EMBEDDED SYSTEMS

BASED ON CORTEX-M4 AND THE RENESAS SYNERGY PLATFORM

2020
PROF. DOUGLAS RENAUX, PHD
PROF. ROBSON LINHARES, DR.
UTFPR / ESYSTECH

RENESAS ELECTRONICS CORPORATION

LAB2 – SAMPLE C PROGRAM

Objectives:

- In Lab 2, the student will develop a simple game. The game’s objective is simply to respond as fast as possible to a visual stimulus. A led goes on after a random time and the player has to press a button. The current response time is presented.
- The main objective of this Lab is to guide the student through a proposed development process that should be used in the following labs.

LAB2 – SAMPLE C PROGRAM

Learning Objectives:

- Project planning
- Problem definition, specification, hardware platform study, software framework study, design
- Project generation
- SSP Configuration
- Threads configuration
- Source code editing
- Compile/Link
- Debug

LAB2 – SAMPLE C PROGRAM

Activities:

1. Plan the phases of the development process
2. Problem definition
3. Specification
4. Hardware platform study
5. Software framework study
6. Design
7. Use the Wizard to generate a Synergy C Project
8. Configure the SSP
9. Use the Editor
10. Compile and link the project
11. Debug on the SK-S7G2 board

further reading for this Lab:
e^2 studio Integrated Development Environment User's Manual: Getting Started Guide
r20ut2771ej0400_e2_start_s.pdf
https://www.renesas.com/us/en/doc/products/tool/doc/006/r20ut2771ej0400_e2_start_s.pdf
(it is also the source of some figures of this section)

LAB2 – ACTIVITY 1

Activity 1 – Plan the phases of the development process

In Lab2, the student is faced with a simple problem to be implemented in C, using many of the functionalities made available in e2studio. The objectives in this lab are not limited to understanding the functionality of the tool, but also to follow a planned approach to a software development activity.

1. Problem definition: the problem statement should clearly define the scope of the problem.
2. Specification: a clear and precise set of statements that define the functionality of the resulting software as well as its non-functional characteristics (performance, ...)
3. Study of the hardware platform: one must have a clear understanding of the features of the MCU and of the board that will be used in this development
4. Software framework study: some of the functionality needed for the solution is available in the form of software components in the SSP, these must be identified and understood
5. Design: identification of the SW components that need to be developed, their interfaces, algorithms and interfaces to other software components.
6. Tool usage: project generation, SSP configuration, source code editing, compilation, linkage, debug

LAB2 – ACTIVITY 2

Activity 2 – Problem Definition

Game objective: to respond as fast as possible to a visual stimulus

Game operation: once the game is turned on, after 1 second an LED is turned on, the player must respond by pressing a button.
Game cycle repeats indefinitely.
Game presents the player response time.

Basic solution: response times are saved in variables and can be visualized in the debugger

LAB2 – ACTIVITY 3

Activity 3 – Specification

S1 - The ResponseTimeGame console shall provide an LED, a push-button and an LCD.

S2 - The operation of the ResponseTimeGame, after power up, is:

1. wait for 1 second
2. turn on LED
3. start measuring time
4. wait for player to press the push-button
5. stop measuring time, save measure as current response time
6. turn off LED
7. a variable should hold the current response time
8. go to step 1

S3 - discard response time measurements above 3 seconds

LAB2 – ACTIVITY 4

Activity 4 – Study the Hardware platform (SK-S7G2 board and S7G2 MCU)

The first manual to be studied is the Starter Kit SK-S7G2 User's Manual.pdf

Relevant information in this manual concerning the current project is:

- Block diagram
- LEDs
- push-button
- LCD

LAB2 – ACTIVITY 4

Relevant blocks of the SK-S7G2 board are marked in red

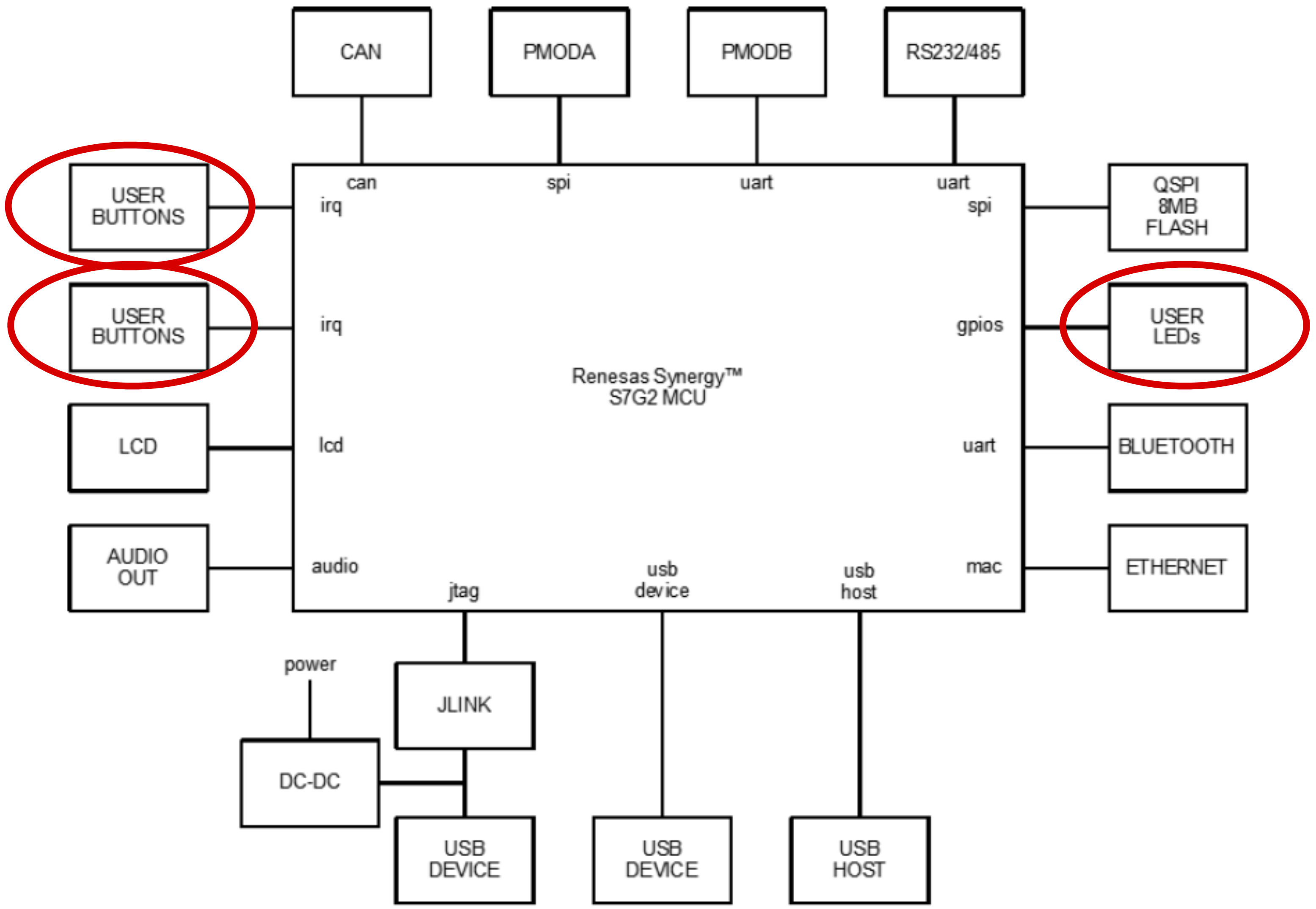


Figure 2: SK-S7G2 block diagram

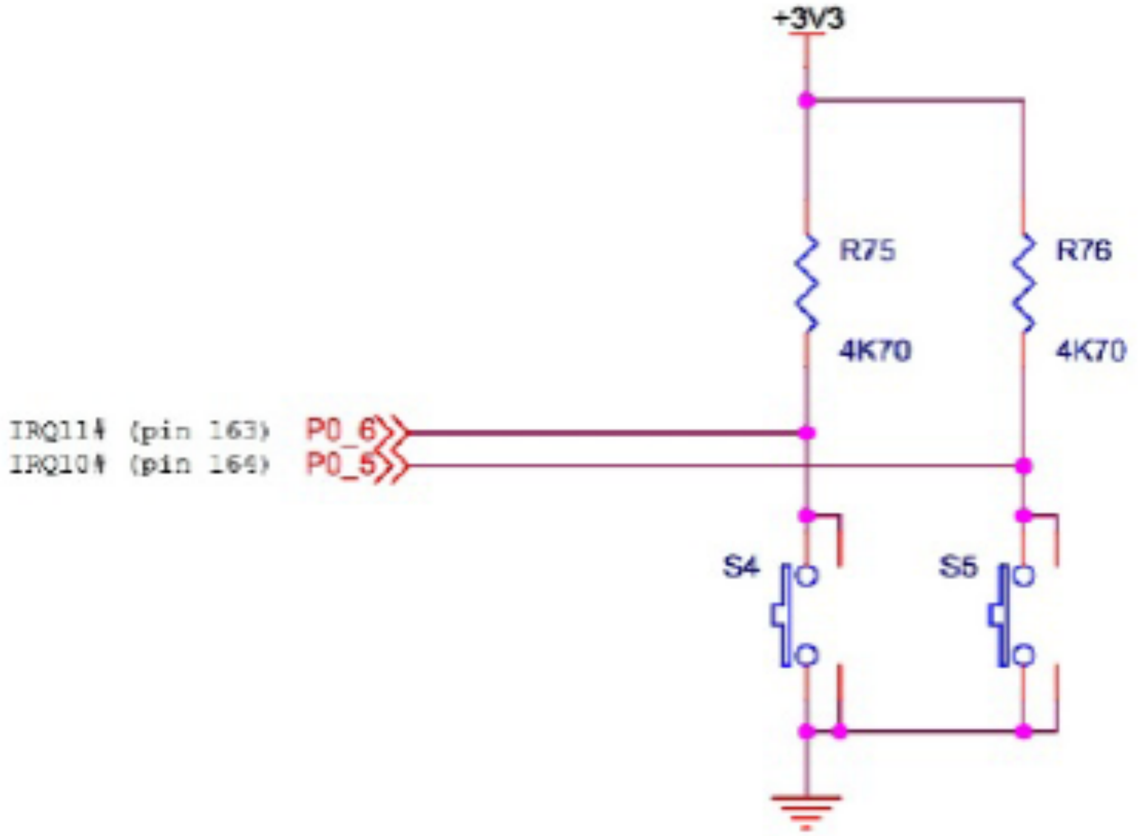
source: Starter Kit SK-S7G2 User's Manual

LAB2 – ACTIVITY 4

There are two push-buttons on the SK-S7G2: S4 and S5.

They are directly connected to GPIO pins: P0.6 and P0.5 respectively. There is no debounce circuit between the push-buttons and the MCU pins.

P0.6 (P006) can generate interrupts at IRQ11.
P0.5 (P005) can generate interrupts at IRQ10



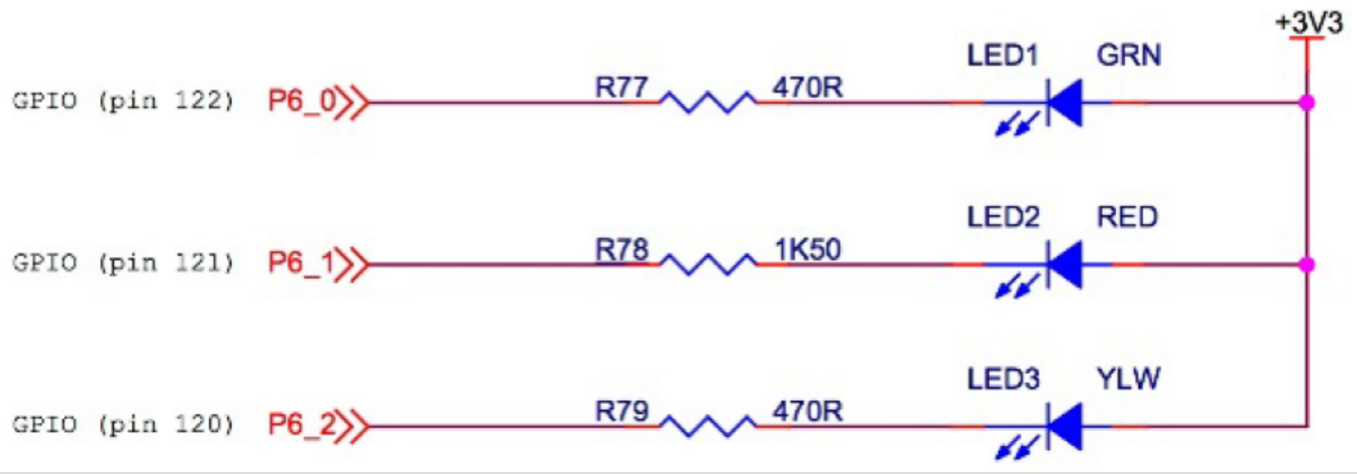
source: Starter Kit SK-S7G2 User's Manual

LAB2 – ACTIVITY 4

There are three LEDs on the SK-S7G2 that are controllable by GPIO pins: LED1 (green), LED2 (red), and LED3 (yellow).

They are connected to GPIO pins P6.0, P6.1 and P6.2 respectively.

The LEDs turn on when a logic level 0 is written to the pin and they turn off writing a logic level 1.



source: Starter Kit SK-S7G2 User's Manual

LAB2 – ACTIVITY 4

Activity 4 – Study the Hardware platform (SK-S7G2 board and S7G2 MCU)

The next manual to be studied is the Renesas S7 Series Microcontrollers User's Manual.pdf

Relevant information in this manual concerning the current project is:

- Overview
- CPU
- Clock Generation
- Event Link Controller
- I/O Ports
- Timers

LAB2 – ACTIVITY 4

S7G2 microcontroller's block diagram with indication of blocks of interest.

Since the user's manual of the S7G2 has more than 2000 pages, it is important to keep focus on what is relevant to this project.

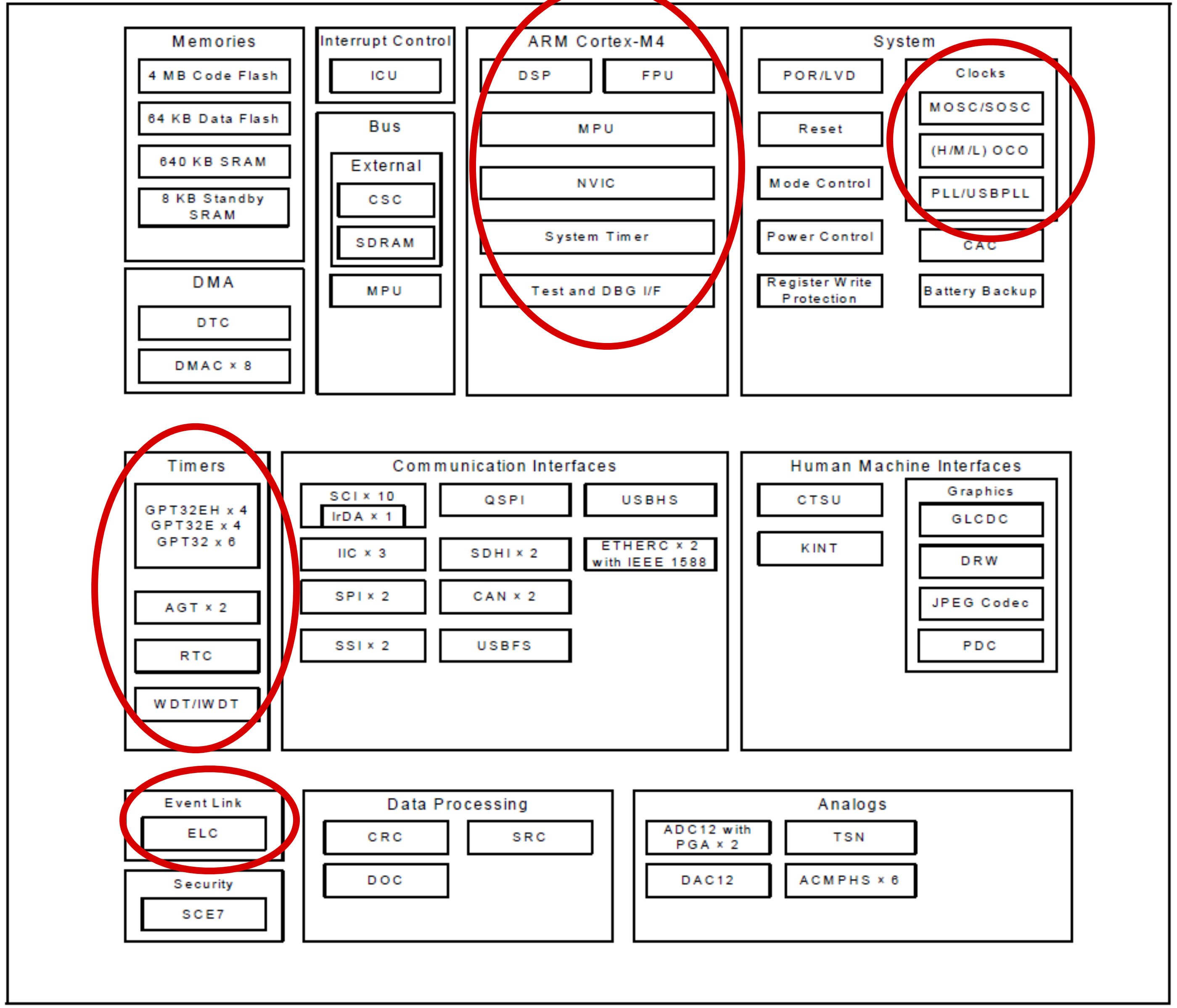


Figure 1.1 **Block diagram**

source: Renesas S7 Series Microcontrollers User's Manual

LAB2 – ACTIVITY 4

The S7G2 microcontroller's clock generation circuit is very flexible and can provide several different clock frequencies to different parts of the MCU. Following limits must be observed:

Flash clock (FCLOCK) - 60 MHz

Core clock (ICLOCK) - 240 MHz

PCLKA - 120 MHz

PCLKB - 60 MHz

PCLKC - 60 MHz

PCLKD - 120 MHz

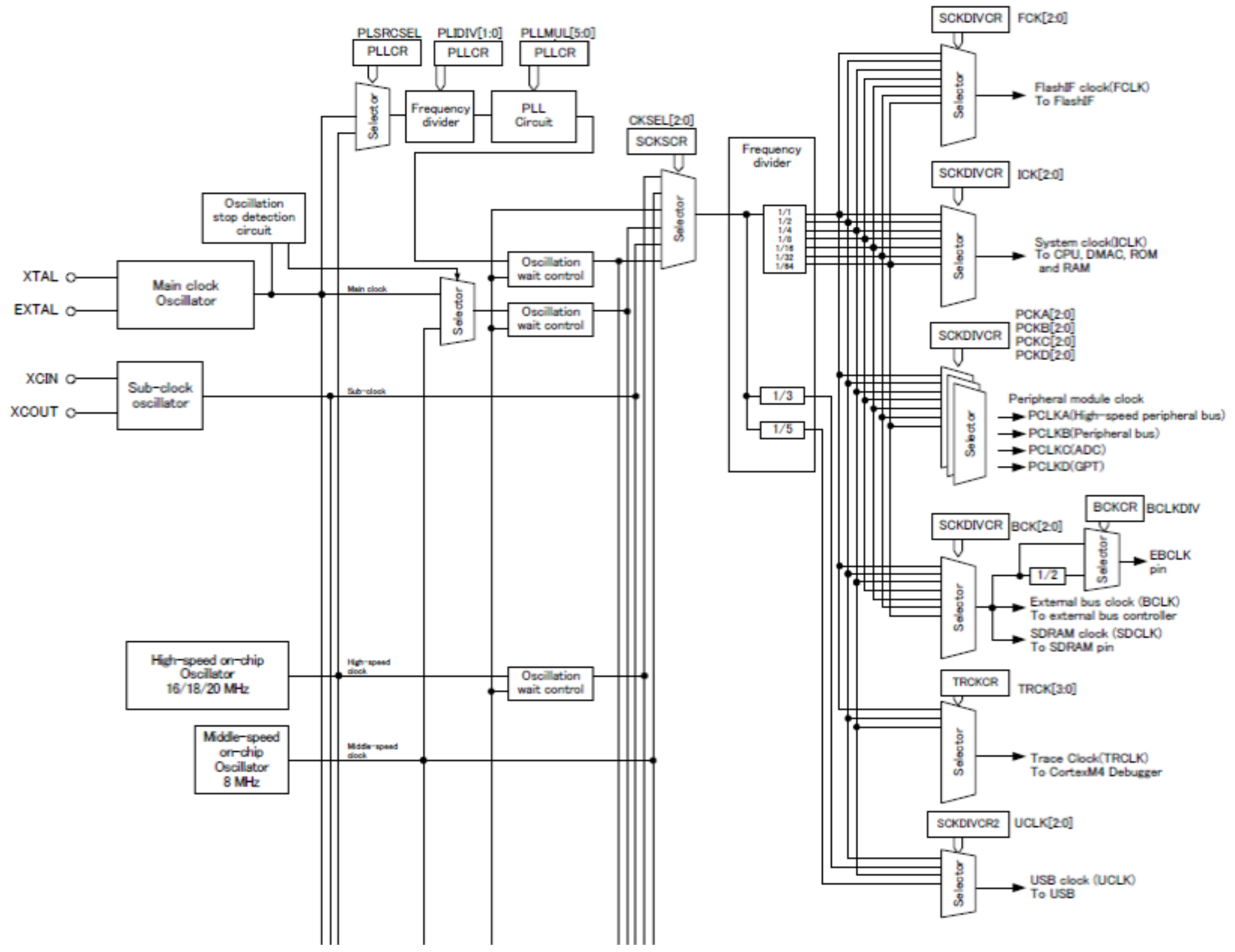


Fig 9.1 (partial)

source: Renesas S7 Series Microcontrollers User's Manual

LAB2 – ACTIVITY 4

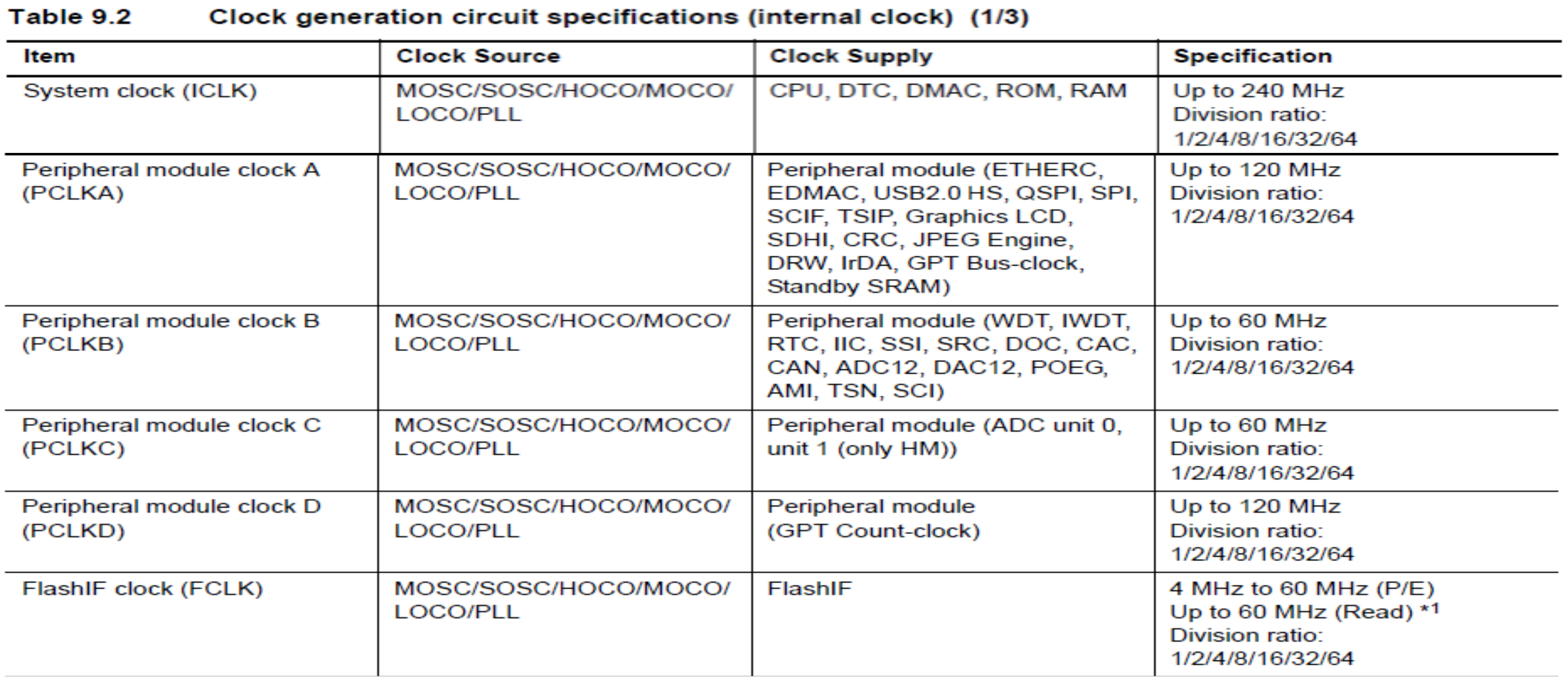


Table 9.2 Clock generation circuit specifications (internal clock) (1/3)

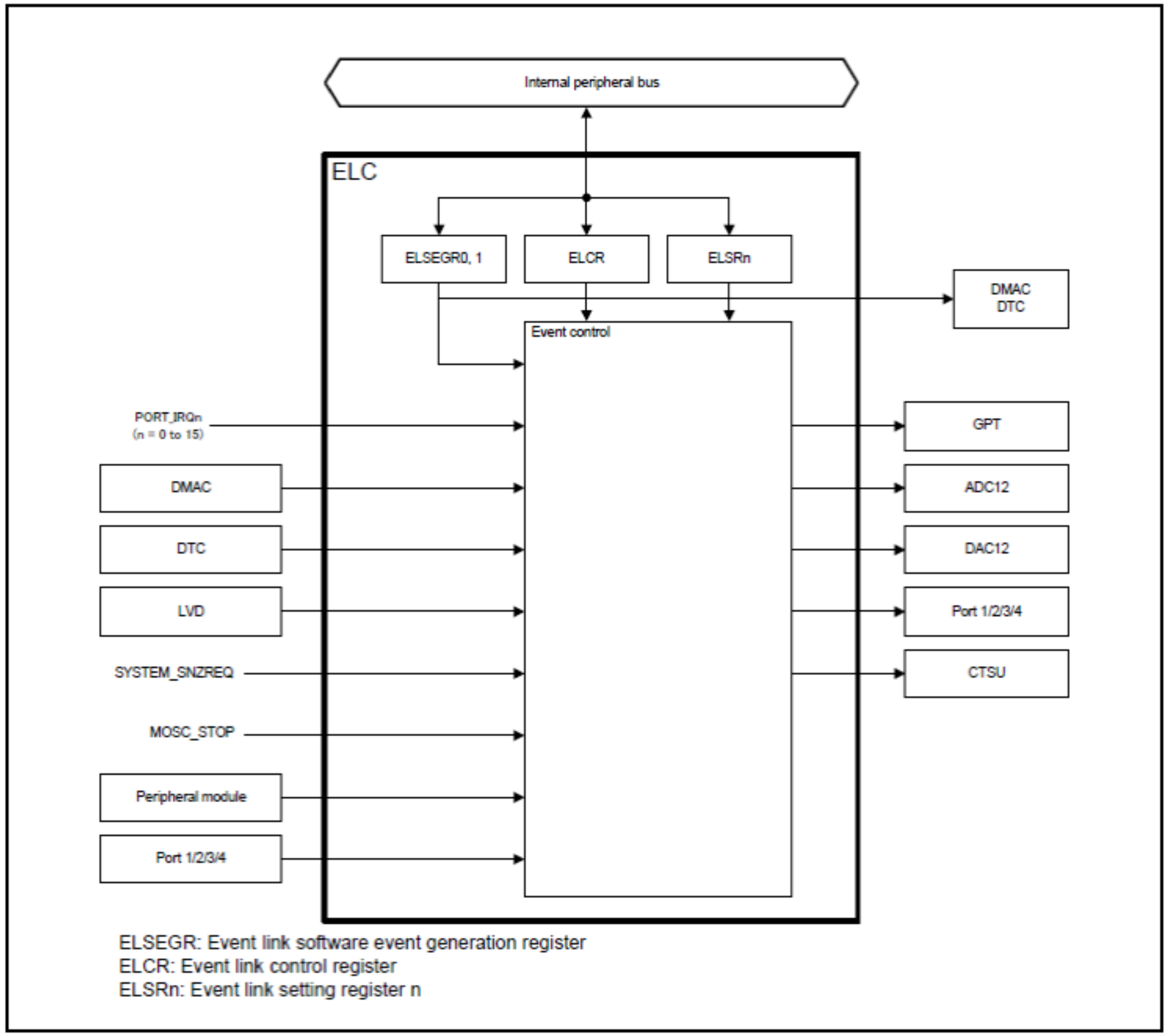
Item	Clock Source	Clock Supply	Specification
System clock (ICLK)	MOSC/SOSC/HOCO/MOCO/LOCO/PLL	CPU, DTC, DMAC, ROM, RAM	Up to 240 MHz Division ratio: 1/2/4/8/16/32/64
Peripheral module clock A (PCLKA)	MOSC/SOSC/HOCO/MOCO/LOCO/PLL	Peripheral module (ETHERC, EDMAC, USB2.0 HS, QSPI, SPI, SCIF, TSIP, Graphics LCD, SDHI, CRC, JPEG Engine, DRW, IrDA, GPT Bus-clock, Standby SRAM)	Up to 120 MHz Division ratio: 1/2/4/8/16/32/64
Peripheral module clock B (PCLKB)	MOSC/SOSC/HOCO/MOCO/LOCO/PLL	Peripheral module (WDT, IWDT, RTC, IIC, SSI, SRC, DOC, CAC, CAN, ADC12, DAC12, POEG, AMI, TSN, SCI)	Up to 60 MHz Division ratio: 1/2/4/8/16/32/64
Peripheral module clock C (PCLKC)	MOSC/SOSC/HOCO/MOCO/LOCO/PLL	Peripheral module (ADC unit 0, unit 1 (only HM))	Up to 60 MHz Division ratio: 1/2/4/8/16/32/64
Peripheral module clock D (PCLKD)	MOSC/SOSC/HOCO/MOCO/LOCO/PLL	Peripheral module (GPT Count-clock)	Up to 120 MHz Division ratio: 1/2/4/8/16/32/64
FlashIF clock (FCLK)	MOSC/SOSC/HOCO/MOCO/LOCO/PLL	FlashIF	4 MHz to 60 MHz (P/E) Up to 60 MHz (Read) *1 Division ratio: 1/2/4/8/16/32/64

source: Renesas S7 Series Microcontrollers User's Manual

LAB2 – ACTIVITY 4

The ELC (Event Link Controller) connects events generated by peripheral modules to other peripheral modules. This direct communication among modules does not require intervention from the CPU.

The ELC is not required in this project, however, it is a compulsory module in the configuration of the SSP.



ELSEGR: Event link software event generation register
ELCR: Event link control register
ELSRn: Event link setting register n

Figure 19.1 ELC block diagram (n = 0 to 18)

source: Renesas S7 Series Microcontrollers User's Manual

LAB2 – ACTIVITY 4

The block diagram of a GPIO pin.

Among the configurable features of a pin are:

- Input or output,
- Enable a pull-up on input,
- Output drive capability: low, medium, high
- Use pin for analog function (ADC or DAC);
- Use pin for peripheral function (e.g. SPI);
- Some inputs are 5V tolerant;
- Pins can generate interrupt on edges of input signal.

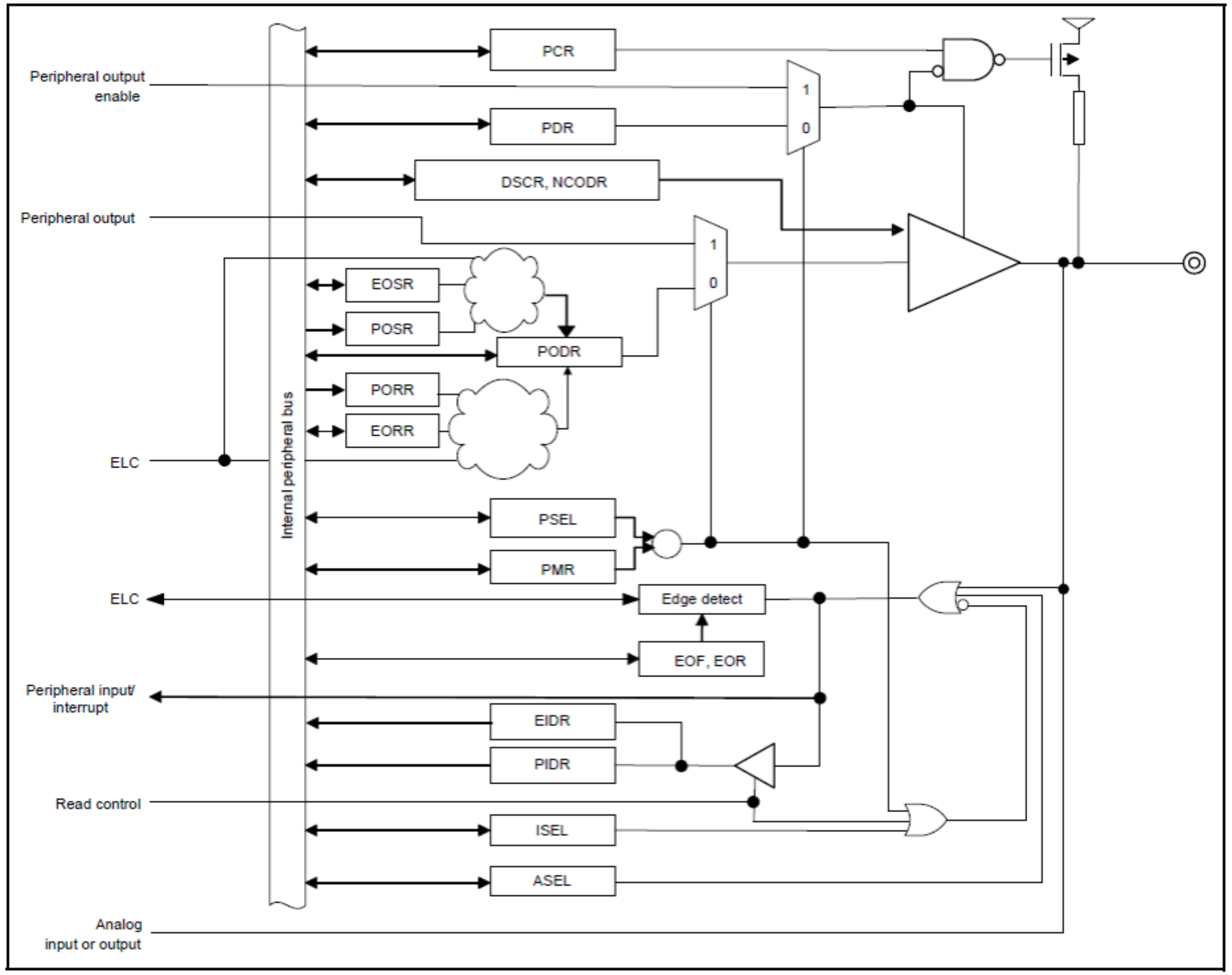


Figure 20.1 **I/O Port registers connection diagram**

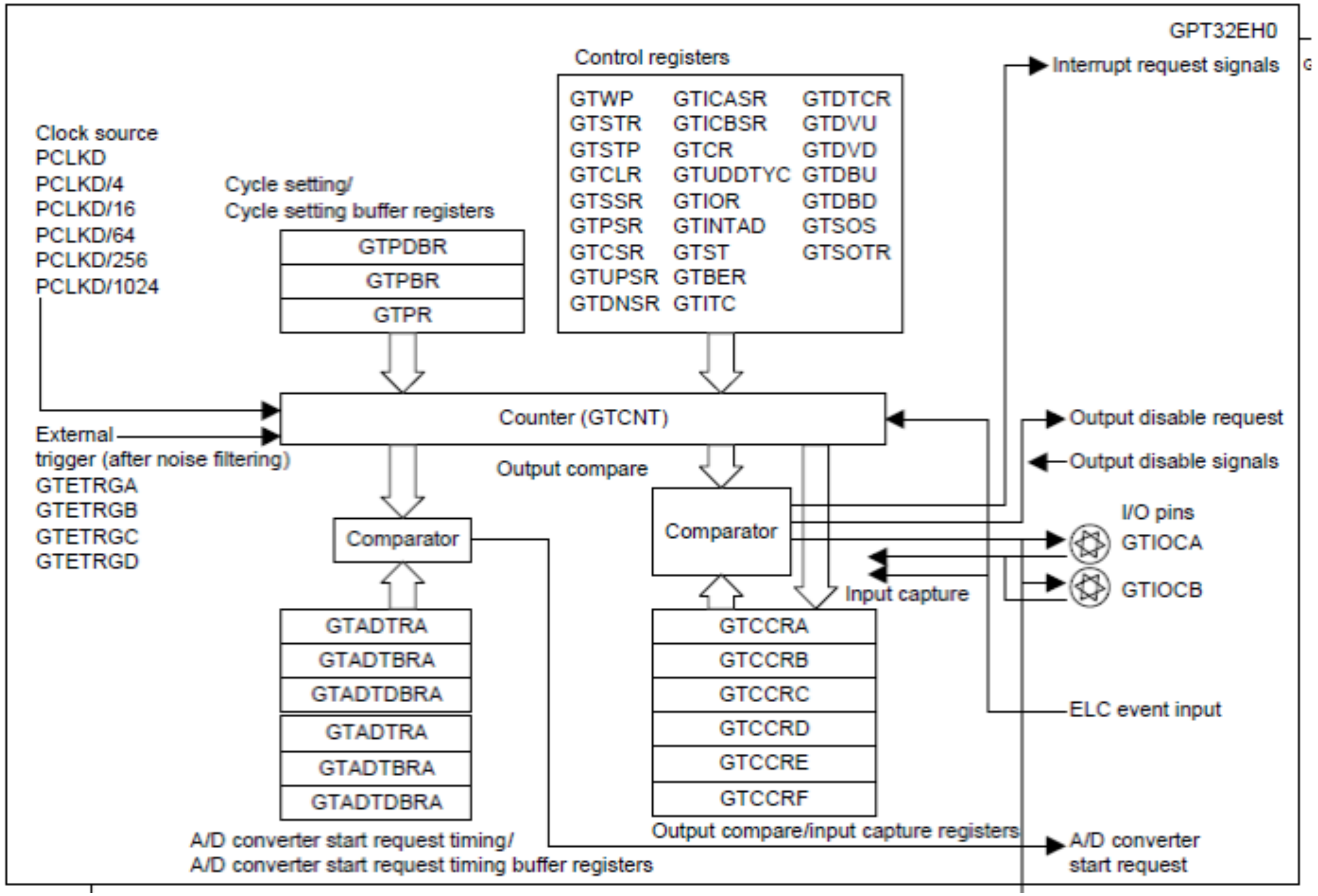
source: Renesas S7 Series Microcontrollers User's Manual

LAB2 – ACTIVITY 4

The block diagram of GPT (General PWM Timer).

GPT characteristics:

- 32-bit counter
- Counts up or down
- Can generate interrupts
- Can start an ADC conversion
- Counts PCLKD pulses
- Periodic or single-shot
- 14 channels, each is a 32-bit counter



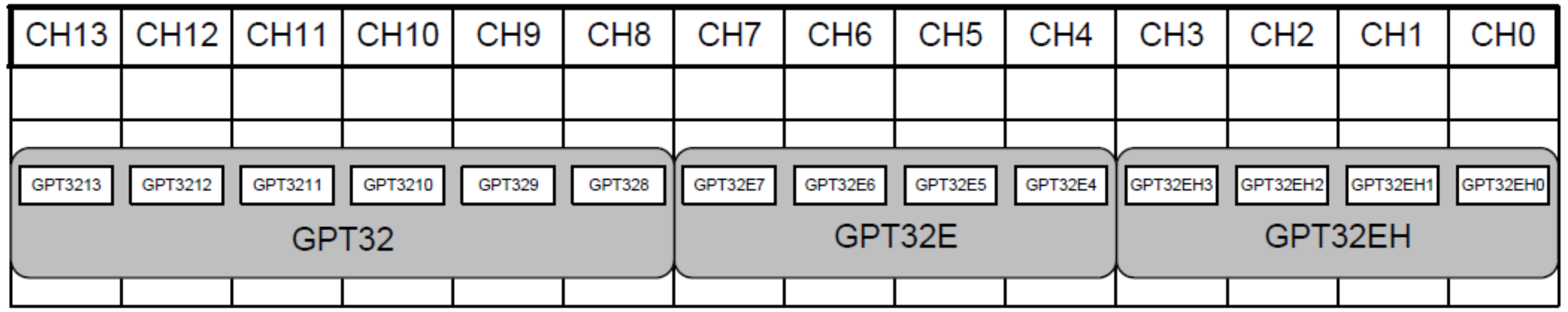
source: Renesas S7 Series Microcontrollers User's Manual

BIG IDEAS FOR EVERY SPACE

LAB2 – ACTIVITY 4

The 14 timer channels are grouped into:

- 4x EH – enhanced high resolution
- 4x E – enhanced
- 6x – conventional



source: Renesas S7 Series Microcontrollers User's Manual

LAB2 – ACTIVITY 5

Activity 5 – Software Framework Study

Study the Renesas Synergy Software Package v1.7.5 (link) available in the Synergy Gallery; specifically Section 4.2.21 about the HAL GPT (General PWM Timer).

https://synergygallery.renesas.com/media/products/1/384/en-US/r11um0140eu0106-synergy-ssp-v175.pdf

Also, study the Module Guide for the GPT HAL (link).

https://www.renesas.com/us/en/doc/products/renesas-synergy/apn/r11an0091eu0101-synergy-gpt-hal-mod-guide.pdf

LAB2 – ACTIVITY 6

Activity 6 – Design

(identification of the SW components that need to be developed, their interfaces, algorithms and interfaces to other software components)

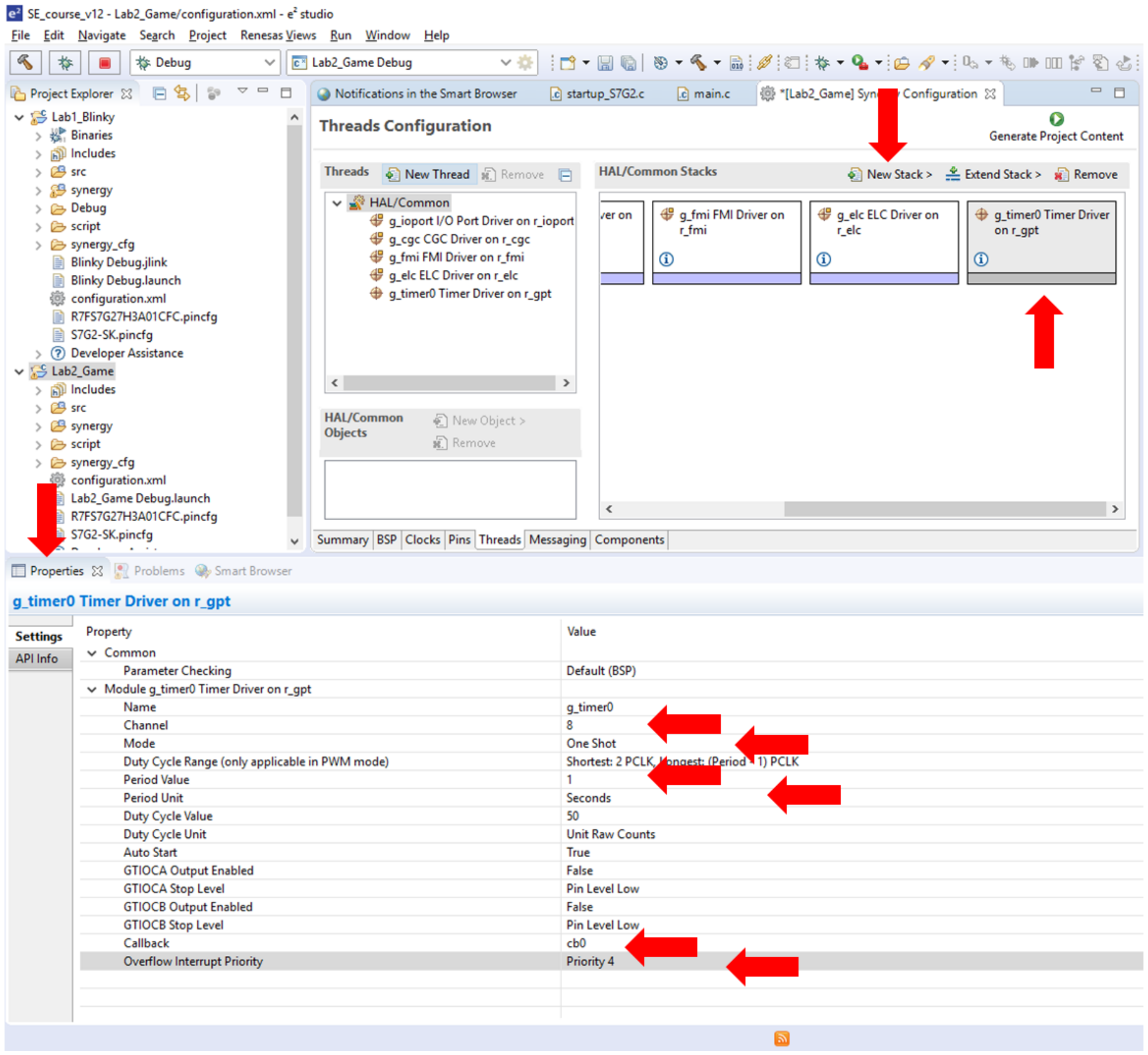
Components of the SSP to be used:

- Timer Driver on r_gpt. This component uses one of the 14 channels of the General PWM Timer. We selected channel 8 for no particular reason.
- External IRQ Driver on r_icu. This component manages the interrupt generated by one of the GPIO lines. Here we select IRQ 11 because on the SK-S7G2 board, the S5 push-button is connected to the IRQ11 pin.

LAB2 – ACTIVITY 6

Timer Driver

on r_gpt



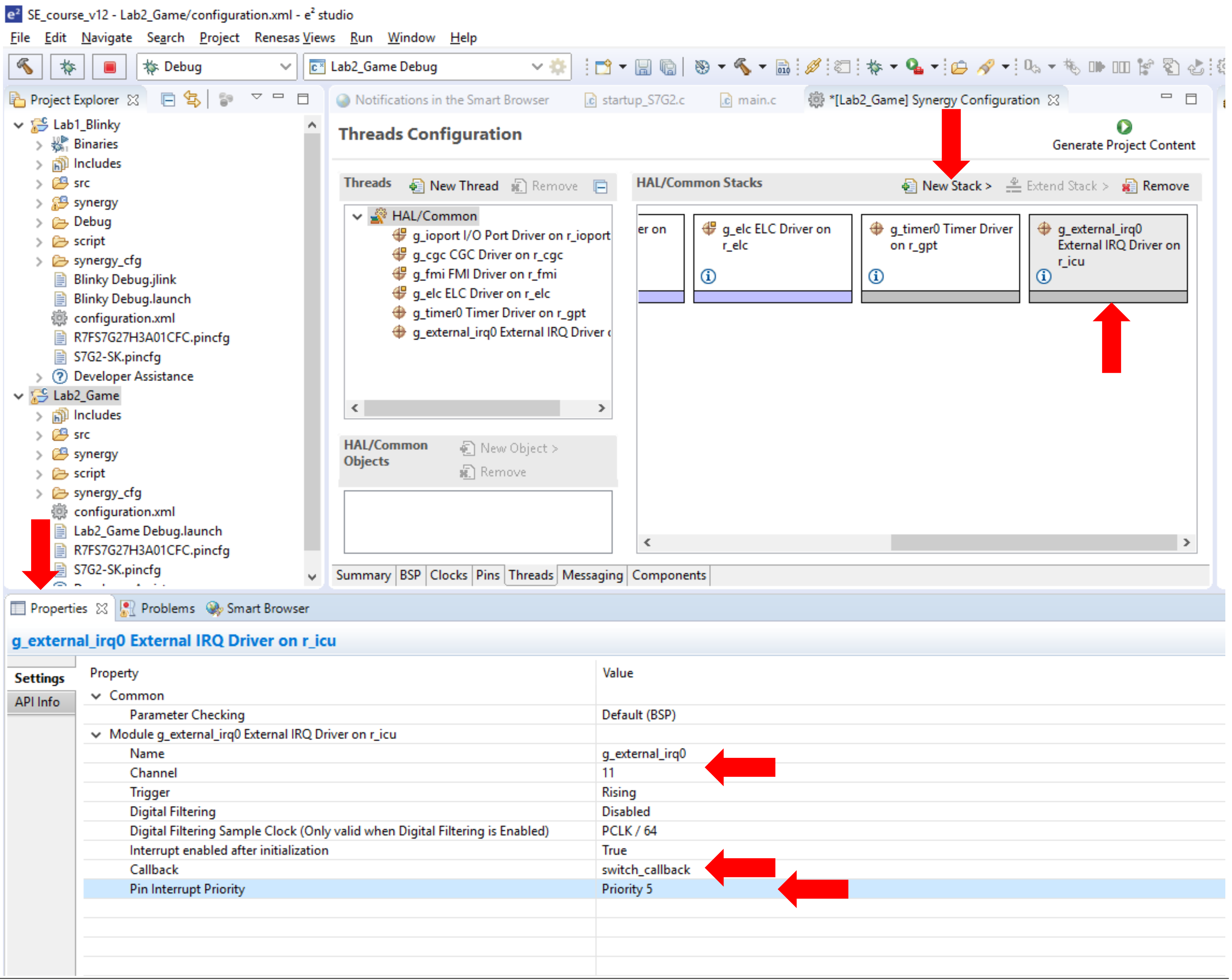
g_timer0 Timer Driver on r_gpt

Property	Value
Common	
Parameter Checking	Default (BSP)
Module g_timer0 Timer Driver on r_gpt	
Name	g_timer0
Channel	8
Mode	One Shot
Duty Cycle Range (only applicable in PWM mode)	Shortest: 2 PCLK, Longest: (Period - 1) PCLK
Period Value	1
Period Unit	Seconds
Duty Cycle Value	50
Duty Cycle Unit	Unit Raw Counts
Auto Start	True
GTIOCA Output Enabled	False
GTIOCA Stop Level	Pin Level Low
GTIOCB Output Enabled	False
GTIOCB Stop Level	Pin Level Low
Callback	cb0
Overflow Interrupt Priority	Priority 4

LAB2 – ACTIVITY 6

External IRQ Driver

on r_icu



g_external_irq0 External IRQ Driver on r_icu

Property	Value
Common	
Parameter Checking	Default (BSP)
Module g_external_irq0 External IRQ Driver on r_icu	
Name	g_external_irq0
Channel	11
Trigger	Rising
Digital Filtering	Disabled
Digital Filtering Sample Clock (Only valid when Digital Filtering is Enabled)	PCLK / 64
Interrupt enabled after initialization	True
Callback	switch_callback
Pin Interrupt Priority	Priority 5

LAB2 – ACTIVITY 6

Algorithm:

1. Initialize the components for the Timer Driver and the External IRQ Driver by calling their open API functions. This also starts the timer on its 1 second period as configured in the Synergy Configuration Tool.
2. Turn off the LEDs
3. Wait for the timer to expire. Its callback function informs via a flag (shared volatile global variable).
4. Reprogram the timer for 3 seconds and restart count (API functions `reset, periodSet, restart`).
5. Wait for the push-button to be pressed. Again, its callback function informs via another flag.
6. Get the current count of the timer (API functions `counterGet`) and save this value to a variable (this variable will be examined with the debugger).

This algorithm is to be implemented by modifying the function `hal_entry`, adding two callback functions and the required global variables.

LAB2 – ACTIVITIES 7 TO 11

Based on the previous study:

- After creating a new Synergy C project (lab2) based on the Blinky template, on the Threads tab of Synergy Configuration, add a Timer Driver (on r_gpt) and configure its properties to: channel 8, single shot, period of 1 second, callback `cb0`, interrupt priority 4.
- Modify the file hal_entry.c to:
  - Turn off all three LEDs, a LED is turned off by writing IOPORT_LEVEL_HIGH to the GPIO pin.
  - Call the GPT API function open(`g_timer0.p_ctrl,g_timer0.p_cfg`) to configure and start the timer. The two parameters of this function are created by the Synergy Configuration tool when the button "Generate Project Content" is pressed.
  - Create a callback function named `cb0`. This function must inform the `hal_entry` function that the timer expired by setting a flag. This flag must be a volatile global variable. Reminder: volatile informs the compiler that its value changes outside the control of the `hal_entry` function.

LAB2 – ACTIVITIES 7 TO 11

Modify the file hal_entry.c to (cont.):

- In function `hal_entry`, wait for the flag to change value, meaning 1 second has passed;
- Turn LEDs on;
- Reset the timer (API function reset);
- Set a new period of 3 seconds (API function `periodSet`);
- Start the timer again (API function `start`);
- Make sure the timer is operating by making successive calls to API function `counterGet`;
- Test if the program is operating correctly up to this point.

LAB2 – ACTIVITIES 7 TO 11

- In the Synergy Configuration tab, add an input driver of type external IRQ driver on r_icu. Configure its properties to: channel 11, callback function `switch_callback`, interrupt priority 5. Recall that, from the study of the board manual, we learned that push-button S4 is connected to IRQ 11.
- Modify the file hal_entry.c to:
  - In the `hal_entry` function, call the open API function of the `g_external_irq0` to configure this component;
  - Create a callback function named `switch_callback`. This function also informs the `hal_entry` function that the timer expired by setting another volatile flag;
  - In the `hal_entry` function, wait for this flag to change then get the number of ticks from the counter up to this point. Save it to a variable;
  - Put a breakpoint right after the variable is updated and play the game.

Renesas.com