DOI: https://doi.org/10.26565/2304-6201-2025-67-04

УДК (UDC) 621.382.002:621.381.821

Horenko Daniiel *Master student of the Institute of Computer Science and Artificial Intelligence, V.N. Karazin Kharkiv National University, Svobody Square, 4, Kharkiv, Ukraine, 61022*
e-mail: horenko2020ku11@student.karazin.ua
https://orcid.org/0009-0004-6910-4622

Kotvytskiy Albert *Candidate of Physical and Mathematical Sciences, Associate Professor, V. N. Karazin Kharkiv National University 4, Svobody Sq., Kharkiv, 61022, Ukraine*
Pavol Jozef Šafárik University in Košice, 2, Šrobárova, Kosice, 041 80, Slovak Republic
e-mail: kotvytskiy@gmail.com;
https://orcid.org/0000-0001-8283-505X

Controlling LEDC timers of the ESP32 microcontroller using registers

Relevance. This paper examines precise generation and control of pulse-width modulation (PWM) signals using the LEDC (LED PWM Controller) subsystem of the ESP32 microcontroller via direct register access. As embedded real-time systems increasingly require fine timing control in LED drivers, motor control and power electronics, standard high-level driver APIs can be insufficient. Direct register manipulation of LEDC enables more precise tuning of frequency, resolution and pulse timing, which is critical for synchronization-sensitive applications.

Objective. To analyze the capabilities of ESP32 LEDC timers when configured through direct register writes, to experimentally evaluate the accuracy and stability of generated PWM signals across representative configurations, and to provide practical recommendations for optimizing LEDC parameters in applied embedded projects.

Methods. The investigation employed low-level register programming under Espressif's ESP-IDF on an ESP32-DevKitC V4 (WROOM-32D). Time-domain characteristics of the PWM outputs were measured with a Logic Analyzer (24 MHz sampling, 8 channels). The study combined theoretical derivations of PWM frequency and period based on clock source, divider (DIV) and counter resolution (RES) with implementation of direct register sequences to configure HSTIMER0 and HS channel 0, and comparative measurements for eighteen distinct configurations covering multiple RES, DIV and DUTY values.

Results. The register-based control method enabled generation of high-frequency PWM in the MHz range with close agreement between calculated and measured values. Across tested configurations the maximum relative deviation did not exceed ±0.03% for frequency and period, and ±0.6% for pulse high-time (duty width). Increasing counter resolution improved duty-cycle granularity, while the prescaler DIV produced a linear change in PWM frequency. The experimental limitations observed at the highest frequencies are attributable to the finite sampling capability of the measurement equipment.

Conclusions. Direct register access to the LEDC allows for obtaining deterministic, high-precision PWM signals with minimal parameter update latency, making them suitable for applications in robotics, power electronics, and other systems with high synchronization requirements. Further research is recommended on the influence of alternative clock sources, low-speed LEDC modes, integration with ISR/FreeRTOS, and extending the approach to other timers and channels.

Keywords: *ATmega, ESP32, LEDC, PWM, register access, logic analyzer*

How to quote: D. V. Horenko, and A. T. Kotvytskiy, "Controlling LEDC timers of the ESP32 microcontroller using registers" *Bulletin of V. N. Karazin Kharkiv National University, series Mathematical modelling. Information technology. Automated control systems*, vol.67, pp. 45-55, 2025. https://doi.org/10.26565/2304-6201-2024-64-01

Як цитувати: Horenko D. V., and Kotvytskiy A. T., Controlling LEDC timers of the ESP32 microcontroller using registers. *Вісник Харківського національного університету імені В. Н. Каразіна, серія Математичне моделювання. Інформаційні технології. Автоматизовані системи управління.* 2025. 67. С.45-55. https://doi.org/10.26565/2304-6201-2024-64-01

Introduction

In modern technological systems, embedded computing modules are becoming increasingly important, providing intelligent process control, data acquisition, and communication between devices. They are the basis of robotic systems, automated production lines, sensor networks, and consumer IoT solutions. A high level of integration, compactness, and energy efficiency makes such systems a key

element of modern electronics [1], [2]. One of the fundamental tasks for embedded controllers remains the precise generation and control of signals, particularly pulse-width modulated signals, which control LEDs, motors, power switches, and other peripheral devices.

Traditionally, for educational and prototyping purposes, microcontrollers of the ATmega series, particularly the ATmega328P and ATmega2560, which are the foundation of the Arduino Uno and Arduino Mega platforms, are widely used. They are distinguished by their simple architecture, extensive library support, and a user-friendly programming environment. However, the use of the high-level Wiring language and the abstraction layers of the Arduino IDE significantly limits the possibilities for precise configuration of timers and pulse-width modulation [3], [4]. In specialized developments, to achieve maximum configuration flexibility, direct access to the microcontroller's registers is used, which allows changing PWM parameters with minimal delay [5].

In recent years, ESP32 microcontrollers from Espressif Systems have become widespread, combining high computational power, built-in Wi-Fi and Bluetooth interfaces, hardware timer modules, and advanced PWM signal control capabilities.

In the microcontroller platform market, the most common boards are the Arduino UNO (ATmega328P) and Arduino MEGA (ATmega2560), with official prices as of October 2025 of 29.30 EUR and 52.80 EUR, respectively, according to the Arduino Store [6], [7].

In contrast, the ESP32-DevKitC module, which is an official product of Espressif Systems, is available in the manufacturer's official store on the AliExpress platform [8] for a price of 8–15 USD, which is several times cheaper while offering significantly higher computing power and integrated wireless interfaces (Wi-Fi, Bluetooth).

Thus, for a comparable price, the user gets a dual-core 32-bit processor with a clock speed of up to 240 MHz, 16 PWM channels, and an advanced clocking system. This makes the ESP32 a cost-effective choice for developers of real-time systems and researchers in the field of precision electronics.

Software development for the ESP32 is possible both in the Arduino IDE environment and using ESP-IDF – the official SDK from Espressif Systems. However, working through the Arduino layer, which is built on the Wiring language, creates additional delays in function calls and conceals the low-level mechanisms for accessing hardware [9]. That is why for tasks related to high-frequency processes, motor control, or the study of timing characteristics, programming in C with direct writes to peripheral registers is advisable, as it allows for achieving maximum performance and precision in signal control.

Among the peripheral subsystems of the ESP32, a special place is held by the LEDC (LED PWM Controller) – a module for generating pulse-width modulated signals, capable of forming up to 16 independent PWM channels with support for high-speed (up to 40 MHz) and low-speed modes. PWM modulation is a basic tool for controlling LEDs, electric motors, audio modules, and other loads, where the stability and precision of the signal parameters determine the operational quality of the entire system:

- in LED drivers – regulating brightness without flickering;
- motor control systems – smoothness of rotation and torque precision;
- in digital audio interfaces – affecting the level of noise and harmonics;
- in synchronization generators – minimization of time fluctuations of the signal.

The use of standard APIs, particularly the ESP-IDF LEDC driver, significantly simplifies programming but does not allow for full control over the timer registers. This limits the precision of configuring the frequency, duty cycle, and the timing of the signal update. In specialized systems, such as robotic controllers or precision power control circuits, such capabilities are insufficient. In these cases, an effective solution is direct access to the LEDC registers, which allows for flexible modification of all necessary parameters, eliminating the overhead of the driver layer.

The purpose of this study is to analyze the capabilities of controlling the LEDC timers of the ESP32 microcontroller through direct access to its registers, and to experimentally evaluate the accuracy and stability of the generated PWM signals.

2 Architecture and Operating Principle of the ESP32 LEDC

In classic microcontrollers of the ATmega family (specifically, ATmega328P, ATmega2560), the implementation of pulse-width modulation (PWM) is based on linking each timer to strictly defined output channels and pins of the chip [10]. For example, Timer0 in the ATmega328P serves only the OC0A and OC0B outputs, and these pins are hardware-fixed. This approach simplifies the hardware logic but significantly limits the flexibility of channel allocation when designing complex control systems [11].

In contrast, the ESP32 microcontroller implements the principle of hardware decoupling between the timer, the channel, and the GPIO output. Each of the 16 channels of the LEDC subsystem can be independently assigned to any of the four available timers and, in turn, routed to any available pin of the microcontroller. This architecture provides exceptional flexibility when implementing complex control systems, such as multi-channel motor drivers, RGB matrices, or power management systems, where not only timing precision but also the efficient use of GPIOs is critical.

Thus, the ESP32 allows for the programmatic reconfiguration of the logical mapping between channels and timers without hardware changes, which significantly expands design possibilities. Furthermore, unlike in AVR, the LEDC features a separation between the logical control level (channel) and the periodicity generator (timer).

- The timer determines the frequency and resolution of the PWM signal.
- The channel is responsible for the duty cycle and the connection to a specific GPIO.

This allows a single timer to be used for multiple channels that will have the same frequency but different duty cycles. Unlike classic AVRs, where the pulse typically begins upon a counter reset, the LEDC architecture allows setting an arbitrary pulse start time using the special HPOINT register, providing flexible control over the phase shift between channels.

2.1 Brief Hardware Context of the ESP32

The ESP32 [12] microcontroller, developed by Espressif Systems, is a high-performance integrated platform for building embedded systems with support for wireless technologies. Its core is a dual-core Tensilica Xtensa LX6 processor, which operates at a clock frequency of up to 240 MHz. The processing cores feature an advanced power-saving system, allowing for dynamic performance adjustments based on the application's needs.

Fig. 1 ESP32 model Wroom-32D
Рис. 1 ESP32 моделі Wroom-32D

The ESP32 is distinguished by a rich set of peripheral modules, including:

- Communication interfaces (UART, SPI, I²C, I²S, CAN),
- Analog-to-digital (ADC) and digital-to-analog (DAC) conversion blocks,
- General-purpose hardware timers,
- Cryptographic accelerators,
- Specialized modules for signal control.

2.2 Structure of the LEDC Subsystem

LEDC is a hardware PWM controller with 16 independent channels that can generate PWM signals of various frequencies and duty cycles. Key features of the LEDC include:

– Support for high-speed and low-speed modes
– High precision (fractional frequency division)
– Fading (automatic duty cycle change)

LEDC has two logical classes of channels:

High-Speed (HS) — 8 channels designed for high-speed PWM generation (using timers HSTIMER0..3);

Low-Speed (LS) — 8 channels oriented towards low-frequency or power-saving modes (using timers LSTIMER0..3).

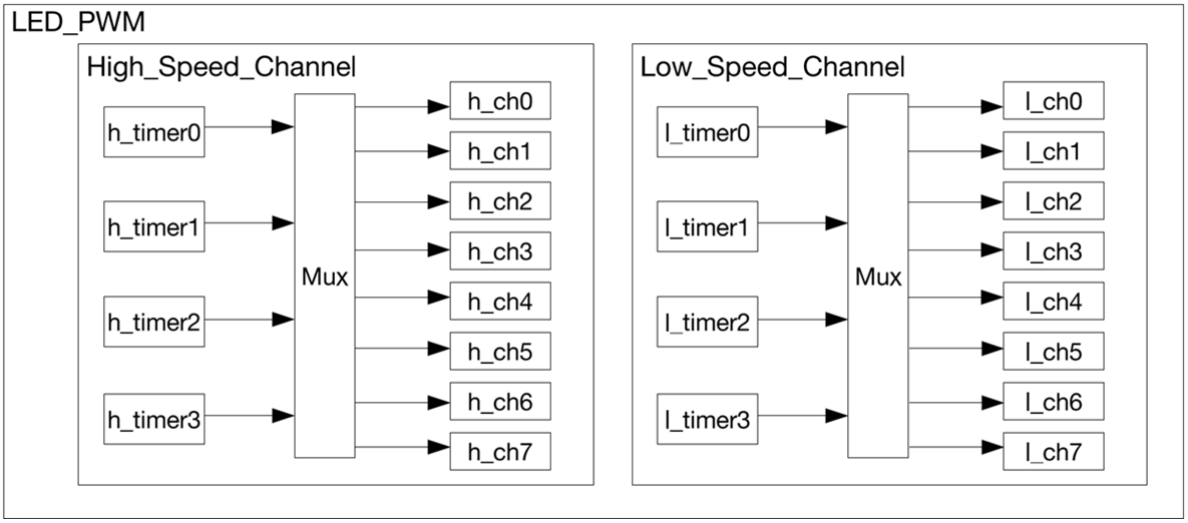

Fig. 2 LED_PWM architecture
Рис. 2 LED_PWM архітектура

As you can see, each output channel (signal), for example h_chn, can operate from any HSTIMERx timer.

2.3 LEDC Registers

The LEDC subsystem of the ESP32 microcontroller manages the pulse-width modulation channels through a set of specialized registers. It is the manipulation at this register level that provides maximum flexibility in configuring signal parameters and minimizing delays.

Let's examine the structure of the high-speed timer and channel.

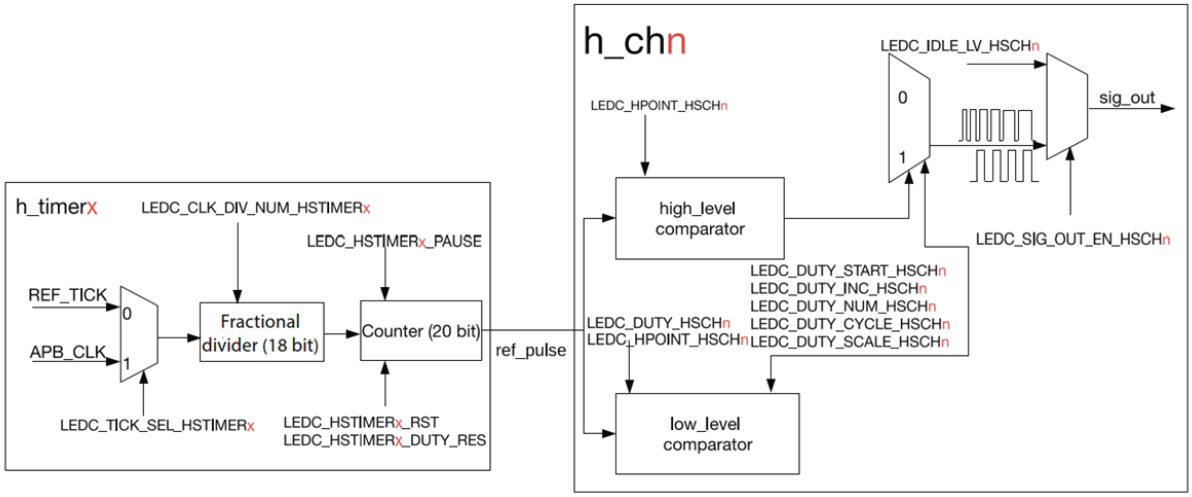

Fig. 3 LED_PWM High-Speed Channel Diagram
Рис. 3 LED_PWM Діаграма високошвидкісного каналу

From this structure, it is clear that the following need to be configured:

1. The h_timerx timer,
2. The h_chn channel,

3. The link between the channel and the timer.

The LEDC_HSTIMERx_CONF_REG register is responsible for configuring the parameters of the high-speed timer and has the following structure:

(reserved)	LEDC_TICK_SEL_HSTIMERx	EDC_HSTIMERx_RST	LEDC_HSTIMERx_PAUSE	LEDC_CLK_DIV_NUM_HSTIMERx		LEDC_HSTIMERx_DUTY_RES
				10 bits of the integer part	8 bits of fractional part	
31 26	25	24	23	22 21 20 19 18 17 16 15 14 13	12 11 10 9 8 7 6 5	4 3 2 1 0
0 0	0	1	0			

Fig. 4 Structure of the LEDC_HSTIMERx_CONF_REG register
Рис. 4 Структура регістру LEDC_HSTIMERx_CONF_REG

LEDC_TICK_SEL_HSTIMERx (bit 25) This bit is used to select the clock source for high-speed timer x, either APB_CLK or REF_TICK.

1: APB_CLK (80 MHz) This is the main peripheral bus of the ESP32. Selecting APB_CLK is typical for tasks requiring the generation of high-frequency PWM signals.

0: REF_TICK (1 MHz) This is a stable but slower reference clock source. REF_TICK is ideal for low-frequency applications.

LEDC_HSTIMERx_RST (bit 24) This bit is used to reset high-speed timer x. The counter value will be "zero" after the reset.

LEDC_HSTIMERx_PAUSE (bit 23) This bit is used to pause the counter in high-speed timer x.

LEDC_CLK_DIV_NUM_HSTIMERx (bits 22–5) This field is used to configure the division factor for the clock divider in high-speed timer x.

The upper 10 bits (22–13): Define the integer part of the divider.

The lower 8 bits (12–5): Define the fractional part of the divider (with a precision of 1/256).

LEDC_HSTIMERx_DUTY_RES (bits 4–0) This field is used to define the bit-width of the counter (from 1 to 20 bits), which is the number of steps in the PWM period for high-speed timer x.

The choice of bit resolution is a trade-off: higher resolution provides smoother control (e.g., of an LED's brightness) but lowers the maximum possible PWM frequency, and vice versa. This relationship is clearly demonstrated by formula (2.1), which integrates all the parameters discussed.

The PWM frequency is calculated by the formula:

$$f_{PWM} = \frac{f_{CLK}}{DIV \cdot 2^{RES}} \quad (2.1)$$

where f_{CRK} – is the clock source (APB_CLK або REF_TICK),

DIV – is the divider,

RES – is the counter resolution.

After configuring the HSTIMER0 timer, the output channel must be configured. The parameters for an individual channel of a high-speed timer x are defined by the LEDC_HSCHn_CONF0_REG register.

(reserved)	LEDC_IDLE_LV_HSCHn	LEDC_SIG_OUT_EN_HSCHn	LEDC_TIMER_SEL_HSCHn
31 4	3	2	1 0
0	0	0	0 0

Fig. 5 Structure of the LEDC_HSCHn_CONF0_REG register
Рис. 5 Структура регістру LEDC HSCHn_CONF0_REG

The register has the following fields:

LEDC_IDLE_LV_HSCHn (bit 3) – determines the signal level on the output when the channel is inactive.

0 — The output is LOW (0 V).

1 — The output is HIGH (3.3 V).

LEDC_SIG_OUT_EN_HSCHn (bit 2) — enables the signal output to the GPIO.

0 — Disables the PWM output (regardless of the timer's operation).

1 — Enables the PWM output (if the timer is running).

LEDC_TIMER_SEL_HSCHn (bits 0-1) — determines which timer controls the PWM channel. The ESP32 has 4 high-speed timers (HSTIMER0 – HSTIMER3).

To describe the relationship between the channel and the timer, it is first necessary to understand how PWM works. The signal is generated as follows:

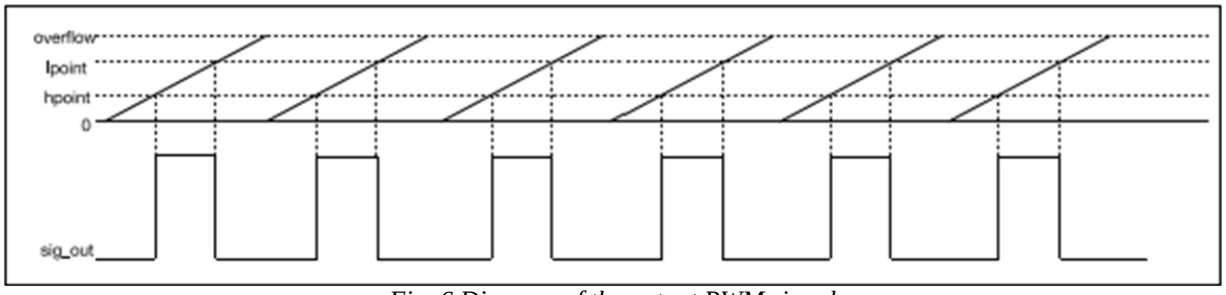

Fig. 6 Diagram of the output PWM signal
Рис. 6 Діаграма вихідного ШІМ сигналу

Each PWM channel receives a 20-bit value from the counter associated with the selected high-speed timer. This value is compared with two registers to generate the signal:

1. LEDC_HPOINT_HSCHn — when the counter reaches this value, the PWM output signal goes HIGH.
2. LEDC_HPOINT_HSCHn + LEDC_DUTY_HSCHn[24:4] — when the counter reaches this sum, the PWM signal returns to LOW.

Thus, HPOINT defines the start time of the pulse, and DUTY sets the duration of the HIGH level, forming the desired duty cycle.

To set the HPOINT value, the LEDC_HSCHn_HPOINT_REG register is used.

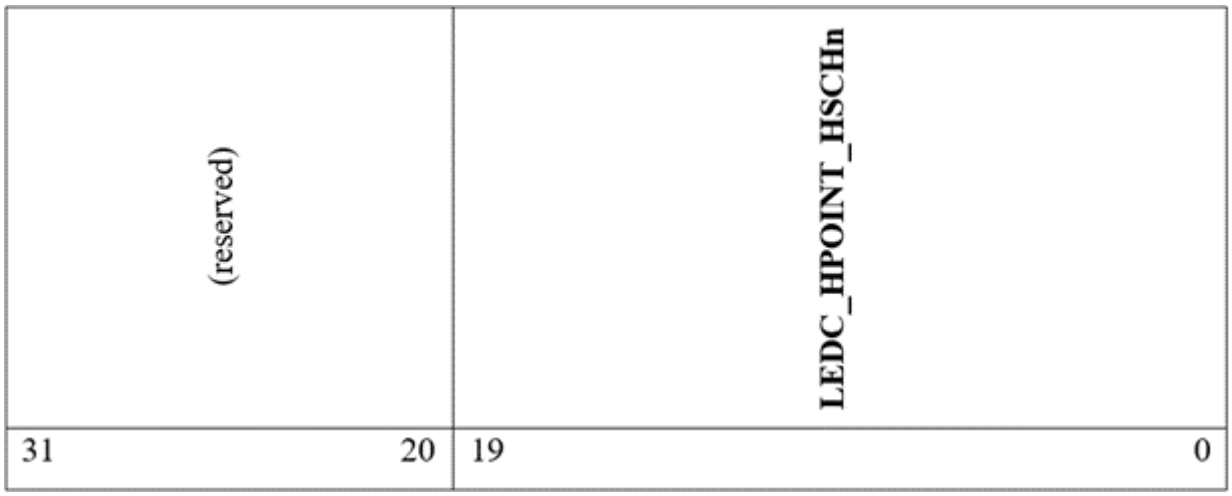

Fig. 7 Structure of the LEDC_HSCHn_HPOINT_REG register
Рис. 7 Структура регістру LEDC_HSCHn_HPOINT_REG

The LEDC_HPOINT_HSCHn field (bits 19 - 0) defines the condition under which the output signal will switch to 1 (HIGH).

This happens when the value of the timer counter, LEDC_HSTIMERx_CNT in the LEDC_HSTIMERx_VALUE_REG register, matches the value of LEDC_HPOINT_HSCHn in the LEDC_HSCHn_HPOINT_REG register. At this moment, a logical one (HIGH) is set on the corresponding PWM output.

To set the DUTY value, the LEDC_HSCHn_DUTY_REG register is used.

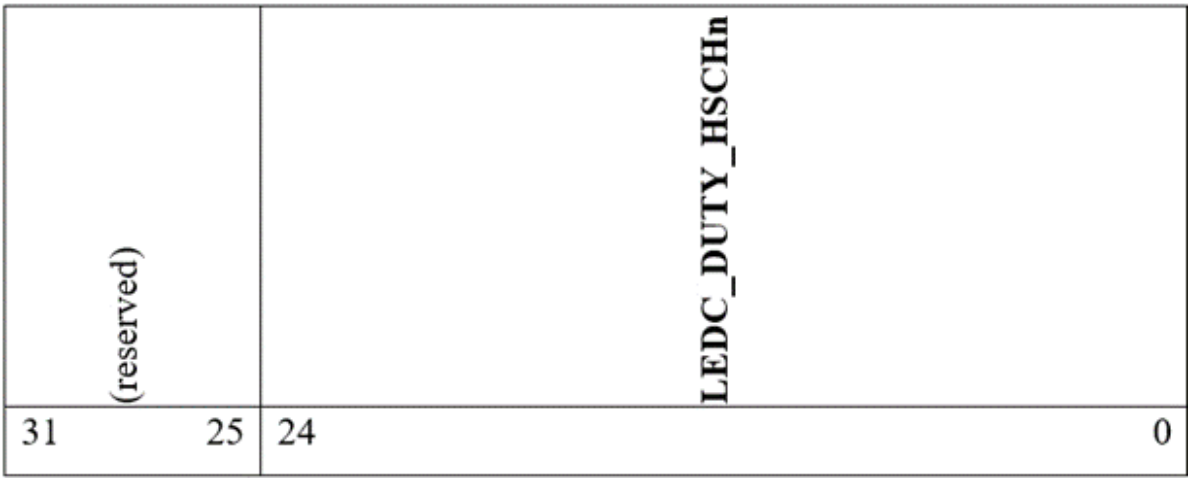

Fig. 8 Structure of the LEDC_HSCHn_DUTY_REG register
Рис. 8 Структура регістру LEDC_HSCHn_DUTY_REG

LEDC_DUTY_HSCHn (bits 24 - 0) defines how long the signal remains high (HIGH) during one PWM period.

When the counter of the hstimerx timer, which is linked to channel n, reaches the value LEDC_LPOINT_HSCHn, the PWM output signal is set to LOW (0).

The duration of the signal's high state is measured in timer ticks. The value of LEDC_LPOINT_HSCHn is calculated by the formula:

$$LEDC\_LPOINT\_HSCHn = LEDC\_HPOINT\_HSCHn1 + LEDC\_DUTY\_HSCHn2 \quad (2.2)$$

or

$$LEDC\_LPOINT\_HSCHn = LEDC\_HPOINT\_HSCHn1 + LEDC\_DUTY\_HSCHn2 + 1 \quad (2.3)$$

depending on the settings. The key point is that the lower 4 bits of the DUTY field are not used

The program implements the enabling of the LEDC clock, the configuration of the HSTIMER0 timer and the HSCH0 channel, as well as outputting the signal to GPIO2:

3 Testing Methodology

3.1 General Methodology

For this research, the ESP32-DevKitC V4 hardware platform, based on the ESP32-WROOM-32D module, was utilized. A Logic Analyzer 24 MHz 8CH served as the measurement tool, allowing for the examination of the timing parameters of pulse signals. Additionally, software control of the ESP32 peripheral clock signal was implemented through the DPORT module, which handles clock enabling/disabling for the LEDC [13].

Timer management was performed via direct access to the microcontroller's registers, using the ESP-IDF libraries:

```
1  #include "soc/ledc_reg.h"
2  #include "soc/io_mux_reg.h"
3  #include "soc/gpio_sig_map.h"  // For LEDC_HS_SIG_OUT0_IDX
4  #include "soc/gpio_reg.h"
5  #include "soc/soc.h"
6  #include "soc/dport_reg.h" // Clock signal control
7  #include "soc/dport_access.h"  //For DPORT_REG_WRITE() and DPORT_REG_READ()
```

Fig. 9 ESP-IDF libraries
Рис. 9 Використані бібліотеки ESP-IDF

The program implements the enabling of the LEDC clock, the configuration of the HSTIMER0 timer and the HSCH0 channel, as well as outputting the signal to GPIO2:

```
11     // Enable LEDC clocking
12     DPORT_REG_WRITE(DPORT_PERIP_CLK_EN_REG, DPORT_REG_READ(DPORT_PERIP_CLK_EN_REG) | DPORT_LEDC_CLK_EN);
13     DPORT_REG_WRITE(DPORT_PERIP_RST_EN_REG, DPORT_REG_READ(DPORT_PERIP_RST_EN_REG) & ~DPORT_LEDC_RST);
14
15     //Connect the LEDC_HS_SIG_OUT0 output signal to GPIO2
16     REG_WRITE(GPIO_FUNC2_OUT_SEL_CFG_REG, LEDC_HS_SIG_OUT0_IDX);//LEDC_HS_SIG_OUT0_IDX = 71 = 0x47
17     REG_WRITE(GPIO_ENABLE_W1TC_REG, (1 << 2));
18
19     //(1<<25) on 80 MHz; (4<<13) integer divider = 4; (6) set PWM to 6 bits
20     REG_WRITE (LEDC_HSTIMER0_CONF_REG, ((1<<25)|(2<<13)|(5)));
21
22     //(1<<2) = 4 sets the ENable enable bit; bits 0 and 1 equal results -> use htimer0
23     REG_WRITE (LEDC_HSCH0_CONF0_REG, (1<<2));
24
25     REG_WRITE (LEDC_HSCH0_HPOINT_REG, 0); // the initial phase is zero
26     REG_WRITE (LEDC_HSCH0_DUTY_REG, (10<<4)); //PWM parameter 10 of 64
27
28     //It is very important to start the timer by writing 0 to the pause bit
29     REG_WRITE(LEDC_HSTIMER0_CONF_REG,REG_READ(LEDC_HSTIMER0_CONF_REG)&~(1<<23));
30     REG_WRITE (LEDC_HSCH0_CONF1_REG, (1 << 31)); //Start updating the duty cycle even though we don't use automatic fade
```

Fig. 10 Main program
Рис. 10 Основна програма

To adjust the configurations, line #19, LEDC_HSTIMER0_CONF_REG, was modified to set the clock source, divider (DIV), and counter resolution (RES). The DUTY value was set in the LEDC_HSCH0_DUTY_REG register on line 25.

During the experiments, the following PWM signal parameters were determined:

- The signal frequency, calculated by formula (2.1), with $f_{CLK} = 80\ MHz$.
- The signal period was calculated by the formula:

$$T_{PWM} = \frac{1}{f_{PWM}} \tag{3.1}$$

- The duration of the high level:

$$t_H = \frac{DUTY}{2^{RES}} * T_{PWM} \tag{3.2}$$

where $DUTY$ is the duty cycle value,
RES is the counter resolution.

3.2 Example of Configuration Analysis

To illustrate the methodology in more detail, let's consider configuration 9 with the following parameters:

RES = 5 bits, **DIV** = 2, **DUTY** = 10

In this configuration, the clock frequency of the LEDC subsystem is $f_{CLK} = 80\ MHz$.. The theoretical frequency is calculated using formula (2.1) and is:

$$f_{PWM} = \frac{f_{CLK}}{DIV \cdot 2^{RES}} = \frac{80 * 10^6 \text{Гц}}{2 * 2^5} = 1.25\ MHz.$$

The signal period is calculated by formula (3.1), and is:

$$T_{PWM} = \frac{1}{f_{PWM}} = \frac{1}{1.25 * 10^6} = 0.8\ \mu s.$$

The duration of the high level is calculated by formula (3.2), and is:

$$t_H = \frac{DUTY}{2^{RES}} * T_{PWM} = \frac{10}{2^5} * 0.8 * 10^{-6} = 0.25\ \mu s$$

Practical results were obtained using a logic analyzer. [14].

Fig. 11 Logic analyzer readings
Рис. 11 Покази логічного аналізатора

The average values of the indicators were calculated over a signal duration of 0.5 seconds and entered into the table.

Table. 1 Configuration 9 Calculations
Табл. 1 Розрахунки конфігурації 9

Configuration 9			
	Frequency, Hz	Period, s	t_H, s
Theoretical	1250000	0.0000008	0.00000025
Practical	1249593.227	0.00000080026	0.0000002495
Error relative to theory	-0.03%	0.03%	-0.18%

The practical results, obtained from the logic analyzer, showed a frequency of 1.2496 MHz, a period of 0.80026 µs, and a high-level duration of 0.2495 µs.

The relative error does not exceed -0.03% for the frequency, 0.03% for the period, and ±0.18% for t_H, which indicates the high accuracy of the implemented register-level control.

4 Results and Analysis

During the experiments, 18 configurations of the LEDC timers were tested with various parameters of resolution (RES = 4, 5, 6 bits), divider (DIV = 1, 2, 4), and duty cycle (DUTY = 10, 33%, 50%). For each configuration, theoretical values of frequency, period, and high-level signal duration were determined, and practical measurements were performed using a logic analyzer. The obtained data demonstrated a high correspondence between the calculations and the experiment, with the maximum error not exceeding ±0.03% for the frequency and period, and ±0.6% for the high-level signal duration.

Analysis of the results showed that as the resolution (RES) increases, the precision of the duty cycle adjustment improves, whereas the PWM frequency is predominantly determined by the value of the divider (DIV).

An increase in DIV inversely proportionally reduces the signal frequency and linearly increases its period. The DUTY parameter directly affects the duration of the high level, t_H, and it was for this indicator that the largest relative deviations were recorded. However, these deviations remain within an acceptable margin of error for practical applications.

5 Conclusions

This work demonstrates the capabilities of direct software control over the registers of the ESP32's LEDC subsystem to generate ultra-high-speed PWM signals. The research confirmed that the signal frequency is inversely proportional to the prescaler value. At the same time, the ratio of the pulse duration

to the period (the duty cycle) remains constant if the DUTY value and the resolution are not changed. Direct register access significantly reduces delays in updating the duty cycle, ensuring more deterministic and stable signal generation, which is crucial for tasks with high requirements for synchronization and error minimization.

The results of this work are of practical significance for embedded systems that require high-speed and precisely regulated PWM signals, particularly in robotics and power electronics, and they also deepen the understanding of the hardware limits of PWM in the ESP32. Further research could be directed towards the application of direct register writes for other LEDC timers and channels, analyzing the impact of different clock sources and low-speed modes on signal quality and power consumption, as well as on the interaction with FreeRTOS cores and ISRs for the synchronous control of multiple channels.

In summary, this work investigated the LEDC architecture, developed an algorithm for the direct configuration of the prescaler, counter resolution, and duty cycle via registers, and conducted a comparative analysis of theoretical and practical calculations which showed a high degree of correspondence. Thus, the scientific problem of controlling LEDC timers has been solved through the development and experimental confirmation of a new approach, which opens up prospects for further optimization and expansion of the ESP32's capabilities in high-speed PWM control.

Funding

This research was supported by the EU NextGenerationEU through the Recovery and Resilience Plan for Slovakia under project No 09I03-03-V01-00119.

REFERENCES

1. Valvano J.W., Yerraballi R. (2014). Embedded Systems — Shape the World: A Cyber-Physical Systems Approach [e-book]. Austin, TX: The University of Texas at Austin. Available at: https://users.ece.utexas.edu/~valvano/Volume1/E-Book/
2. Barr M. (1999). Programming Embedded Systems in C and C++. O'Reilly. 174 p. Available at: https://archive.org/details/programmingembed0000barr
3. Microchip Technology Inc. (2015). ATmega328P — 8-bit AVR Microcontroller with 32K Bytes In-System Programmable Flash. Datasheet. [Electronic resource]. Available at: https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf (accessed: 22.09.2025).
4. Microchip Technology Inc. (2014). ATmega640/1280/1281/2560/2561 — 8-bit AVR Microcontroller. Datasheet. [Electronic resource]. Available at: https://ww1.microchip.com/downloads/en/devicedoc/atmel-2549-8-bit-avr-microcontroller-atmega640-1280-1281-2560-2561_datasheet.pdf (accessed: 22.09.2025).
5. Kotvytskyi, A. T. (2024). Intellectual capital as a basis for innovative development: robotic systems. In Monographic series «European Science» (Book 28, Part 1). Karlsruhe: ScientificWorld-NetAkhatAV. 168 p. ISBN 978-3-98924-041-4. DOI: 10.30890/2709-2313.2024-28-01. Available at: https://desymp.promonograph.org/index.php/sge/issue/view/sge28-01/sge28-01
6. Arduino Store (2025). Arduino UNO Rev3 – Official Product Page. Arduino AG, Italy. [Electronic resource]. Available at: https://store.arduino.cc/collections/uno (accessed: 07.10.2025).
7. Arduino Store (2025). Arduino MEGA 2560 Rev3 – Official Product Page. Arduino AG, Italy. [Electronic resource]. Available at: https://store.arduino.cc/collections/giga (accessed: 07.10.2025).
8. Espressif Systems Official Store (2025). Official Manufacturer Page on AliExpress. Espressif Systems. [Electronic resource]. Available at: https://www.aliexpress.com/store/1100220184 (accessed: 07.10.2025).
9. Arduino (n.d.). Getting Started with Arduino — official documentation (Arduino Docs). [Electronic resource]. Available at: https://docs.arduino.cc/learn/starting-guide/getting-started-arduino/ (accessed: 01.10.2025).
10. All About Circuits (2021). Pulse-width Modulation (PWM) Timers in Microcontrollers. [Electronic resource]. Available at: https://www.allaboutcircuits.com/technical-articles/introduction-to-microcontroller-timers-pwm-timers/ (accessed: 01.10.2025).

11. University of Washington (2010). Lecture 7: ATmega328 Timers and Interrupts (Course CSE P567: Embedded Systems). Seattle: University of Washington. 32 p. Available at: https://courses.cs.washington.edu/courses/csep567/10wi/lectures/Lecture7.pdf
12. Espressif Systems (2020). Technical Reference Manual for ESP32. Version 5.5. 661 p. [Electronic resource]. Available at: https://www.espressif.com/sites/default/files/documentation/esp32_technical_reference_manual_en.pdf (accessed: 01.10.2025).
13. Espressif Systems (2024). LED Control (LEDC) – Programming Guide for ESP32. ESP-IDF v5.5.1. [Electronic resource]. Available at: https://docs.espressif.com/projects/esp-idf/en/stable/esp32/api-reference/peripherals/ledc.html (accessed: 01.10.2025).
14. Saleae Support (2025). Using Logic. [Electronic resource]. Available at: https://support.saleae.com/user-guide/using-logic (accessed: 01.10.2025).

Горенко Даніель Васильович *Магістр ННІ комп'ютерних наук та штучного інтелекту, Харківський національний університет імені В.Н. Каразіна, майдан Свободи, 4, Харків, Україна, 61022*

Котвицький Альберт Тадеушевич *Кандидат фізико-математичних наук, доцент, Харківський національний університет імені В.Н. Каразіна, майдан Свободи, 4, Харків, Україна, 61022*
Pavol Jozef Šafárik University in Košice, 2, Šrobárova, Kosice, 041 80, Slovak Republic

Керування LEDC таймерами мікроконтролера ESP32 за допомогою регістрів

Актуальність. У статті розглядаються питання точного формування та керування широтно-імпульсними (ШІМ) сигналами на базі підсистеми LEDC мікроконтролера ESP32 шляхом прямого доступу до регістрів. Через зростаючі вимоги до точності таймінгу у світлодіодних драйверах, керуванні двигунами та силовій електроніці, а також обмеження високорівневих драйверних інтерфейсів, дослідження є актуальним для розробників вбудованих реального-часу систем.

Мета дослідження. Проаналізувати можливості керування LEDC-таймерами ESP32 через прямий запис у регістри, експериментально оцінити точність та стабільність сформованих PWM-сигналів для ряду конфігурацій і розробити практичні рекомендації щодо оптимізації параметрів.

Методи дослідження. Застосовано регістрове програмування в середовищі ESP-IDF на платі ESP32-DevKitC V4 (WROOM-32D), експериментальні вимірювання часових характеристик вихідних сигналів логічним аналізатором (Logic Analyzer, 24 MHz, 8ch), порівняльний аналіз теоретичних розрахунків (формули частоти, прескейлера, розрядності лічильника) та практичних вимірювань для набора з 18 конфігурацій (RES, DIV, DUTY).

Результати. Розглянуто архітектуру LEDC та структуру відповідних регістрів таймерів і каналів; реалізовано алгоритм налаштування HSTIMER0 та HS-каналу через пряме записування в регістри і виведення сигналу на GPIO. Експериментально підтверджено високу відповідність розрахунків і вимірювань: максимальна відносна похибка частоти і періоду не перевищувала ±0,03%, а тривалості високого рівня — ±0,6%.

Висновки. Прямий регістровий доступ до LEDC дозволяє отримати детерміновані, високоточні ШІМ-сигнали з мінімальною затримкою оновлення параметрів, що є придатними для застосувань у робототехніці, силовій електроніці та інших системах з високими вимогами до синхронізації. Рекомендовано подальші дослідження на тему впливу альтернативних джерел такту, режимів низької швидкості LEDC, інтеграції з ISR/FreeRTOS та розширення підходу на інші таймери й канали.

Ключові слова: *ATmega, ESP32, LEDC, ШІМ, регістрове керування, логічний аналізатор*

Надійшла у першій редакції 02.09.2025, в останній - 05.10.2025.
The first version has been received on 02.09.2025, the final version – on 05.10.2025.