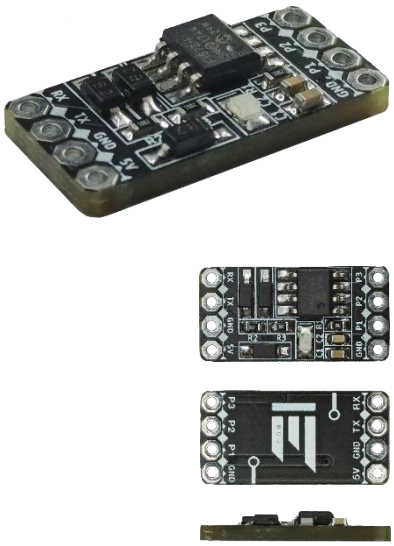
LENTARK®

PWM-CORE

PS504F0A-02W30

FEATURES



- Programmable 3 Pwm Output: P1, P2, P3
- Accurate Frequency and Duty Cycle.
- Frequency Range: 4 Hz – 500 KHz
  - Frequency Tolerance (Max): ±0.2%
  - Frequency Stability (80°C): 100 ppm
- Tunable Period Range: 250.00ms – 120.00s
- Duty Cycle Range: 0.00% – 100.00%
- Easy communication: UART (Rx, Tx)
- Programmable Baud Rate: 9600 / 19200 / 57600 / 115200
- Ability to Save Data to Internal Memory.
- Input Supply Voltage Range: 5V
- Low Output Voltage Swing: 0 – 125 µV

APPLICATIONS

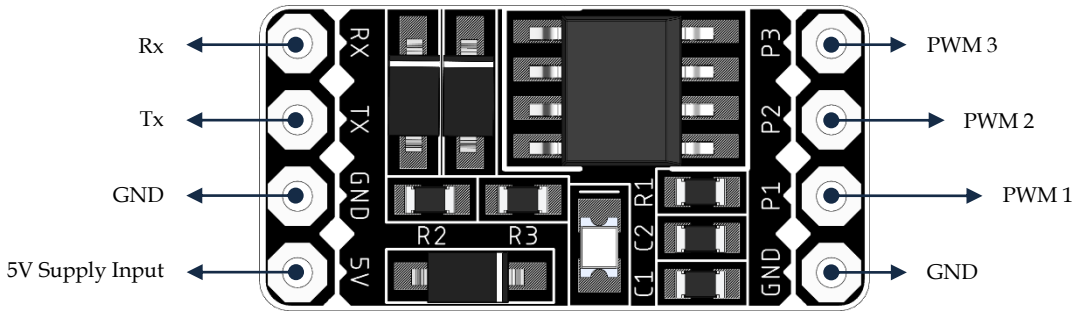
- P & N Channel MOSFET, IGBT Switching.
- Lighting Applications.
- Motor Control.
- Audio Applications.
- 16 Bit PWM.
- R, L and/or C Load Control.
- Hobby.

GENERAL DESCRIPTION

PS504F0A-02W30 is a PWM core with 5V input voltage and three channel signal output. **The frequency and duty cycle characteristics of the PWM channels can be defined separately**. Signal outputs have amplitudes of 5V. It has a maximum tolerance of ± 0.2% in the 4 Hz - 500 KHz band and also has the ability to be calibrated for operations in lower frequency bands.

The communication is quite convenient and simple. All control operations are provided with the codes to be sent to the Rx pin of PS504F0A-02W30. The **Protocols** are shown together with the examples below. The device can save all signal configurations in its internal memory. Thus, the desired initial values can be determined at each restart of the device.

PINOUTS



ELECTRICAL SPECIFICATIONS

∏ Pushing the device to operate above the "Max." listed in the table below may cause the device to overheat and to take up permanent damage. It is inconclusive that the device will function beyond the operating limits as set out in this technical document. Prolonged exposure to work under "maximum" rating conditions may affect device reliability.

Table 1: *Electrical Specifications.*

Conditions: Unless Otherwise Noted, $T_O = +25^oC$ and $V_{IN} = 5V$.						
Parameters	**Sym**	**Min**	**Typ**	**Max**	**Units**	**Condition**
Input						
Input Voltage	V_{IN}	4.5	5	5.5	V	DC
Input Current [No Load]	I_{IN}	– 3.29	2.24 –	– 4.11	mA	$f = 1\ \ Hz$ $f = 500\ KHZ$
Output						
Output Voltage, High	$V_{OUT,HIGH}$	4.200	5	5.100	V	$V_{IN} = 5V$
Output Voltage, Low	$V_{OUT,LOW}$	0	0	0.615	V	
Output Resistance, High	$R_{OUT,HIGH}$	–	–	200	Ω	
Output Resistance, Low	$R_{OUT,LOW}$	75	–	–	Ω	
Total Output Current*	I_{OT}	–	30	45	mA	
Switching						
Rise Time	t_R	–	15	32	ns	C_L = 0 pF
			30	65		C_L = 50 pF
Fall Time	t_F	–	15	30	ns	C_L = 0 pF
			30	60		C_L = 50 pF
Output Power Dissipation	W_{PD}	–	100	720	mW	**Note 1**

* **:** It refers to the total current that can be drawn from the signal outputs.

Note1 : It refers to the power that can be consumed in the sum of the signal outputs.

Tolerance & Sensitivity						
Parameters	**Sym**	**Min**	**Typ**	**Max**	**Units**	**Condition**
Frequency Tolerance *		–	0.05	0.2	% [percent]	250 ms – 2 µs
		–	–	5		120 s – 250 ms
Frequency Sensitivity		–	0.0312	–	µs	$T > 2$ µs
Duty Cycle Sensitivity (0.00 – 100.00)	D	0.01	–	0.01	% [percent]	$T > 312$ µs
		0.01	–	0.1		$T > 31$ µs
		0.1	–	1		$T < 31$ µs

* **:** The period between 120 s - 250 ms can be calibrated manually. See "**Fast Examples of Protocols**" to see examples.

PIN DESCRIPTIONS

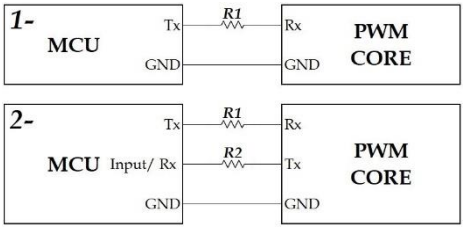
Pin	*Description*	*Notes*	*Types of Connections*
Rx	Used to read data from any MCU. (9600 Baud Rate – 8 Bit Buffer)	It connects to the Tx port of any MCU.	 **Figure 1:** *Typical Connection Schemes.*
Tx	This pin informs whether Pwm-Core's Rx pin is ready to read new data. (1: Ready — 0: Busy)	It connects to the Rx or Input port of any MCU. (0 ≤ R1,2,3 ≤ 470 Ω)	
GND	GND	–	
5V	5V Supply Voltage Input	–	
PWM1	Signal Output 1	For detailed information, see also **Electrical Characteristics table.**	
PWM2	Signal Output 2		
PWM3	Signal Output 3		

Table 2: *Pin Descriptions.*

The device operates the PWM channels asynchronously. This means that no phase difference is sought between PWM channels. However, asynchronous operation provides the opportunity to dynamically assign value, provide continuous (without interruption or restart) PWM. In this way, changing the parameters of any PWM channel does not require any action on other channels, and the new values are written to the PWM channel after the period of the relevant PWM channel is completed.

Saving the current values of the PWM channels is done on command. As explained in Table 3, if there is a '/' character at the end of the command line, all current values including the incoming command value are recorded. In this way, the initial values can be determined in case of restarting the device.

While there is a value waiting to be written with the completion of the period of the PWM channel, in the case of new commands to change this value, the last command is written to the PWM channel. For example; The period of the 2nd PWM channel is 500 µs and the working rate is 24%. Before the PWM period of this channel is completed, if commands are sent to change the operating rate of the channel to 24.15%, 24.3%, 24.45% respectively, the last sent value is written to the PWM channel before the period is completed. The new values generated by period commands and operating rate commands are stored separately in memory. For example; 3. If commands are sent to change both the period and the working rate of this channel before the period of the channel is completed, the last sent working rate value and the last sent period value are written simultaneously to the PWM channel when the period of the channel is completed.

PROTOCOLS

The connections of the Pwm-Core should be done as specified in the **Pin Description** title. The UART communication protocol of the MCU that will send the command must be configured according to the information given in the **Protocol Information** header and the String / Char * / Char [] variable must be prepared in accordance with the following two rules for the command to be sent.

Note: The device can keep all signal configurations in its memory. Thus, it does not need to be reconstructed in every use. See **Table 3**.

Rule 1 – Format:

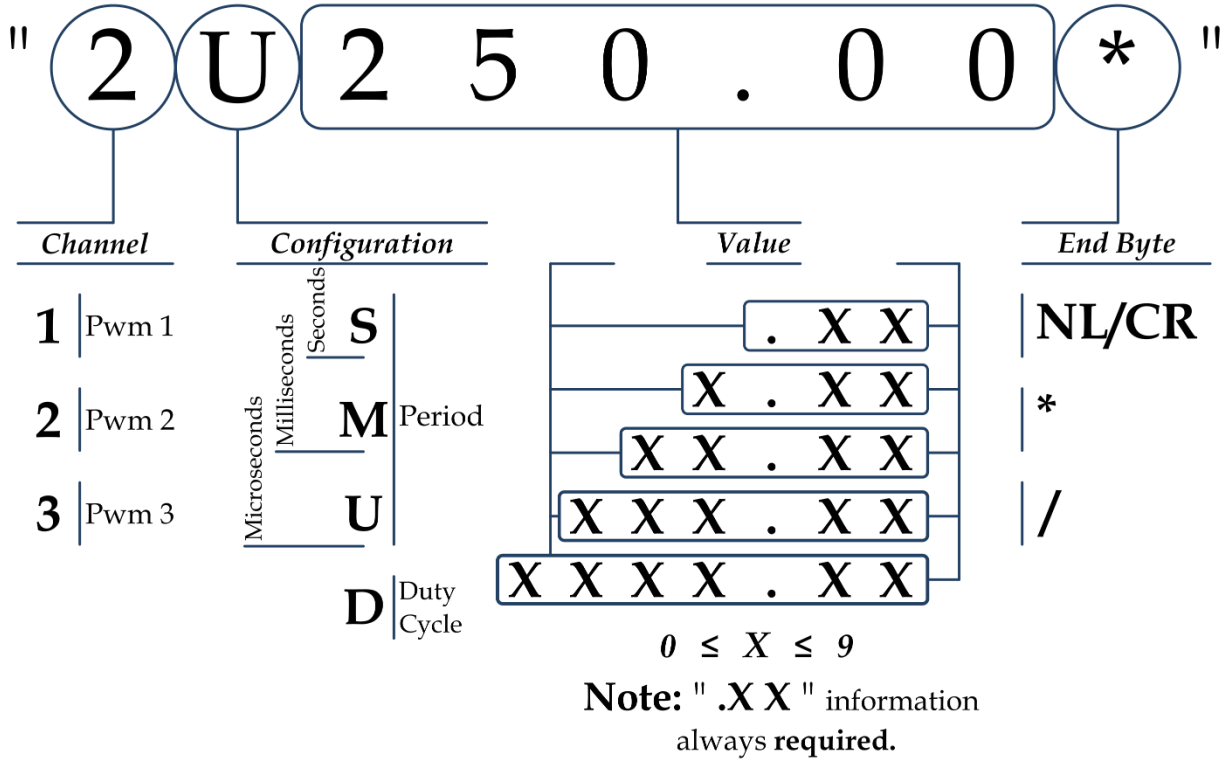


Figure 2: *Format.*

Table 3: *Explanation of End Bytes.*

End Bytes	Explanation	Examples
'*' or NL/CR	**The command completion byte.** The sent command is processed and the new value is written to the PWM channel after the relevant PWM channel completes its period.	1P1.23* 2D74.11*
'>'	**The command completion and saving byte**. The command sent is processed, all values are stored in the memory and new value is written to the PWM channel after the relevant PWM channel completes its period.	1U100.10/ 2D47.25/ 3U720.00/

Table 4: *Demonstration of Required Commands for Baud Rate Programming.*

Configuration	Explanation	Values	
B	By default, the device is at 9600 Baud Rate. This configuration is used to program Baud Rate when it wants to operate at high communication speeds.	1B4.00/ 1B3.00/ 1B2.00/ 1B1.00/	9600 Baud Rate [Default] 19200 Baud Rate 57600 Baud Rate 115200 Baud Rate

Rule 2 – Protocol, Processing Time and Limits:

Protocol Information:

Mode: Asynchronous
Baud Rate: 9600 / 19200 / 57600 / 115200
Data Polarity: Active-High
Rx Read Bits: 8 Bits

Limits:

	En Düşük Değer	En Yüksek Değer
S	0.50	120.00
M	0.01	500.00
U	2.00	2000.00
D	0.00	100.00

Processing_Time:

If Pwm-Core's Tx[Rx_Rdy] pin is not to be used, Pwm-Core's processing time must be taken into consideration in order to must be kept waiting while the first data is being processed. The processing times specified in **Table 5** show the highest and typical processing times determined. However, since each configuration goes through a different set of mathematical operations, it cannot be guaranteed that it will not exceed the specified processing times.

Configuration	*Typical*	*Max.*
Period	420 µs	770 µs
Duty Cycle	230 µs	280 µs
+ Saving Comd.	+ 3,25 ms	+ 3,35 ms

Table 5: *Processing Times.*

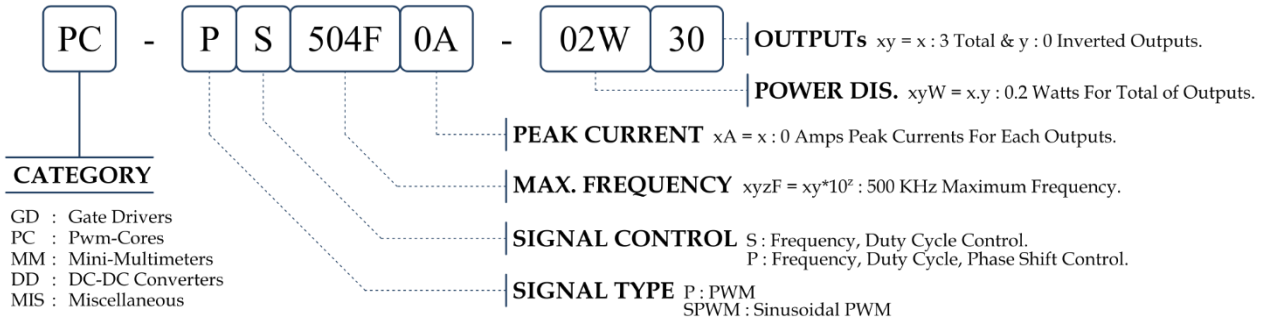
Fast Examples of Protocols:

- If Channel 1's Period will be, 6 milliseconds; → "1M6.00*"
- If Channel 1's Period will be, 128.5 microseconds; → "1U128.50*"
- If Channel 2's Period will be, 68.431 seconds; → "2S68.43*"
- If Channel 2's Period will be, 620 milliseconds; → "2S0.62*"
- If Channel 3's Period will be, 2 milliseconds; → "3M2.00*"
- If Channel 3's Period will be, 774.05 microseconds; → "3U774.05*"
- If Channel 1's Duty Cycle will be 21.8%; → "1D21.80*"
- If Channel 1's Duty Cycle will be 47.05%; → "1D47.05*"
- If Channel 2's Duty Cycle will be 11.1%; → "2D11.10*"
- If Channel 2's Duty Cycle will be 0%; → "2D0.00*"
- If Channel 3's Duty Cycle will be 77.05%; → "3D77.05*"
- If Channel 3's Duty Cycle will be 100%; → "3D100.00*"

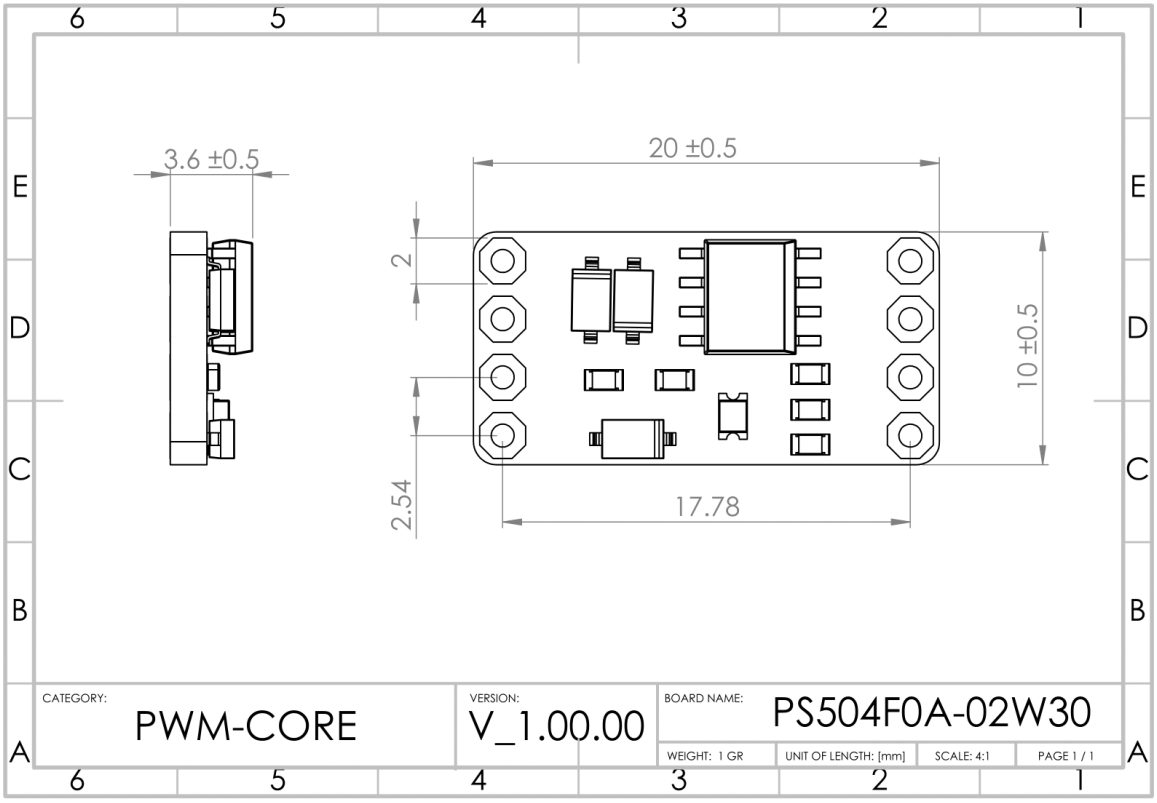
Examples For Coding:

```
//PS504F0A-02W30
#define Rx_Ready 10 //Any Input Pin
#include <SoftwareSerial.h>
SoftwareSerial _mySerial(10, 11);// RX, TX

//----------Baud Rate--------------------
//Baud Rate: 4 -> 9600 | 3 -> 19200 | 2 -> 57600 | 1 -> 115200
uint16_t cong_Baud_[2]= {4, 0};
//-----------Period Type--------------------------
// S: Seconds | M: Milliseconds | U: Microseconds
uint8_t channel_1_SMU_= 'M';
uint8_t channel_2_SMU_= 'M';
uint8_t channel_3_SMU_= 'U';
//------------Period-----------------------------
//{Integer Part, Floating Part}
//Exp: {245,75} => if _config == 'U' than Period = 245.75 microseconds
uint16_t channel_1_period_[2]= {5, 0};
uint16_t channel_2_period_[2]= {5, 0};
uint16_t channel_3_period_[2]= {500, 0};
//-----------Duty Cycle---------------------------
uint16_t channel_1_duty_[2]= {50, 0};
uint16_t channel_2_duty_[2]= {50, 0};
uint16_t channel_3_duty_[2]= {0, 0};

void setup() {
Serial.begin(9600); while (!Serial) {;}
_mySerial.begin(9600); // Set Baud Rate
pinMode(Rx_Ready, INPUT_PULLUP); // Set Rx_Ready to input & Turn on pull-up resistors
//send_configuration(1, 'B', cong_Baud_, '/');
send_configuration(1, channel_1_SMU_, channel_1_period_, '*');
send_configuration(2, channel_2_SMU_, channel_2_period_, '*');
send_configuration(3, channel_3_SMU_, channel_3_period_, '*');
send_configuration(1, 'D', channel_1_duty_, '*');
send_configuration(2, 'D', channel_2_duty_, '*');
send_configuration(3, 'D', channel_3_duty_, '/');
}

void loop() {

  /*/-------------------*Channel 1*-----------------------------
    //0.1% Duty each Cycle
  if((channel_1_duty_[1]+=10) >=90) {channel_1_duty_[1] = 0;
    if(channel_1_duty_[0]++ >=99) channel_1_duty_[0] = 0;
  }
  send_configuration(1, 'D', channel_1_duty_, '*');
  while(!digitalRead(Rx_Ready)); // Wait for the Pwm-Core to be ready

  //10 micro seconds each cycle.
  if(channel_1_SMU_== 'U'){
  if((channel_1_period_[0]+=10) >=2000 && channel_1_SMU_== 'U') {channel_1_period_[0] = 2;channel_1_SMU_= 'M';}}

  if(channel_1_SMU_== 'M'){
    if(channel_1_period_[1]++ >=99) {channel_1_period_[1] = 0;
      if(channel_1_period_[0]++ >=9 ) {channel_1_period_[0] = 500;channel_1_SMU_= 'U';}}
  }
  send_configuration(1, channel_1_SMU_, channel_1_period_, '*'); */

  /*/-------------------*Channel 3*-----------------------------
  //0.1% Duty each Cycle
  if(channel_3_duty_[0]++ >=99) channel_3_duty_[0] = 0;
  send_configuration(3, 'D', channel_3_duty_, '*');
  while(!digitalRead(Rx_Ready)); // Wait for the Pwm-Core to be ready

  //10 micro seconds each cycle.
  if(channel_3_SMU_== 'U'){
  if((channel_3_period_[0]+=10) >=2000 && channel_3_SMU_== 'U') {channel_3_period_[0] = 2;channel_3_SMU_= 'M';}}

  if(channel_3_SMU_== 'M'){
    if(channel_3_period_[1]++ >=99) {channel_3_period_[1] = 0;
      if(channel_3_period_[0]++ >=9 ) {channel_3_period_[0] = 500;channel_3_SMU_= 'U';}}
  }
  send_configuration(3, channel_3_SMU_, channel_3_period_, '*'); */

  if (Serial.available()) {int a = Serial.read();_mySerial.write(a);Serial.write(a);}
}

void send_configuration (uint8_t _ch, uint8_t _conf, uint16_t* _value, uint8_t _end)
{char _buffer[12]; // Buffer
// Prepare the buffer Note: \n\r is not necessary
sprintf(_buffer, "%d%c%d.%02d%c\n\r",_ch,_conf,_value[0],_value[1],_end);
while(!digitalRead(Rx_Ready)); delayMicroseconds(50); // Wait for the Pwm-Core to be ready
_mySerial.write(_buffer); // Send buffer to Pwm-Core
Serial.write(_buffer);
}

// End of CODE
// Outputs:
//    1M5.00*
//    2M5.00*
//    3U500.00*
//    1D50.00*
//    2D50.00*
//    3D0.00/
```

PRODUCT CODE



TECHNICAL DRAWING



CONTACT INFORMATION

Lentark Electronics

Website : www.lentark.com
E-mail : info@lentark.com