



Санкт-Петербургский государственный университет
Кафедра системного программирования

Использование СПО в сфере АСУ ТП на примере ОС РВ Embox

Пилюк Дмитрий

Санкт-Петербург
2025

- АСУ ТП — система технических и программных средств, предназначенных для автоматизации управления технологическим процессом.
- ПЛК — программируемый логический контроллер.
 - ▶ Классические ПЛК
 - ▶ "Мягкие" ПЛК

МЭК(IEC) 61131-3

IEC 61131-3 — раздел международного стандарта IEC 61131, описывающий языки программирования для программируемых логических контроллеров.

- LD – Ladder Diagram
- FBD – Function Block Diagram
- SFC – Sequential Function Chart
- ST – Structured Text
- IL – Instruction List

Свободные IDE

- Beremiz
- OpenPLC editor
- Eclipse 4diac

The screenshot shows the Beremiz software interface for ladder logic programming. The project is named 'Blink'. The left sidebar shows the project structure with a single file 'Unnamed' and a resource named 'resource1'. The main workspace displays a ladder logic program for a blinking LED. The program consists of the following components and connections:

- Timers:** TONB (TON) and TOFB (TOF) are used to control the LED. TONB is triggered by the rising edge of 'blink_led' and has an output 'IN 0' connected to the 'IN' of TOFB. TOFB has an output 'IN 0' connected to the 'IN' of the LED.
- LEDs:** The program uses two sets of LED drivers. The first set, controlled by TONB, includes an LED driver (LED_ONB) with 'EN' connected to 'IN 0' and 'LED_SUCCESS' output. The second set, controlled by TOFB, includes an LED driver (LED_OFFB) with 'EN' connected to 'IN 0' and 'LED_SUCCESS' output.
- Relays:** A relay (T#15) is used to switch between the two sets of LED drivers. Its coil is controlled by the 'N' contact of the TONB timer. The normally closed contact of T#15 is connected to the 'IN' of the LED_ONB driver, and the normally open contact is connected to the 'IN' of the LED_OFFB driver.
- Outputs:** The outputs of the LED drivers are connected to the 'LED_ON' and 'LED_OFF' contacts of the relay T#15, which in turn control the physical LED.

The right sidebar contains a library of functions and blocks, including standard functional blocks, additional functional blocks, and the 'Blink' library which contains blocks for LED_ON, LED_OFF, and T#15.

OpenPLC Editor

OpenPLC Editor

File Edit Display Help Recent

example

main

Description :

Class Filter : All

#	Name	Class	Type	Location	Initial Value	Documentation	Debug
0	blink_time	Input	TIME				
1	out	Output	BOOL				

Start typing to add a comment to this rung

```
graph LR; in(( )) ---|out| TON[TON  
IN PT ET  
--- ("TIME")]; TON ---|Q| TOF[TOF  
IN PT ET  
--- ("TIME")]; TOF ---|Q| out(( ))
```

Create new rung

Library

- Additional Function Blocks
- Arduino Function Blocks
- Communication Blocks
- Jaguar
- MQTT
- PIAM
- Sequent Microsystems Modules
- Standard Function Blocks
- Arithmetic
- BitShift
- Bitwise
- CharacterString
- Comparison

Console

Clear console

No file selected

Matiec (MAT IEC) — это свободный компилятор языка программирования IEC 61131-3, используемый для программируемых логических контроллеров (ПЛК).

- ST, IL, SFC -> ANSI C
- Исполняемые файлы:
 - ▶ iec2c
 - ▶ iec2iec

Протоколы взаимодействия

- Modbus
- EtherNet/IP
- EtherCAT
- CANopen
- Bacnet
- DeviceNet

Modbus

Modbus — это открытый протокол связи (клиент-сервер или master-slave), разработанный компанией Modicon в 1979 году для промышленных контроллеров (ПЛК).

Виды:

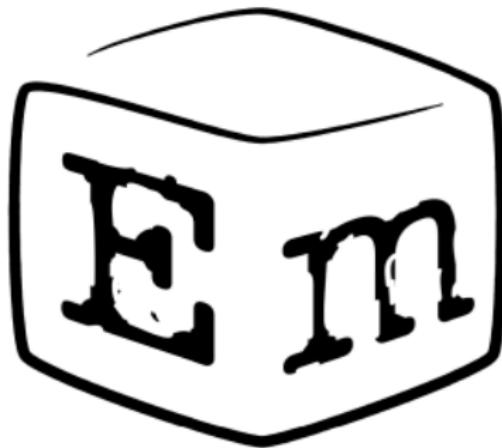
- RTU
- TCP
- ASCII

libmodbus — это свободная библиотека программного обеспечения реализующая взаимодействие по протоколу Modbus.

Embox

ОС РВ Embox — свободная операционная система реального времени.

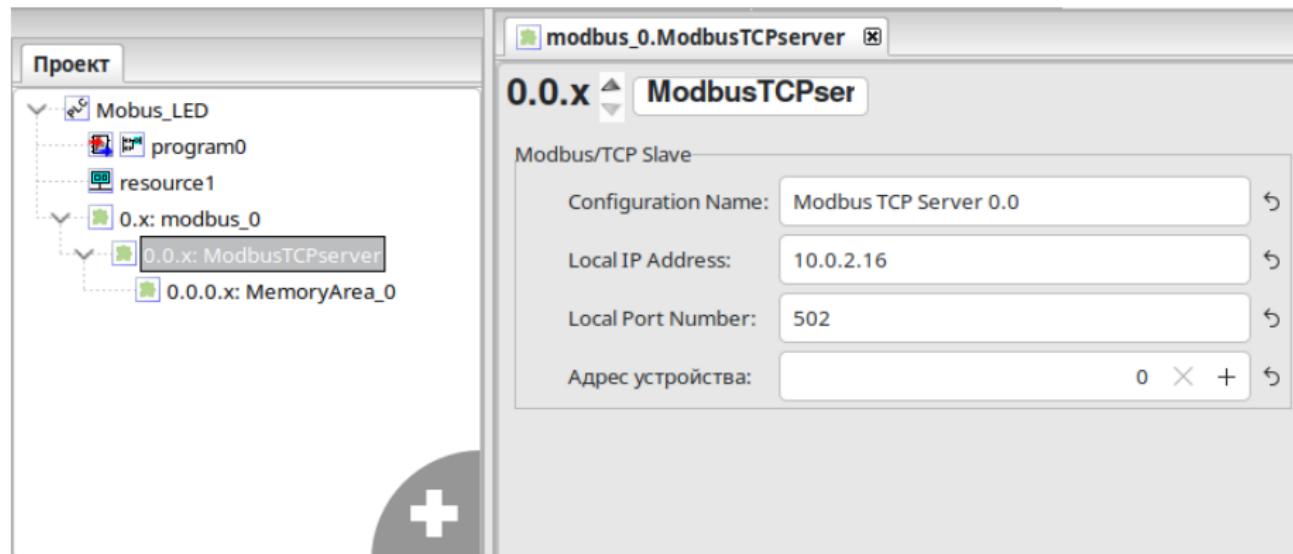
- Модульная архитектура
- Поддержка POSIX
- Кроссплатформенность



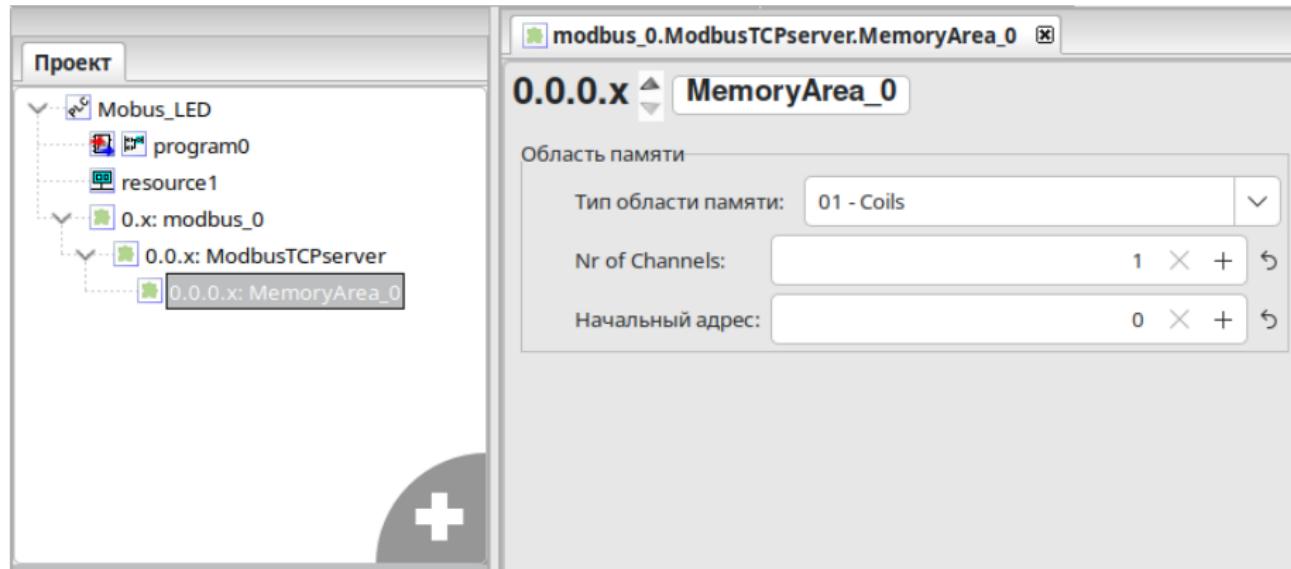
Matiec в Embox

```
project > plc > examples > ☰ Mybuild
1  package project.plc.examples
2
3  @BuildDepends(project.plc.matiec)
4  module blink extends project.plc.core {
5      source "blink.st"
6
7      @NoRuntime depends project.plc.matiec
8      @NoRuntime depends project.plc.matiec_lib.led
9  }
10
```

Modbus в Embox. Подход 1



Modbus в Embox. Подход 1



Modbus в Embox. Подход 1

```
24 |
25 #include "modbus.h"
26
27
28 typedef struct _server_node_t {
29     const char *ip_adress;
30     uint16_t port;
31     uint8_t slave_id;
32     modbus_mapping_t mem_area;
33     modbus_t *ctx;
34 } server_node_t;
35
36
37 /* Values for instance 0 of the modbus plugin */
38
39 #define NUMBER_OF_TCPSERVER_NODES 1
40
41 #define NUMBER_OF_SERVER_NODES NUMBER_OF_TCPSERVER_NODES
42
43
44 /*initialization following all parameters given by user in application*/
45
46 static server_node_t server_nodes[NUMBER_OF_SERVER_NODES] = {
47 /*node 0.0*/
48 {"0.0.0.0", 502, 0, {1, 0, 0, 0, 0, 0, 0, 0}}
49 }
50 ;
51
52 ****
53 /*located variables*/
54 ****
55
56 BOOL __QX0_0_0_0;
57
58 #define LOC_VARS_INIT __QX0_0_0_0 = &server_nodes[0].mem_area.tab_bits[0];
59
```

Modbus в Embox. Подход 2

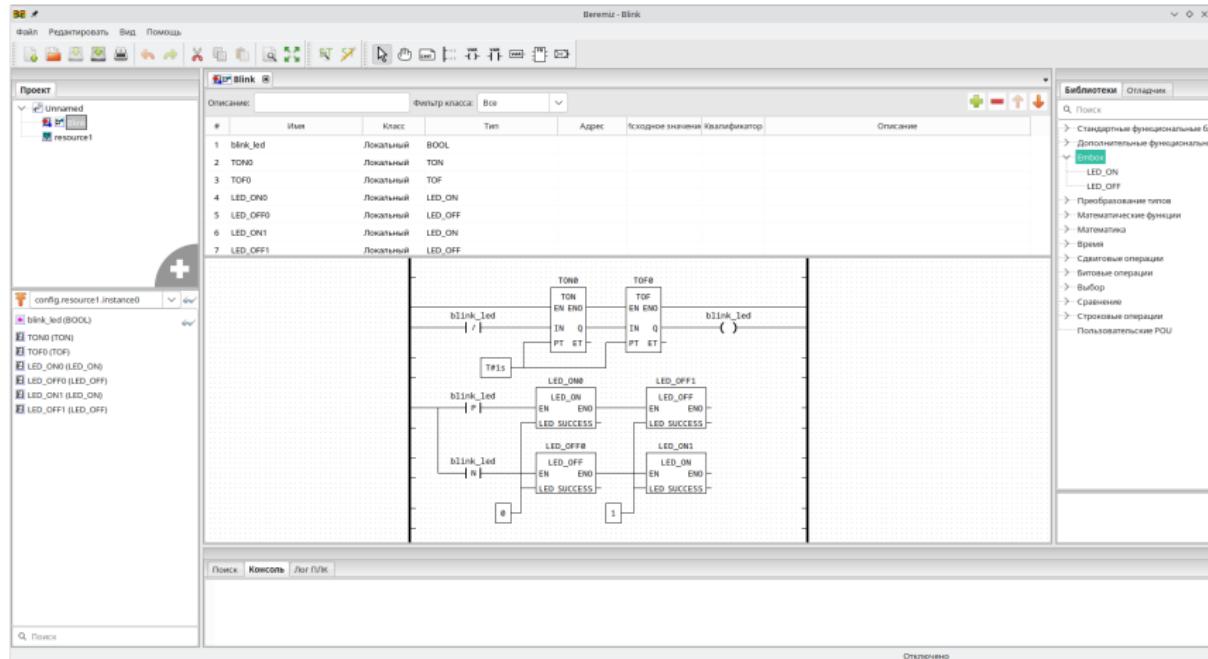
project > plc > templates > nucleo_f207zg > rootfs > [t] modbus.toml

```
1  host = '192.168.2.128'  
2  port = 502  
3  mb_addr = 0  
4  
5  addr_bits = 0  
6  nb_bits = 1  
7  start_bits = 0  
8  
9  addr_input_bits = 1  
10 nb_input_bits = 0  
11 start_input_bits = 0  
12  
13 addr_input_registers = 2  
14 nb_input_registers = 0  
15 start_input_registers = 0  
16  
17 addr_registers = 3  
18 nb_registers = 0  
19 start_registers = 0  
20 |
```

Modbus в Embox. Подход 2

```
20  @AutoCmd
21  @Cmd(name="mb_service", help="")
22  @Build(stage=2)
23  @BuildDepends(project.plc.modbus)
24  module mb_service{
25      source "mb_service.c"
26
27      @NoRuntime depends project.plc.modbus
28  }
```

Blink



Modbus blink

Проект

Modbus_LED

- program0
- resource1
- 0.x: modbus_0
- 0.0.x: ModbusTCPserver
- 0.0.0.x: MemoryArea_0

Описание: Фильтр класса: Все

#	Имя	Класс	Тип	Адрес
1	Coil0	Локальный	BOOL	%QX0.0.0.0
2	LED_ON0	Локальный	LED_ON	
3	LED_OFF0	Локальный	LED_OFF	
4	R_TRIGGER	Локальный	R_TRIGGER	
5	F_TRIGGER	Локальный	F_TRIGGER	

config.resource1

instance0 (program0)

```
graph LR; Coil0[Coil0] --> R_TRIGGER[R_TRIGGER  
CLK 0]; Coil0 --> F_TRIGGER[F_TRIGGER  
CLK 0]; R_TRIGGER --> LED_ON[LED_ON  
EN ENO  
LED SUCCESS]; F_TRIGGER --> LED_OFF[LED_OFF  
EN ENO  
LED SUCCESS];
```