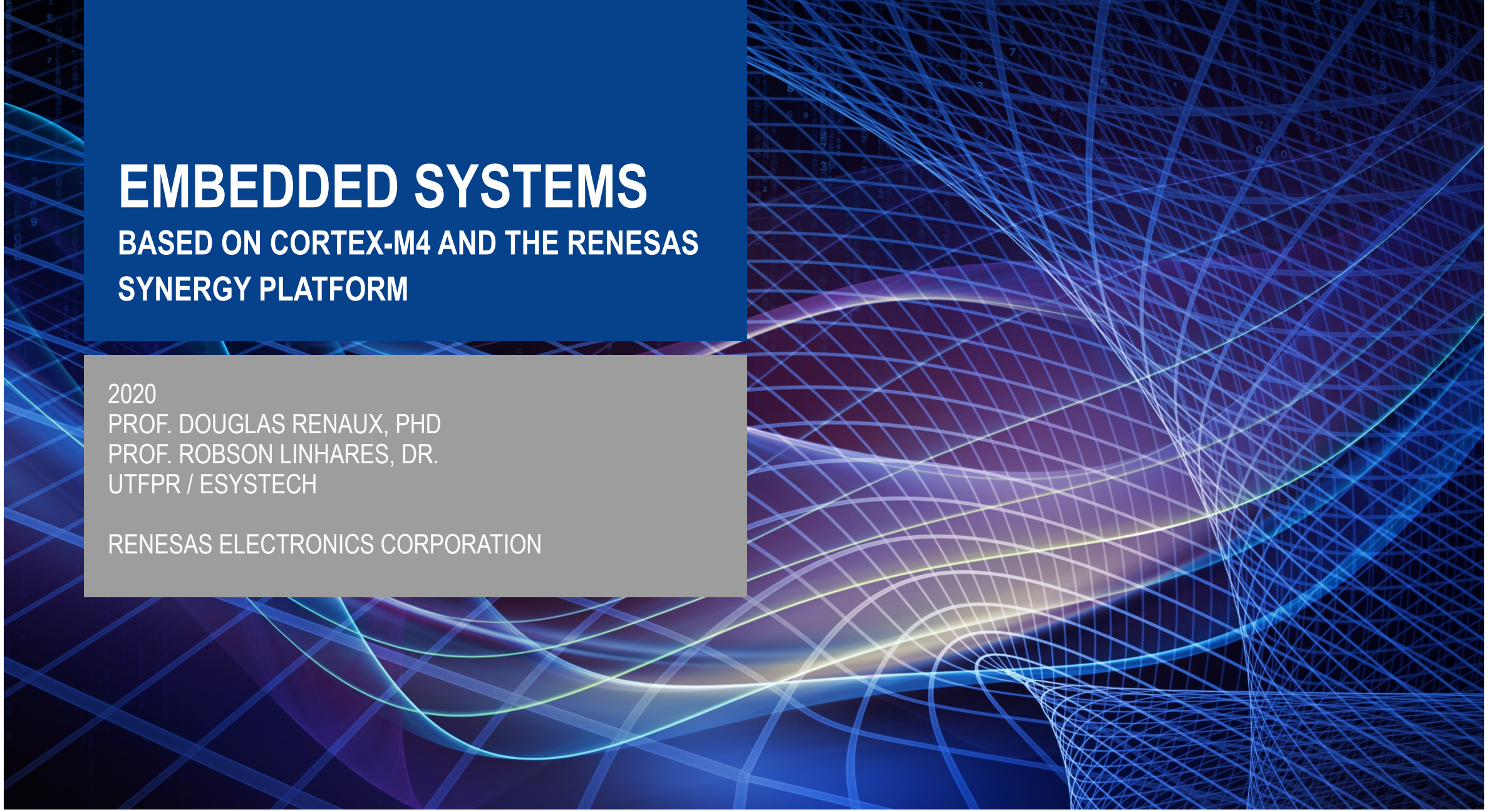


EMBEDDED SYSTEMS

BASED ON CORTEX-M4 AND THE RENESAS SYNERGY PLATFORM

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

RENESAS ELECTRONICS CORPORATION

5 – TIMER AND GPIO

1. Introduction - Peripherals
2. Timer
3. PWM – Pulse Width Modulations
4. GPIO
5. Low-power Drivers (LED, Relay)
6. Power Drivers (DC Motor)

5.1 – PERIPHERALS

A Microprocessor-based system consists of three types of components:

1. **Microprocessor** (the **core** of the MCU) - executes the instructions of a program;
2. **Memory** - store code and data;
3. **Peripherals** - perform specific functions particularly related to I/O. Peripherals provide the means for the system to sense, actuate and communicate with the world.

A microprocessor-based system without peripherals would be unable to interact with the rest of the world!

PERIPHERALS

Classes of peripherals in a microprocessor-based system:

- Digital I/O: input/output of digital signals,
- Analog I/O: input/output of analog signals,
- Timing: pulse generation, pulse measurement, PWM, ...
- Storage: non-volatile memory, file system, ...
- Communications: RS-232, SPI, I2C, CAN, USB, Ethernet, ...
- HMI: touch-screen, keyboard, ...
- Imaging: camera interface, ...
- System management: clock generation, watchdog, power management, ...

5.2 – TIMERS/COUNTERS

Timers/Counters are essential components of a Microcontroller Unit (MCU). They consists of a **digital counter circuit** that counts pulses on their input. **Timers** count clock pulses and Counters count pulses of a signal present on their input pin. **Timers/Counters** are used for:

- Measuring time, particularly time intervals,
- Counting events,
- Keeping track of current date and time (Real-Time Clock - RTC),
- Generating digital waveforms (PWM - Pulse-Width Modulation),
- Generating periodic interrupts to the MCU (Operating System Clock),
- Generating periodic signals to other peripheral, such as the command to start an Analog-to-Digital Conversion,
- Restart the MCU if software is unable to periodically restart a WATCHDOG timer.

TIMERS

- Timers/Counters vary in their characteristics, such as:
  - edge of the input signal used for counting: positive, negative, or both edges;
  - count direction: up, down, or both;
  - what is being counted: clock pulses (timers) or other input signal (counters);
  - periodic timers vs single-shot;
  - a number of actions can be taken at the end of the timing period: change state of GPIO pin, generate IRQ, stop the timer, reload the timer;
  - number of bits of the counter circuit: typically 32 bits for ARM MCUs;
  - since the frequency of the input signal and number of bits of the counter determine the maximum timing period, a prescaler may be used to reduce the input frequency.

TIMING SAMPLE PROBLEM 1

Consider the problem where an LED, connected to a pin on an MCU, must be on during 0.9 second.

The first solution to be presented is a software-based solution:

1. Turn the LED on by activating the MCU pin connected to it;
2. Execute a loop N times, where N is chosen so that the execution of the loop takes 0.9 second;
3. Turn the LED off by deactivating the MCU pin connected to the LED

SOFTWARE-BASED SOLUTION

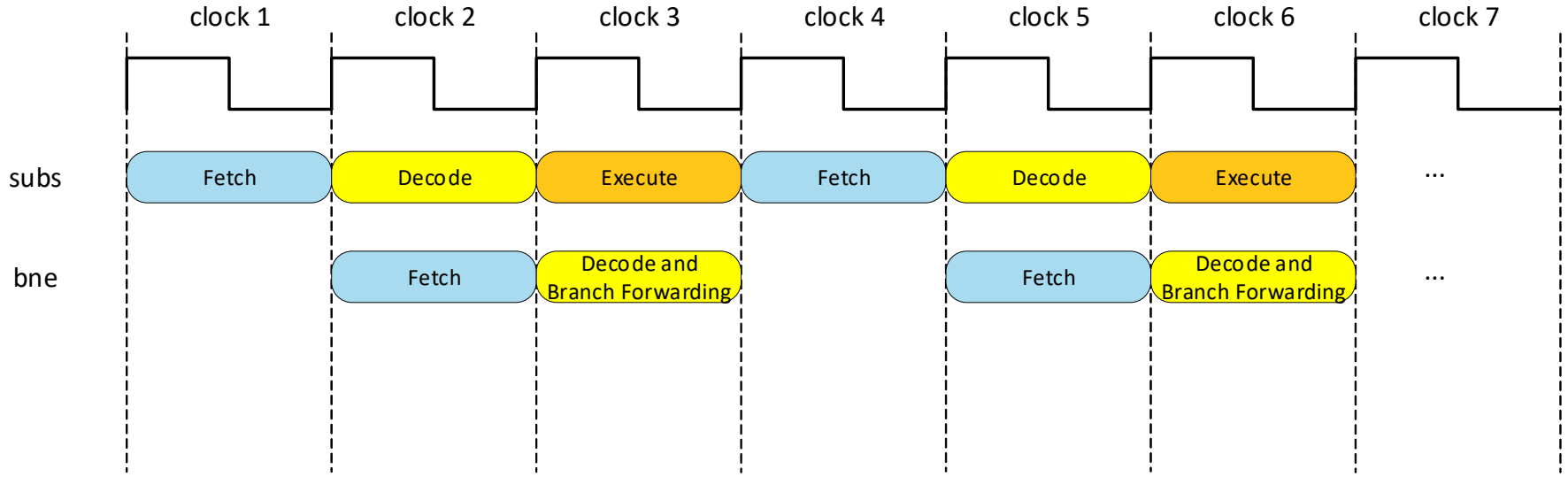
```
void hal_entry(void)
{
    // pins 0,1,2 are output, reset pin 0 to 0 to turn green led ON
    R_IOPORT6->PCNTR1 = 0x00060007;

    __asm("     ldr r4,=72000000-2");
    __asm("loop:                  ");
    __asm("     subs r4,r4,#1     ");
    __asm("     bne loop          ");

    // pins 0,1,2 are output, set pin 0 to 1 to turn green led OFF
    R_IOPORT6->PCNTR1 = 0x00070007;
}
```

SOFTWARE-BASED SOLUTION

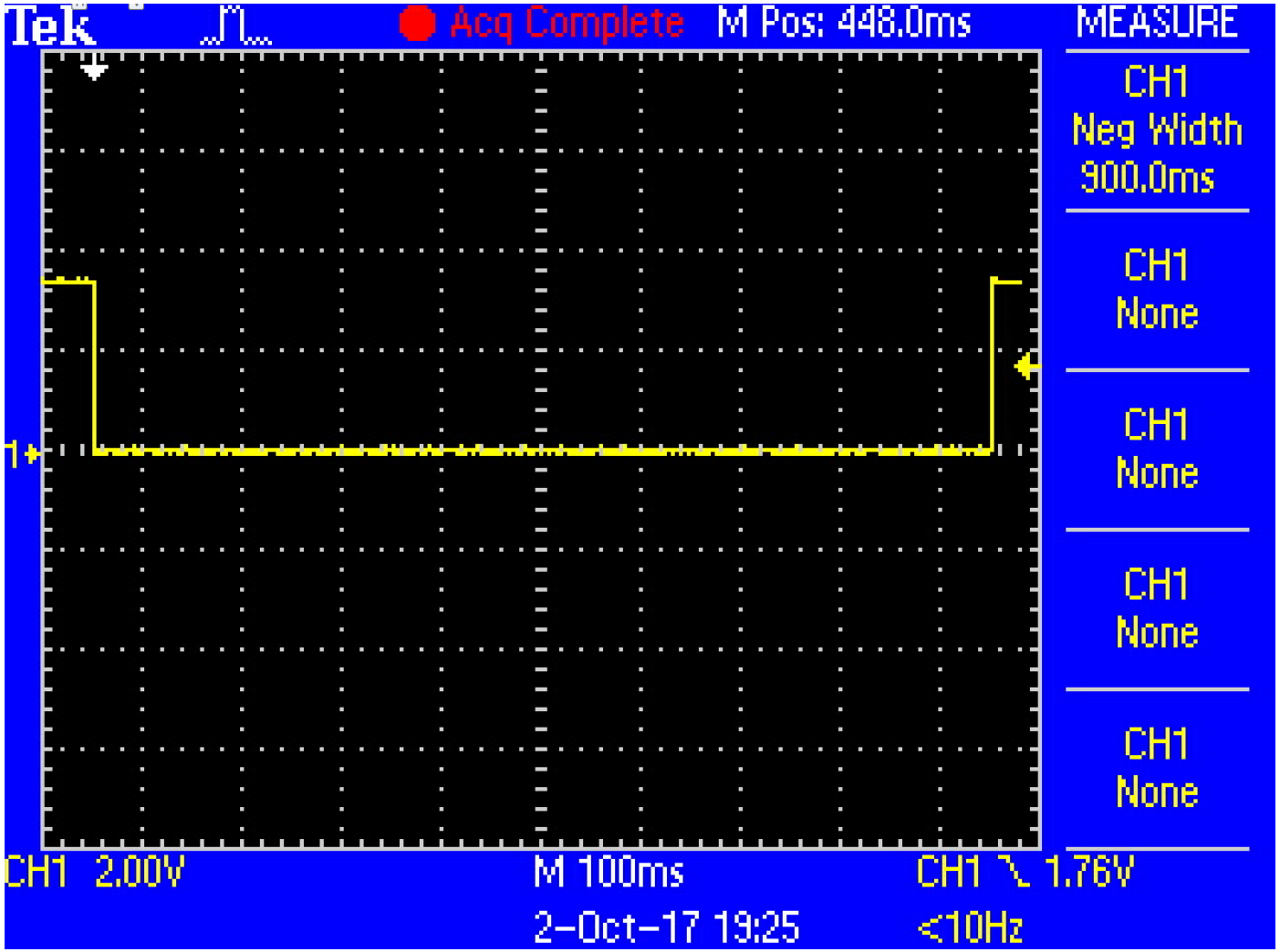
- This is the Timing Diagram for the execution of the code of the previous slide.
- Due to branch forwarding at the decode stage of the BNE instruction, each loop takes 3 clock cycles.
- At 240 MHz (period of 4.16ns), it takes 12.5 ns for each loop; hence, 0.9 second takes 72,000,000 loops.
- Remark: the loop_count must be reduced by 2 to compensate for GPIO and load instructions.



source: Authors

SOFTWARE-BASED SOLUTION

- Connecting a scope to the LED allows us to verify the timing of the software-based solution.



source: Authors

SOFTWARE-BASED SOLUTION

Evaluation of the software-based solution:

- 100% of the MCU processing capability was used for counting so that the exact timing of 0.9 seconds was obtained.
- No other activities where executed by the processor during this time.
- If interrupts where allowed, there would have been an error in timing.
  - This software-based solution is called busy-wait. It should be avoided as it wastes processor cycles and energy.

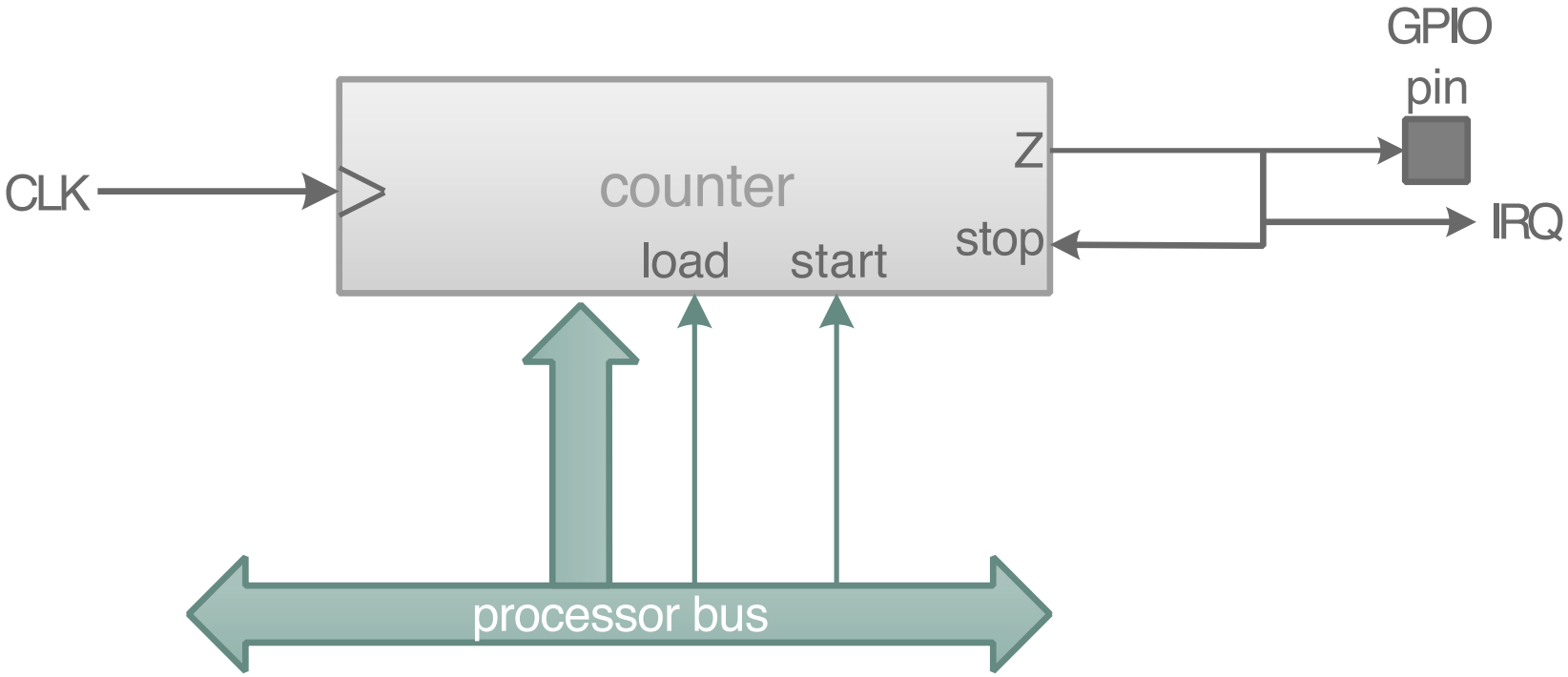
A VERY SIMPLE TIMER

- To avoid the drawbacks of the software-based solution, a small piece of hardware is added to an MCU: a counter.
- This circuit performs the function of counting clock cycles, freeing the processor to perform other tasks.

A VERY SIMPLE TIMER

Description:

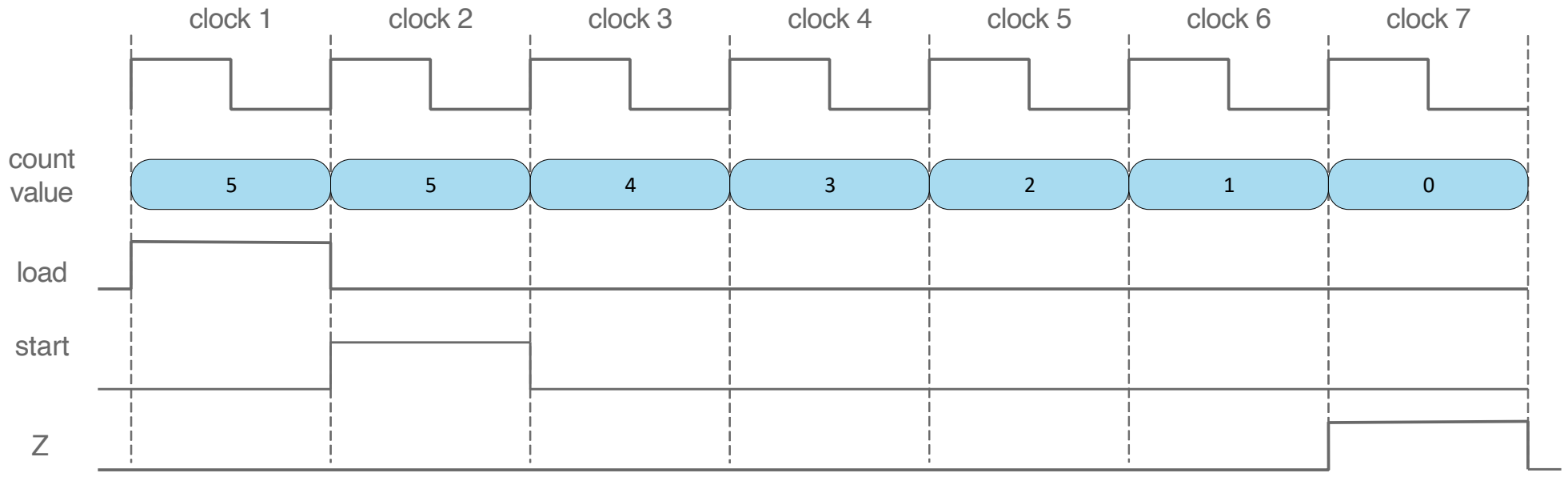
- The counter decrements its value on every positive edge of the CLK.
- The processor can write an initial count value by performing a load followed by a start command.
- When the count reaches 0 the Z output is activated.
- The Z output may command a change to the GPIO pin, generate an Interrupt Request (IRQ), and stop the counter.



source: Authors

A VERY SIMPLE TIMER

- clk 1 - the value 5 is loaded to the counter.
- clk 2 - the processor commands the start of the counting.
- clk 3 to 7 - count value is decremented on every positive edge of the CLK.
- clk 7 - count reaches 0 activating the Z output.



source: Authors

SOLVING THE TIMING SAMPLE PROBLEM 1 USING THE SIMPLE TIMER

Solution:

- Load the value 72,000,000 to the timer and start it.
- When the count reaches zero:
  - Option 1: timer commands LED pin to turn LED off;
  - Option 2: generate an IRQ and its service routine turns the LED off.
- Remark: for option 2, the value loaded to the timer should be lower to compensate for the entry into the interrupt service routine.

SOLVING THE TIMING SAMPLE PROBLEM 1 USING THE SIMPLE TIMER

Evaluation:

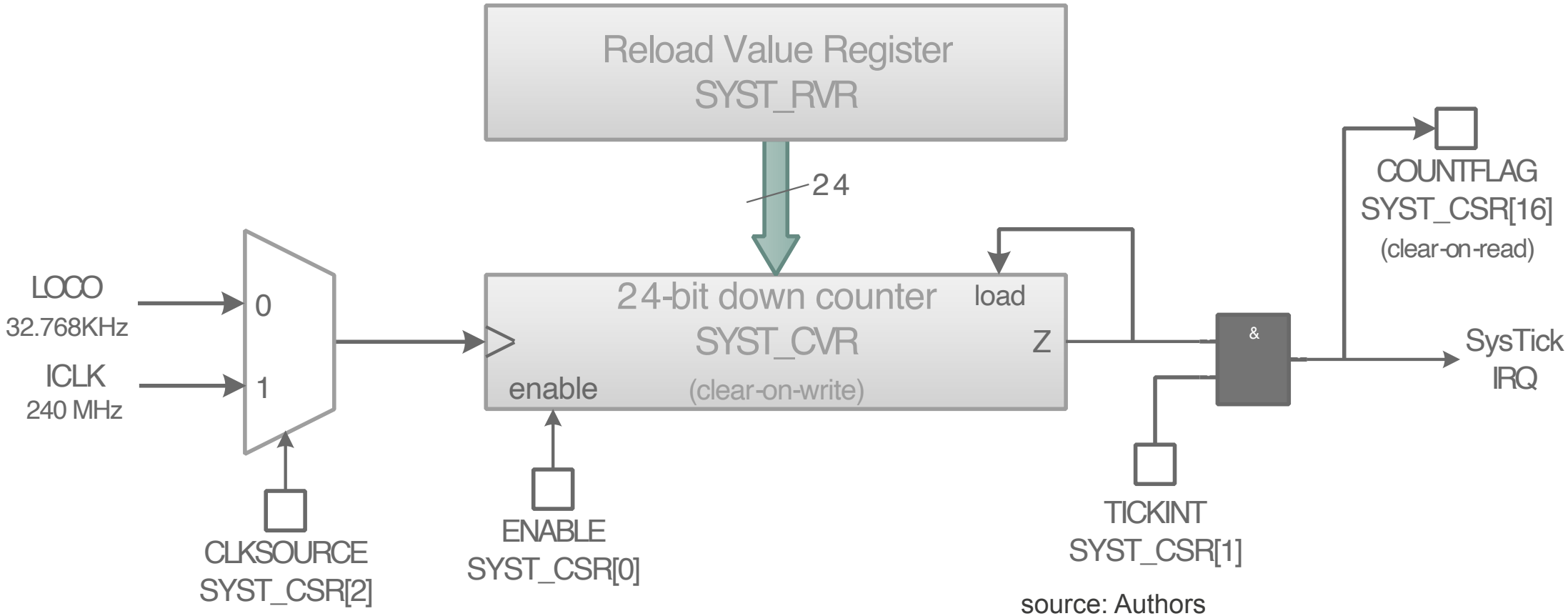
- The solution using a timer requires significantly less processing power, just a few cycles to configure the timer and to perform an action when the timing period finishes.
- Processor is free to perform other tasks OR
  processor can be put into a low energy sleep state.
- If interrupts are serviced during the timing period, this will not affect the timing precision.

TIMER CASE STUDY 1 – SYSTICK

- The SYSTICK is a timer available in all Cortex-M processors.
- It is a simple 24-bit counter with auto-reload.
- Its main use is to generate periodic interrupts required by most Embedded Operating Systems.
- Since its structure and operation is defined by ARM, its interface is standard, regardless of MCU supplier.

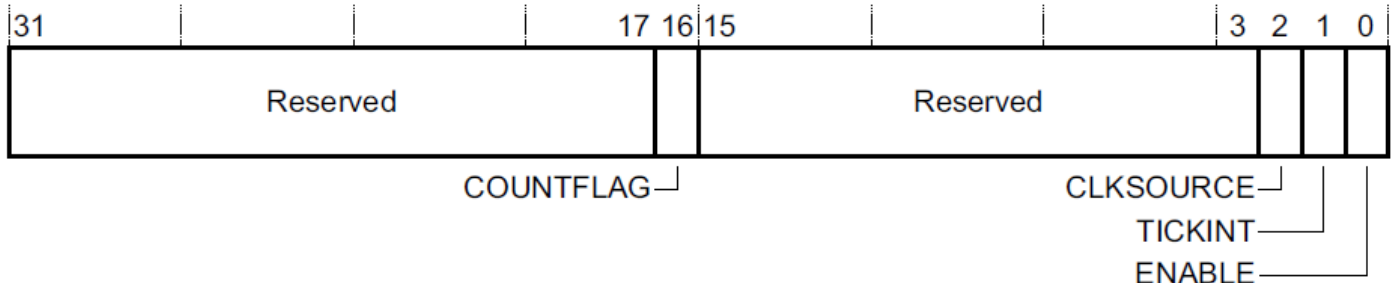
TIMER CASE STUDY 1 – SYSTICK

SysTick auto-reload has a period of SYST_RVR + 1 clock cycles.

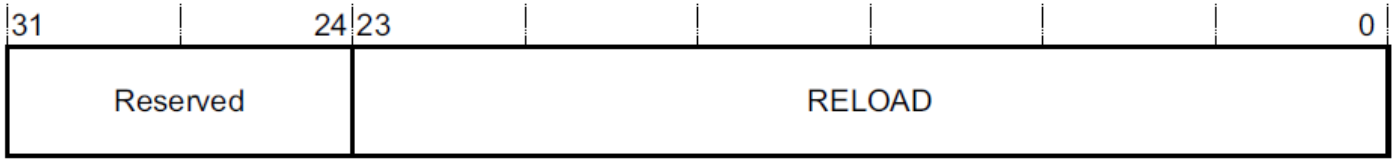


source: Authors
(based on ARM documentation:
DDI0403D ARMv7-M Architecture Reference Manual)

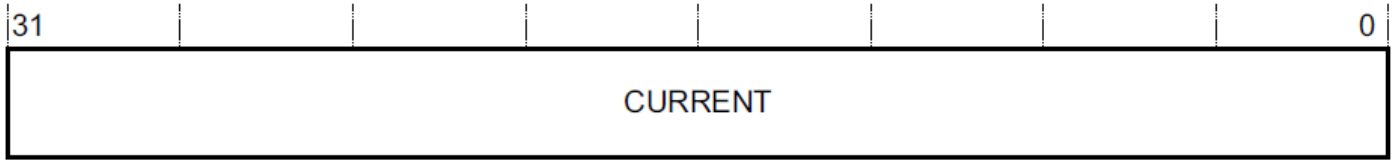
TIMER CASE STUDY 1 – SYSTICK



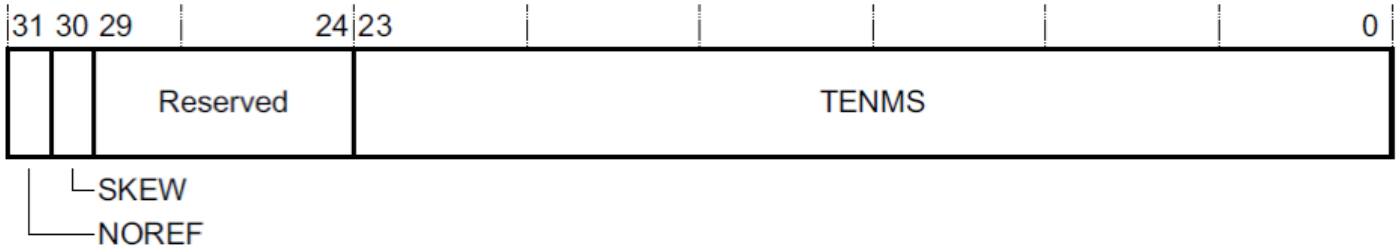
SYST_CSR controls de operation of SysTick. When this register is read, COUNTFLAG is cleared.



SYST_RVR holds de 24-bit reload value. Writing a 0 to this register disables SysTick.



SYST_CVR is the current count value. By writing any value to it the register is cleared.

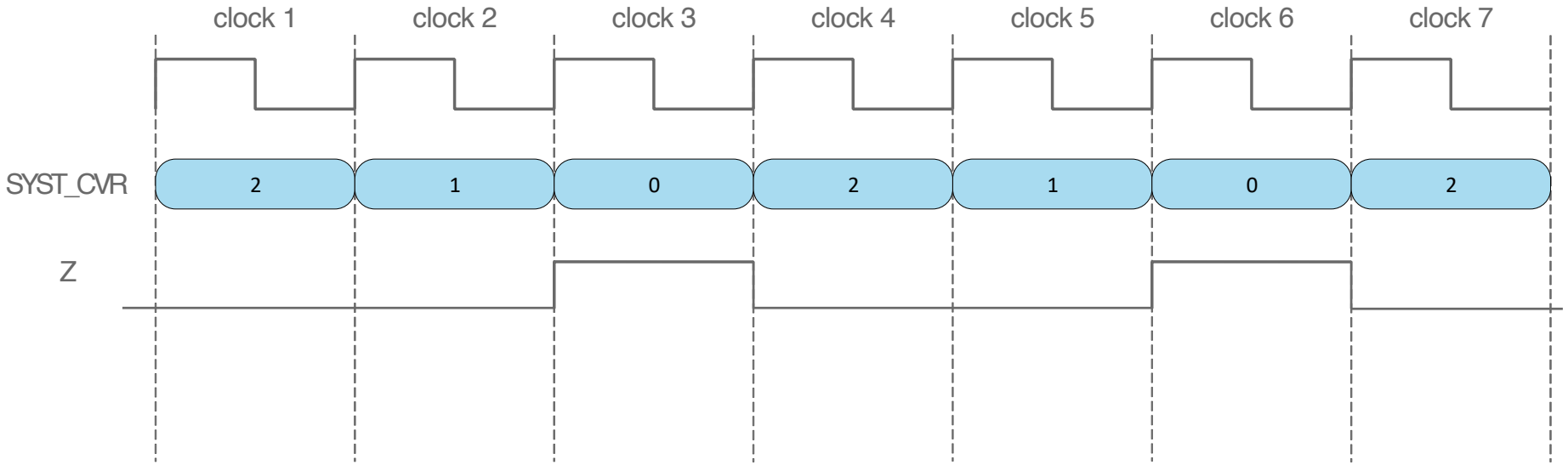


SYST_CALIB values are factory defined. TENMS, if available, holds the reload value corresponding to 10 ms for the reference clock.

source: DDI0403D ARMv7-M Architecture Reference Manual

TIMER CASE STUDY 1 – SYSTICK

- Timing diagram for the scenario where SYST_RVR holds the value 2.
- As the reload occurs on the next positive edge of the clock, the SysTick period is of 3 clock cycles.
  If enabled, every reload generates a SysTick interrupt request (IRQ).



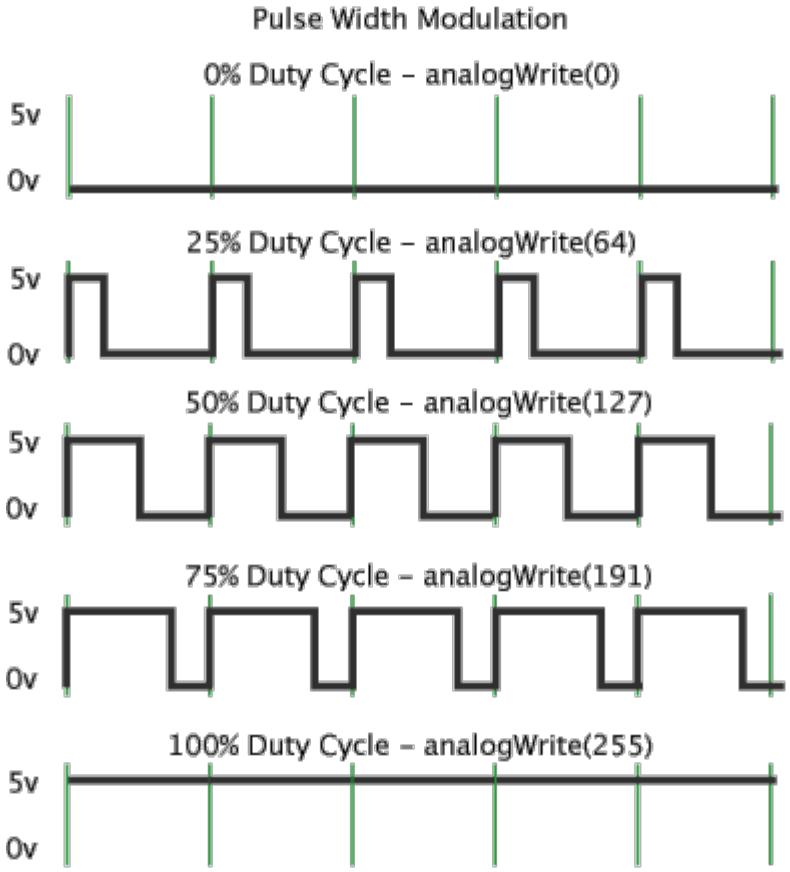
source: Authors

5.3 – PWM

- **Pulse Width Modulation** (PWM) is a modulation technique that allows the encoding of analog information in a binary digital signal.
  This is achieved by encoding the information in width of a pulse while maintaining the pulse frequency constant.
- A low pass filter with cut-off frequency way below the PWM frequency restores the analog information.
- PWM has several applications including: control of power electronics including switched power supplies, motor control, temperature control and light dimmering; audio power-amplifiers; and analog signal generation.
- Often, a low-pass filter is not required as the load itself performs this function.

PWM

The effect of 5 different duty cycles after passing a low-pass filter:



0 V

1.25 V

2.5 V

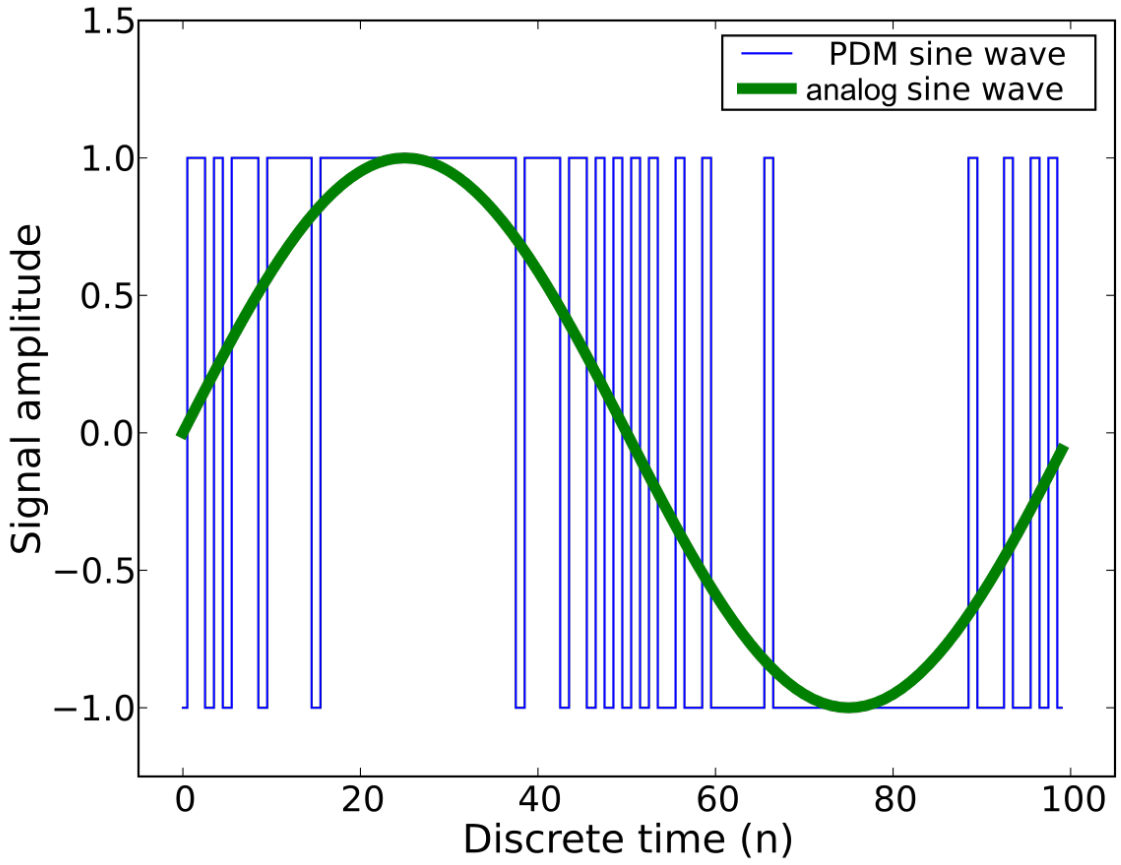
3.75 V

5V

source: commons.wikimedia.org (CC)

PWM

- If the duty cycle of a PWM signal is varied on every PWM period, an analog signal can be produced (after a low-pass filter).
- The higher the frequency of the PWM signal when compared to the frequency of the analog signal, the better (less noisy) the analog signal will be.

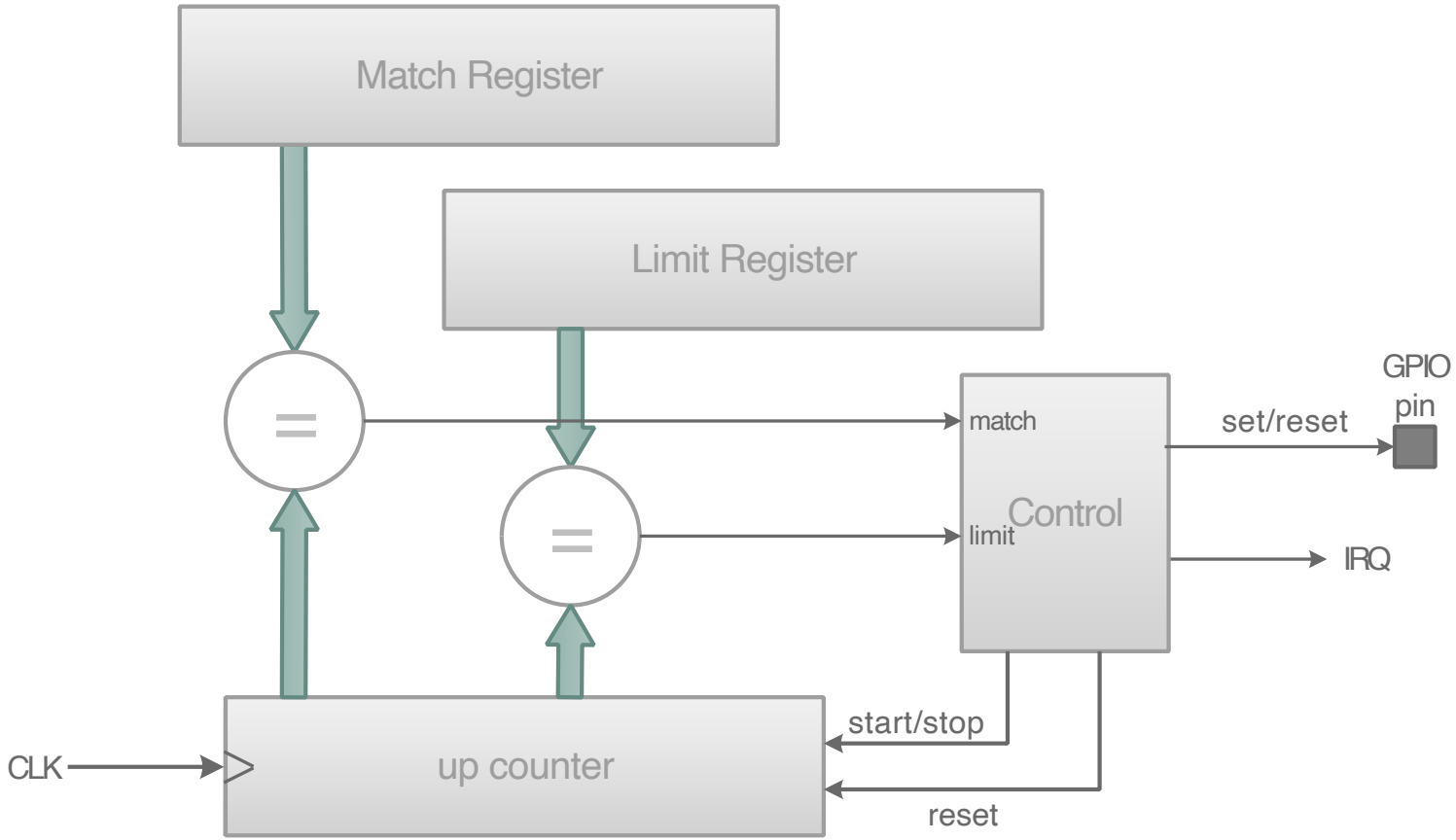


source: commons.wikimedia.org (CC)

A PWM TIMER

Operation:

- at start of period set the GPIO pin.
- counts up from 0.
- when counter matches Match Reg then reset GPIO pin.
- when counter matches Limit Reg then:
  set GPIO pin;
  generate IRQ;
  reset counter.
- interrupt service routine may reprogram the match register.



source: Authors

TIMER CASE STUDY 2 – S7G2 GPT

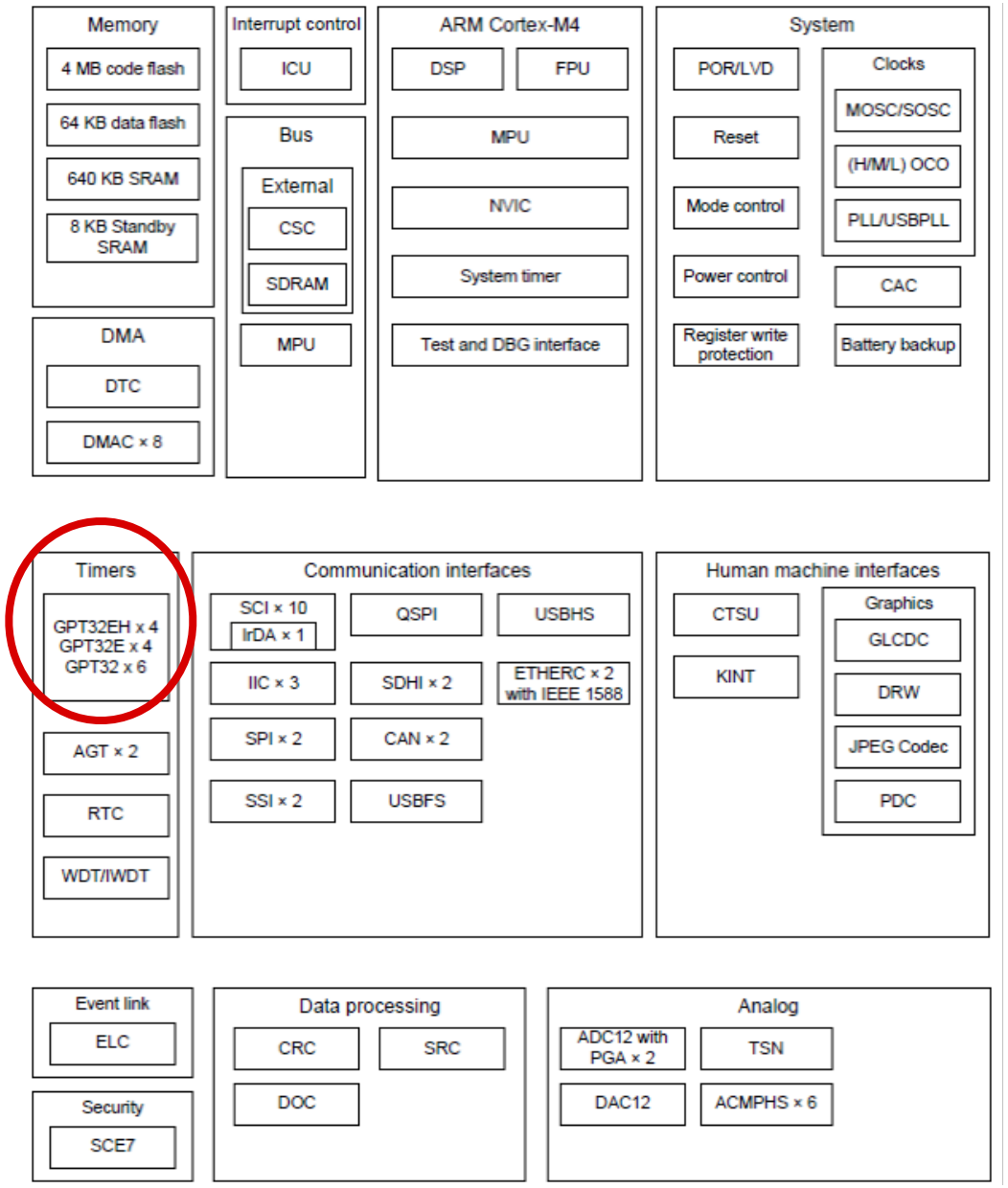
The Renesas S7G2 MCU has 14 timers of the type GPT (General PWM Timer).

Each one is a complex circuit that provides significant flexibility for a variety of applications.

TIMER CASE STUDY 2 – S7G2 GPT

Diagram shows the hardware modules of the S7G2 MCU.

General PWM Timers (GPT) are identified.

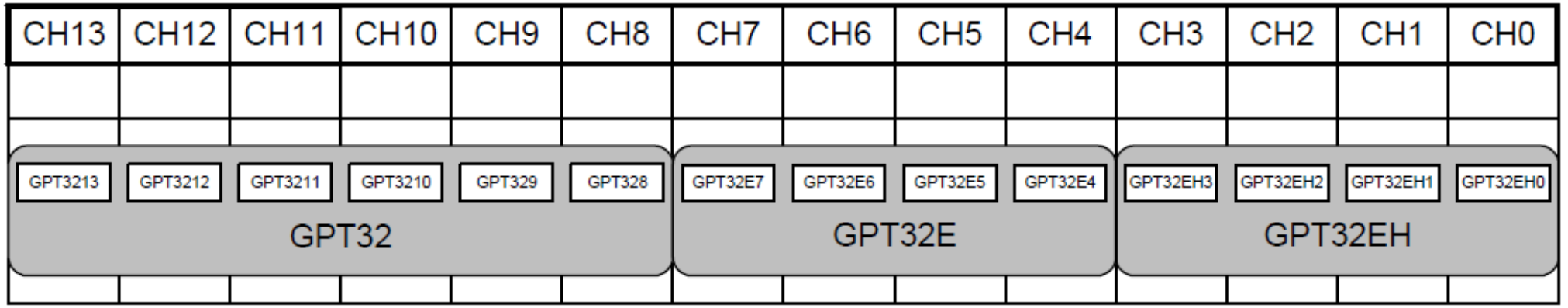


source: Renesas S7 Series Microcontrollers User's Manual

TIMER CASE STUDY 2 – S7G2 GPT

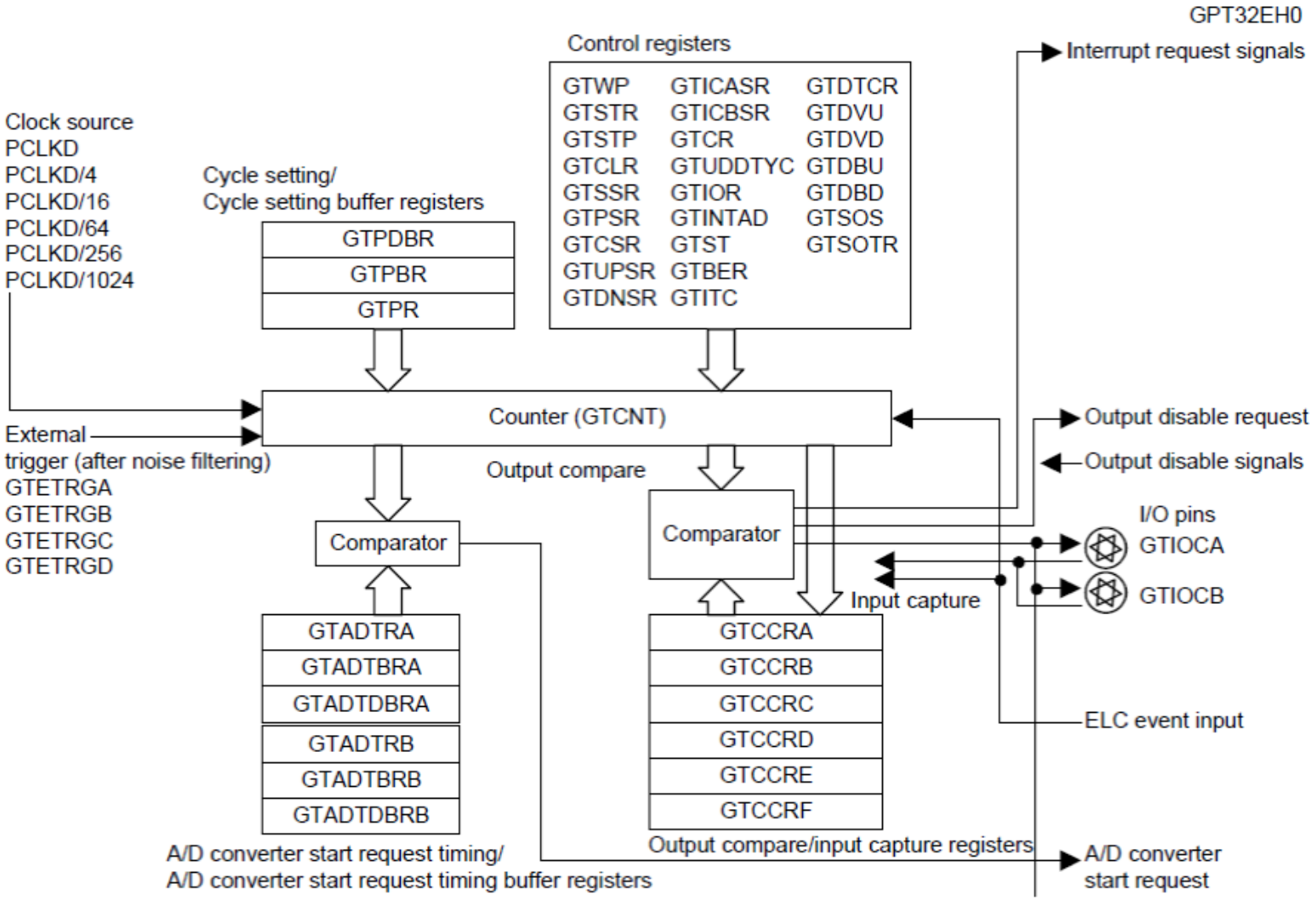
There are 14 timers, grouped into:

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



source: Renesas S7 Series Microcontrollers User's Manual

TIMER CASE STUDY 2 – S7G2 GPT



source: Renesas S7 Series Microcontrollers User's Manual

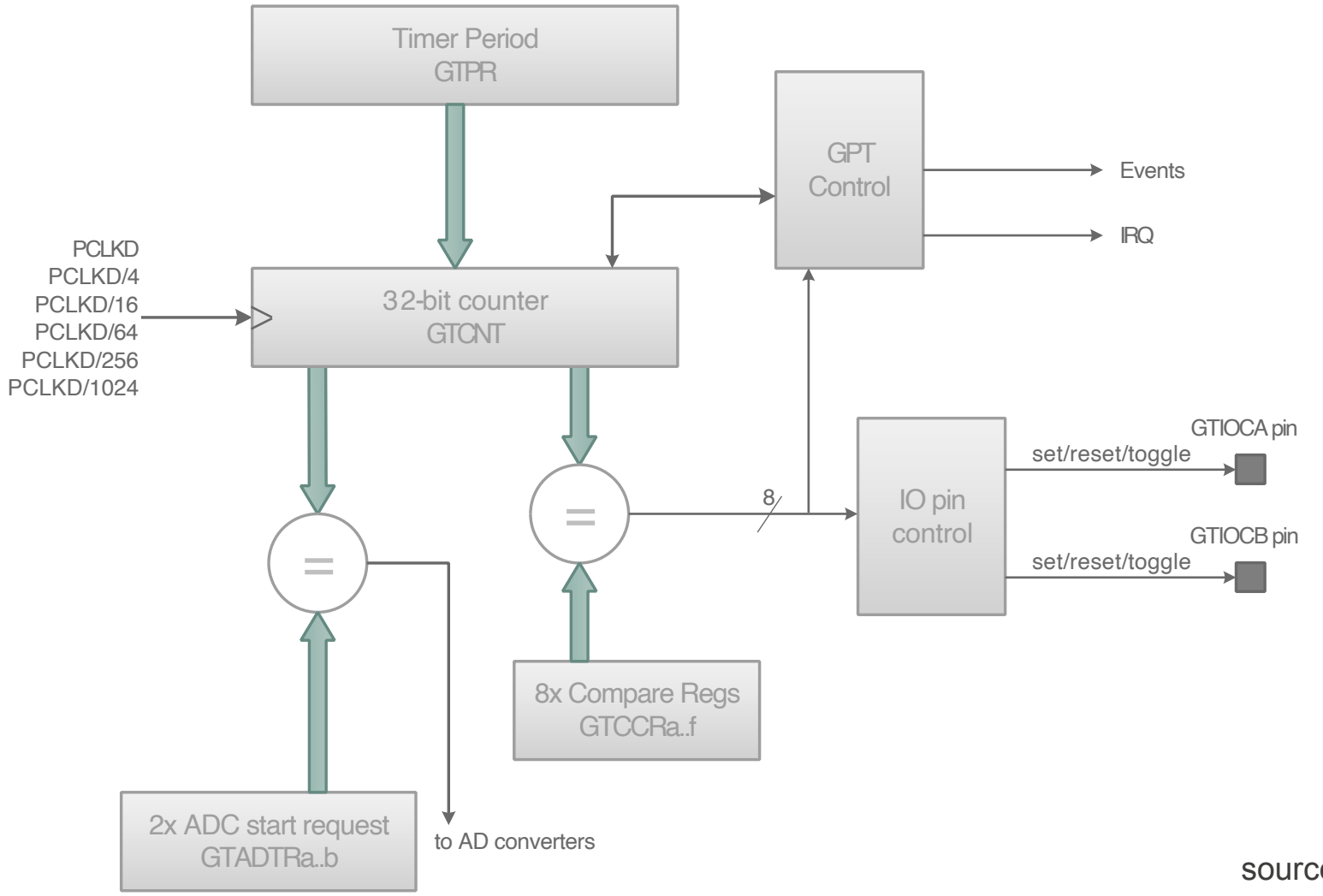
TIMER CASE STUDY 2 – S7G2 GPT

GTWP	: General PWM Timer Write-Protection Register
GTSTR	: General PWM Timer Software Start Register
GTSTP	: General PWM Timer Software Stop Register
GTCLR	: General PWM Timer Software Clear Register
GTSSR	: General PWM Timer Start Source Select Register
GTPSR	: General PWM Timer Stop Source Select Register
GTCSR	: General PWM Timer Clear Source Select Register
GTUPSR	: General PWM Timer Up Count Source Select Register
GTDNSR	: General PWM Timer Down Count Source Select Register
GTICASR	: General PWM Timer Input Capture Source Select Register A
GTICBSR	: General PWM Timer Input Capture Source Select Register B
GTCR	: General PWM Timer Control Register
GTUDDTYC	: General PWM Timer Count Direction and Duty Setting Register
GTIOR	: General PWM Timer I/O Control Register
GTINTAD	: General PWM Timer Interrupt Output Setting Register
GTST	: General PWM Timer Status Register
GTBER	: General PWM Timer Buffer Enable Register
GTITC	: General PWM Timer Interrupt and A/D Converter Start Request Skipping Setting Register
GTCNT	: General PWM Timer Counter
GTCCRA	: General PWM Timer Compare Capture Register A
GTCCRB	: General PWM Timer Compare Capture Register B
GTCCRC	: General PWM Timer Compare Capture Register C
GTCCRD	: General PWM Timer Compare Capture Register D
GTCCRE	: General PWM Timer Compare Capture Register E
GTCCRF	: General PWM Timer Compare Capture Register F
GTPR	: General PWM Timer Cycle Setting Register
GTPBR	: General PWM Timer Cycle Setting Buffer Register
GTPDBR	: General PWM Timer Cycle Setting Double-Buffer Register
GTADTRA	: General PWM Timer A/D Converter Start Request Timing Register A
GTADTBRA	: General PWM Timer A/D Converter Start Request Timing Buffer Register A
GTADTDBRA	: General PWM Timer A/D Converter Start Request Timing Double-Buffer Register A
GTADTRB	: General PWM Timer A/D Converter Start Request Timing Register B
GTADTBRB	: General PWM Timer A/D Converter Start Request Timing Buffer Register B
GTADTDBRB	: General PWM Timer A/D Converter Start Request Timing Double-Buffer Register B
GTDTCR	: General PWM Timer Dead Time Control Register
GTDVU	: General PWM Timer Dead Time Value Register U
GTDVD	: General PWM Timer Dead Time Value Register D
GTDBU	: General PWM Timer Dead Time Buffer Register U
GTDBD	: General PWM Timer Dead Time Buffer Register D
GTSOS	: General PWM Timer Output Protection Function Status Register
GTSOTR	: General PWM Timer Output Protection Function Temporary Release Register
OPSCR	: Output Phase Switching Control Register

source: Renesas S7 Series Microcontrollers User's Manual

TIMER CASE STUDY 2 – S7G2 GPT

Simplified view of a GPT



source: Authors

TIMER CASE STUDY 2 – S7G2 GPT

GPT characteristics and operation:

- The counter (GTCNT) can count up (from 0 to GTPR), count down (from GTPR to 0) or up and down (from 0 to GTPR and back to 0).
  Hence, the period is either GTPR+1 or 2x GTPR;
- Due to a prescaler, the clock source can be selected from PCLKD to PLCKD/1024;
- A GPT can be used to trigger the start of an AD conversion, registers GTADTR are used to determine the timing to command the start;
- 8 compare registers are available. On match, a number of actions can take place: set, reset, toggle an IO pin, generate an interrupt request, generate an event, ...

5.4 – GPIO

General Purpose Input/Output (GPIO) is possibly the simplest form of I/O in a system.

An input pin can be connected to a key, a push-button, or any other digital signal. By reading the pin the program can detect if it is on high or low level and take appropriate action.

Input pins can generate interrupt requests (IRQ) when a required transition or level is detected.

An output pin can be connected to an LED, to a switch (transistor, relay) or any other device to be controlled by.

GPIO CASE STUDY – S7G2 I/O PORTS

The MCU used in the lab experiments (kit SK-S7G2) is the R7FS7G27H3A01CFC. Its packaging is a 176 pins LQFP (Low-profile Quad Flat Package).
Of its 176 pins, 126 are available either for I/O or to be used by the integrated peripherals.

The I/O pins are grouped into 12 ports
(P0 to PB) each with up to 16 pins.

Port	pins		Port	pins
P0	13		P6	16
P1	16		P7	8
P2	10		P8	7
P3	16		P9	6
P4	16		PA	5
P5	11		PB	2

GPIO CASE STUDY – S7G2 I/O PORTS

source: Renesas S7 Series Microcontrollers User's Manual

I/O pins can be configured in several ways:

- inputs may have an internal pull-up;
- outputs may be open-drain;
- output current capacity may be selectable: 2, 4, 16/20 mA;
- some inputs are 5V tolerant.

Remarks:

- total current provided by the MCU is restricted to 80 mA;
- configuration capabilities vary from pin to pin, see table on the right.

Table 20.2 I/O port functions

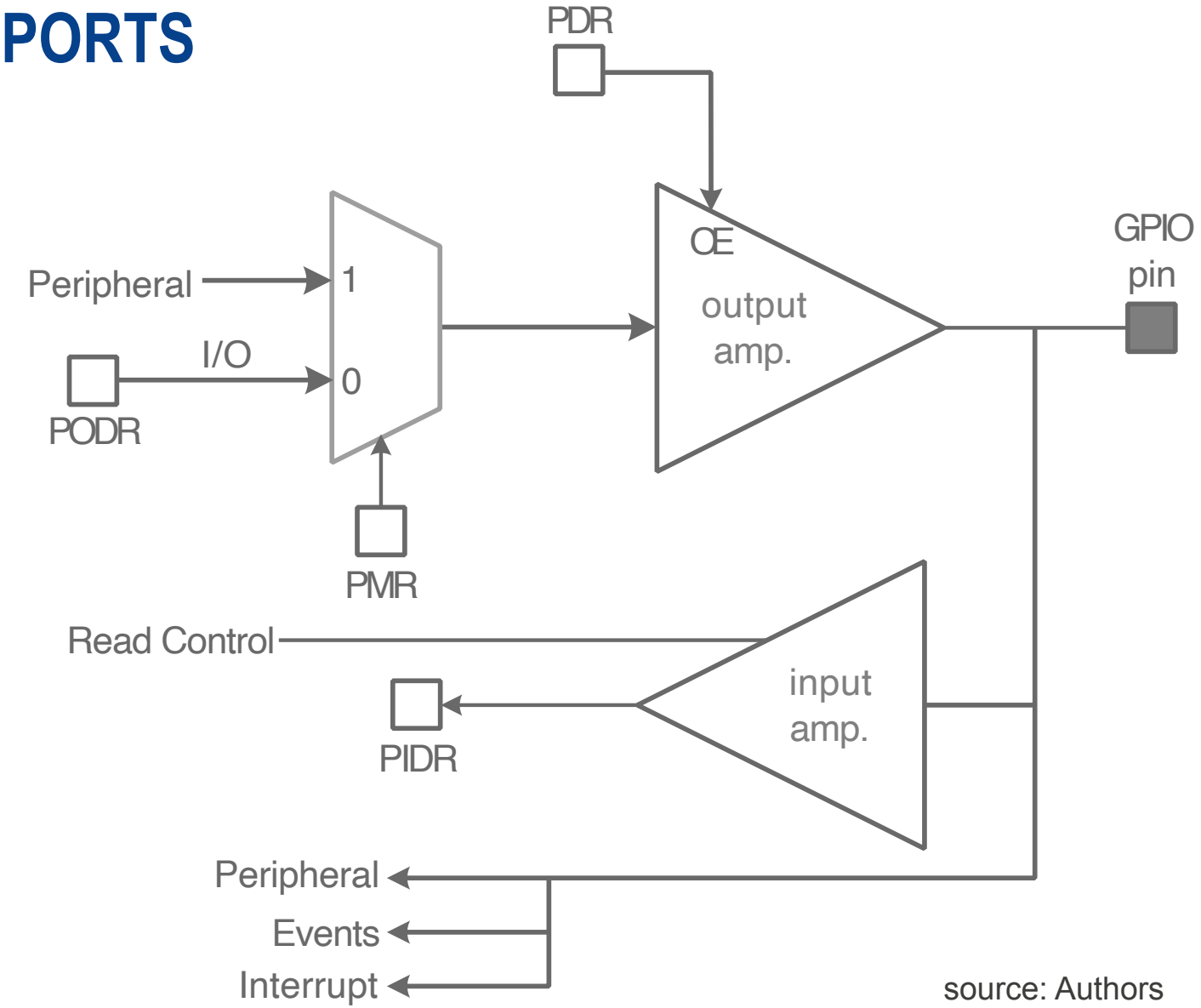
Port	Port name	Input pull-up	Open-drain output	Drive capacity switching	5-V tolerant
Port 0	P000 to P007	-	-	-	-
	P008 to P011, P014, P015	✓	✓	-	-
Port 1	P100 to P115	✓	✓	Low, middle, high	-
Port 2	P200	✓	-	-	-
	P202 to P204, P207	✓	✓	Low, middle, high	-
	P205, P206	✓	✓	Low, middle, high	✓
	P201, P212	✓	✓	Low	-
	P213	✓	✓	High	-
Port 3	P300 to P315	✓	✓	Low, middle, high	-
Port 4	P400, P401	✓	✓	Middle	✓
	P402 to P404	✓	✓	Low, middle	✓
	P405 to P406	✓	✓	Low, middle, high	-
	P407	✓	✓	Low, middle, high	✓
	P408 to P415	✓	✓	Low, middle, high	✓
Port 5	P500 to P510, P513 to P515	✓	✓	Low, middle, high	-
	P511, P512	✓	✓	Middle	✓
Port 6	P600 to P615	✓	✓	Low, middle, high	-
Port 7	P700 to P707	✓	✓	Low, middle, high	-
	P708 to P713	✓	✓	Low, middle, high	✓
Port 8	P800 to P813	✓	✓	Low, middle, high	-
Port 9	P900 to P915	✓	✓	Low, middle, high	-
Port A	PA00 to PA15	✓	✓	Low, middle, high	-
Port B	PB00, PB02 to PB07	✓	✓	Low, middle, high	-
	PB01	✓	✓	Low, middle, high	✓

rem: not all pins listed above are available on the part number R7FS7G27H3A01CFC

GPIO CASE STUDY – S7G2 I/O PORTS

Simplified block diagram

of an I/O pin



Reg	Description
PODR	Port output data
PDR	Port direction
PIDR	Port input data
PMR	Port mode: I/O vs periph.

source: Authors

GPIO CASE STUDY – S7G2 I/O PORTS

source: Renesas S7 Series Microcontrollers User's Manual

Block diagram for an I/O pin

Reg	Description		Reg	Description
PCR	Pull-up control		PODR	Port output data
PDR	Port direction		PSEL	Peripheral sel.
DSCR	Drive capabil.		PIDR	Port input data
NCODR	open-drain ctr		ISEL	interrupt enable
EOSR	evt output set		ASEL	analog select
EORR	event output reset		PMR	Port mode: I/O vs periph.
POSR	port output set		EIDR	event input data
PORR	port output reset		EOF/ EOR	event on falling/ rising edge

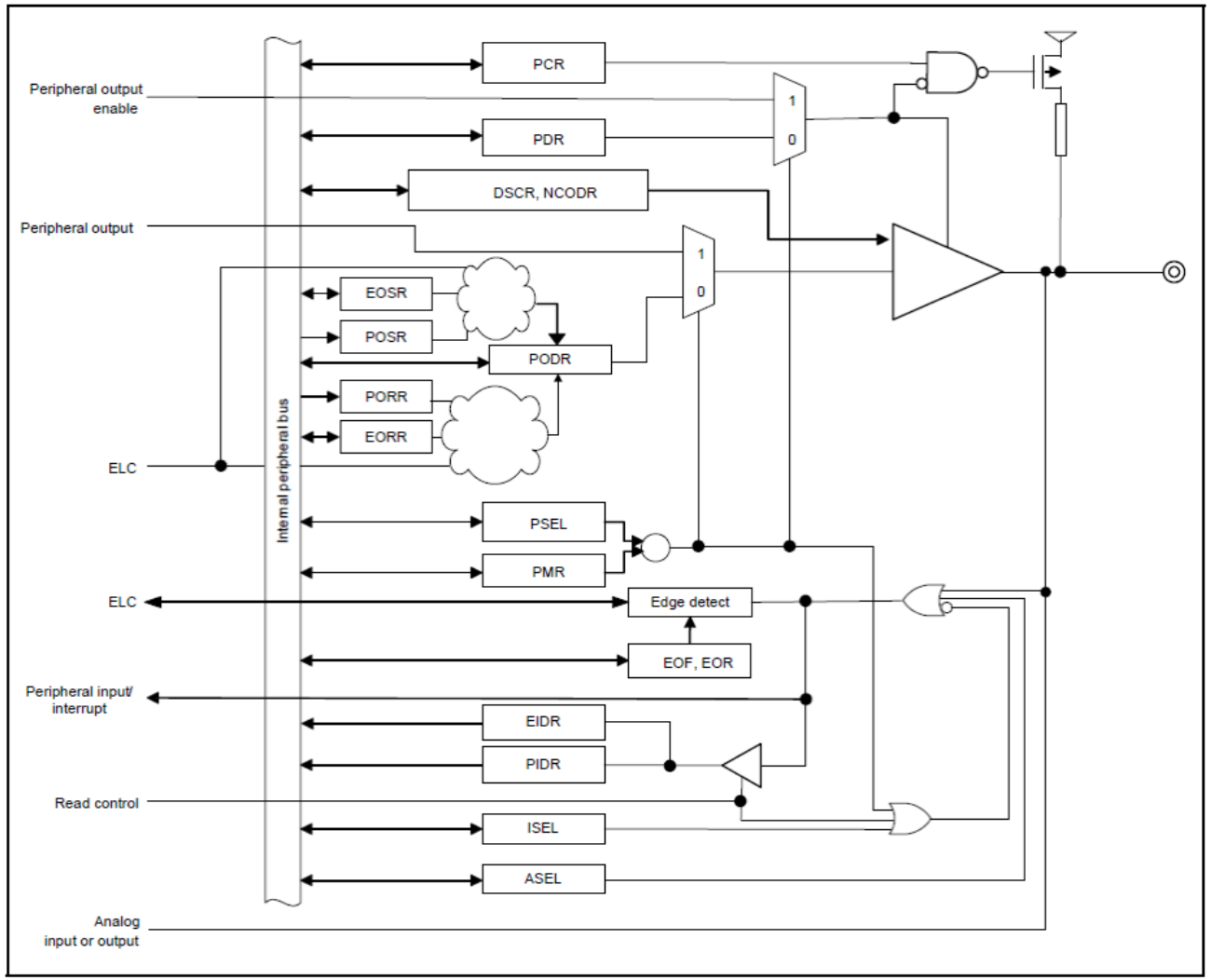


Figure 20.1 Connection diagram for I/O port registers

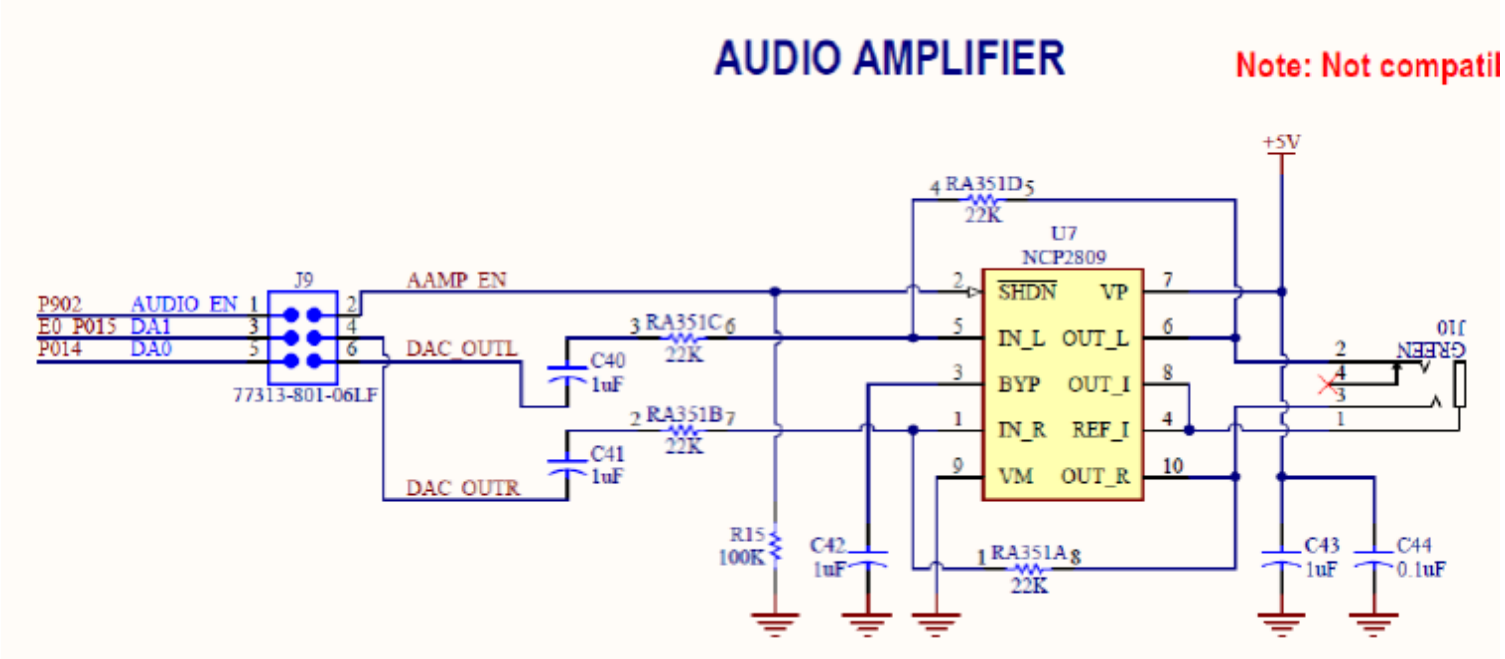
5.5 – LOW-POWER DRIVERS

When a device connected to an output pin has low input current requirements (low typically means below 1 mA) then it can be directly connected to the I/O pin.

Example: the Renesas Development Kit for S7G2 has an audio amplifier and a speaker connector.

This audio amplifier is controlled by an output pin:

- P902 is an I/O pin of the S7G2 microcontroller that controls if the amplifier is enabled or not.



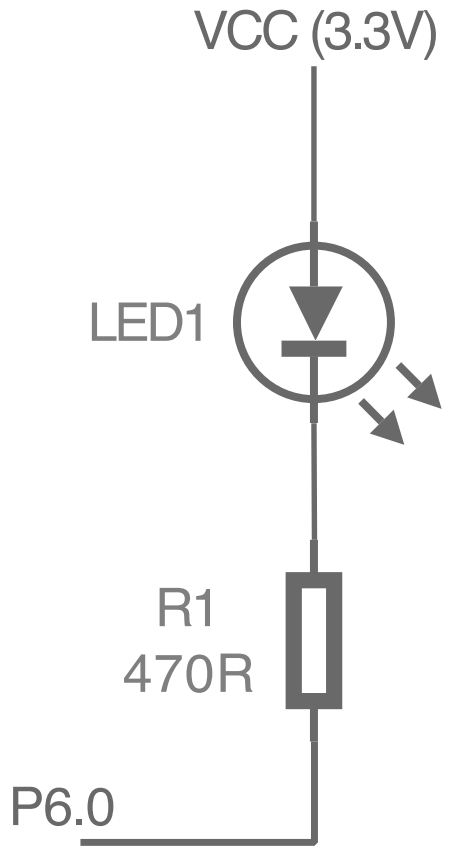
source: Renesas Development Kit S7G2 (DK-S7G2) User's Manual

LOW-POWER DRIVERS

In the case of a small LED, which typically requires current around 2 to 5 mA to light up, a limited number can still be connected directly to output pins.

The S7G2 MCU has the same source and sink capability, meaning that an output pin configured for middle drive capacity can source 4mA when its output is high and can sink 4mA when its output is low.

This symmetry is not a rule, many digital logic ICs are capable to sink much higher currents than to source them. Hence, quite often, devices such as LEDs are turned on by an output pin sinking current, i.e. with a logic level 0.



P6.0 = 0 (0V) LED is ON
P6.0 = 1 (3.3V) LED is OFF

source: Authors

5.6 – POWER DRIVERS

Loads that require higher currents than in the previous examples are controlled by output pins connected to interface circuits, such as:

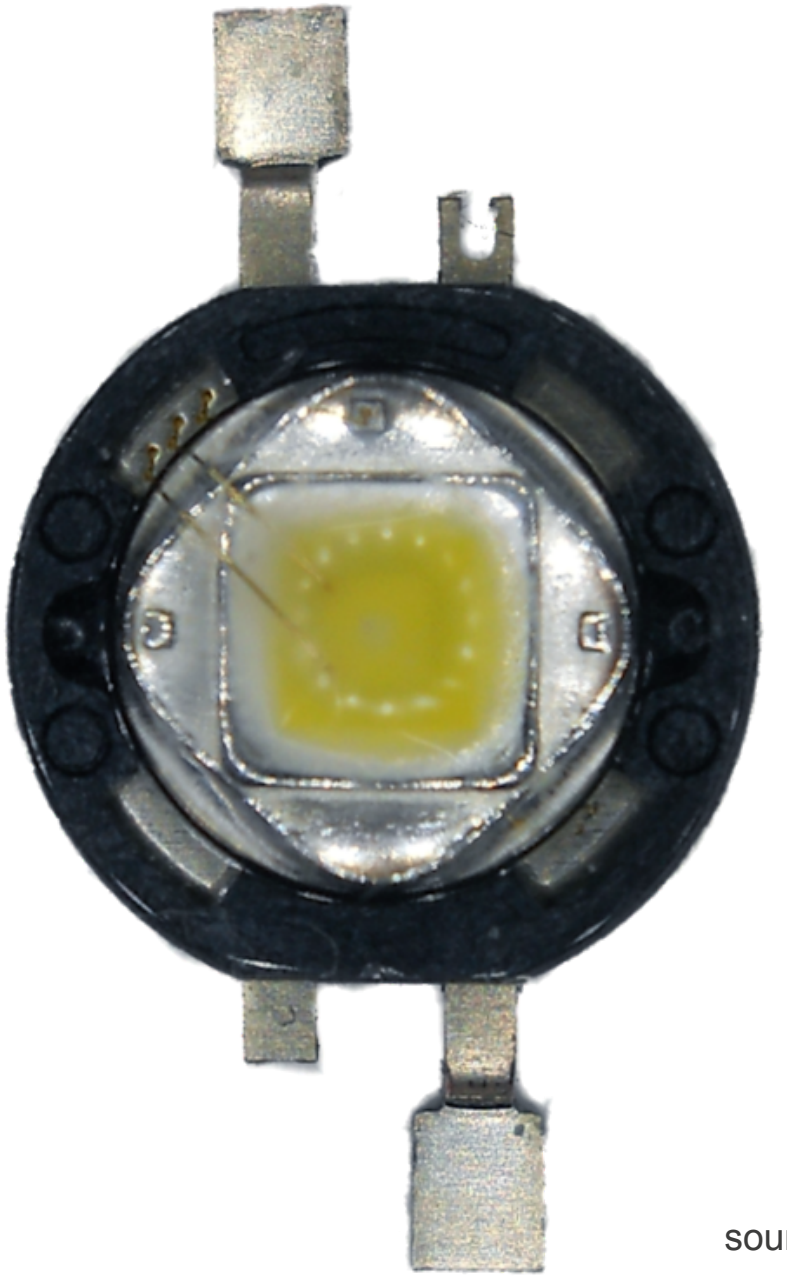
- Transistor (bipolar, MOS, or BICMOS),
- Relay,
- H-Bridge,
- Other power electronics circuits.

Such interface circuits also allow the use of much higher voltages on the load.

POWER DRIVERS

Example 1 - Power LED

Consider a Power LED requiring a forward current of 500mA with a forward voltage of 3.2V ± 0.15V (varies with ambient temperature and component sample).



source: wikimedia.org (CC)

POWER DRIVERS

Example 1 - Power LED

A low-cost solution consists of a transistor, acting as a power switch, and a series resistance, to limit the forward current.

When P6.0 is at logic level 1, Q1 is on and LED is on.

When P6.0 is at logic level 0, Q1 and LED are off.

Drawbacks:

R1 dissipates 1W when LED is on.

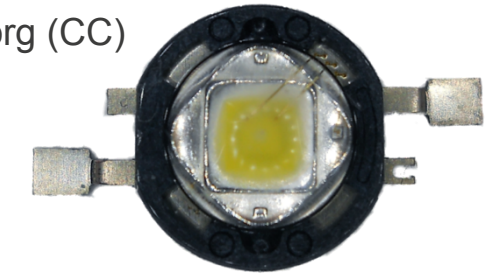
LED current, and brightness, vary with LED forward voltage.

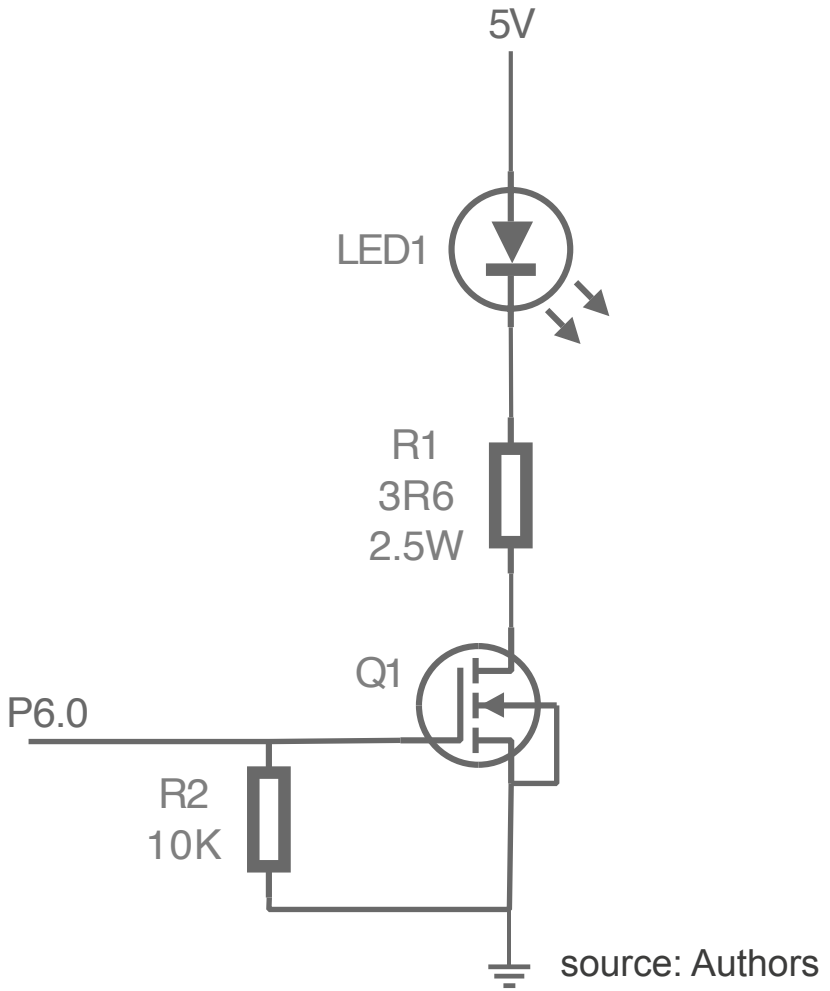
Advantages:

Load can operate from a power source with different voltage.

(in this example, MCU VDD is 3.3V and LED is powered from 5V)

source: wikimedia.org (CC)





source: Authors

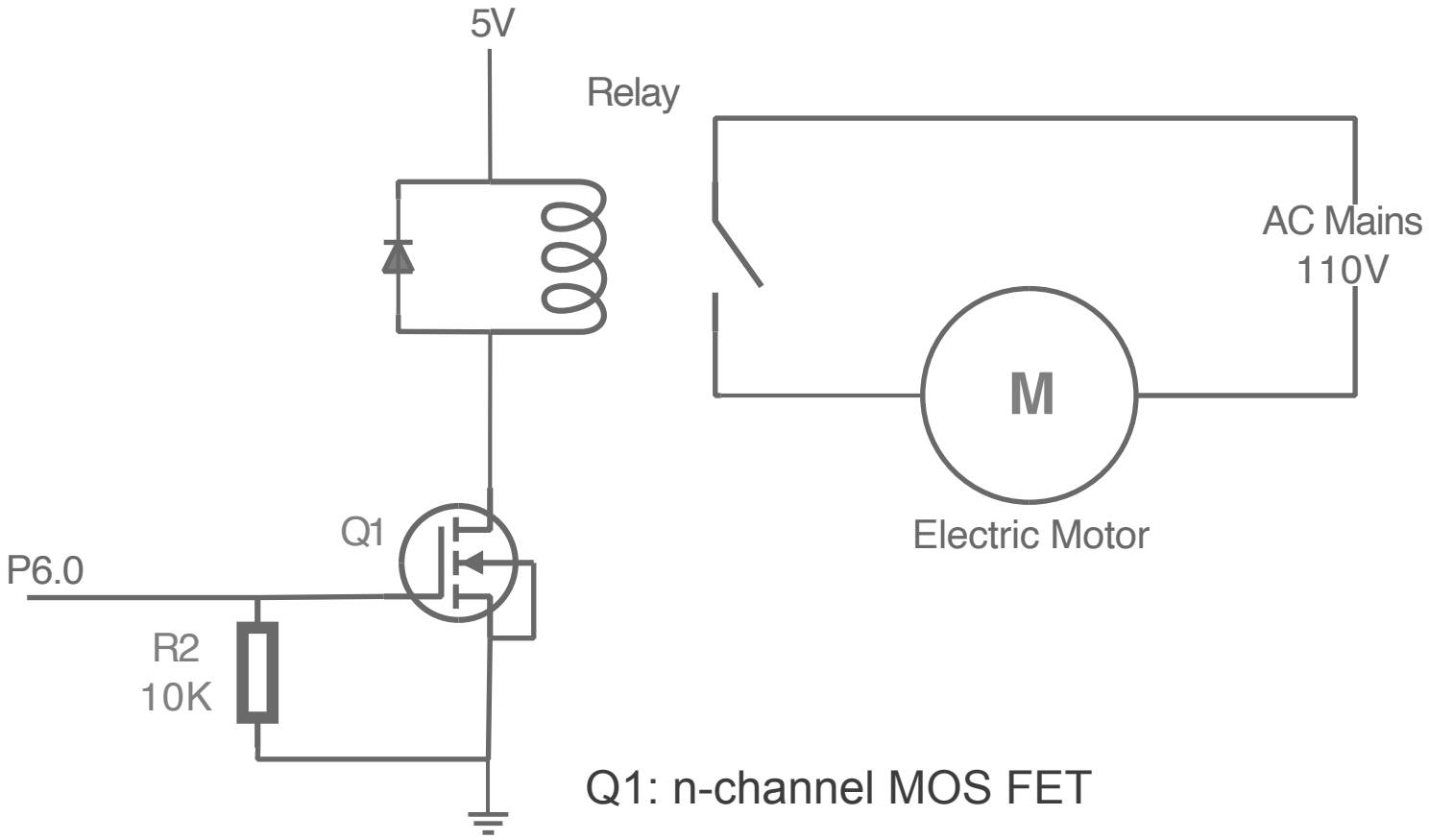
POWER DRIVERS

Example 2 - Relay

If galvanic isolation between the MCU and the load is required, a solution is the use of a relay.

When P6.0 is at logic level 1, the transistor is ON, current flows through the relay coil and the contact closes, turning the electric motor ON.
When P6.0 is at 0, the motor is off.

Since the relay coil is an inductive load to the transistor, the diode is required to avoid voltage spikes when the transistor switches off.



Q1: n-channel MOS FET

source: Authors

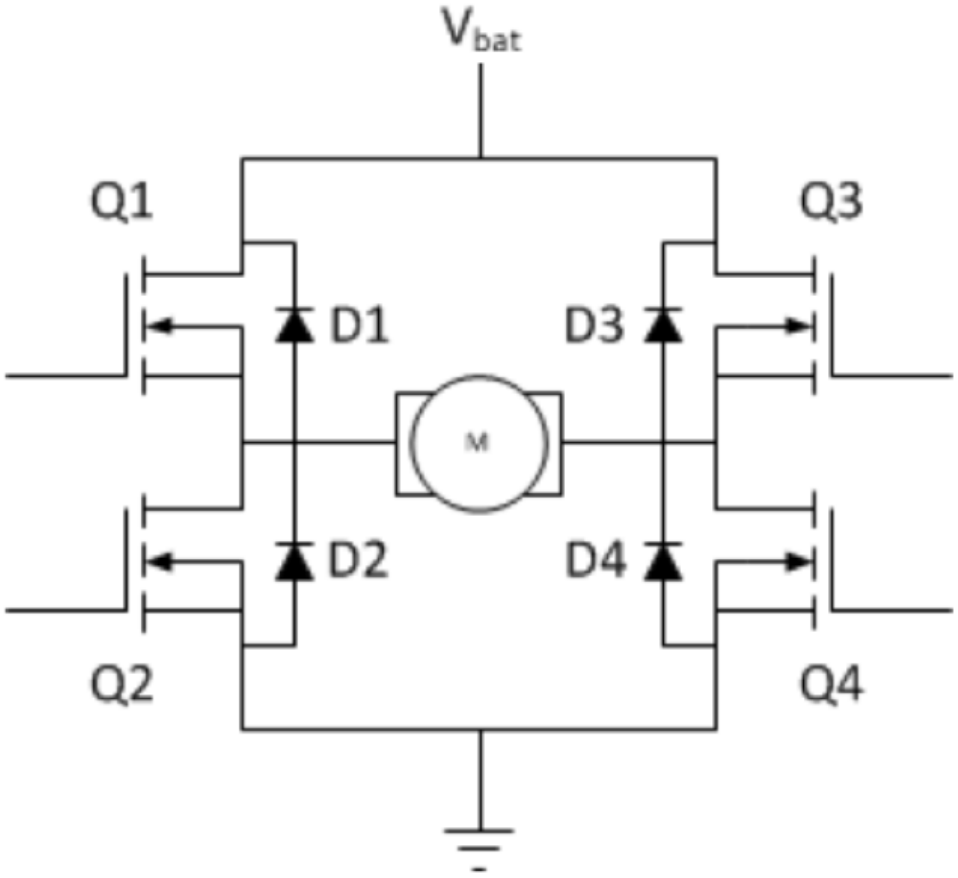
POWER DRIVERS

Example 3 - H-Bridge

An H-Bridge consists of four switches (in this case four n-mos). The load is a DC motor. When Q1 and Q4 are ON, the motor is energized. When Q2 and Q3 are ON the polarity is reversed.

Q1 and Q2 should never be ON at the same time as this would short circuit the power supply. For the same reason, Q3 and Q4 should never be ON at the same time.

Typically, the four switches are driven by PWM signals to control the current and the speed of the motor.



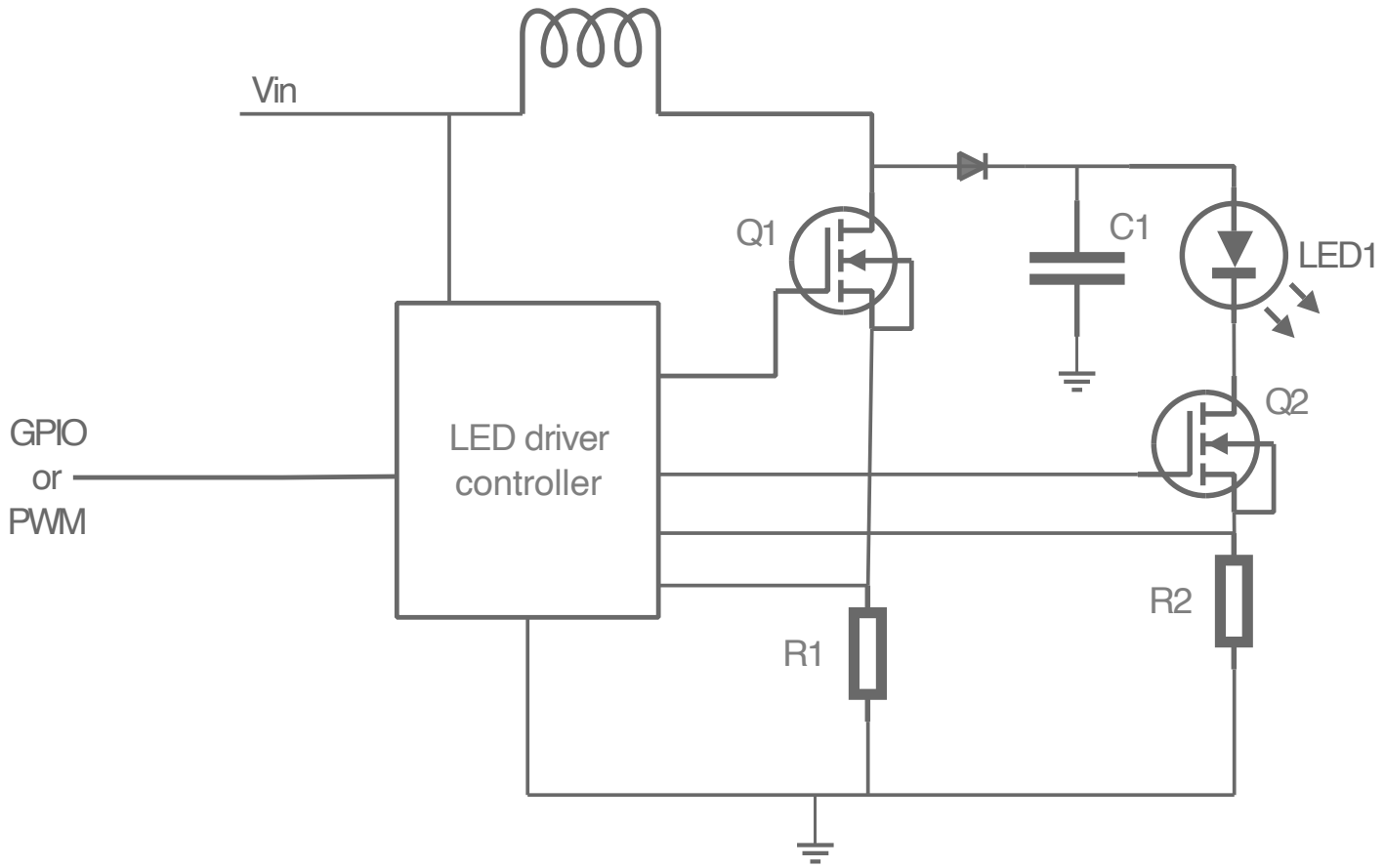
POWER DRIVERS

Example 4 -
Power LED driver using a switching power supply.

GPIO can turn LED on or OFF.

Alternatively, PWM can dimmer the LED.

When compared to Example 1, this circuit does not waste energy on a current limiting resistor, hence, it is energy efficient.



source: Authors

Renesas.com