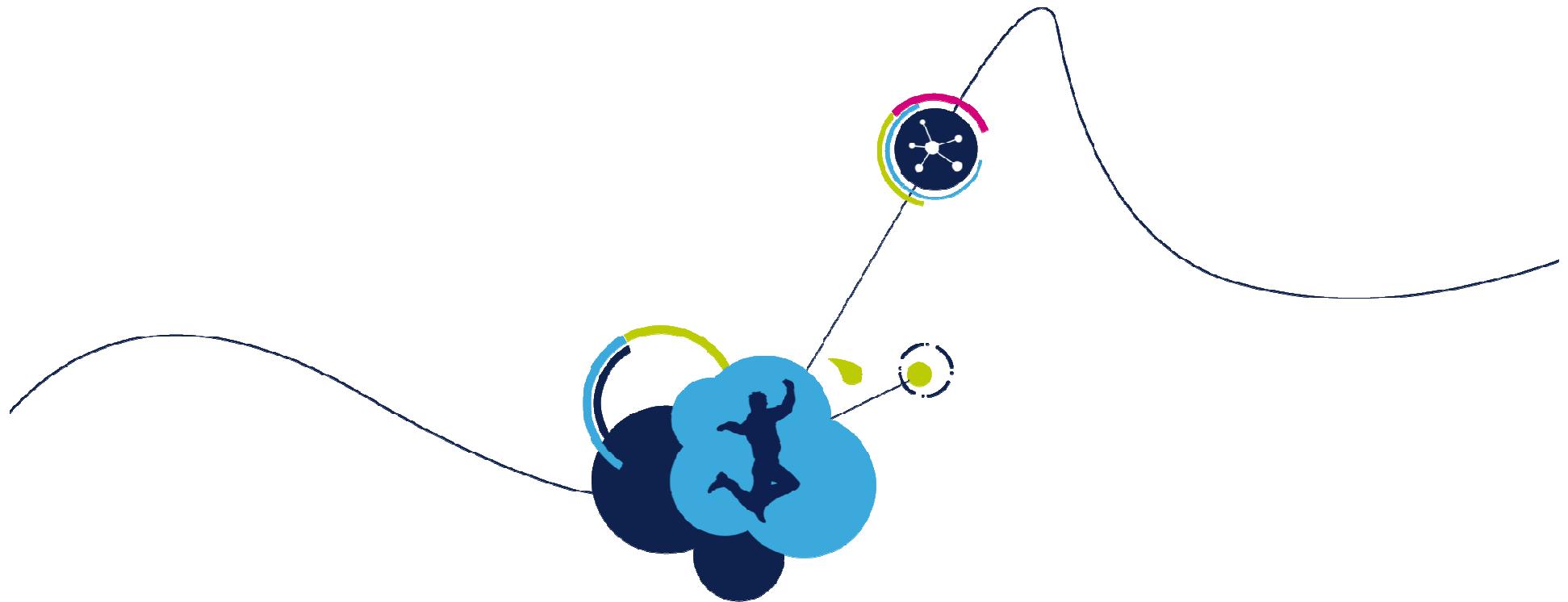


- As soon as you are using ST-LINK

- look at
<http://www.st.com/web/catalog/tools/FM146/CL1984/SC720/SS1450/PF251168?searchtype=partnumber>
- « Related Tools and Software » section to check or update firmware

Related Tools and Software

Related Tools and Software	
Part Number	Description
STSW-LINK004	STM32 ST-LINK utility
STSW-LINK005	ST-LINK/V2 firmware upgrade
STSW-LINK009	ST-Link, ST-Link/V2, ST-Link/V2-1 USB driver signed for XP, Windows7, Windows8



Software setup

Quick description of tools

13



Toolchain



SCILAB

High level language for complex calculation



STM32Cube
Embedded Software

Collection of embedded software components, highly portable from one STM32 to another

XCos

Graphical development environment for simulation

STM32CubeMX



Configuration software tool on the PC, able to generate initialization C code versus user choices



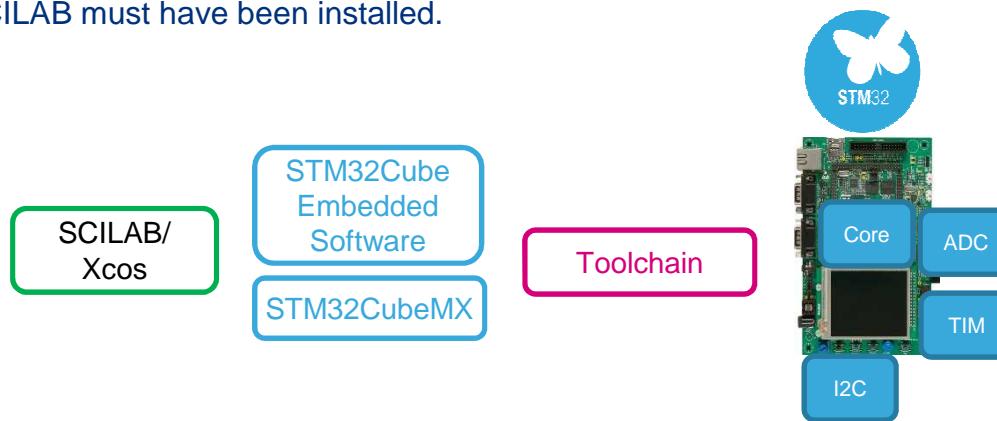
Tools usage

14



- **Code Generation for STM32**

- Data (input or output) obtained within STM32 through its peripherals (ADC, Timers, ...) and algorithm fully executed on STM32.
- Code generated from Xcos diagram.
- Needed tools: SCILAB/Xcos, STM32CubeMX, one of supported toolchains and STM32 toolbox for SCILAB must have been installed.



Step #1 – Software installation

15



- Install SCILAB 5.5.2 software
 - SCILAB/Xcos is mandatory
 - <http://www.scilab.org>
- Install STM32CubeMX
 - Download and documents available from : www.st.com/microxplorer
- Install toolchain (Cf Slide 4 : « Systems Check »)
 - Cf Slide 3 « Systems Check » to get link to supported 3rd parties download area.

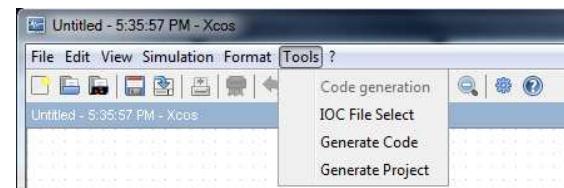
Step #2 – Software installation

16



- Install STM32 for SCILAB

- This toolkit is mandatory to be able to design Xcos graphical application for STM32.
- Set ATOMS Config Proxy and Network parameters to download using atomsGui() command
 - atomsSetConfig('parameter', 'value') for parameters :
 - useProxy , proxyHost, proxyPort, proxyUser, proxyPassword and offline
- Or download from <https://atoms.scilab.org/> (Xcos or Real-Time) and enter the install command to install xcos_stm32_toolbox
- STM32 is available from Xcos diagram
 - Functionalities have been added to the « Tools » menu :



Step #4 – STM32 for SCILAB integration

17



- STM32 Help document available for SCILAB

- Enter help from Scilab 5.5.2 Console
|-->help

- Help Browser window opens

- STM32 dedicated Help is available

Help regarding STM32 functionalities

The screenshot shows the Scilab Help Browser window. On the left is a tree view of help categories, and on the right is a detailed list of topics under the selected category. A red arrow points from the text "Help regarding STM32 functionalities" to the "STM32 Help" category in the tree view.

STM32 Help

- **STM32 Getting Started**
 - STM32 Environment Loading —
 - STM32 Macros —
- **STM32 Tools**
 - STM32_Code_Generation —
 - STM32_IOC —
 - STM32_Preferences —
 - STM32_Project_Generation —
- **STM32 Drivers**
 - ADC_Read —
 - GPIO_Exti —
 - GPIO_Read —
 - GPIO_Write —
 - TIMER —
 - USART_Receive —
 - USART_Send —

Step #4 – STM32 for SCILAB integration

18



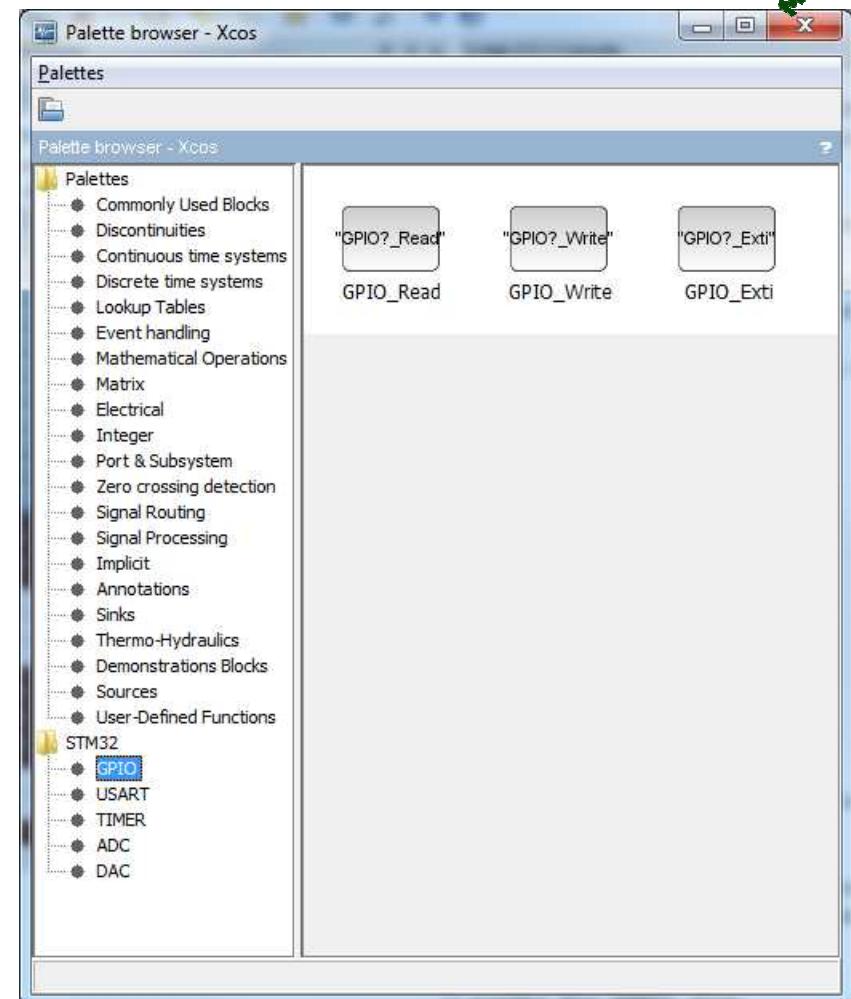
- STM32 Palette for STM32 peripherals integrated to Palette browser

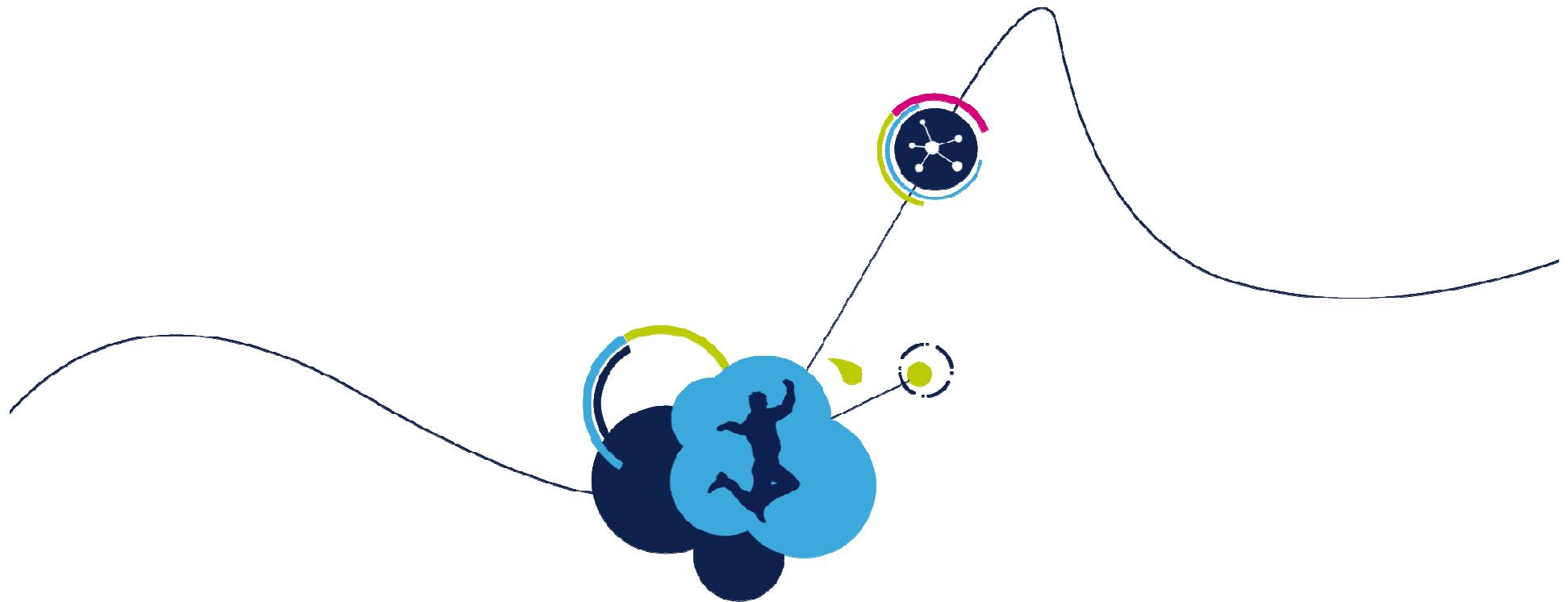
1. Available drivers:

- GPIO
 - Read, Write, External Interrupt
- USART
 - Send, Receive
- TIMER
 - Output PWM,
- ADC
 - Read
- DAC
 - Write



Look at release note for restrictions and not supported functionalities.





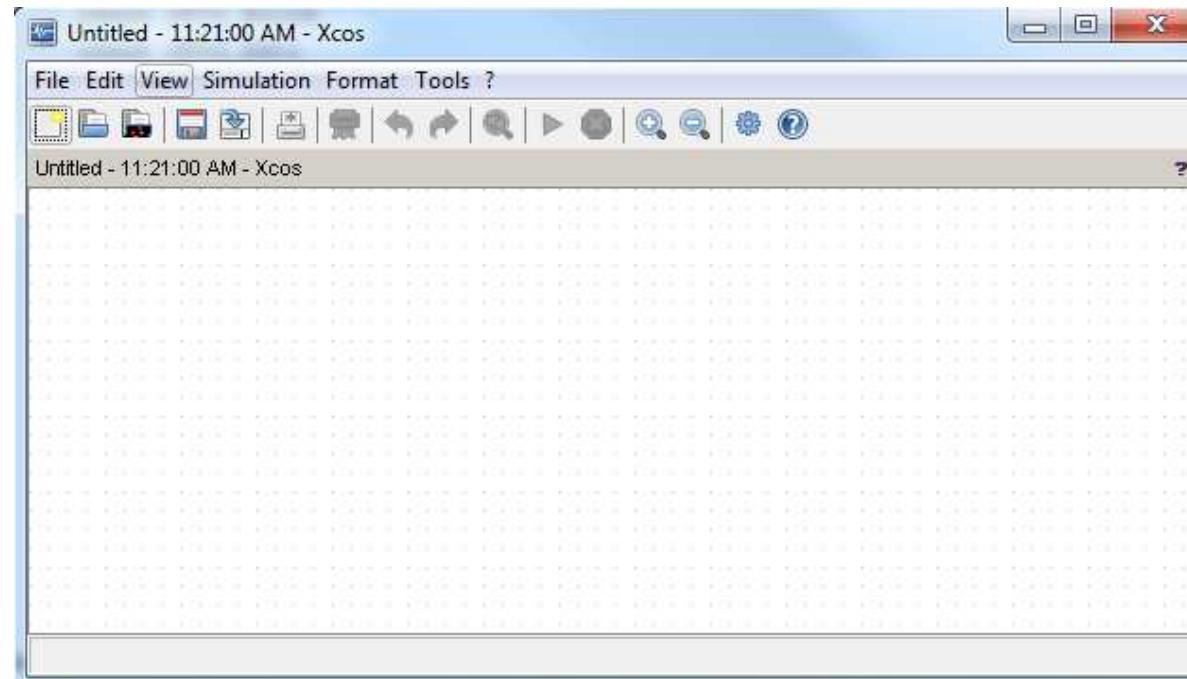
Xcos diagram setting

Xcos diagram Setting 1/5

20



- Enter xcos command from Scilab 5.5.2 Console `|-->xcos` or click xcos icon . 
- Then new xcos diagram is opened

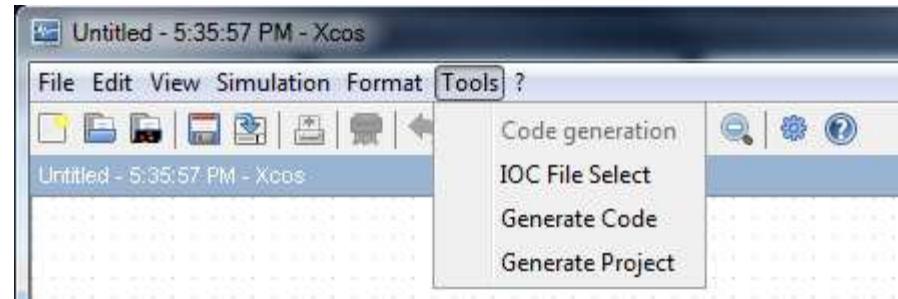


Xcos diagram Setting 2/5

21



- Tools tab gives possibility to :



1. Select .ioc file.
 - .ioc file is STM32 configuration done using [STM32CubeMX](#). (cf slide 6)
2. Generate C code for this diagram.
 - Based on STM32 HAL C code libraries
3. Generate project for this diagram.
 - STM32CubeMX generates project for the selected toolchain.



Toolchain is selected from STM32CubeMx
Project settings (Alt+P).

Xcos diagram Setting 3/5

22



- Save diagram :

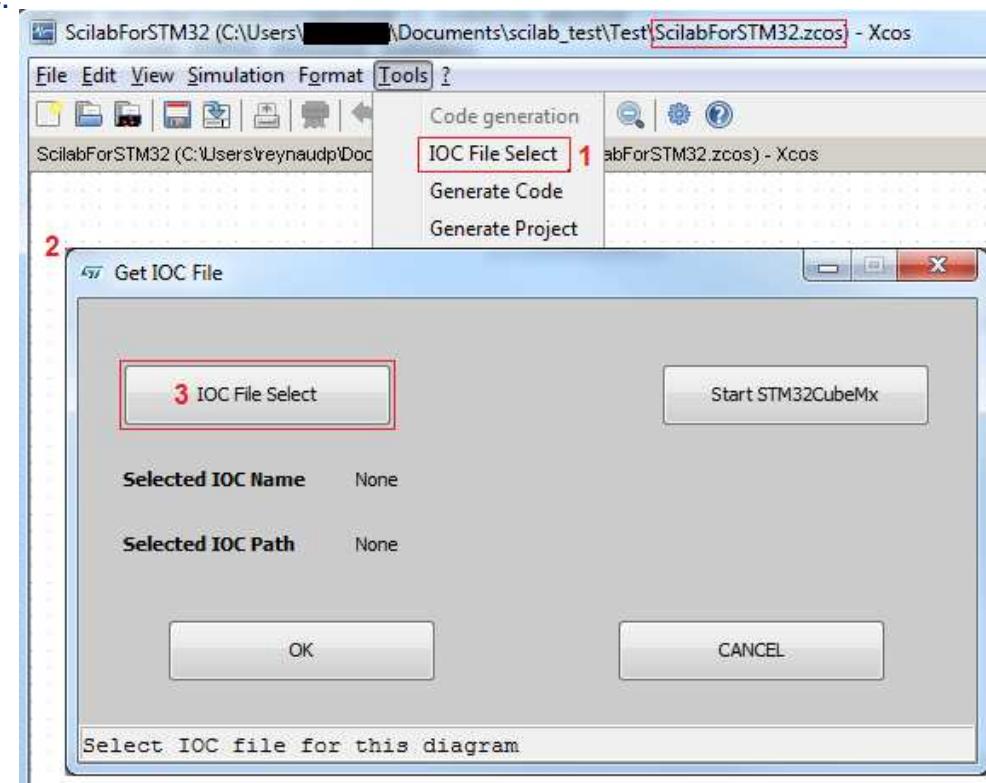
- C:\Users\xx\AppData\Roaming\Scilab\scilab-5.5.2\ « diagramName.conf » contains informations about STM32 used and ioc file (path and name) attached to the diagram.
- This file is created when you attach an ioc file to diagram from « Tools > IOC File Select »
- It is better to save the file before selecting the ioc.

- Example:

1. Save diagram as ScilabForSTM32.zcos
2. Select « IOC File Select »
3. Get ioc File window opens
4. Push IOC File Select button.
5. Browser opens
6. Select an ioc file
7. Push OK button



- ioc file is generated from STM32CubeMx tool.
- It contains STM32 configuration (hardware and peripherals settings) used for the diagram.
- See following Xcos application example for details.



Xcos diagram Setting 4/5

23



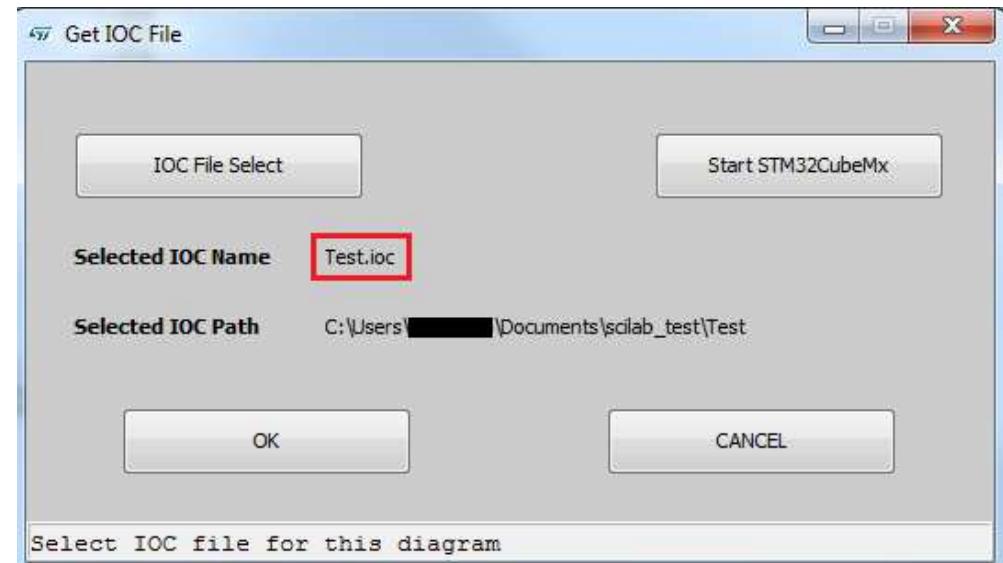
- IOC file MUST be selected every time diagram is opened
 - Selected IOC Name and Selected IOC Path are automatically updated with parameters saved in C:\Users\xx\AppData\Roaming\Scilab\scilab-5.5.2\« DiagramName.conf »
 - Example:
 1. Diagram saved as ScilabForSTM32.zcos
 2. Select « IOC File Select »
 3. Selected IOC Name and Selected IOC PathAre updated with parameters from:

AppData\Roaming\Scilab\scilab-5.5.2\ScilabForSTM32.conf

ScilabForSTM32.conf file contents:

```
Test.ioc
C:\Users\[REDACTED]\Documents\scilab_test\Test
STM32F4
```

1. loc file name
2. loc file path
3. STM32 family



- Convention : Same name for ioc file and its repository.

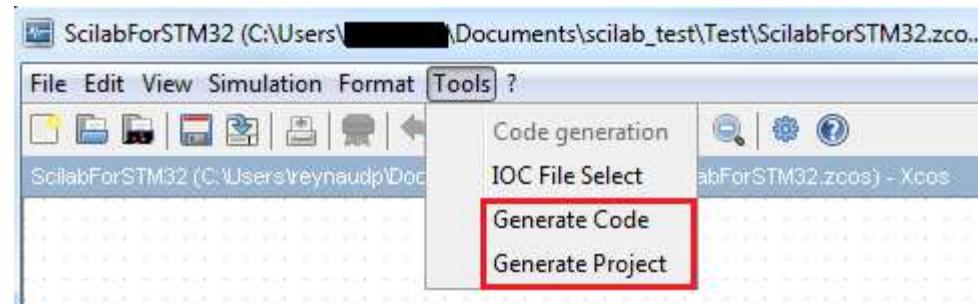


Xcos diagram Setting 5/5

24



- Generate Code and Generate Project



Cf slides:

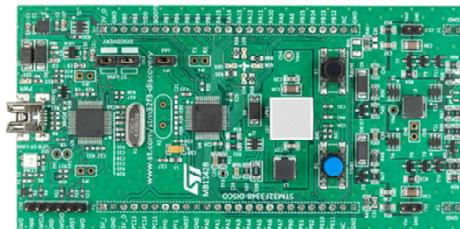
code generation
and
project generation

Xcos application example



- **Hardware :**

- Example based on STM32F3348-DISCO
- Configuration :
 - Leds (LED3/4/5/6)
 - Push Button (User blue button)
 - USART2 Virtual Com Port (SB14&SB16 soldered)
 - ADC1
 - TIM1 & TIM6



- **Software application :**

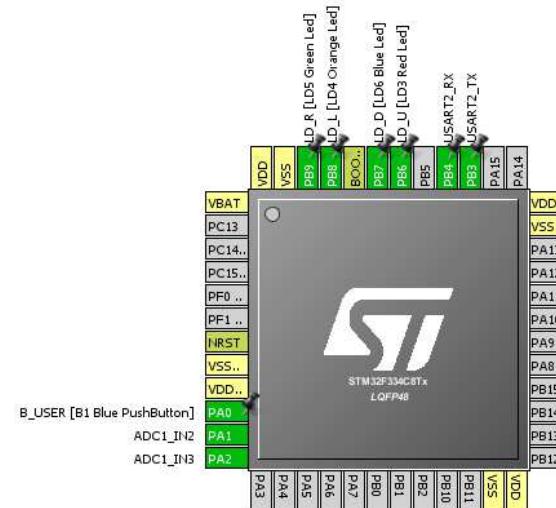
- Use TIM1 to blink LED3 at 1Hz
- Use TIM6 to blink LED4 at 2Hz
- Use TIM6 to trig ADC1 channels 2&3 conversion
- Blink Led6 when user push button is pressed
- Send ADC1 channel 3 values on USART2 when user push button is pressed

STM32CubeMX STM32F3348 Pinout



- Hardware pinout configuration

- PA0 : GPIO_EXTI0
- PA1 : ADC1_IN1
- PA2 : ADC1_IN2
- PB3 : Usart2_Tx
- PB4 : Usart2_Rx
- PB6 to PB9 : GPIO_Output

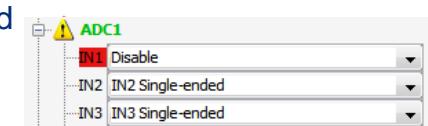


- Hardware setting

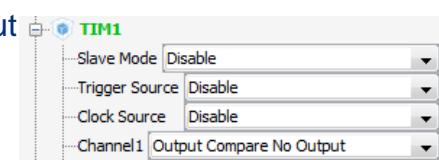
- USART2 is Asynchronous



- ADC1 IN2 & IN3 Single-ended



- TIM1 Channel1 as Ouput Compare No output



- TIM6 Activated (No Output)



STM32CubeMX Peripheral settings 1/2

27



- Peripheral configuration :

- USART2

- Baud Rate : 115200
- Word Length: 8 Bits
- Parity: None
- Stop Bits: 1
- Enable global interrupt

USART2 Configuration

Parameter Settings	User Constants	NVIC Settings	GPIO Settings	DMA Settings
Interrupt Table				
USART2 global interrupt / USART2 wake-up interrupt through EXT line...		Enabled	Preemption Priority	Sub Priority
		<input checked="" type="checkbox"/>	0	0

- ADC1

- Injected Channels 2&3
- Interrupt at end of sequence of conversion
- Conversion triggered from Timer6
- Interrupt Enabled

ADC1 Configuration

Parameter Settings	User Constants	NVIC Settings	
Interrupt Table	Enabled		
ADC1 and ADC2 interrupts		<input checked="" type="checkbox"/>	

ADC1 Configuration

Configure the below parameters :

ADCx_Common_Settings	
Mode	Independent mode
ADC_Settings	
Clock Prescaler	ADC Asynchronous clock mode
Resolution	ADC 12-bit resolution
Data Alignment	Right alignment
Scan Conversion Mode	Enabled
Continuous Conversion Mode	Disabled
Discontinuous Conversion Mode	Disabled
DMA Continuous Requests	Disabled
End Of Conversion Selection	End of sequence of conversion
Overrun behaviour	Overrun data overwritten
Low Power Auto Wait	Disabled
ADC_Regular_ConversionMode	
Enable Regular Conversions	Disable
ADC_Injected_ConversionMode	
Enable Injected Conversions	Enable
Number Of Conversions	2
External Trigger Conversion Edge	Trigger detection on the rising edge
External Trigger Source	Timer 6 Trigger Out event
Injected Conversion Mode	None
Queue Injected Context	Disabled
Rank	
Channel	Channel 2
Sampling Time	7.5 Cycles
Offset Number	No offset
Injected Offset	0
Rank	
Channel	Channel 3
Sampling Time	7.5 Cycles
Offset Number	No offset
Injected Offset	0



STM32CubeMX Peripheral settings 2/2

- Peripheral configuration :
 - TIM1**
 - Default configuration
 - TIM1 Update interrupt enabled
 - TIM6**
 - Trigger event :Update Event
 - TIM6 global interrupt enabled

TIM6 Configuration

Parameter Settings			User Constants		NVIC Settings		DMA Settings	
Interrupt Table			Enabled	Preemption				
TIM6 global and DAC1 underrun error interrupts			<input checked="" type="checkbox"/>	0				

TIM1 Configuration

Parameter Settings			User Constants		NVIC Settings		DMA Settings	
Interrupt Table			Enabled	Preemption Priority				
TIM1 break and TIM15 interrupts			<input type="checkbox"/>	0				
TIM1 update and TIM16 interrupts			<input checked="" type="checkbox"/>	0				
TIM1 trigger and commutation and TIM17 interrupts			<input type="checkbox"/>	0				
TIM1 capture compare interrupt			<input type="checkbox"/>	0				

TIM6 Configuration

Configure the below parameters :

Counter Settings	
Prescaler (PSC - 16 bits value)	0
Counter Mode	Up
Counter Period (AutoReload Register - 16 bits va...)	0
Trigger Output (TRGO) Parameters	
Trigger Event Selection	Update Event

Pin Configuration

GPIO							
Pin Name	Signal on Pin	GPIO mode	GPIO Pull Up ...	Maximum out...	Fast Mode	User Label	Modified
PA0	n/a	External Interrupt	No pull up pull down	n/a	n/a	B_USER [B1 Bl...	<input checked="" type="checkbox"/>
PB6	n/a	Output Push Pull	No pull up pull down	Low	Disable	LD_U [LD3 Red...	<input checked="" type="checkbox"/>
PB7	n/a	Output Push Pull	No pull up pull down	Low	Disable	LD_D [LD6 Blu...	<input checked="" type="checkbox"/>
PB8	n/a	Output Push Pull	No pull up pull down	Low	Disable	LD_J [LD4 Oran...	<input checked="" type="checkbox"/>
PB9	n/a	Output Push Pull	No pull up pull down	Low	Disable	LD_R [LD5 Gre...	<input checked="" type="checkbox"/>

PA0 Configuration :

GPIO mode : External Interrupt Mode with Falling edge trigger detection

GPIO Pull Up Pull Down : No pull up pull down

User Label : B_USER [B1 Blue PushButton]

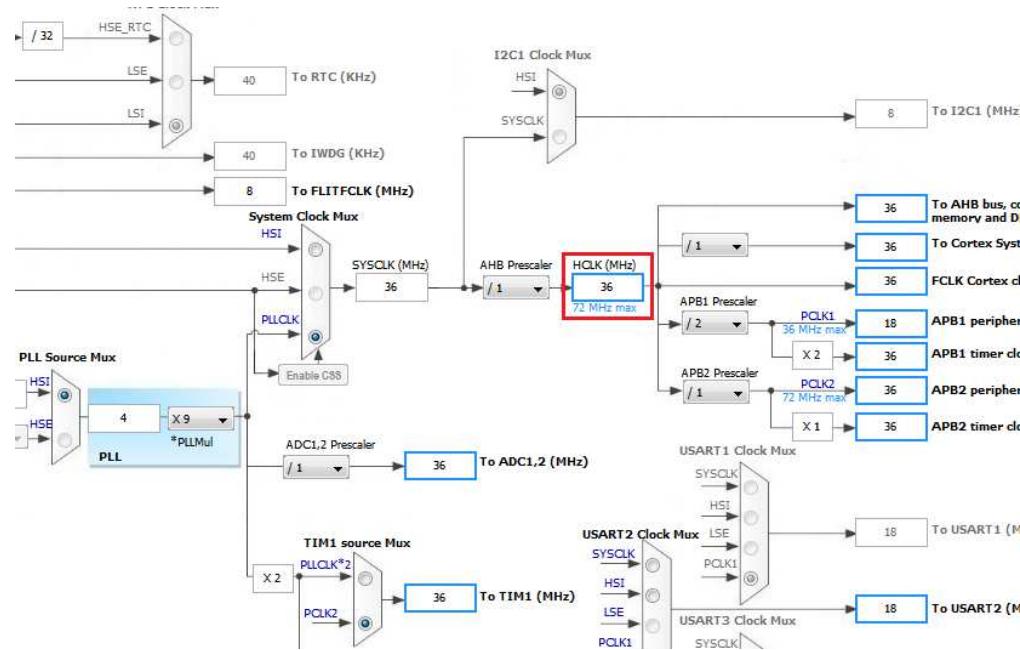
Group By IP: Apply Ok Cancel

STM32CubeMX Clock Configuration

29



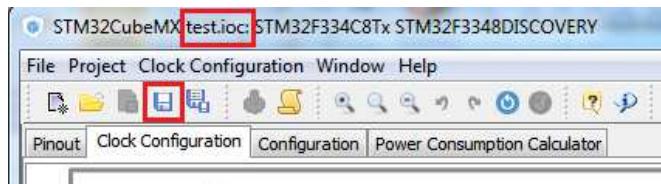
- Clock Default Configuration:
 - 16 MHz
- Modification not mandatory
 - Can be 36 MHz for example



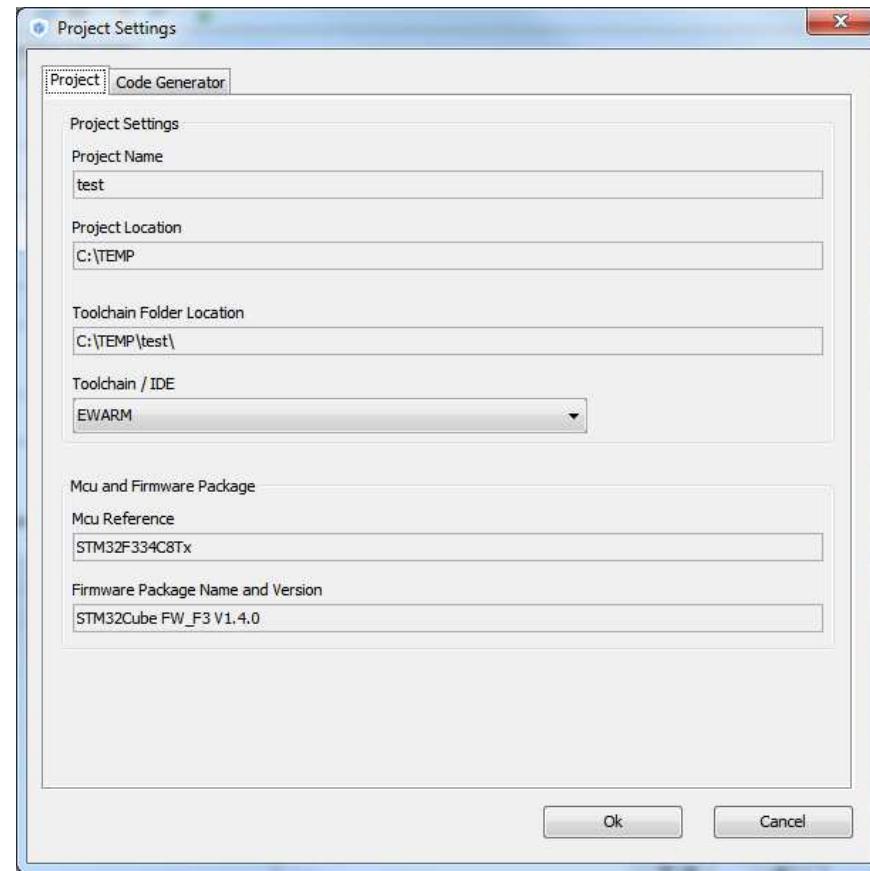


STM32CubeMX project Settings

- Project Name:
 - «test» for this example
- Project Location :
 - C:\TEMP for this example
- Save the current project
 - test.ioc file is available from c:\TEMP\test repository



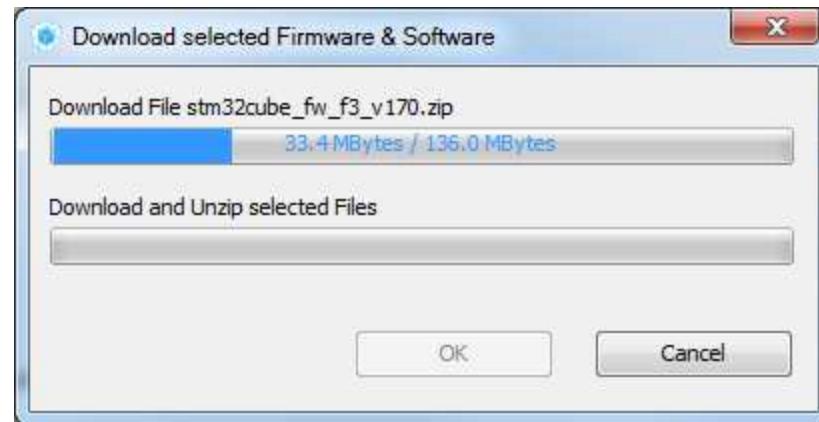
- You can save ioc file anywhere but
- It is preferable to save ioc file in same repository as diagram that will use it. Repository and ioc file must have same name.



STM32CubeMX project Settings



- Project Name:
 - «test» for this example
- Project Location :
 - C:\TEMP for this example
- Save the current project
 - test.ioc file is available from c:\TEMP\test repository



- Generated code is based on STM32 HAL C libraries.
- It is automatically downloaded/updated if it is necessary.

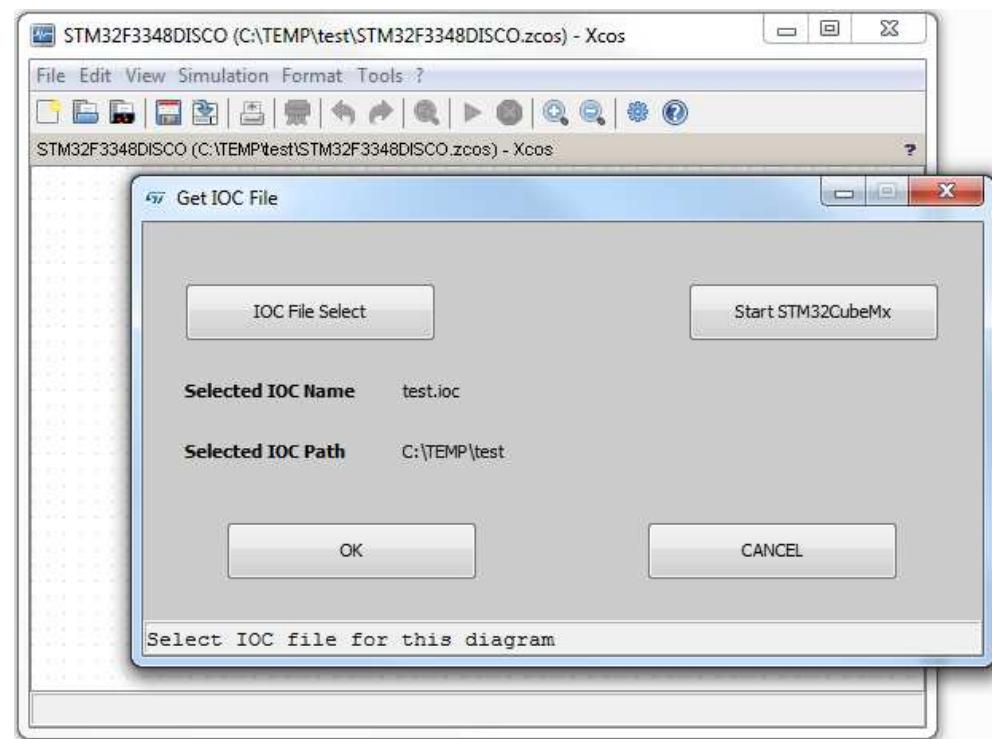
Xcos diagram creation



- Open new xcos diagram from Scilab 5.5.2.
- Save xcos diagram as C:\Temp\test\STM32F3348DISCO.zcos
- Link diagram to previously saved C:\Temp\test\test.ioc file



Tools > IOC File Select



USE TIM1 to Blink LED3 at 1Hz

33

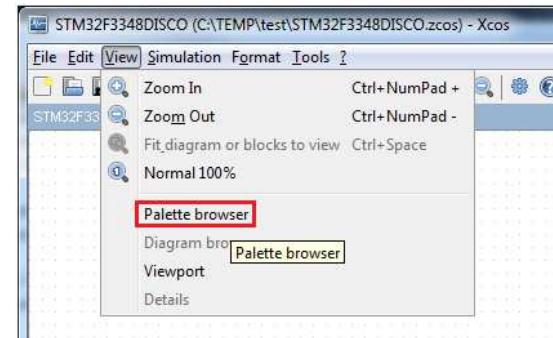


- Software application example:
 - **Use TIM1 to blink LED3 at 1Hz**
 - Use TIM6 to blink LED4 at 2Hz
 - Use TIM6 to trig ADC1 channels 2&3 conversion
 - Blink Led6 when user push button is pressed
 - Send ADC1 channel 3 values on USART2 when user push button is pressed



TIM1 Selection & Configuration

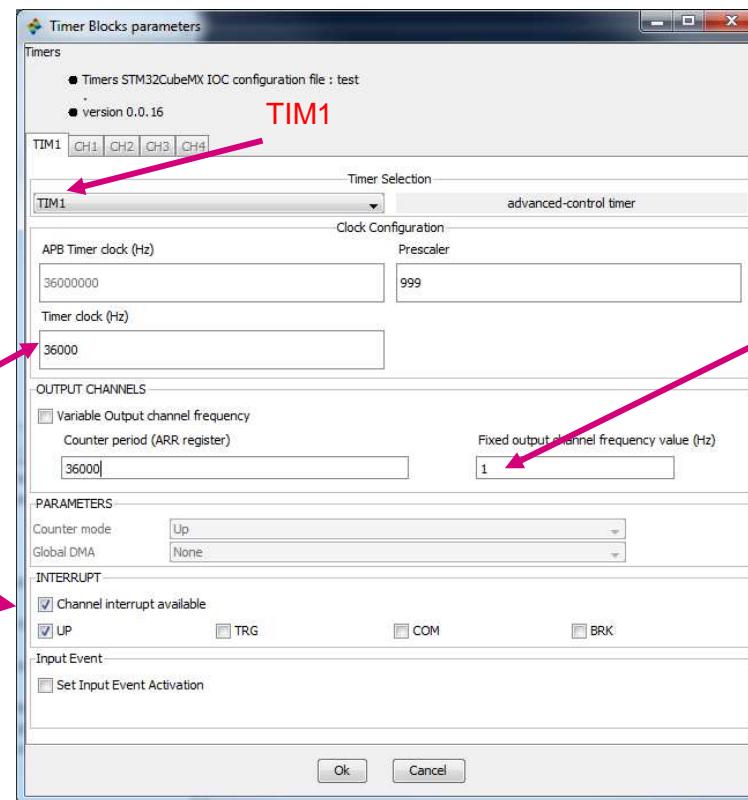
- Palette browser
 - Select View>Palette browser if it is not visible
- TIM1 Selection
 - Drag&Drop TIMER block from Palette browser
- TIM1 Configuration
 - Open (double click) TIMER block parameters window
 - Select TIM1 and set parameters.



- Prescaler must be set with constraint that ARR max value is 65535 regarding needed output channel frequency.

Validate Update
interrupt

Prescaler or
Timer Clock



1Hz for output
frequency

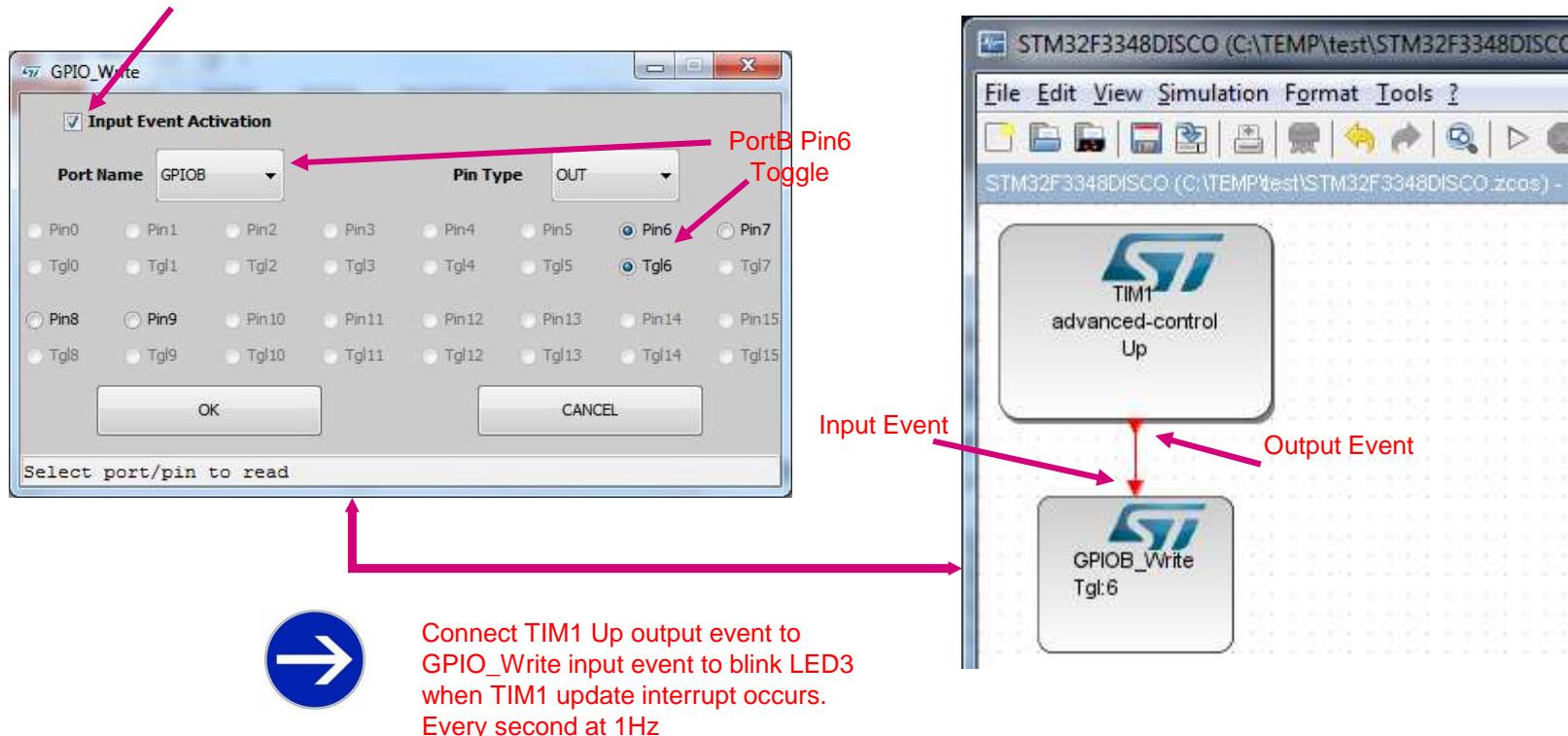
TIM1 Application

35



- TIM1 toggle LED3 at 1Hz
 - Drag&Drop GPIO_Write block to diagram and open (double click) GPIO_Write block parameters.

Check Input Event Activation.
GPIO_Write block will be activated by
the connected event



USE TIM6 to Blink LED4 at 2Hz

36



- Software application example:
 - Use TIM1 to blink LED3 at 1Hz
 - **Use TIM6 to blink LED4 at 2Hz**
 - Use TIM6 to trig ADC1 channels 2&3 conversion
 - Blink Led6 when user push button is pressed
 - Send ADC1 channel 3 values on USART2 when user push button is pressed

TIM6 Application

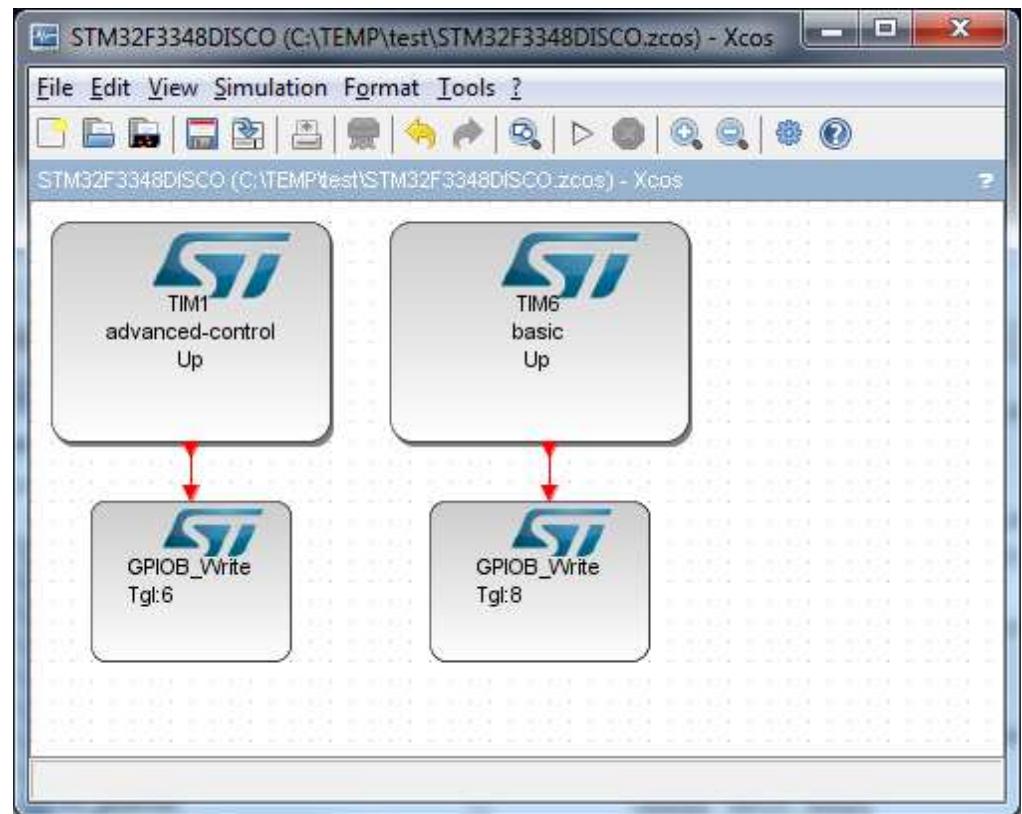
37



- TIM6 toggle LED4 at 2Hz
 - Make the same thing as for TIM1 but frequency is 2Hz and PortB Pin8 toggle as it is connected to Led4



Connect TIM6 Up output event to
GPIO_Write input event to blink LED4
when TIM6 update interrupt occurs.
Every 0.5 second at 2Hz



USE TIM6 to trig ADC1 channels 2&3



- Software application example:
 - Use TIM1 to blink LED3 at 1Hz
 - Use TIM6 to blink LED4 at 2Hz
 - **Use TIM6 to trig ADC1 channels 2&3 conversion**
 - Blink Led6 when user push button is pressed
 - Send ADC1 channel 3 values on USART2 when user push button is pressed

ADC1 Selection & Configuration

39



- TIM6 is configured to trig ADC1 from STM32CubeMx

ADC1 Selection

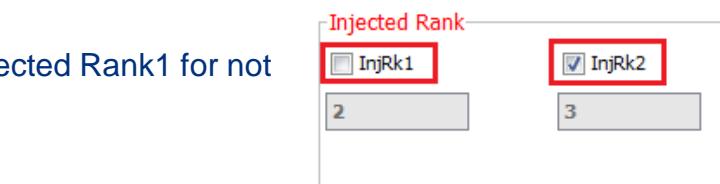
- Drag&Drop ADC block from Palette Browser and open ADC block parameters

ADC1 Configuration

- We will need ADC Ch3 value, uncheck Injected Rank1 for not needed Ch2

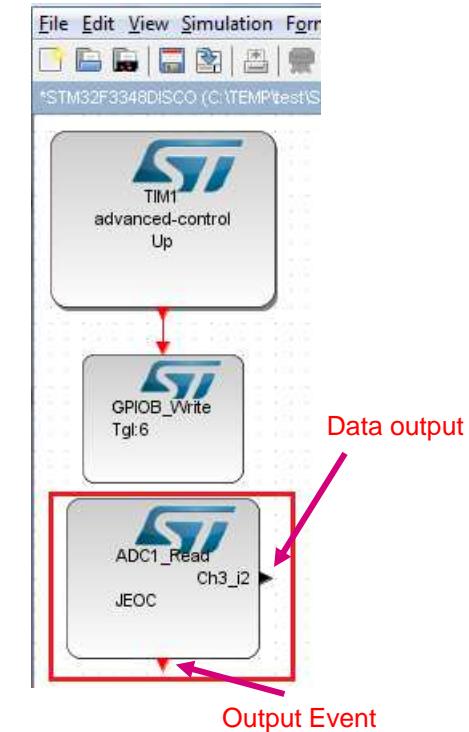
- Select JEOC/S as interrupt output trigger

Injected end of conversion trigger



ADC1_Read End of Injected Conversion (JEOC) event is available to trig process

Ch3 value is available as output



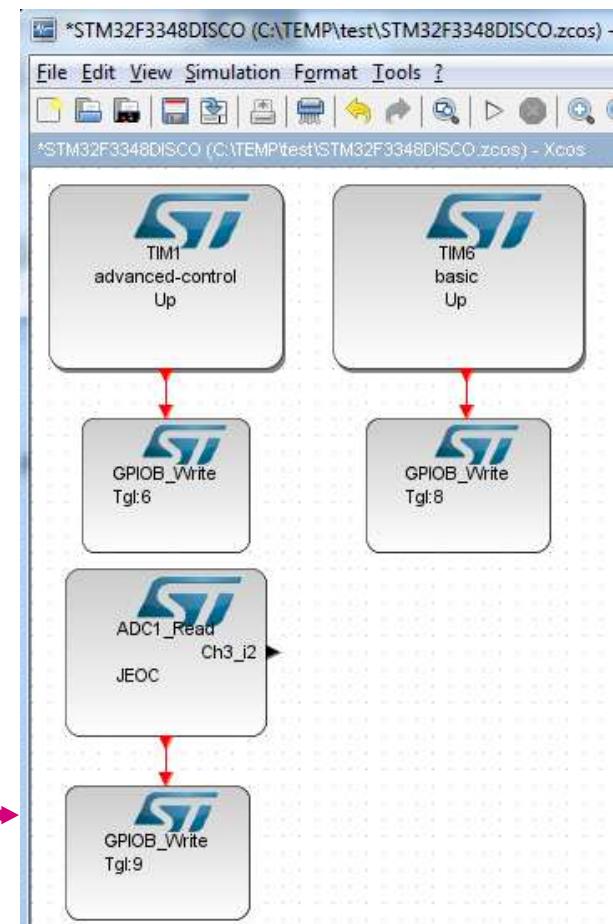
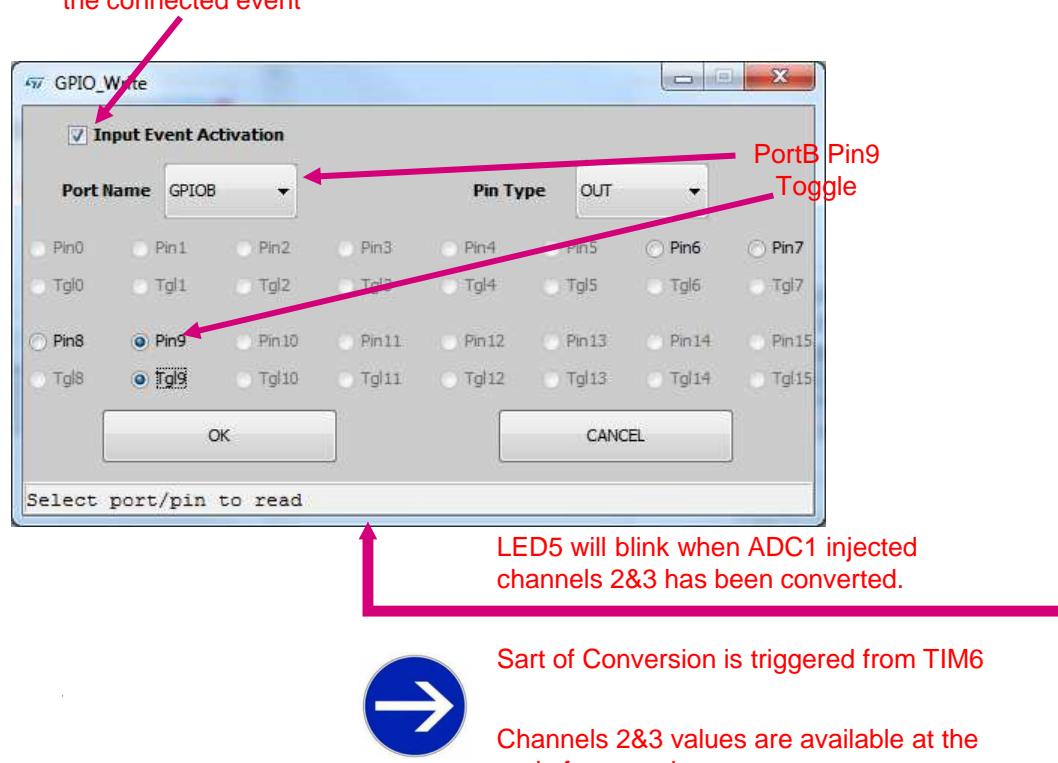
ADC1 Application

40



- TIM6 trig ADC1 channels conversion
 - Blink LED5 at end of ADC1 conversion to verify that TIM6 has triggerer it.
 - Drag&Drop GPIO_Write block.
 - Set GPIO_Write block parameters window to toggle Pin9 (LED5 is connected to Pin9)

Check Input Event Activation.
GPIO_Write block will be activated by
the connected event



Push Button functions

41



- Software application example:
 - Use TIM1 to blink LED3 at 1Hz
 - Use TIM6 to blink LED4 at 2Hz
 - Use TIM6 to trig ADC1 channels 2&3 conversion
 - **Blink Led6 when user push button is pressed**
 - **Send ADC1 channel 3 values on USART2 when user push button is pressed**

EXTI Selection & Configuration

42



- EXTI0 Selection

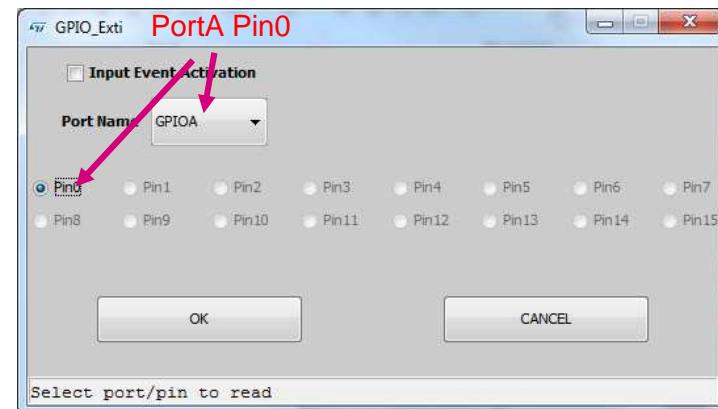
- Push Button is connected to External Interrupt 0 on PA0
- External interrupt Event will be generated for action on Push Button.
- Drag&Drop GPIO_Exti block from palette Browser
- Open (double click) GPIO_Exti block parameters window and select GPIOA pin0



External Event on PA0 available
on External Interrupt block.



External Interrupt Event
Trig LED6 blink &
USART_Send

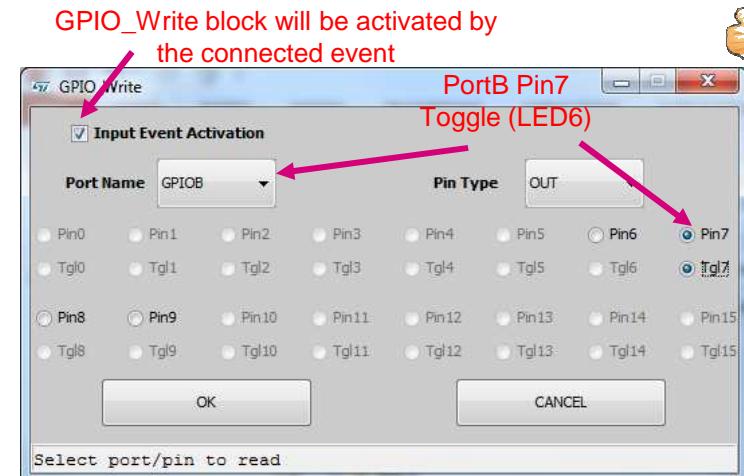


Push Button Action 1/2



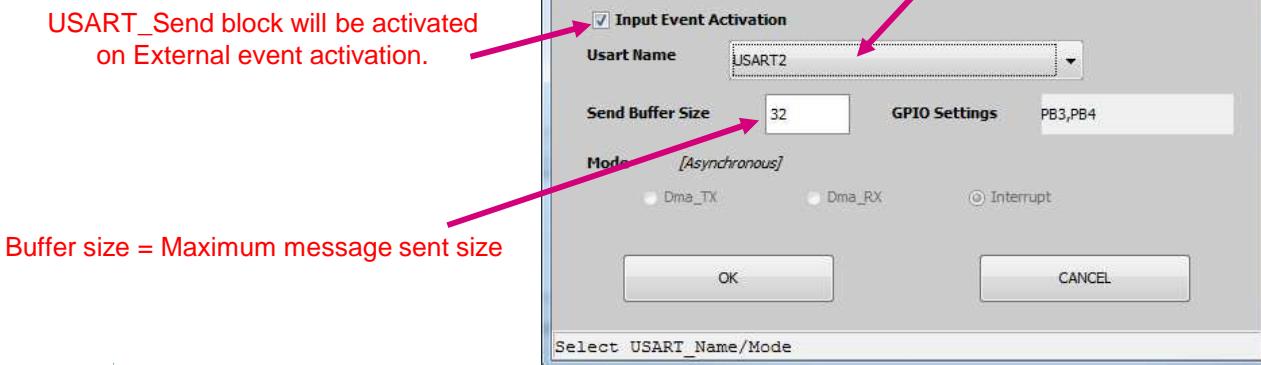
- Blink LED6

- Drag&Drop GPIO_Write block from palette browser to diagram.
- Open (double click) GPIO_Write block parameters window and select Pin7 (LED6 is connected to Pin7)
- Connect GPIOA_Exti0 output event to GPIOB_Write Tgl:7 input event. PB7 will blink every button pushed.



- USART2 Settings

- Drag&Drop USART_Send block from Palette Browser and open parameters window. Connect GPIOA_Exti0 output event to USART2 input event.

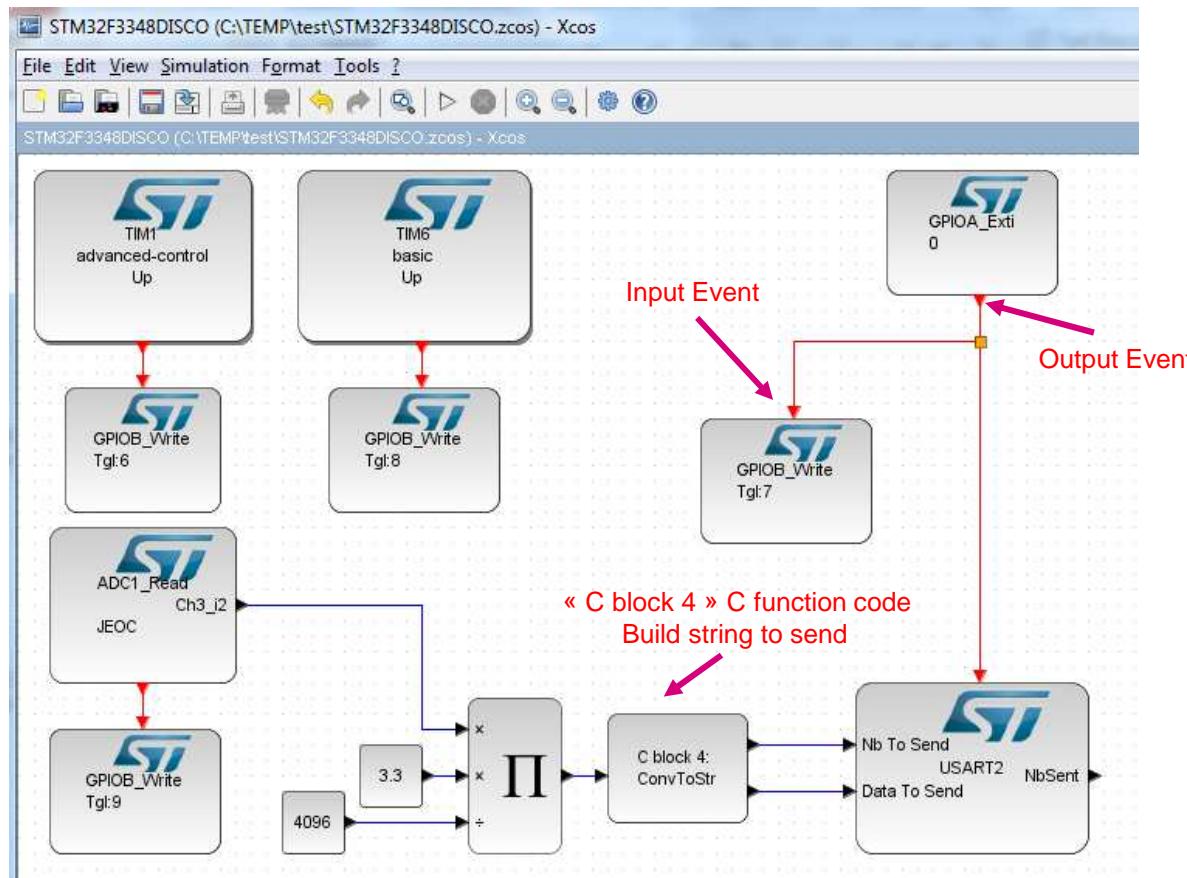


It is mandatory to set Buffer Size as close as messages sent in order to avoid memory waste.

Push Button Action 2/2



- String to send on USART2
 - Build string with ADC1 Ch3 value
 - ADC1 Ch3 output is 12bits value. It should be converted to 0-3.3V value. Connect converted value to « C block 4 » to build string to send.
 - Connect ConvToStr block outputs to USART2 Send block.



« C block 4 » configuration 1/2



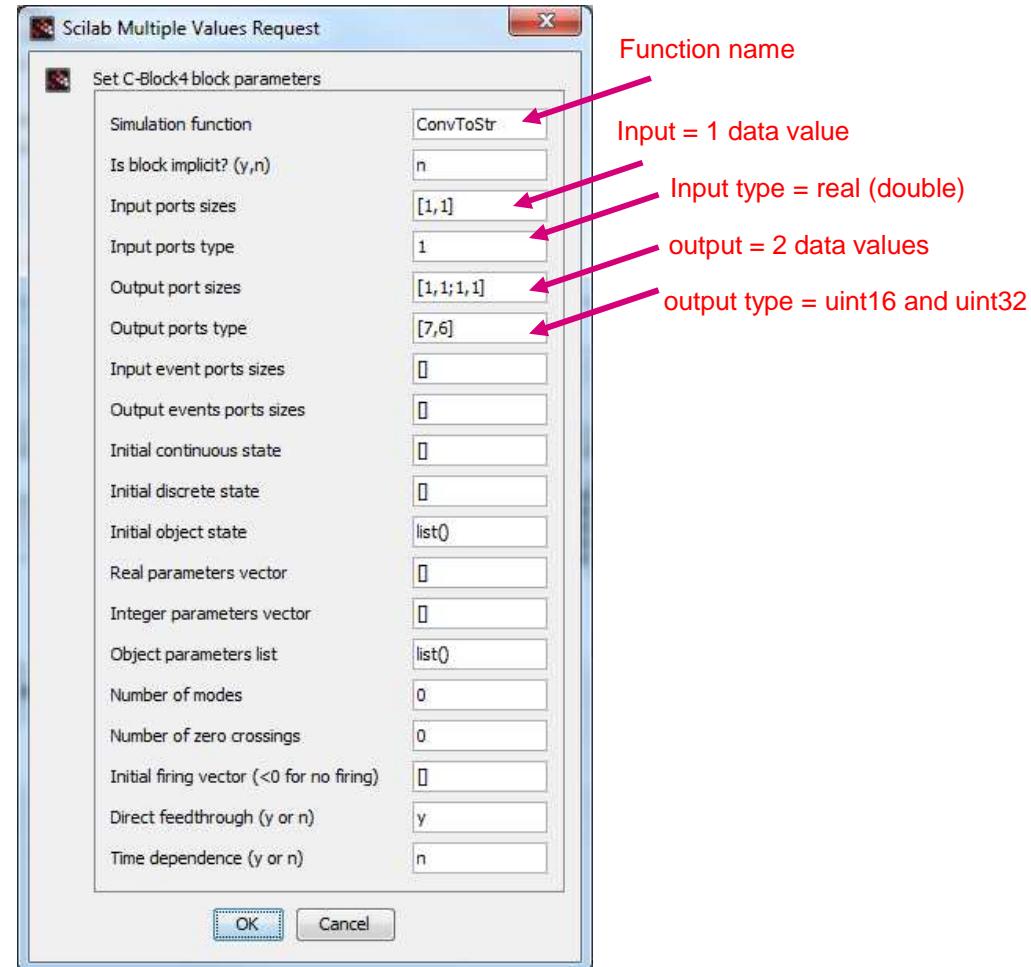
- C function parameter description

- Drag&Drop « C block 4 ».
- It is used to include C code to SCILAB.
- Set « C block 4 » parameters.
- Click OK



Data types:

- 1 real
- 2 complex
- 3 int32
- 4 int16
- 5 int8
- 6 uint32
- 7 uint16
- 8 uint8



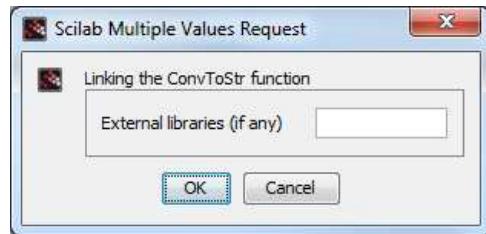
« C block 4 » configuration 2/2

46



- C function write

- Write C code application processing input(s) to generate output(s)
- Click OK.
- « External libraries » not processed for the moment.



Flag input value:

- -5 Error
- 0 Derivative state
- 1 Output state
- 2 State update
- 3 Output Event Timing
- 4 initialization
- 5 Ending
- 6 Reinitialization
- 7 Continuous property Update
- 9 Zero Crossing
- 10 Jacobian

outputs →

input →

```
#include "scicos_block4.h"

void ConvToStr(scicos_block *block,int flag)
{
    static char str[15];
    unsigned short* strSize;
    char** str_pt;
    /* init */
    strSize = block->outptr[0];
    str_pt = block->outptr[1];
    *str_pt = str;
    if (flag == 4) { Initialization
        sprintf(str,"ADC_VAL=%2.2f",*(double*)block->inptr[0]);
        *strSize = strlen(str);
        /* output computation */
    } else if(flag == 1) { Output update
        sprintf(str,"ADC_VAL=%2.2f",*(double*)block->inptr[0]);
        *strSize = strlen(str);
        /* ending */
    }
}
```

OK Cancel

Build Application flow

47



- Build Xcos Application Process

- Build application has been divided in 2 steps:
 - Generate Code
 - Generate Project



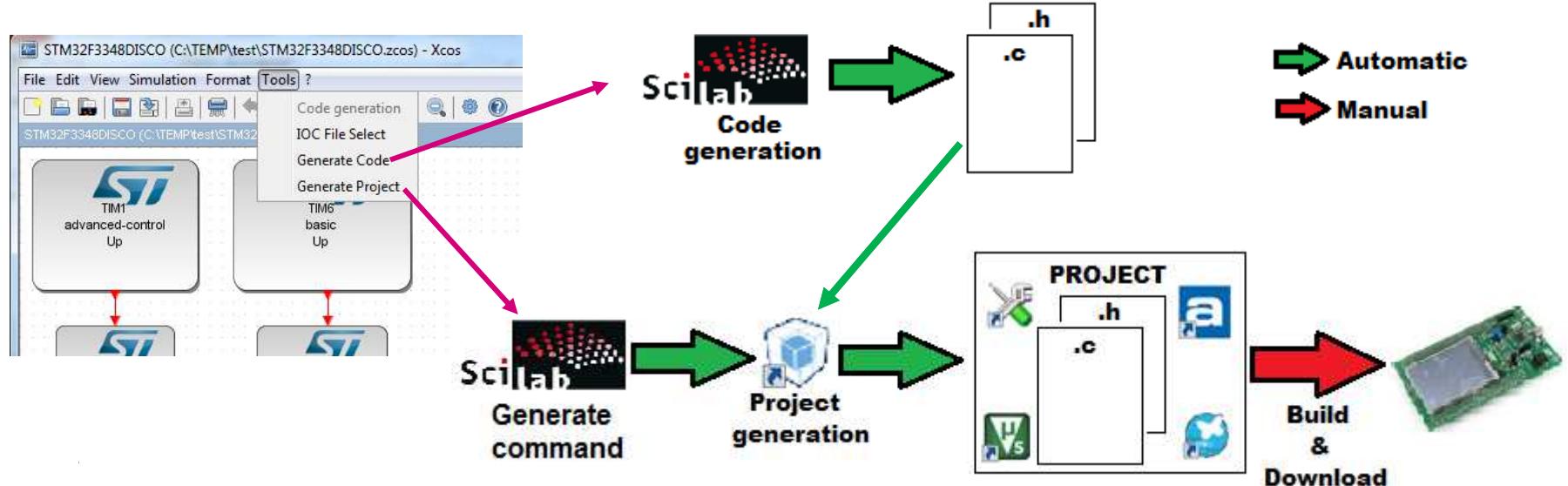
IOC File Selection is mandatory.
IOC file must have been selected before
generating code / generating project

- Generate Code

- Press « Generate Code » to automatically call `stm32BuildProject` function that will generate C code for the diagram.

- Generate Project

- Press « Generate Project » to automatically call `stm32GenerateProject` function.
- It is used to retrieve all C generated files and generate STM32CubeMx command.
- Project is generated from STM32CubeMx depending on project settings parameters.



Generate Code

48



- **Code generation flow**

- Entry point: `stm32BuildProject` function.
- The process gives a « flat image » of the diagram, which is a data base of all blocks connections, inputs, outputs, events etc...
- Then, generated code is Synchronous or Asynchronous.
- Asynchronous code manages interrupts where Synchronous code is called every regular steps.
- Generated code depends on block input event connection.
 1. When block input event is connected to an STM32 output event. Generated code is asynchronous (interrupts)
 2. When block input event is connected to a Palettes>Event handling block. Generated code is synchronous.
 3. When block do not have input event:
 - If data flow (input/output connection) including block is connected to a block with input event, cf 1 or 2.
 - If data flow do not include any block with input event, Generated code is synchronous. Minus synchronous step time used or default Tick Handler value. (cf help `STM32_Preferences`)

- **Synchronous code**

- The process gets all activation clocks and computes STM32 systick value (PGCD or default Tick Handler when diagram do not have any clock event)
- A stepi ($i = 1 \dots n$) function is generated for every activation clock.

- **Asynchronous code**

- Only STM32 block generates asynchronous code. Code of blocks connected to an STM32 output event is generated in STM32 peripheral HAL interrupt Callback function.
- Then, code will be called every time STM32 peripheral event occurs.

Generate Project

49



• Project generation flow

- When « Generate Code » is called, .c/.h files are created as well as main.c.
- Then, STM32CubeMX is called to integrate peripherals initialization to main.c and to generate project including all .c/.h files.
- Interface files with STM32CubeMX are .mlproject and .script files.



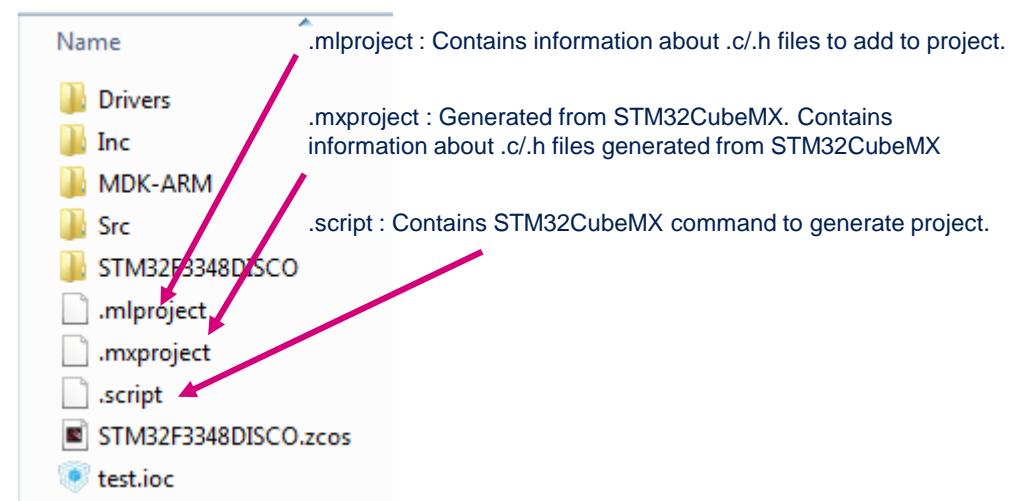
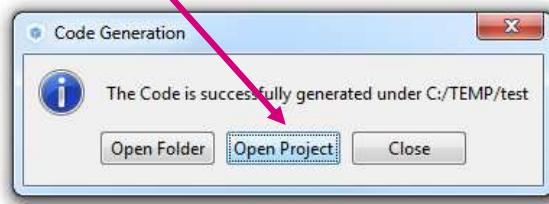
« Generate Project » is mandatory every time
« Generate Code » has been called.

• Generate Project

- Press « Generate Project » to automatically call STM32CubeMX.
- Press « Open Project » to start toolchain.
- Project can be built, downloaded into STM32 target and run or debug.



Click « Open Project » to automatically open project using selected toolchain.

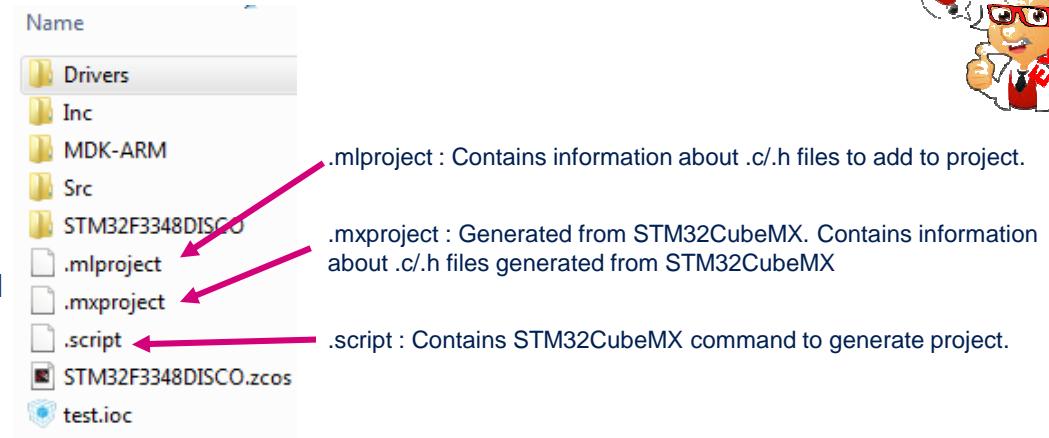


STM32CubeMX Code Generation

50



- STM32CubeMX project generation
 - Project is generated in same repository as ioc file. (cf « test » project example)
- STM32CubeMX project contains
 - Drivers : Contains STM32 selected library and CMSIS files



- EWARM/MDK-ARM: Contains toolchain project files (For example)



- Inc & Src: Contains STM32CubeMX generated or modified files.



- test (name: Contains all .c/h files generated from MATLAB®

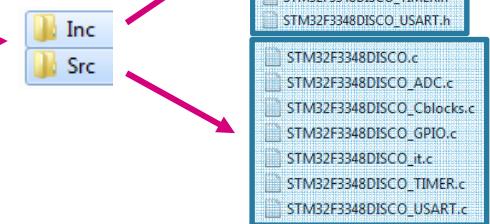
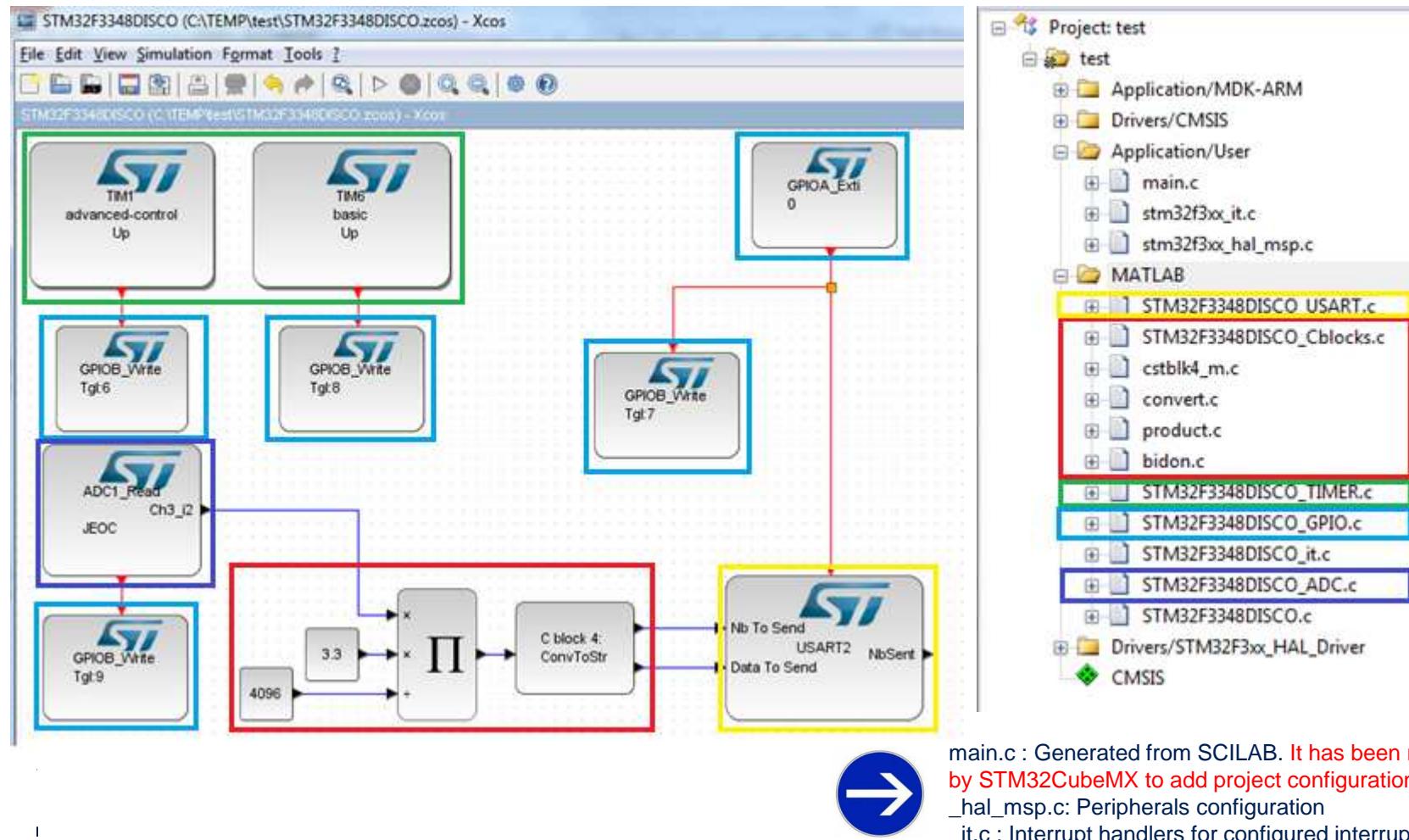




Diagram vs Generated Code

- Every STM32 peripherals generate initialization code in .c/.h files which name is created using name of the diagram and peripheral name. Asynchronous code (interrupts) is generated in "DiagramName"_it.c file and Synchronous code is generated in "DiagramName".c file.

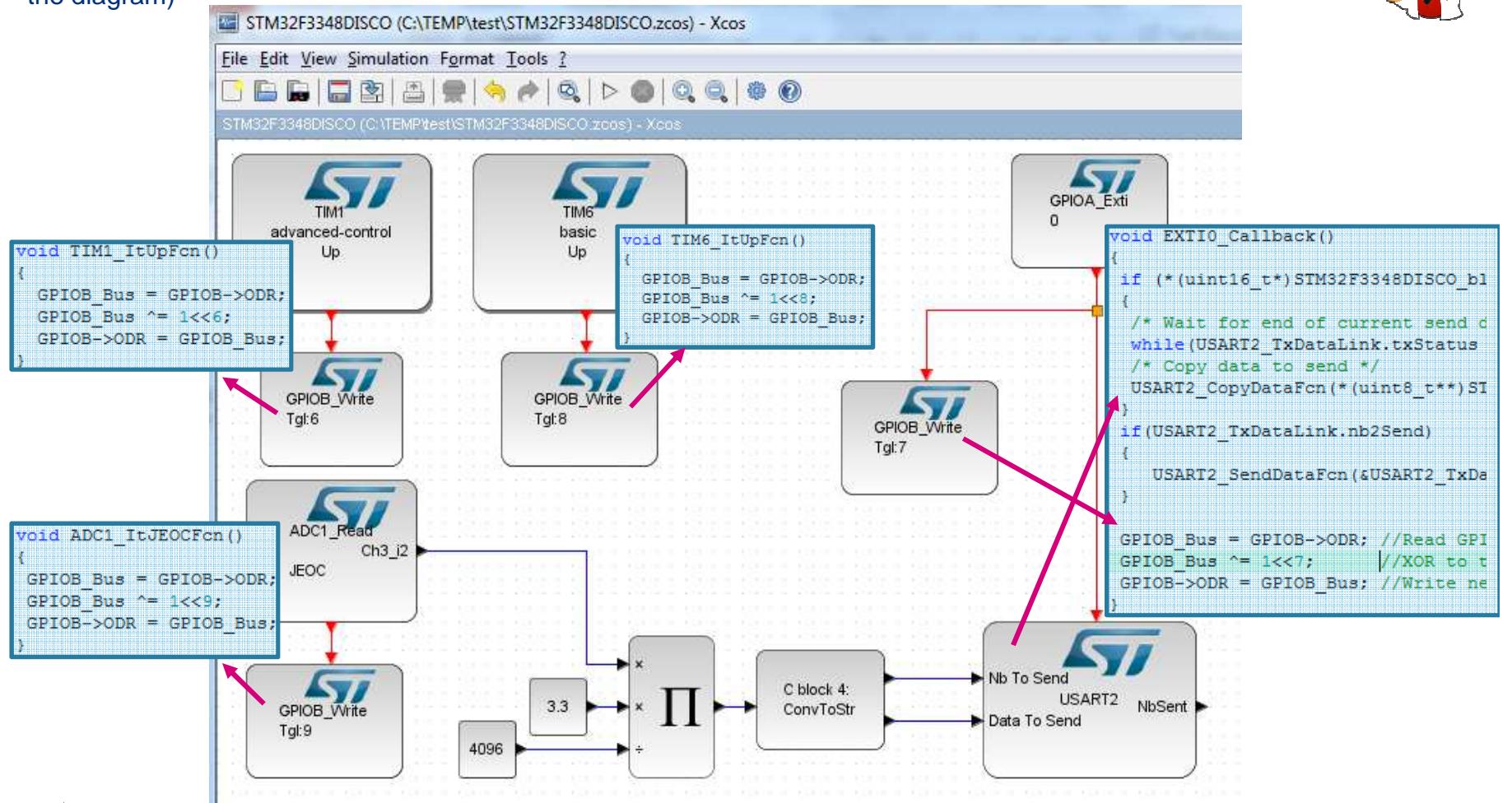


Asynchronous Generated Code

52



- Asynchronous code is generated in xx_it.c file from blocks connected to STM32 event. (XX is name of the diagram)

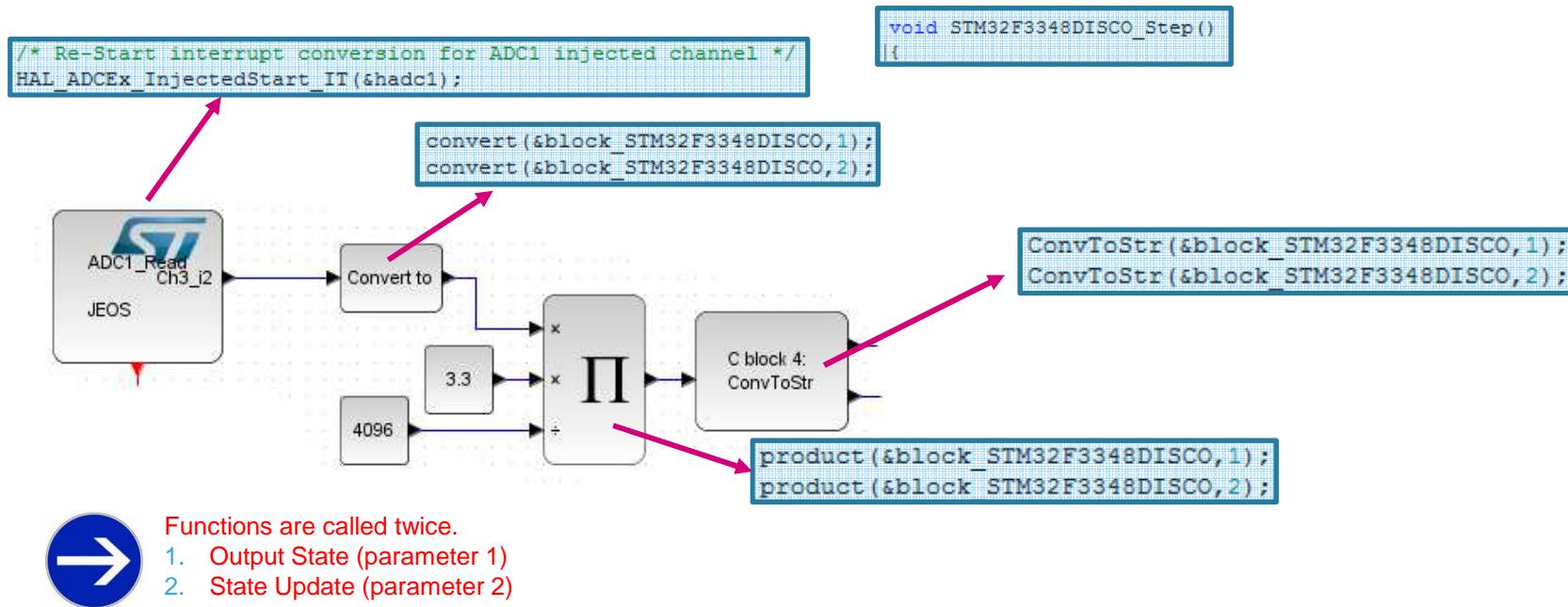


Synchronous Generated Code

53



- Synchronous code is generated in "DiagramName".c file from blocks connected to « Event Handling » blocks. Each Event is associated to a Step function including all code it manages.



- Step functions are scheduled from main depending on « Event Handling » step parameter time or default tick handler step when there is no « Event Handling » in the diagram.

```

/* Systick configuration and enable SysTickHandler interrupt */
if (SysTick_Config((uint32_t)(SystemCoreClock * 0.001)))

```



Step function is scheduled at default 1ms tick handler value. (cf preferences)

```

/* Infinite loop */
/* Real time from systickHandler */
while(1) {
    /*Process tasks every solver time*/
    if(remainAutoReloadTimerLoopVal_S == 0) {
        remainAutoReloadTimerLoopVal_S = autoReloadTimerLoopVal_S;
        /* Step the model for base rate */
        STM32F3348DISCO_Step();
    } //End if
} //End while

```



It is required to know toolchain functionalities.
Keil µVision used here as example

Toolchain Project

54



- Toolchain settings

- STM32CubeMX has automatically generated project including mandatory settings. It is exactly same project at it should be generated « by hand ». Possibility to tune all settings.

- Toolchain Actions

- Build project
- Start Debug Session (Ctrl+F5)



STM32 board must be connected to PC
when you click « Start Debug Session »

BUILD
project

```
1  */
2  ****
3  * File Name      : main.c
4  * Description    : Main program body
5  ****
6  *
7  * COPYRIGHT(c) 2017 STMicroelectronics
8  *
9  * Redistribution and use in source and binary forms, with or without modification,
10 * are permitted provided that the following conditions are met:
11 *   1. Redistributions of source code must retain the above copyright notice,
12 *      this list of conditions and the following disclaimer.
13 *   2. Redistributions in binary form must reproduce the above copyright notice,
14 *      this list of conditions and the following disclaimer in the documentation
15 *      and/or other materials provided with the distribution.
16 *   3. Neither the name of STMicroelectronics nor the names of its contributors
17 *      may be used to endorse or promote products derived from this software
18 *      without specific prior written permission.
19 *
20 * THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
21 * AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
22 * IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
23 * DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE
24 * FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
25 * DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
26 * SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
27 * CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
28 * OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
29 * OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
30 *
31 ****
32 */
33 /* Includes -----
34 #include "main.h"
35 #include "stm32f3xx_hal.h"
36
37 /* USER CODE BEGIN Includes */
```



It is required to know toolchain functionalities.

Run Project

55

- STM32F3348DISCO example results

- Project is started and waits at 1st main instruction.
- Click «Go»
- LD3/LD4/LD5 are blinking
- LD6 is alternatively ON and OFF when you press User button. ADC value set on PA2 (ADC1 Ch3) is sent to PC through USART.

You can see ADC value on PC using PuTTY for example.

```
0.60Volt  
1.02Volt  
2.28Volt  
3.30Volt  
0.00Volt
```



Example using µVision (KEIL) toolchain



The screenshot displays the µVision IDE interface. The left pane shows the project structure for 'test' with files like Application/MDK-ARM, Drivers/CMSIS, and MATLAB. The right pane shows the code editor with the 'main.c' file open. The code includes initialization for peripherals such as GPIO, DMA, ADC, TIM, and USART. A call stack and locals table are visible at the bottom.

```
#define NB_SOLVER_LOOP_FOR_STEP1 1 //Nb of solver loop to do before triggering Step function
static uint32_t autoReloadTimerLoopStep1_S = NB_SOLVER_LOOP_FOR_STEP1; //Step function called from main
/* USER CODE END 0 */

int main(void)
{
    /* USER CODE BEGIN 1 */
    /* Data initialization */
    //int_T i;
    /* USER CODE END 1 */

    /* MCU Configuration-----*/
    /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
    HAL_Init();

    /* Configure the system clock */
    SystemClock_Config();

    /* Initialize all configured peripherals */
    MX_GPIO_Init();
    MX_DMA_Init();
    MX_ADC1_Init();
    MX_TIM1_Init();
    MX_TIM6_Init();
    MX_USART2_UART_Init();
    MX_TIM7_Init();
}

/* USER CODE BEGIN 2 */
/* USER CODE END 2 */
```

Diagram and Code variation 1/3

56



- One step function scheduled from « Event Handling »

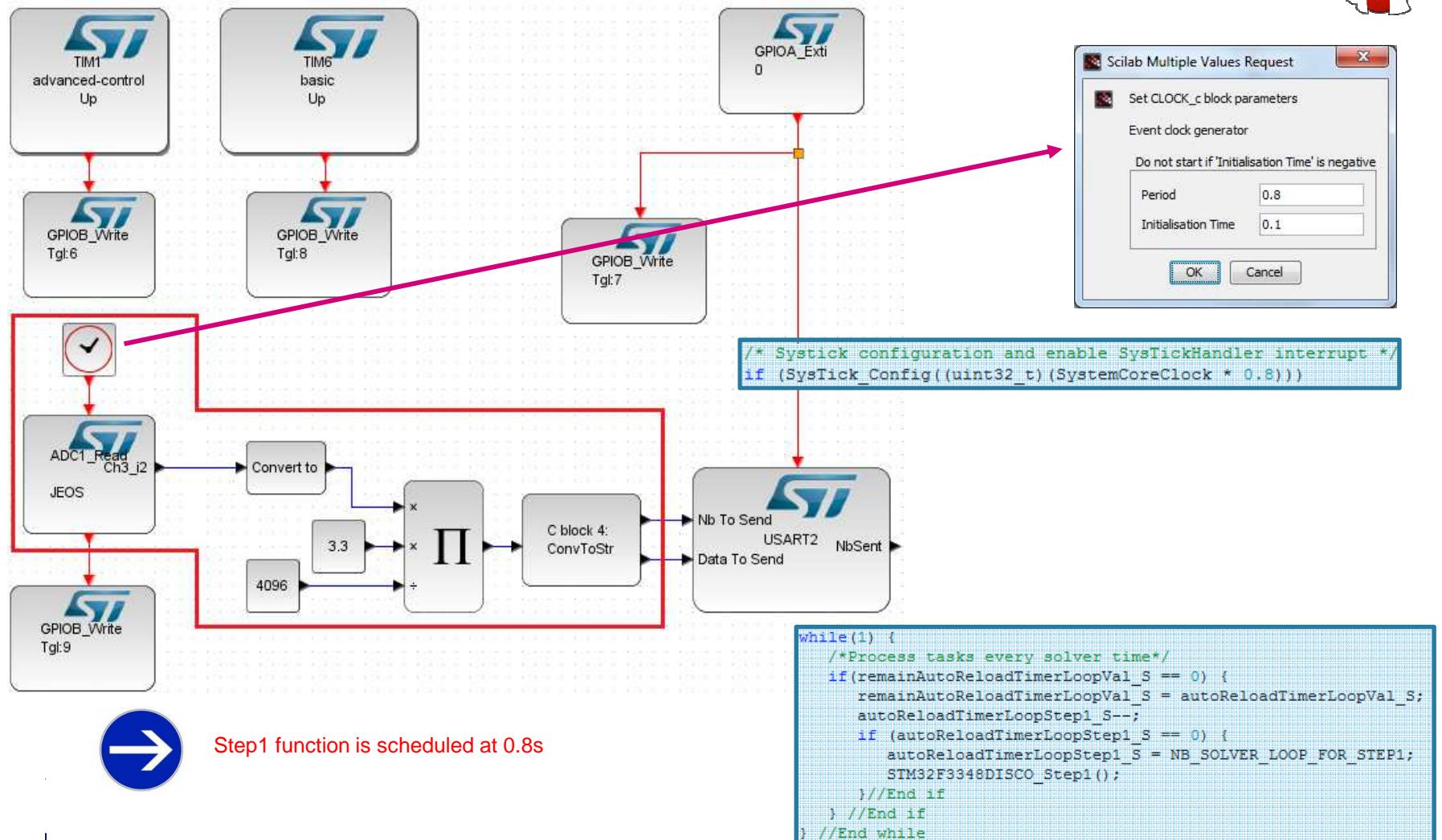


Diagram and Code variation 2/3

57



- Two steps functions scheduled from « Event Handling »

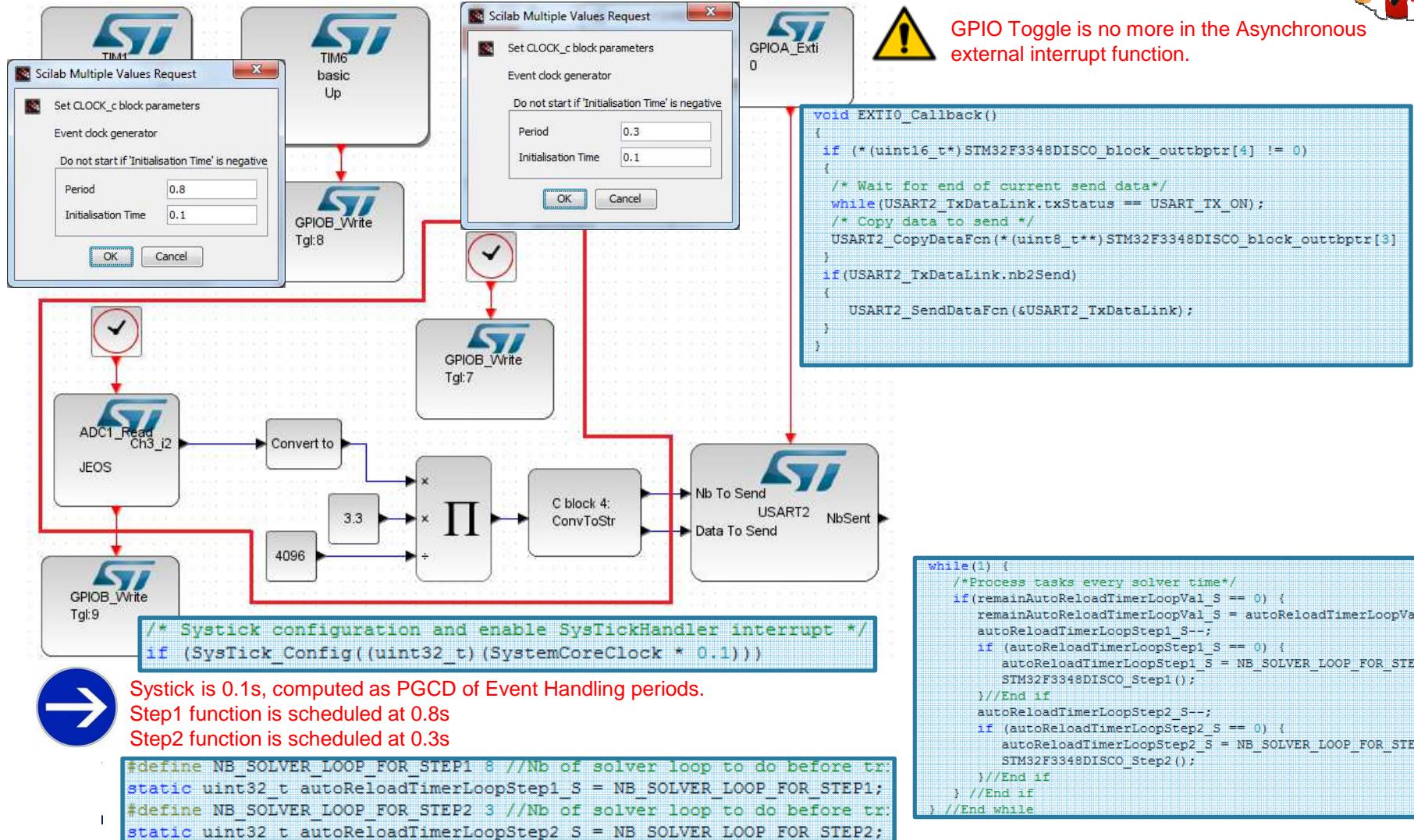
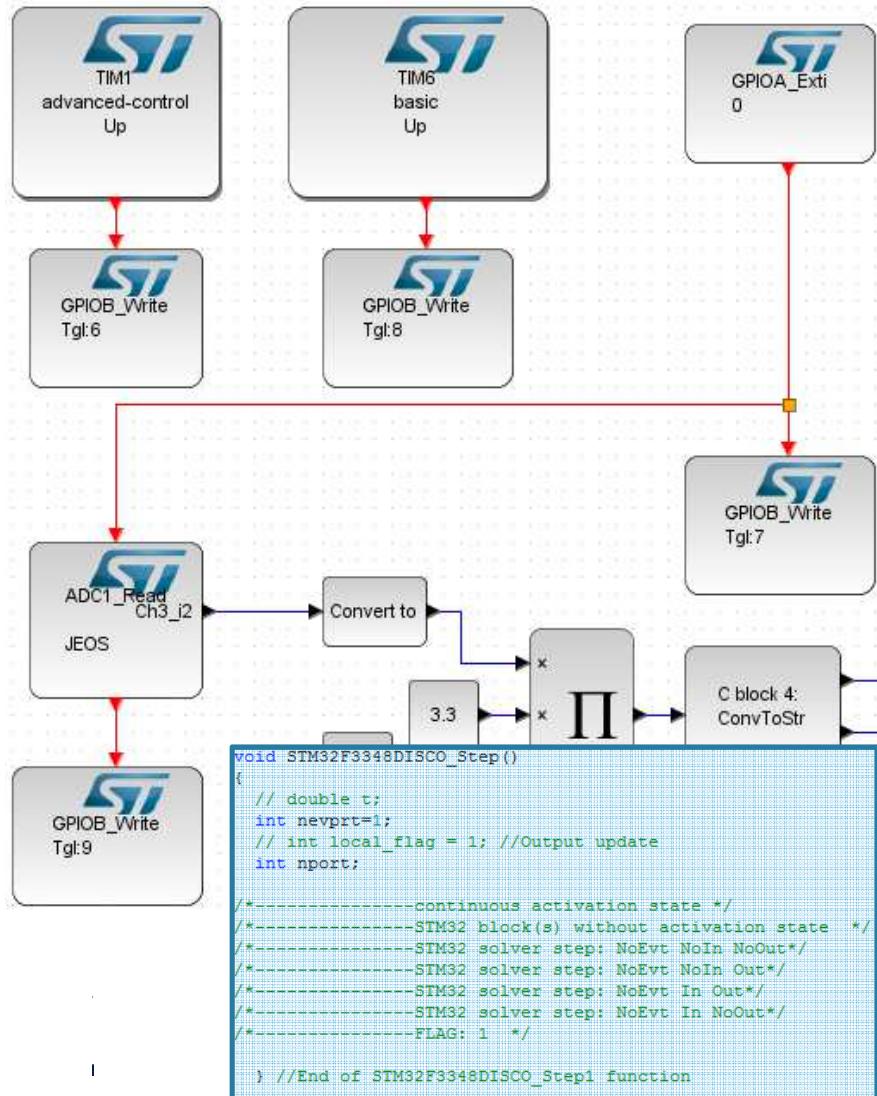


Diagram and Code variation 3/3

58



- No Synchronous function



For this configuration:

- Step function is scheduled at default 1ms tick handler value. (cf preferences)
- BUT Step function is empty!
- All generated code is Asynchrone

```

void EXTI0_Callback()
{
    GPIOB_Bus = GPIOB->ODR; //Read GPIO
    GPIOB_Bus ^= 1<<7; //XOR to toggle !
    GPIOB->ODR = GPIOB_Bus; //Write new

    /* Re-Start interrupt conversion for ADC1 */
    HAL_ADCEx_InjectedStart_IT(&hadc1);

    convert(&block_STM32F3348DISCO,1);
    convert(&block_STM32F3348DISCO,2);

    product(&block_STM32F3348DISCO,1);
    product(&block_STM32F3348DISCO,2);

    ConvToStr(&block_STM32F3348DISCO,1);
    ConvToStr(&block_STM32F3348DISCO,2);

    if (*(uint16_t*)STM32F3348DISCO_block_outtbptr[4] != 0)
    {
        /* Wait for end of current send data*/
        while(USART2_TxDataLink.txStatus == USART_TX_ON);
        /* Copy data to send */
        USART2_CopyDataFcn((uint8_t**)STM32F3348DISCO_block_outtbpt);
    }
    if(USART2_TxDataLink.nb2Send)
    {
        USART2_SendDataFcn(&USART2_TxDataLink);
    }
}

```

END 59

* *

*