Перв. примен.	

УТВЕРЖДЕН
643.04832915.10189-
01 addition xx-ЛУ

ИЗДЕЛИЕ Logic Box

Программное обеспечение управления

Вводное руководство

Дополнительный документ

643.04832915.10189-01 addition xx

Листов 16

Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата	

2023

АННОТАЦИЯ

Программное обеспечение управления программируемых логических контроллеров серии Logic Box предназначено для настройки и сопровождения систем промышленной автоматизации на базе модульных ПЛК Logic Box LB241 и Logic Box LB340. Настоящий документ представляет собой краткое практическое введение, объясняющее на примерах принципы организации, возможности и приемы работы с ПЛК серии Logic Box. Более полно система документирована в документе ПО Logic Box: Руководство программиста, а также Руководстве по эксплуатации ПЛК Logic Box LB340 и Руководстве по эксплуатации УСО Logic Box LB241.

СОДЕРЖАНИЕ

1. УСТАНОВКА И ПЕРВАЯ ПРОВЕРКА

Программное обеспечение управления ПЛК Logic Box с открытым исходным кодом доступно на GitHub [1]. Для установки ПО управления в ОС Linux нужно выполнить следующие шаги:

1) установить систему управления версиями `git` и зависимости ПО LogicBox – библиотеки языка Perl5: `Socket6.pm`, `JSON.pm`, `YAML.pm`, `Data::HexDump.pm`. Они могут быть или не быть в системе по умолчанию. В случае ОС Debian GNU/Linux для установки достаточно выполнить команды:

```
$ sudo apt-get update
$ sudo apt-get install git git-gui
$ sudo apt-get install libsocket6-perl libjson-perl libyaml-perl
$ sudo apt-get install libdata-hexdump-perl
$ sudo apt-get install build-essential autoconf
```

2) загрузить копию ПО с github:

```
$ git clone https://github.com/it-npc/logicbox
$ cd logicbox
$ git submodule update --init --recursive
```

3) (не требуется для работы с УСО LB241BC, только для ПЛК): установить средства для сборки транслятора `matiec` (`autoconf`, `GNU toolchain`) и собрать его:

```
$ sudo apt-get update
$ sudo apt-get install build-essential autoconf
$ make -C matiec
```

4) (не требуется для работы с УСО LB241BC, только для ПЛК): установить ESP-IDF – среду разработки для микроконтроллеров ESP32. Инструкция по установке под Linux: [2]. На шаге 3 («Step 3. Set up the Tools») для экономии дискового пространства достаточно выбрать две модели микроконтроллера: `esp32,esp32s3`.

5) добавить каталог `logicbox/bin` в путь поиска команд:

```
$ export PATH="$PATH:`pwd`/bin"
```

6) подключить один или несколько ПЛК Logic Box к локальной сети Ethernet, к которой подключен управляющий компьютер. Если компьютер имеет несколько интерфейсов локальной сети, ПЛК можно подключить к любой из них;

7) Выполнить команду `lbdiscover`:

```
$ lbdiscover
<noname> LB241BC [192.168.1.250] f4:12:fa:d5:51:e3 eth0
bc100 LB241CPU [192.168.1.245] f4:12:fa:d5:51:11 eth1
```

В данном примере команда lbdiscover обнаружила два ПЛК, причем на разных интерфейсах (`eth0` и `eth1`). Первый из ПЛК по-видимому не имеет загруженной конфигурации, поскольку обнаружен с именем `<noname>` – возможно, это новое устройство с завода. Второй ПЛК имеет загруженную конфигурацию, в которой есть его имя: `bc100`. Оба обнаруженных ПЛК имеют адреса IPv4; возможно они заданы статически в конфигурации, или получены по DHCP, а могли и отсутствовать (`[]`). Адреса IPv4 не требуются для работы управляющего ПО, которое при отсутствии в фале конфигурации статически заданного адреса IPv4 для данного ПЛК использует протокол IPv6.

Все модули Logic Box имеют уникальные MAC-адреса (например, `f4:12:fa:d5:51:e3`), которые кроме основного применения для связи по сети Ethernet используются для идентификации физического устройства в файле конфигурации. MAC-адрес указан также на этикетке устройства. Зная MAC-адрес головного модуля ПЛК, можно приступать к его конфигурации.

2. МИНИМАЛЬНАЯ КОНФИГУРАЦИЯ

При помощи текстового редактора создадим в каталоге `conf` файл в формате YAML [3] с именем `conf.yml`. Запуск редактора `emacs`, имеющего встроенную поддержку синтаксиса YAML:

```
$ emacs conf/conf.yml
```

Если запустить редактор не получается с диагностикой `emacs: command not found`, можно воспользоваться другим редактором (например, `nano`), или установить редактор `emacs`:

```
$ sudo apt-get update
$ sudo apt-get install emacs
```

В редакторе нужно ввести в файл `conf/conf.yml` следующую минимальную конфигурацию в формате YAML:

```
plc1:
  slot-1:
    macaddr: f4:12:fa:d5:51:e3
```

Здесь `plc1` это имя ПЛК в системе (можно выбрать мнемоничное, отражающее роль данного ПЛК в системе автоматизации). MAC-адрес получен из вывода команды discover. После сохранения файла, даже до загрузки конфигурации в ПЛК, появляется возможность обращаться к ПЛК по имени, например:

получить время в секундах с момента последней перезагрузки ПЛК:

```
$ lbcmd plc1 sys.uptime
2024-08-25 13:04:26.529 sys.uptime=748
```

получить версию встроенного ПО головного модуля ПЛК:

```
$ lbcmd plc1 sys.version
2024-08-25 13:05:49.407 sys.version=20240815235144-cedb76f-dirty
```

считать текущую конфигурацию ПЛК (но она пока пустая):

```
$ lbcmd plc1 --getconf
$
```

загрузить конфигурацию в ПЛК (и сохранить в flash-памяти):

```
$ lbconf plc1
Configuration status: OK
```

считать текущую конфигурацию ПЛК:

```
$ lbcmd plc1 --getconf
---
plc1:
  slot-1:
    macaddr: f4:12:fa:d5:51:e3
```

3. ЯЗЫК КОНФИГУРАЦИИ НА ПРИМЕРАХ

Чтобы двигаться дальше, нужно составить конфигурацию, описывающую конкретную аппаратную конфигурацию ПЛК (типы и порядок модулей) и настройки на конкретную задачу (режимы портов ввода-вывода, выполняемые ПЛК прикладные программы). Для освоения языка конфигурации имеются следующие ресурсы:

1) в каталоге `conf/examples` есть несколько полных работоспособных примеров конфигурации;

2) в файле `conf/conf-full.yml` приведена абстрактная конфигурация, в которой представлены примеры всех допустимых параметров и синтаксических конструкций;

3) далее в этом разделе приводятся и комментируются фрагменты конфигурации.

3.1. Общая структура описания конфигурации

Описание конфигурации представляет собой древовидную структуру данных в формате YAML [3] с ассоциативным массивом (ключ–значение) на верхнем уровне. Ключи на первом (верхнем) уровне дерева это имена устройств (ПЛК) в распределенной системе автоматизации. Ключи на втором уровне это имена программных подсистем базовых модулей ПЛК, или имена вида `slotN`, соответствующие модулям расширения ПЛК:

```
bc1:
  clock: {...}
  slot-1: {...}
  slot1: {...}
  slot2: {...}
plc202:
  clock: {...}
  modbus\_client: {...}
  task0: {...}
  task1: {...}
  slot-2: {...}
  slot-1: {...}
  slot1: {...}
  slot2: {...}
  ...
plc203:
  ...
```

Номера слотов (разъемов расширения LB340 или модулей в сборке LB241) присваиваются слева направо. В сборке LB241 номер «−1» соответствует самому правому из «базовых» модулей (LB241CPU, LB241BC), т.е. отрицательными индексами N нумеруются базовые модули, а положительными – модули ввода-вывода. Нулевой индекс не используется.

3.2. Взаимодействие между подсистемами: var, var_out

Взаимодействие между подсистемами осуществляется в терминах именованных переменных по схеме «издатель–подписчик». Одна из подсистем («издатель») время от времени присваивает некоторой переменной новое значение; оно поступает в подсистему «брокер», которая рассылает всем подсистемам–подписчикам данной переменной уведомления с ее новым значением, и по этому событию подсистемы–подписчики выполняют те или иные действия. Пример:

```
plc1:
  modbus_server:
    server: 192.168.1.60:1502
    holding5: {var: myvar}
  modbus_client:
    addr: 10
    holding0x300: {var_out: myvar, period: 1s}
    response_timeout: 0.1s
```

В этом примере ПЛК с именем plc1 работает как шлюз Modbus TCP—Modbus RTU. По внешнему запросу к ПЛК по протоколу Modbus TCP с функцией 0x6 Write Single Register и адресом регистра 0x0005 обновляется значение переменной myvar. Значение переменной myvar регулярно (с частотой 1Гц) записывается по протоколу Modbus RTU в сервер (slave) с адресом 10 функцией 0x6 Write Single Register и адресом регистра 300 (шестнадцатеричное). Эта передача данных реализуется на системном уровне, без участия прикладных программ ПЛК. Развитие этого примера с использованиям интервалов регистров и переменных:

```
plc1:
  modbus_server:
    server: 192.168.1.60:1502
    holding0..15: {var: myvar0..15}
  modbus_client:
    addr: 10
```

```
    holding1000..1003: {var_out: myvar3..6, period: 1s}
    holding2000..2001: {var_out: myvar12..13, period: 5s}
    response_timeout: 0.1s
  modbus_client2:
    addr: 20
    holding1..2: {var_out: myvar3..6, period: 2s}
    holding2000..2003: {var_out: myvar10..13, period: 500ms}
    response_timeout: 50ms
```

В этом примере по внешнему запросу по протоколу Modbus TCP с функцией 0x6 Write Single Register или 0x16 Write Multiple Registers с адресами регистров в диапазоне 0..15 (десятичное) обновляются значения переменных `myvar0`, `myvar1`, ... `myvar15`. Значение некоторых из переменных `myvar0..15` регулярно (но с разными периодами) записывается по протоколу Modbus RTU в серверы (slave) с адресами 10 и 20 функцией 0x16 Write Multiple Registers.

3.3. Прикладные программы ПЛК

ПЛК Logic Box поддерживают два способа прикладного программирования:

1) Кодирование на языке ST (IEC 61131-3 Ed.2) [4], компиляция программы в бинарный код процессора ПЛК, запуск программы в ПЛК на одном из 5 уровней приоритета (`task0`...`task4`) с контролем времени цикла;

2) Графическое программирование распределенных систем управления в терминах функциональных блоков стандарта IEC 61499 (FBD) в программном комплексе Eclipse 4diac™. Встроенное ПО ПЛК LogicBox включает в себя среду времени выполнения стандарта IEC 61499 (4diac FORTE).

Программы, созданные обоими способами, могут одновременно выполнятся на одном ПЛК, взаимодействуя через брокер в терминах переменных.

3.3.1. Пример 1: генератор меандра на языке ST

```
PROGRAM oscillator
  VAR_EXTERNAL
    out1: BOOL;
    out2: BOOL;
    start: BOOL;
  END_VAR

  VAR
    ton1 : TON;
```

```
    ton2 : TON;
  END_VAR

  if start then
  ton1(
       IN := not ton2.q,
       PT := T#1000ms);
  ton2(
       IN := ton1.q,
       PT := T#1000ms);
  end_if;

  out1 := ton1.q;
  out2 := not out1;
END_PROGRAM
```

В этом примере программа на языке ST оперирует тремя скалярными переменными: входная переменная START типа BOOL разрешает работу генератора, выходные переменные OUT1 и OUT2 типа BOOL представляют собой выходы генератора, работающие в противофазе. Изначально генератор остановлен (START $= 0$). Конфигурация для запуска этой программы:

```
plc1:
  var:
    START: 0
    OUT1: 1
    OUT2: 1
  task0:
    period: 1ms
    maxrun: 5ms
    prog1:
      enable: y
      file: "examples/01-oscillator.st"
  on_stop:
    maxrun: 100ms
    prog991:
      enable: y
      file: "examples/01-onstop.st"
```

Параметр period задает периодичность цикла запуска программ задачей task0. В данном примере в цикле задачи task0 запускается единственная программа prog1. Обязательный параметр maxrun задает ограничение на максимально допустимое астрономическое время выполнения цикла задачи task0. Если ограничение maxrun оказывается превышено – из-за ошибки прикладного программиста (например зацикливание) или системных причин, например перегрузки процессора ПЛК другими задачами – выполняется

аварийная остановка всех задач `task0...taskN`, после чего однократно выполняются все программы в задаче `on_stop` для корректной остановки технологического процесса, которым управляет данный ПЛК:

```
PROGRAM on_stop
  VAR_EXTERNAL
    out1: BOOL;
    out2: BOOL;
  END_VAR
  out1 := TRUE;
  out2 := TRUE;
END_PROGRAM
```

Для запуска прикладной программы на ПЛК нужно выполнить следующие шаги:

1) с помощью текстового редактора записать приведенный выше фрагмент конфигурации в файл `conf.yml`. Загрузка этой конфигурации пока не удастся, потому что в ПЛК ещё не загружены программы `st/examples/01-oscillator.st` и `st/examples/01-onstop.st`:

```
    $ lbconf plc1
    $ Configuration status: task0: prog1: not loaded yet, cannot enable
```

2) загрузить все программы, упомянутые в конфигурации с флагом `enable: y` (в данном примере это программы `prog1` и `prog991`):

```
    $ lbupload plc1 1
    $ segment sizes: textsz=868 datasz=72 bsssz=0
    $ GETPLACE returned text=0x428c0000 data=0x3c87ff98 bss=0x00000000
    $ chunk upload: 246@0
    $ chunk upload: 250@246
    $ chunk upload: 250@496
    $ chunk upload: 194@746
    $ OK uploaded prog1 from /home/user/logicbox/st/examples/01-oscillator.st
    $
    $ lbupload plc1 991
    $ segment sizes: textsz=116 datasz=56 bsssz=0
    $ GETPLACE returned text=0x428c1000 data=0x3c87ff60 bss=0x00000000
    $ chunk upload: 172@0
    $ OK uploaded prog991 from /home/user/logicbox/st/examples/01-onstop.st
```

3) загрузить конфигурацию:

```
    $ lbconf plc1
    $ Configuration status: OK
```

4) разрешить выполнение прикладных программ и проверить, что разрешение действует, т.е. не произошло немедленной аварийной остановки:

```
$ lbcmd plc1 sys.run=1
$ lbcmd plc1 sys.run
2024-08-27 13:49:21.313 sys.run=1
```

5) при помощи отладочных функций считывания и установки переменных, предоставляемых утилитой `lbcmd`, проконтролировать работу программы `prog1`:

```
$ lbcmd plc1 START OUT1 OUT2 1
2024-08-27 13:53:56.572 START=0 OUT1=0 OUT2=1
2024-08-27 13:53:57.583 START=0 OUT1=0 OUT2=1
2024-08-27 13:53:58.593 START=0 OUT1=0 OUT2=1
2024-08-27 13:53:59.605 START=0 OUT1=0 OUT2=1
^C
$ lbcmd plc1 START=1
$ lbcmd plc1 START OUT1 OUT2 1
2024-08-27 13:56:37.116 START=1 OUT1=1 OUT2=0
2024-08-27 13:56:38.126 START=1 OUT1=0 OUT2=1
2024-08-27 13:56:39.139 START=1 OUT1=1 OUT2=0
2024-08-27 13:56:40.149 START=1 OUT1=0 OUT2=1
2024-08-27 13:56:41.158 START=1 OUT1=1 OUT2=0
^C
```

6) посмотреть данные о фактической длительности выполнения циклов прикладных программ, которые полезны для коррекции параметра конфигурации `maxrun`. Значения выводятся в микросекундах:

```
$ lbcmd plc1 --stats
prog1.lastleft = 9831
prog1.lastrun = 168
prog1.maxrun = 3037
prog1.maxrun2 = 3031
prog1.minleft = 6962
prog1.minrun = 58
prog1.ticks = 1242269
```

7) запретить выполнение прикладных программ и проверить, что программа `prog1` остановлена, а задача `on_stop` отработала и установила сигналы `OUT1`, `OUT2` в «нейтральное» состояние, заданное в программе `prog991`:

```
$ lbcmd plc1 sys.run=0
$ lbcmd plc1 START OUT1 OUT2 1
2024-08-27 14:04:34.565 START=1 OUT1=1 OUT2=1
2024-08-27 14:04:35.575 START=1 OUT1=1 OUT2=1
2024-08-27 14:04:36.585 START=1 OUT1=1 OUT2=1
2024-08-27 14:04:37.595 START=1 OUT1=1 OUT2=1
^C
```

3.3.2. Пример 2: использование массивов и структур

(WIP)

ПЕРЕЧЕНЬ ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. ЭЛЕКТРОПРИВОД НПЦ ИТ. LogicBox PLC control software. — Access mode: https://github.com/it-npc/logicbox.

2. Standard Toolchain Setup for Linux and macOS - ESP32 — ESP-IDF Programming Guide latest documentation. — Access mode: https://docs.espressif.com/projects/esp-idf/en/latest/esp32/get-started/linux-macos-setup.html.

3. The Official YAML Web Site. — Access mode: https://yaml.org.

4. ГОСТ Р МЭК 61131-3-2016 КОНТРОЛЛЕРЫ ПРОГРАММИРУЕМЫЕ Часть 3. Языки программирования. — 2016.

ПЕРЕЧЕНЬ СОКРАЩЕНИЙ

DHCP – Dynamic Host Configuration Protocol

IPv4 – протокол Интернет версии 4

IPv6 – протокол Интернет версии 6

ПО – программное обеспечение

ПЛК – программируемый логический контроллер

УСО – устройство связи с объектом