Yacht Devices

Руководство пользователя

Устройства для яхт NMEA 2000 Bridge YDNB-07

также охватывает модели

YDNB-07N, YDNB-07R

Версия прошивки

1.42

2022

Содержание

Комплектация

Устройство	1 шт.
Настоящее руководство	1 шт.
Наклейки для герметизации слота MicroSD	6 шт.

Введение

NMEA-мост Yacht Devices объединяет две физические сети NMEA 2000 в одну логическую сеть, обеспечивая бесперебойный обмен сообщениями между ними. Устройство также позволяет организовать произвольную фильтрацию, обработку и маршрутизацию любых сообщений шины CAN.

Это позволяет решать следующие задачи:

1. **Обход физических ограничений сетей NMEA 2000**, касающихся длины сетей (100 метров для обычного кабеля и 250 метров для кабеля тяжелого или среднего типа) и максимального количества (50) физических устройств, подключенных к сети. В сети с адресной емкостью 252 можно задействовать несколько мостов для расширения до примерно 250 физических устройств.

2. **Изолируйте устройства друг от друга.** С помощью простого фильтра можно заблокировать передачу всех или выбранных сообщений от определенного устройства в отдельной подсети.

3. **Обеспечьте правильную работу оборудования.** Корректируйте смещение датчика эхолота или «удаляйте» недостоверные данные в сообщениях от оборудования, работающего лишь частично, с помощью 2- или 3-строчного скрипта.

4. **Для обеспечения совместимости оборудования** разных поколений. Вы можете создавать и отправлять сообщения NMEA 2000 любого типа, используя данные из других сообщений в сети.

5. **Диагностика неисправностей в сети NMEA 2000.** Устройство может записывать сетевые сообщения и отладочные данные из пользовательских программ на карту MicroSD в текстовый файл. Эти данные можно просматривать в стандартном текстовом редакторе на смартфоне или планшете с слотом для карт MicroSD — компьютер не требуется. Вы даже можете создавать и редактировать программы для устройства прямо на своем телефоне!

6. **Создавайте пользовательские шлюзы**, не соответствующие стандартам NMEA 2000. Один из CAN-интерфейсов устройства имеет высоковольтную гальваническую развязку и может работать при более высоком напряжении питания.

7. **Создавайте пользовательские шлюзы,** соединяющие две сети CAN с произвольными протоколами. Поддерживаются как 11-битные (CAN A), так и 29-битные (CAN B) кадры. Одна из сетей может работать на любой скорости передачи данных от 50 кбит/с до 1000 кбит/с. Хотя язык программирования устройства не был разработан для полноценных приложений CAN, можно создать, например, шлюз между системой пропульсии на базе CAN, зарядным устройством аккумулятора или сервоприводом и NMEA 2000.

8. **Создайте собственное устройство обработки данных NMEA 2000,** например, реализуйте сложные правила цифрового переключения для оборудования цифрового переключения NMEA 2000 от Yacht Devices, такие как: переключение с задержкой и по времени, автоматическое отключение, условное переключение или вывод предупреждений на картплоттере, когда определенные значения данных превышают пороговое значение.

Программирование устройства требует знания NMEA 2000. Копию стандарта NMEA 2000 можно приобрести у Национальной ассоциации морской электроники: http://www.nmea.org.

Компания Yacht Devices обращает ваше внимание на то, что в сети NMEA 2000 могут находиться такие важные устройства, как эхолот, магнитный компас и автопилот. Выход из строя или некорректная работа этих устройств может привести к серьезным авариям и гибели людей. При программировании устройства вы должны полностью осознавать все возможные последствия. Перед проведением морских испытаний необходимо организовать обязательное обучение экипажа судна.

Гарантия и техническая поддержка

1. Гарантия на устройство действует в течение двух лет с даты покупки. Если устройство было приобретено в розничном магазине, при обращении по гарантии может потребоваться предъявить чек.

2. Гарантия на Устройство аннулируется в случае нарушения инструкций, приведенных в данном Руководстве, повреждения корпуса, а также ремонта или модификации Устройства без письменного разрешения производителя.

3. Если заявка на гарантийное обслуживание принята, неисправное Устройство необходимо отправить производителю.

4. Гарантийные обязательства включают ремонт и замену товара и не включают расходы на установку и настройку оборудования, а также на отправку неисправного Устройства производителю.

5. Ответственность производителя в случае любого ущерба, возникшего в результате эксплуатации или установки Устройства, ограничивается стоимостью Устройства.

6. Производитель не несет ответственности за любые ошибки и неточности в руководствах и инструкциях других компаний.

7. Устройство не требует технического обслуживания. Корпус устройства не подлежит разборке.

8. В случае неисправности, прежде чем обращаться в службу технической поддержки, ознакомьтесь с Приложением А.

9. Производитель принимает заявки по гарантии и оказывает техническую поддержку исключительно по электронной почте или через официальных дилеров.

10. Контактные данные производителя и список официальных дилеров опубликованы на сайте: http://www.yachtd.ru/.

I. Технические характеристики

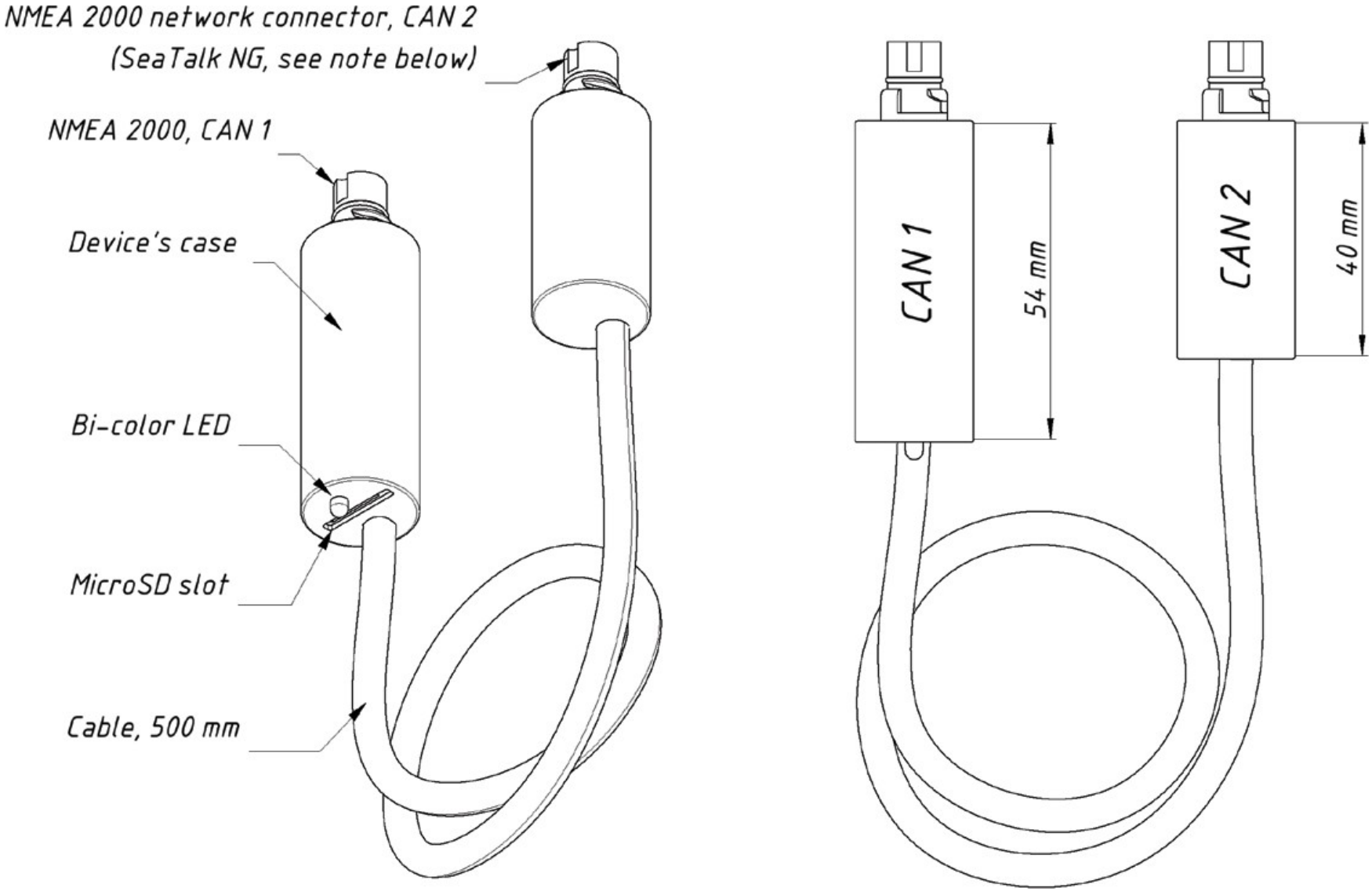

Рисунок 1. Чертеж моста модели YDNB-07R

Наши устройства поставляются с разъемами NMEA 2000 различных типов. Модели с суффиксом R в конце названия оснащены разъемами NMEA 2000, совместимыми с Raymarine SeaTalk NG (как показано на изображении выше). Модели с суффиксом N оснащены разъемами NMEA 2000 Micro Male (см. Приложение C).

Параметр устройства	**Значение**	**Единица**
Напряжение питания, интерфейс CAN1	9..16	В
Средний ток потребления, CAN1	38	мА
Коэффициент эквивалентности нагрузки, CAN1	1	LEN
Напряжение питания, интерфейс CAN2	9..30	В
Средний ток потребления, CAN2	13	мА
Коэффициент эквивалентности нагрузки, CAN2	1	LEN
Гальваническая развязка, прочность изоляции между CAN1 и CAN2	2500	Врмс
Защита от обратной полярности на обоих интерфейсах CAN1 и CAN2	Да	—
Диапазон рабочих температур	-20..55	°C
Длина кабеля	500	мм
Длина корпуса устройства (без разъема) CAN1 / CAN2	54/40	мм
Вес без карты MicroSD	52	г

Компания Yacht Devices Ltd заявляет, что данный продукт соответствует основным требованиям директивы по электромагнитной совместимости 2004/108/EC.

Утилизируйте данный продукт в соответствии с директивой WEEE. Не смешивайте электронные отходы с бытовыми или промышленными отходами.

II. Установка и подключение к сетям NMEA 2000 или CAN

Ни в коем случае не подключайте оба разъема устройства к одной и той же сети NMEA 2000 или CAN. Это может привести к перегрузке сети бесконечной пересылкой сообщений и вызвать временный сбой в работе сети.

Устройство не требует технического обслуживания. При выборе места установки устройства выбирайте сухое место. Избегайте мест, где устройство может оказаться под водой; это может привести к его повреждению.

Устройство можно подключить непосредственно к разъему магистрали сети. Модель YDNG-07N также можно подключить через стандартный отводной кабель «DeviceNet NMEA 2000 Micro». Если у вас есть шина CAN с нестандартным физическим уровнем, подключите шины CAN и шины питания в соответствии с распиновкой устройства, приведенной в Приложении C. Перед подключением устройства отключите питание шины. Если у вас есть вопросы по использованию разъемов, обратитесь к документации производителя:

- Справочное руководство SeaTalk NG (81300-1) для сетей Raymarine
- Техническое руководство по продуктам Garmin NMEA 2000 (190-00891-00) для сетей Garmin. После подключения

устройства закройте защелку соединения, чтобы обеспечить водонепроницаемость и надежность.

Микроконтроллер устройства питается от интерфейса CAN1. Устройство не будет работать, пока интерфейс CAN1 не получит питание. Если вы хотите запустить устройство для ознакомления или реализуете пользовательское устройство обработки данных, которое по замыслу не требует обмена данными между интерфейсами CAN1 и CAN2, интерфейс CAN2 можно оставить неподключенным.

На устройстве имеется светодиод, который мигает красным или зеленым цветом. При подаче питания на интерфейс CAN1 устройства светодиод выдаст серию из 2 вспышек с интервалом 5 секунд. Если этого не произошло, см. Приложение A.

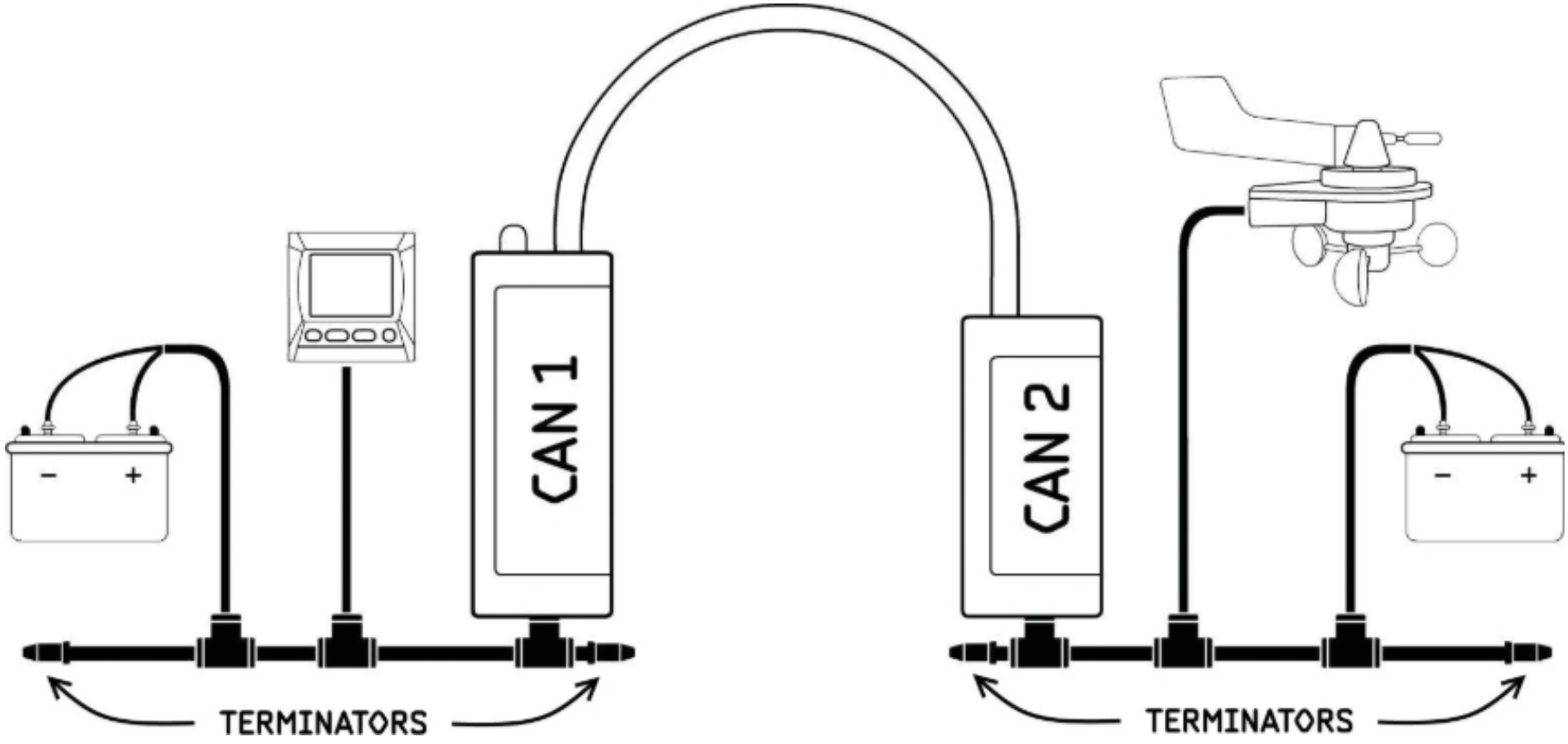

Рисунок 2. Типичная схема сети приложения моста NMEA 2000

Помните, что для сети NMEA 2000 требуется отдельный источник питания и два терминатора — по одному на каждом конце магистрали. Если устройство используется в качестве моста, соединяющего два существующих сегмента сети NMEA 2000, необходимо установить терминаторы на каждом сегменте магистрали и обеспечить питание обоих сегментов от источника питания 12 В. Более подробную информацию по этой теме можно найти в перечисленных выше документах компаний Raymarine и Garmin.

III. Светодиодные сигналы

1. **Сигнал с периодичностью 5 секунд: светодиод мигает два раза.** Первый миг указывает на состояние сети интерфейса CAN1. Зеленый миг светодиода означает, что в течение последних 5 секунд по интерфейсу CAN1 были получены или успешно отправлены данные; красный — в противном случае. Второй миг указывает на состояние сети интерфейса CAN2.

Устройство можно настроить на прием только ограниченного набора сообщений NMEA 2000 (см. раздел VII.3), остальные сообщения отфильтровываются на аппаратном уровне. В связи с этим в некоторых сетях NMEA 2000 красный индикатор может гореть большую часть времени, даже если сеть функционирует нормально. В этом случае, чтобы проверить соединение, выключите и снова включите одно из устройств, находящихся в сети (например, картплоттер). Состояние сети будет отображаться миганием зеленого индикатора в течение некоторого времени, пока устройство включается и подключается.

2. **Три мигания (цвет может отличаться), один раз после установки карты MicroSD в устройство.** См. раздел V.

3. **Длительная вспышка (3 секунды), красная или зеленая.** Режим диагностики запущен/завершен, см. раздел X.

4. **Пять зеленых вспышек при включении сети NMEA 2000.** В устройство вставлена карта MicroSD с обновлением прошивки, прошивка обновляется (см. раздел XI).

IV. Слот MicroSD и совместимость карт

Устройство имеет слот для карты MicroSD, который позволяет настраивать устройство (см. раздел V) и обновлять прошивку (см. раздел XI).

Слот устройства оснащен механизмом «push-push», работающим на пружине и обеспечивающим надёжную фиксацию карты. Неправильная загрузка или извлечение карты (слишком быстрое извлечение пальца или отсутствие ожидания щелчка) может привести к выбросу карты из устройства на расстояние до 5 метров. Во избежание возможных травм глаз, потери или повреждения карты, а также других опасностей, вставляйте и извлекайте карту с осторожностью.

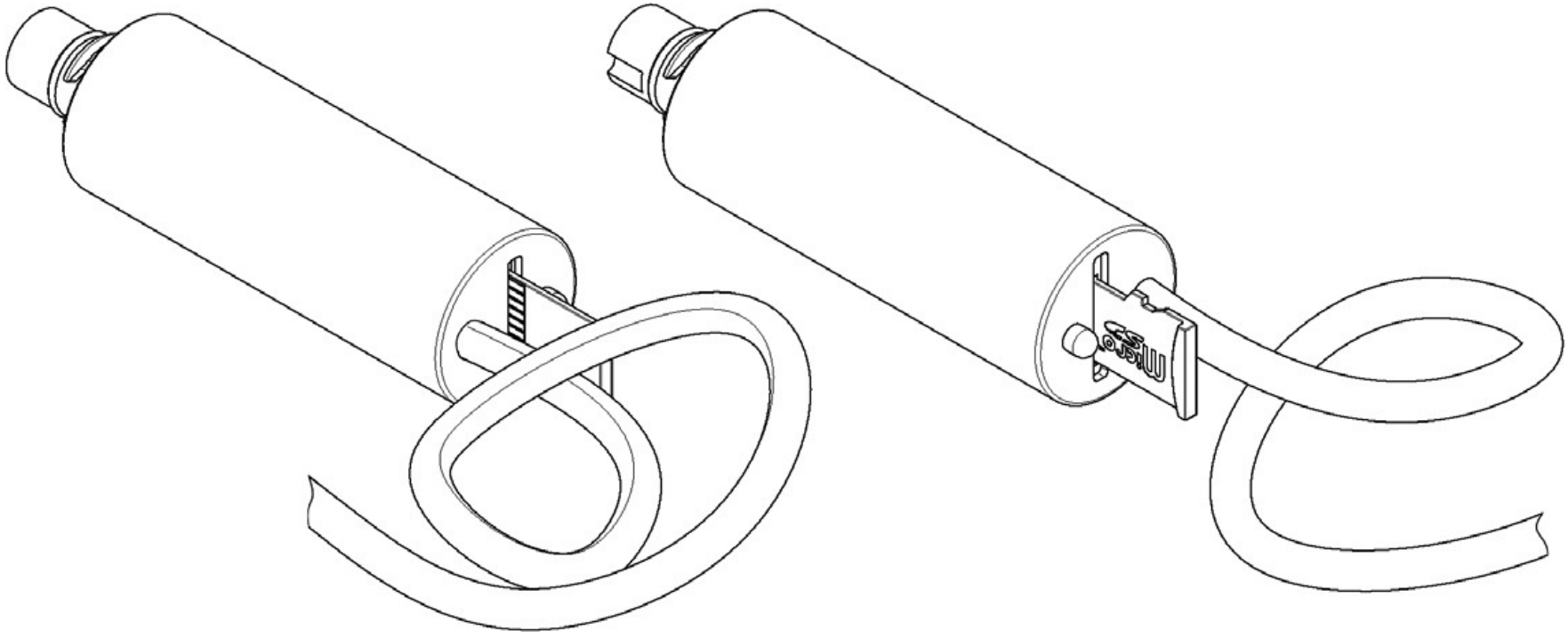

Рисунок 1. Устройство с картой MicroSD (сторона с контактными площадками видна слева, сторона с этикеткой — справа)

Поскольку слот для карт MicroSD обычно не используется во время работы устройства, мы рекомендуем закрыть его наклейкой, входящей в комплект поставки, или кусочком клейкой ленты, чтобы предотвратить попадание воды в устройство через этот слот.

Устройство поддерживает карты памяти MicroSD любой емкости и класса. Перед использованием в устройстве карту MicroSD необходимо отформатировать на персональном компьютере. Устройство поддерживает следующие файловые системы: FAT (FAT12, FAT16, MS-DOS) и FAT32. Оно не поддерживает exFAT, NTFS или любые другие файловые системы.

Будьте осторожны при установке карты MicroSD в устройство. Карту следует вставлять так, чтобы сторона с надписью была обращена к светодиоду, а сторона с контактами — к кабелю.

V. Загрузка программ в устройство

Поместите файл YDNB.CFG с программой в корневой каталог карты MicroSD с файловой системой FAT или FAT32. Вставьте карту MicroSD в устройство. Через несколько секунд вы должны увидеть три мигания светодиода:

1. **Три красных мигания** означают, что карта не считывается.

2. **Один зеленый мигающий сигнал, за которым следуют два красных,** указывает на то, что файл YDNB.CFG не найден на карте памяти, и текущая конфигурация устройства была сохранена в файл YDNBSAVE.CFG.

3. **Один красный, за которым следует один зеленый и еще один красный мигающий сигнал,** указывает на то, что файл YDNB.CFG содержит ошибки и не был принят устройством. В корневом каталоге карты памяти создан текстовый файл YDNBERR.TXT, содержащий журнал ошибок.

4. **Три зеленых вспышки** означают, что файл успешно загружен в устройство. В корневом каталоге карты создан текстовый файл YDNBSAVE.CFG, содержащий текущую программу и используемые настройки.

Устройство выполняет компиляцию текста программы в байт-код. Перед тем как сохранить программу в файле YDNBSAVE.CFG, устройство декомпилирует байт-код обратно в текст. Именно поэтому содержимое файлов YDNB.CFG и YDNBSAVE.CFG может отличаться друг от друга.

Файл YDNB.CFG должен содержать хотя бы одну интерпретируемую строку кода (настройку, фильтр и т.д.) без ошибок, чтобы его можно было загрузить в устройство.

Чтобы изменить текущую программу, вставьте в устройство карту памяти, на которой нет файла YDNB.CFG. Светодиод устройства будет мигать зеленым, красным, красным. Это означает, что файл YDNBSAVE.CFG, содержащий текущую программу, был сохранен на карту. Эту программу можно изменить, сохранить как YDNB.CFG и загрузить обратно в устройство.

VI. Структура и основной синтаксис программы

Программа определяет алгоритмы и правила обработки и пересылки сообщений NMEA 2000 или других сообщений CAN, которые устройство принимает через интерфейсы CAN1 и CAN2.

Программа состоит из настроек, встроенных функций, подпрограмм фильтров и комментариев. Настройки, функции и фильтры подробно описаны в последующих разделах данного руководства.

Комментарии в программе добавляются после символа #. Комментарии могут располагаться как в начале строки, так и после интерпретируемого текста программы.

Параметры можно задавать в любом месте программы, за исключением подпрограмм фильтров. Тем не менее, мы рекомендуем объявлять параметры перед фильтрами.

Пример 1.

```
# Пример N1

FW_CAN1_TO_CAN2=ON                  # Разрешить пересылку всех несовпадающих сообщений
FW_CAN2_TO_CAN1=OFF                 # Настройка для другого направления

match(CAN2,0x01FD0600,0x01FFFF00) # 1-й фильтр
{
      # Пустая подпрограмма, совпадающие сообщения будут отбрасываться
}

match(CAN2,0x00000010,0x000000FF) # 2-й фильтр
{
      send(CAN1)                    # Пересылка совпавшего сообщения на интерфейс CAN1
}

match(CAN1,0x00000020,0x000000FF) # 3-й фильтр (для CAN1)
{
      # Без send(), совпадающие сообщения будут отбрасываться
}
# Конец программы
```

Фильтр состоит из заголовка, который начинается с ключевого слова *match()* с параметрами, определяющими «критерии сопоставления» входящих CAN-фреймов, за которым следует подпрограмма обработки данных на специально созданном языке программирования.

Порядок фильтров имеет значение. Устройство будет сопоставлять принимаемые сообщения с фильтрами в точном соответствии с тем порядком, в котором они указаны в программе. После успешного сопоставления сообщение будет обработано соответствующей подпрограммой фильтра и не будет обрабатываться последующими фильтрами.

Если сопоставления с каким-либо фильтром не происходит, устройство следует настройкам пересылки. Если пересылка сообщений включена для интерфейса (по умолчанию — да), данное сообщение будет отправлено на другой интерфейс CAN. Если пересылка отключена, сообщение будет отброшено.

Далее мы объясним работу этой программы.

Программа в *примере 1* выше имеет только две настройки пересылки и три фильтра.

Во-первых, у нас есть два правила переадресации по умолчанию — они будут применяться к сообщениям, для которых не было определено соответствующего подпрограммы `match()`. Параметр FW_CAN1_TO_CAN2=ON указывает устройству перенаправлять все сообщения, полученные по интерфейсу CAN1, на интерфейс CAN2. Параметр FW_CAN2_TO_CAN1=OFF указывает устройству не перенаправлять (отбрасывать) все сообщения, полученные по интерфейсу CAN2, на интерфейс CAN1.

Первый фильтр перехватит PGN 0x1FD06, полученный на интерфейсе CAN2 — с любого адреса устройства, однако, поскольку его подпрограмма пуста, этот PGN будет отброшен.

Второй фильтр перехватит любой PGN, полученный на интерфейсе CAN2 — но только от устройства с адресом 0x10 — и, как предписывает подпрограмма, отправит сообщение на интерфейс CAN 1. Поскольку PGN 0x1FD06 уже был обработан первым фильтром, пересылка отключена с помощью FW_CAN2_TO_CAN1=OFF, а других фильтров для CAN2 нет, в результате все PGN от устройства CAN2 с адресом 0x10 будут пересылаться на CAN1, за исключением 0x1FD06.

Третий фильтр будет перехватывать все PGN-сообщения, поступающие по интерфейсу CAN1 от устройства с адресом 0x20, и блокировать их. Поскольку пересылка с CAN1 на CAN2 включена с помощью параметра FW_CAN1_TO_CAN2=ON, в итоге все PGN-сообщения с CAN1 будут пересылаться на CAN2, за исключением сообщений от устройства 0x20.

VII. Описание настроек

Обратите внимание, что в приведенных ниже описаниях для разделения альтернативных значений настроек используется вертикальная черта (труба).
ON|OFF означает, что для данного параметра возможны два значения — ON или OFF.

1. Пересылка сообщений

```
FW_CAN1_TO_CAN2=ON|OFF FW_CAN2_TO_CAN1=ON|OFF
```

Устанавливает правила пересылки по умолчанию — определяет поведение для сообщений, не имеющих соответствующих `фильтров match()`
фильтров. ON — пересылать, OFF — не пересылать (отбрасывать).

2. Сборка сообщений NMEA 2000 с быстрыми пакетами

```
PGNS_TO_ASSEMBLY=x
```

x — от одного до пяти PGN, введенных в виде десятичных или шестнадцатеричных значений, разделенных запятыми.

Стандарт NMEA 2000 определяет специальное расширение протокола для PGN с общей длиной полезных данных более 8 байт (от 9 до 223 байт) — полезные данные передаются в виде последовательности стандартных CAN-кадров, каждый из которых содержит от 1 до 8 байт полезных данных (см. стандарт NMEA 2000, раздел «3.1 Сообщения с быстрой передачей пакетов»).

Устройство может «собирать» PGN-сообщения NMEA 2000 с быстрыми пакетами из серии CAN-фреймов и предоставлять программе полную полезную нагрузку PGN. Сообщения, не собранные полностью, будут отбрасываться. Однако сборка требует большого объема оперативной памяти, поэтому для сборки можно выбрать не более пяти PGN.

Обратите внимание, что сборка также занимает время, так как устройство будет ждать, пока не будет получен последний кадр CAN, прежде чем передать собранный PGN программе, что приведет к задержке. По возможности используйте метод обработки данных по кадрам (см. раздел IX).

Стандарт NMEA 2000 не предписывает строгого порядка передачи кадров быстрых пакетов по шине. Контроллер CAN устройства имеет 3 «ящика исходящих сообщений», поэтому кадрам CAN с быстрыми пакетами можно помещать в разные «ящики» в исходящей очереди, что может привести к отправке кадров с быстрыми пакетами не в том порядке. Некоторое оборудование NMEA 2000 не может хорошо справляться с этой ситуацией; в таких случаях используйте следующую настройку управления потоком данных:

```
STRICT_QUEUE=OFF|CAN1|CAN2|ON
```

Эта настройка принудительно устанавливает строгий порядок *исходящих* быстрых пакетов для интерфейсов CAN1, CAN2 или обоих (ON). По умолчанию эта функция отключена (OFF). Включение этой функции приводит к небольшому снижению производительности.

3. Аппаратные фильтры

```
CAN1_HARDWARE_FILTER_y=f,m CAN2_HARDWARE_FILTER_y=f,m
```

y — номер аппаратного фильтра, десятичное число от 1 до 7;
f — значение фильтра, десятичное или шестнадцатеричное число (29 значимых битов);
m — маска фильтра, десятичное или шестнадцатеричное число (29 значимых битов).

Устройство может фильтровать сообщения, поступающие с интерфейсов CAN1 и CAN2, на аппаратном уровне, что в некоторых случаях позволяет снизить нагрузку на микропроцессор в 100 раз. Отбор сообщений осуществляется по 29-битному идентификатору кадра CAN (CAN ID), который содержит приоритет сообщения, PGN, адрес отправителя и (в некоторых случаях) адрес получателя.

Формальное выражение сравнения для аппаратных и программных фильтров:

```
if ( ( CAN_FRAME_ID AND mask ) == filter ) then MATCH else NO_MATCH
```

Сообщения передаются в программу только в том случае, если они соответствуют одному из аппаратных фильтров. Для каждого интерфейса можно настроить до 7 пользовательских аппаратных фильтров с номерами от 1 до 7.

Устройство также имеет системный аппаратный фильтр № 0, который пользователь не может изменить:

```
CAN1_HARDWARE_FILTER_0=0x00E80000, 0x01F90000 CAN2_HARDWARE_FILTER_0=0x00E80000, 0x01F90000
```

Фильтр (первый параметр) задает биты для сравнения с идентификатором сообщения, а маска (второй параметр) указывает биты, результат сравнения которых является значимым. (1 — соответствующий по позиции бит идентификатора сообщения CAN должен точно совпадать с битом, установленным в фильтре; 0 — может быть любым).

Таким образом, аппаратный фильтр системы пропускает только сообщения со следующими PGN: 0xE800 (подтверждение ISO), 0xEA00 (запрос ISO), 0xEC00 (транспортный протокол ISO), 0xEE00 (запрос адреса ISO).

Если в программе не определён пользовательский аппаратный фильтр для интерфейса, для данного интерфейса автоматически добавляется фильтр, пропускающий все сообщения:

```
CAN1_HARDWARE_FILTER_1=0,0
```

Таким образом, если в программе не задан пользовательский аппаратный фильтр, все сообщения будут передаваться в программу.

4. Экземпляр NMEA 2000

```
DEVICE_INSTANCE=x SYSTEM_INSTANCE=y
```

x — число от 0 до 255, y — число от 0 до 15.

Эти настройки позволяют задать значения NMEA 2000 «Device Instance» и «System Instance» устройства, которые используются для генерации NMEA 2000 «Device Information» (NAME), передаваемой в сообщении ISO Address Claim.

Значение по умолчанию для обоих параметров равно 0. Эти настройки не будут сохранены в файл YDNBSAVE.CFG, если они имеют значения по умолчанию.

Обратите внимание, что при скорости передачи данных CAN2, установленной на 250 (CAN2_SPEED=250), установка SYSTEM_INSTANCE=8 (или выше) предотвращает отправку собственного PGN «Address Claim» устройства по CAN2

5. Инициализация слотов сообщений

`SLOTx=aabbccdd…` или `SLOTx=aa bb cc dd…`

x — номер слота, 1 – 3;
aabbccdd — последовательность байтов (1 – 229), шестнадцатеричные значения.

Устройство имеет 3 буфера, называемых «SLOTS», которые могут использоваться в качестве шаблонов исходящих CAN-фреймов или в качестве буферов хранения данных. Настройки SLOTx позволяют заполнять буферы слотов заранее заданными данными до запуска программы. Формат буфера описан в VIII.2. Содержимое слота можно загрузить в главный буфер сообщений (DATA) с помощью функции `load()` и перезаписать содержимое слота с помощью функции `save()` (см. VIII.8)

Пример неадресованного запроса ISO на заявку на адрес ISO:

```
SLOT1= 00FFEA18 FF 03 00EE00
```

Здесь 00FFEA18 — это CAN-идентификатор для «ISO Address Claim» в порядке байтов с младшим битом впереди (приоритет сообщения + номер PGN + адрес устройства = 18EAFF00); FF — маркер PGN для одиночного кадра, 03 — длина полезных данных в байтах, 00EE00 — полезные данные PGN.

6. Скорость интерфейса CAN2

`CAN2_SPEED=x`

x — скорость в кбит/с, заводская настройка — 250, допустимые значения — 50, 125, 250, 500 и 1000.

Скорость интерфейса CAN2 можно установить в диапазоне от 50 до 1000 кбит/с (NMEA 2000 имеет скорость 250 кбит/с). Это обеспечивает совместимость с различными сетями CAN. Скорость интерфейса CAN1 не может быть изменена, она имеет фиксированную скорость 250 кбит/с.

VIII. Программирование устройства

Мы создали специальный язык программирования для устройства, похожий на распространенные языки сценариев. Существует ряд ограничений, налагаемых задачами и вычислительными возможностями микроконтроллера.

Программа состоит из текста в формате ASCII, каждая строка представляет собой либо оператор, либо комментарий, либо пустую строку. Каждый новый оператор должен начинаться с новой строки. «Переносы строк» не поддерживаются. Специального символа «окончания оператора» нет.

Язык программирования поддерживает целые и рациональные числа, как положительные, так и отрицательные. Разделителем десятичной части для рациональных чисел является точка. Шестнадцатеричные числа должны иметь префикс 0x (x строчной буквы).

Допускается использование 7-битных символов ASCII в одинарных кавычках (типа 'a', '1'), они будут интерпретироваться как соответствующий номер символа ASCII (например: 'a' == 97).

Зарезервированные идентификаторы, такие как константы (DATA, M_PI), идентификаторы интерфейсов (CAN1, CAN2) и идентификаторы типов данных (UINT32, FLOAT и т. д.), должны вводиться заглавными буквами.

Язык программирования поддерживает только одну пользовательскую функцию `func()`, кроме того, можно вызывать встроенные функции `get(), set(), cast(), send(), sin()` и другие.

1. Определение фильтров

Фильтр определяется ключевым словом `match()` с тремя параметрами в скобках и кодом подпрограммы в фигурных скобках. Подпрограмма может быть пустой:

```
match(interface_id, filter, mask)
{
}
```

interface_id — идентификатор интерфейса: CAN1, CAN2 или ANY; filter и mask — числовые значения фильтра и маски.

Формальное выражение сравнения для аппаратных и программных фильтров:

```
if ( ( CAN_FRAME_ID AND mask ) == filter ) then MATCH else NO_MATCH
```

Например, этот фильтр

```
match(ANY, 0x00000000, 0x01FFF800)
{
}
```

будет соответствовать всем 11-битным кадрам CAN (для всех идентификаторов CAN в диапазоне 0x000...0x7FF будет найдено совпадение, поскольку для всех таких идентификаторов выражение AND 0x01FFF800 дает в результате 0)

Ключевое слово `match()` не транслируется в байт-код, поэтому сравнение указанных фильтров с полученными сообщениями происходит очень быстро.

Программа может содержать до 20 фильтров.

2. Буфер сообщений

Подпрограмма имеет доступ к сообщению и его 29-битному идентификатору CAN, которые хранятся в буфере, описанном в таблице 1.

Таблица 1. Структура буфера сообщений

Смещение	Длина в байтах	Описание
0	4	29-битный идентификатор сообщения CAN (младший бит первым)
4	1	0xFF – индикатор однокадрового сообщения NMEA 2000 (сообщение CAN с 29-битным идентификатором); 0xfe – индикатор сообщения CAN с 11-битным идентификатором; 0x00..0xE0 с шагом 0x20 – индикатор быстрого пакета NMEA 2000 (счетчик последовательности); остальные значения зарезервированы
5	1	Длина сообщения (1..223)
6	223	Данные сообщения

Изменяя значения в буфере, подпрограмма может модифицировать сообщения и даже создавать новые. Независимо от фактической длины полученного сообщения, подпрограмма имеет доступ ко всем 229 байтам буфера.

Обратите внимание: если PGN быстрого пакета NMEA 2000 не включен в список PGNS_TO_ASSEMBLY, программа будет вызываться для каждого CAN-сообщения в последовательности, а байт со смещением 4 будет установлен в 0xFF.

3. Отправка сообщения

Для отправки сообщения, хранящегося в буфере, используется встроенная функция `send()`:

```
send(interface_id)
```

interface_id — интерфейс, через который отправляется сообщение, CAN1 или CAN2. Этот параметр можно опустить, в этом случае сообщение будет отправлено через интерфейс, противоположный тому, с которого было получено обрабатываемое сообщение.

Чтобы избежать досадных ошибок с непредсказуемыми последствиями, мы рекомендуем всегда использовать `send()` без параметра, где это возможно.

Обратите внимание, что подпрограмма фильтра отвечает за пересылку обрабатываемого сообщения. Если ваша подпрограмма не содержит вызова `send()`, то обработанное сообщение не будет переслано.

4. Переменные, операторы и выражения

Программисту доступно 26 глобальных переменных с именами от A до Z. При написании имен переменных не учитывается регистр. Переменные могут иметь один из следующих типов, которые определяются неявно и автоматически преобразуются в необходимый тип при выполнении операций.

- INT8 — 1-байтовое целое число со знаком;
- UINT8 — 1-байтовое целое число без знака;
- INT16 — 2-байтовое целое число со знаком;
- UINT16 — 2-байтовое целое число без знака;
- INT32 — 4-байтовое целое число со знаком;

- UINT32 — 4-байтовое целое число без знака;
- FLOAT — 4-байтовое число с плавающей запятой (IEEE 754-1985, одинарная точность).

При включении устройства все переменные инициализируются значением 0 типа INT32.

Поддерживаются следующие арифметические, логические и двоичные операции (перечислены в порядке приоритета):

- * (умножение), / (деление), % (остаток от целочисленного деления)
- + (сложение), - (вычитание), << (бинарный сдвиг влево), >> (бинарный сдвиг вправо), | (логическое ИЛИ), ^ (XOR), & (логическое AND)
- = (присваивание)

С помощью скобок можно изменить порядок выполнения действий:

```
A = (3 + 5) * 2 # Эквивалентно A = 16
```

При присвоении значений переменным тип результата выбирается автоматически:

```
A = 1                   # INT32
B = 2.0                 # FLOAT
C = '1'                 # 0x31, UINT8
D = B * C               # 0x31 * 2.0 = 49 * 2.0 = 98.0, FLOAT
E = 0xFF00AA3F          # UINT32
```

Встроенная функция cast() позволяет явно задать тип выражения:

```
[type] cast(expression,type)
```

Где *type* — один из типов, перечисленных выше. Например:

```
B = cast(1.5, INT8)                 # 1, INT8
C = cast(B << 2, FLOAT)             # 4.0, FLOAT
```

Обратите внимание, что компилятор не оптимизирует алгоритмы, не пытается выполнять вычисления эффективно.

5. Изменение буфера сообщений

Для изменения сообщения в буфере и чтения данных из него предусмотрены, соответственно, две встроенные функции `set()` и `get()`:

```
set(выражение1, тип, выражение2) [тип]
get(выражение1, тип)
```

выражение1 — выражение, преобразованное в тип UINT8, смещение первого байта данных в буфере; тип — идентификатор типа данных (см. VIII.4);
выражение2 — выражение, результат которого будет приведен к указанному типу и помещен в буфер со смещением, указанным в выражении1.

Адресация данных в буфере начинается с нуля. Для более удобного доступа к данным рекомендуется использовать встроенную константу DATA, которая указывает на первый байт полезной части сообщения.

Пример 2.

```
match(CAN1,0x1F50B00,0x1FFFF00)
{
      A = get(DATA+1, UINT32)              # получить 4-байтовое целое число из сообщения
      set(DATA+1, UINT32,A + 20) # установить исправленное значение
      send()                                    # отправить сообщение на CAN2
}
```

В примере 2 реализован фильтр для параметра «Глубина воды» (PGN 0x1F50B). Подпрограмма извлекает значение глубины (четырехбайтовое целое число, байты 1–4 данных полезной нагрузки входящего сообщения) и записывает это значение обратно в буфер, прибавив 20. После этого сообщение отправляется (на интерфейс CAN2). В результате показания глубины в сети CAN 2 будут увеличены на 20 сантиметров.

6. Операторы управления потоком `if()` и `while()`

Условная инструкция `if()` и инструкция цикла `while()` поддерживают следующие операторы сравнения: > (больше), < (меньше), == (равно), != (не равно), <= (меньше или равно), >= (больше или равно). Блок *else* является опциональным:

```
if (выражение1 оператор выражение2) { # выполнить действия
}
else
{

        # выполнить другие действия
}
```

В примере 2 выше не учтено, что датчик глубины может передавать специальные значения глубины: 0xFFFFFFFF («не доступно»), 0xFFFFFFFE («вне диапазона») или 0xFFFFFFFD («зарезервировано»). С помощью оператора `if()` мы проверим, находится ли значение данных в допустимом диапазоне:

Пример 3.

```
match(CAN1,0x1F50B00,0x1FFFF00)
{
        A = get(DATA+1, UINT32)                 # получить значение глубины
                                                # в виде 4-байтового целого числа из буфера

        if (A < 0xFFFFFFFD-20)                  # значение + требуемое смещение 20 см допустимо?
        {
                set(DATA+1, UINT32,A + 20) # добавить смещение 20 см
        }
        send()                          # отправить сообщение на CAN2

}
```

Устройство поддерживает оператор управления потоком `while()` для создания циклов предварительной проверки:

```
while (выражение1 оператор выражение2) { # выполнить действия
}
```

Операторы сравнения такие же, как и в операторе `if()`. Поскольку `while()` может создавать бесконечный цикл, через 3 секунды выполнение цикла будет прервано, и выполнение программы продолжится за пределами

цикла. Если достигнуто максимальное время выполнения, устройство выдает сообщение «FUNC Execution timeout (3000 ms)» в текстовый журнал диагностики.

7. Функции реального времени

Большинство элементов данных NMEA 2000 имеют короткий срок действия. Необходимо отслеживать временные метки полученных элементов данных, чтобы определить, остаются ли данные актуальными (например, чтобы не использовать устаревшие значения в случае, если устройство-источник данных отключается от шины и/или прекращает передачу данных)

```
UINT32 timer()
UINT32 timediff(выражение)
```

Функция `timer()` возвращает количество миллисекунд, прошедших с момента включения устройства. Значение таймера переполняется (сбрасывается в нуль) примерно каждые 49 дней. Функция `timediff()` возвращает разницу (в миллисекундах) между текущим значением таймера и значением временной метки, переданным в качестве аргумента. Функция может обрабатывать переполнение таймера и вернет правильное значение разницы во времени — если разница составляет менее 49 дней. См. также: `heartbeat()`, VIII.11.

8. Буферы хранения

```
save(slot) load(slot)
```

Функция `Save(slot)` сохраняет данные из буфера сообщений в указанный буфер SLOTx. Функция `Load(slot)` перезаписывает текущий буфер сообщений данными, полученными из указанного буфера SLOTx. Поддерживаются три идентификатора SLOT: SLOT1, SLOT2, SLOT3. Примечание: содержимое SLOT можно задать в начале программы (см. VII.5).

9. Получение адреса моста

```
UINT8 addr()
```

Функция `addr()` возвращает текущий адрес устройства NMEA 2000 моста; она имеет одинаковое значение как на интерфейсе CAN1, так и на CAN2.

Используйте, когда вам нужно изменить адрес устройства-источника PGN и сделать так, чтобы они выглядели так, будто они были отправлены самим мостом.

Пример 4.

```
# Обработка сообщений о фактическом давлении (PGN 0x1FD0A) и генерация # сообщений о параметрах окружающей среды (PGN 0x1FD06)

match(CAN2,0x1FD0A00,0x1FFFF00)
{
      send()                                                  # Переслать исходное сообщение
      A = get(DATA + 2, UINT8)                                # Извлечь тип данных
      if (A == 0)                                             # Это атмосферное давление?
      {
            B = get(DATA + 3, INT32)                          # Извлечь значение давления
            if (B < 0x7FFFFFFD)                               # Проверить достоверность значения
            {
                  set(1, UINT8,6)                             # Изменить PGN с 0x1FD0A на 0x1FD06 set(DATA + 1,
                  UINT32, 0xFFFFFFFF)                         # Установить значения в неиспользуемые поля set(DATA + 5,
                  UINT16, B / 1000)                           # Преобразовать и установить давление send(CAN1)          #
                  Отправить параметры окружающей среды на CAN1 set(0, UINT8, addr())              # Заменить адрес
                  на адрес
                                                              # моста
                  send(CAN2)                                  # Отправить параметры окружающей среды на CAN2
            }
      }
}
```

Программа из примера 4 принимает PGN «Фактическое давление» (0x1FD0A) от интерфейса CAN2 и пересылает его на интерфейс CAN1. Предположим, что в подсети CAN1 имеется устаревшее устройство, которое не понимает PGN «Фактическое давление», поэтому Bridge сгенерирует дополнительный PGN «Параметры окружающей среды» 0x1FD06 для атмосферного давления и отправит его в CAN1, используя свой собственный адрес устройства. Исходный PGN 0x1FD0A будет отправлен «как есть», без изменения адреса устройства.

10. Инициализация программы

Как и фильтры, раздел инициализации определяется ключевым словом `init()` без параметров, после чего в фигурных скобках следует его подпрограмма:

```
init()
{
}
```

Функция `init()` может использоваться для инициализации переменных программы. Код этой подпрограммы будет выполняться при включении устройства до выполнения любого другого кода. Отправить данные с помощью `send()` из функции `init()` невозможно, поскольку код `init()` выполняется до того, как устройство завершит процедуры «заявки на адрес» NMEA 2000. Все вызовы `send()` из функции `init()` будут игнорироваться.

11. Heartbeat

Как и фильтры или `init()`, heartbeat определяется ключевым словом `heartbeat(ms)`, за которым в фигурных скобках следует его подпрограмма. Heartbeat требует целочисленного параметра (константы), который указывает интервал повторения в миллисекундах.

```
heartbeat(1000)
{ # Этот код будет выполняться каждую секунду
}
```

Функцию `heartbeat()` можно использовать, когда код необходимо выполнять периодически. Функция `heartbeat()` впервые вызывается из пользовательской программы сразу после вызова `init()`.

12. Пользовательская функция: func() и call()

Устройство поддерживает одну пользовательскую функцию `func()`, которую можно вызвать из любой другой подпрограммы с помощью `call()`. Как и фильтры или `init()`, пользовательская функция определяется ключевым словом `func()`, за которым в фигурных скобках следует ее подпрограмма. Пример 5 показывает вызов `func()` из функции `heartbeat()` с помощью `call()`.

Пример 5.

```
# Генератор последовательностей Фибоначчи DIAGNOSTICS=45 #

запись первых 45 чисел init(){

      # задать два первых числа в последовательности Фибоначчи A=cast(0,UINT32)
      # F0
      B=cast(1,UINT32) # F1
}

# вычислить следующее число Фибоначчи и записать его в файл журнала func(){

      C = A + B                 # F(n) = F(n-1) + F(n-2)
      log(C)                    # записать F(n) в файл журнала
}

# генерировать следующее число каждую секунду, всего 45 heartbeat(1000){

      call()                    # вычислить следующее число Фибоначчи
      A = B                     # сохранить F(n-2) для следующего раунда
      B = C                     # сохранить F(n-1) для следующего раунда
}
```

Рекурсия — вызов `call()` внутри `func()` — также разрешена, но максимальная глубина рекурсии ограничена 10.

Если достигнут максимальный уровень рекурсии, устройство выдает сообщение «Достигнута максимальная глубина рекурсии» в диагностическом журнале TEXT.

13. Математические функции и константы

Константы с плавающей запятой:

```
M_PI = π = 3,14159...
π
M_PI2 = π / 2
```

Константы с плавающей запятой высокой точности:

```
M_180P = 180 / π M_2PI = 2 *
M_1800P = M_180P / 10
M_P180 = π / 180
```

Тригонометрические функции:

```
FLOAT sin(x)          FLOAT cos(x)          FLOAT tan(x)
```

Где аргумент x — угол в радианах.

Обратные тригонометрические функции:

```
FLOAT asin(y)         FLOAT acos(y)         FLOAT atan(y)
```

Возвращает угол в радианах. Примечание: при вводе значений аргументов, выходящих за пределы области определения функции, возвращается ноль. Квадратный корень:

```
FLOAT sqrt(z)
```

Примечание: при передаче отрицательного значения в `sqrt()` возвращается ноль. Арктангенс с двумя аргументами

```
FLOAT atan2(x,y)
```

Возвращает угол на евклидовой плоскости, в радианах, между положительной осью x и лучем к точке (x, y). Примечание: `atan2(0,0)` возвращает ноль.

14. Программный сброс

Вызовите функцию *reset()* для перезагрузки моста; её можно использовать для перезапуска программы и сброса настроек обоих интерфейсов CAN, например, при получении определённого кадра CAN или по таймеру.

15. Передача параметров в программу в режиме реального времени

```
UINT8 param(slot)
```

Функция `param()` может обрабатывать специальную команду «YD:PARAM», введенную в «Строку описания установки 2» (PGN 126998) моста через шлюз NMEA 2000-PC. Это может пригодиться, когда необходимо настроить некоторые параметры программы и наблюдать за реакцией в режиме реального времени, например, при настройке значений ПИД на CAN-сервоприводе.

Формат команды «YD:PARAM»:

```
YD:PARAM hexits_string
```

Где `hexits_string` — это поле данных — последовательность из не более 60 шестнадцатеричных цифр (hexits) — символов в диапазоне [0–9, A–F]. Этот механизм позволяет отправлять до 30 произвольных байтов для обработки устройством.

Если команда принята, Bridge ответит и добавит «OK» в «Строку описания установки 2». Если в данных передачи обнаружена ошибка, например недопустимый символ или общее количество шестнадцатеричных знаков превышает 60, устройство ответит «YD:PARAM ERR».

Вы можете вводить шестнадцатеричные идентификаторы, разделенные пробелами или без них, с ведущим нулем или без него, в верхнем или нижнем регистре. Устройство удалит все пробелы в ответе и преобразует шестнадцатеричные идентификаторы в верхний регистр, поэтому любая из приведенных ниже команд:

```
YD:PARAM 5 6 7 1A f
YD:PARAM 05 06 07 1a 0F YD:PARAM
0506071A0F
```

даст одинаковый ответ и одинаковое содержимое SLOT:

```
YD:PARAM 0506071A0F OK
```

Примечание: если введены все 30 байт, устройство не добавит «OK» в ответ из-за ограничения на размер строки «Installation Description String».

Функция `param()` принимает один из SLOT в качестве аргумента и возвращает количество байтов, прочитанных из полезной нагрузки команды «YD:PARAM». Целевой SLOT будет заполнен до 30 байтами данных из команды

Поле данных. Чтобы получить байты поля данных, просто вызовите функцию `load()` для этого SLOT, чтобы скопировать содержимое в рабочий буфер. SLOT может содержать до 229 байтов, но данный метод настройки позволяет задать только первые 30 байтов; все остальные байты в SLOT будут очищены.

Если команда не была задана, полезные данные команды пусты или команда неверна, SLOT не будет изменен, а `param()` вернет 0.

Пример 6.

```
# Симулятор уровня топлива в баке. Отправляет PGN 127505 «Fluid Level».
# Позволяет установить произвольный «уровень топлива» с помощью команды «YD:PARAM yy», #
где yy — желаемый уровень топлива (шестнадцатеричное значение, в %)

# подготовка PGN 127505 «Fluid Level» с заполненным на 100% баком дизельного топлива, экземпляр жидкости = 0 SLOT1 =
0111F219 FF 08 00A861FFFFFFFFFF


# Основной цикл, выполняемый при каждом сигнале пульса
(1000)

{

      # получены новые настройки уровня бака через "YD:PARAM"?
      if (param(SLOT2) == 1)             # прочитан только один байт (правильная длина полезных данных)?
      {
         load(SLOT2)                     # скопировать слот в рабочий буфер
         A = get(0, INT8) * 250 # получить новый уровень жидкости в %, преобразовать в разрешение N2K load(SLOT1)
                                         # загрузить PGN 127505 в рабочий буфер
         set(DATA+1, INT16, A) # обновить поле данных «уровень жидкости %»
         сохранить(SLOT1)                # сохранить SLOT1 с измененными данными «уровень жидкости %»
      }
      # Отправить PGN 127505 «Уровень жидкости»
      load(SLOT1)
      send(CAN1)
}
```

IX. Оптимизация и производительность

Время передачи типичного однокадрового сообщения CAN с полной длиной полезной нагрузки 8 байт составляет около 520 микросекунд. Устройство не тратит ресурсы ЦП на обработку связи на канальном уровне шины CAN, эти задачи переносятся на аппаратное обеспечение контроллера CAN.

Задержка между приемом и передачей сообщения — для сообщений, не соответствующих ни одному фильтру, — составляет не более 100 микросекунд, даже при использовании максимального количества фильтров (20).

Из общего времени в 100 микросекунд только 40 микросекунд уходит на сравнение с фильтрами (выполнение программы). Остальное время тратится на обработку прерываний контроллера CAN, помещение полученного сообщения в очередь входа программы, извлечение сообщений из очереди выхода программы и последующую передачу их контроллеру CAN для отправки.

Время выполнения (для сообщений, соответствующих фильтру) в примере 2 составляет 240 микросекунд, а в примере 9 — 515 микросекунд. Общие задержки в устройстве составляют 300 и 575 микросекунд соответственно.

Устройство имеет программную очередь на 100 входящих и 100 исходящих сообщений, при этом максимальное время нахождения сообщений в очереди составляет 50 миллисекунд. Производительность устройства достаточна для практических применений даже в сетях с высокой нагрузкой. При условии, что не допускаются грубые ошибки программирования, проблем с перегрузкой возникнуть не должно.

1. Используйте аппаратные фильтры

Один из наиболее распространённых вариантов использования моста — фильтрация данных между двумя сегментами сети. Использование аппаратных фильтров, показанное в примере 7 ниже, является чрезвычайно эффективным.

Пример 7.

```
FW_CAN1_TO_CAN2=ON
FW_CAN2_TO_CAN1=ON
CAN1_HARDWARE_FILTER_1=0x00000000, 0x00FFFFFF match(CAN2,0x1FD0A00,0x1FFFF00)
{
      # здесь выполняется необходимая обработка...
}
```

Благодаря аппаратным фильтрам системы (см. раздел VII.3) сообщения, необходимые для нормальной работы сети NMEA 2000 (подтверждение ISO, запрос ISO, заявка на адрес ISO), будут пересылаться между интерфейсами CAN1 и CAN2 без изменений.

В данной программе аппаратный фильтр для интерфейса CAN2 не задан, поэтому Device добавит фильтр по умолчанию № 1, который будет принимать все сообщения с CAN2 и передавать их фильтрам для дальнейшей обработки (см. раздел VII.3). Для интерфейса CAN1 настроен CAN1_HARDWARE_FILTER_1, чтобы блокировать все PGN на интерфейсе CAN1. В результате все сообщения, полученные по CAN1, будут отбрасываться до того, как они достигнут фильтров `match()`, что значительно экономит ресурсы ЦП.

2. Используйте match() вместо нескольких if()

Может возникнуть соблазн использовать несколько операторов `if()` для обработки разных PGN внутри одного фильтра, как в примере 8.a ниже.

Этот метод уступает использованию фильтров `match()`, поскольку фильтры `match()` используют аппаратное ускорение сравнения CAN-идентификаторов — добавление дополнительных фильтров `match()` не приводит к дополнительным накладным расходам или потере ресурсов ЦП. С другой стороны, `оператор if()` выполняется на виртуальной машине устройства, поэтому работает в несколько раз медленнее.

Пример 8.a (неправильный)

```
match(CAN1,0x00000010,0x000000FF)                    # отправлено с адреса 0x10?

{
      p = (get(0,UINT32) >> 8) & 0x1FFFF # получить PGN
      if (p == 0x1FD0A)                              # это сообщение «Фактическое давление»?
      {

        # обработать данные

      }

}
```

Пример 8.b (правильный)

```
match(CAN1,0x1FD0A10,0x1FFFFFF) # PGN «Фактическое давление» с адреса 0x10?
{
        # обработать данные
}
```

3. Используйте оптимизации вместо того, чтобы полагаться на компилятор

Текущая версия компилятора Device не поддерживает оптимизацию выражений.

Например, в следующем коде:

```
A = time()                          # получить время работы устройства в мс
B = A / ( 60 * 1000 )               # преобразование в минуты
C = A / ( 60 * 60 * 1000 ) # преобразование в часы
```

Умножение будет выполняться 3 раза, чего можно легко избежать:

```
A = time()                          # получить время работы устройства в миллисекундах
B = A / 60000                       # преобразование в минуты
C = A / 3600000                     # преобразовать в часы
```

Если в вашей программе много вычислений, попробуйте их оптимизировать.

В будущих обновлениях программного обеспечения особое внимание будет уделяться вопросам оптимизации при компиляции в байт-код.

4. По возможности избегайте сборки PGN с быстрым пакетом

В большинстве случаев данные в сообщении можно изменить без предварительной сборки. Рассмотрим, например, взлом лога расстояний (уголовно наказуемое преступление в большинстве стран) с использованием сообщения Distance Log (PGN 0x1F513).

Пример 9.

```
PGNS_TO_ASSEMBLY=0x1F513 match(CAN1,0x1F51300,0x1FFFF00)
{
      set(DATA+6, UINT32, get(DATA+6, UINT32) — 1852000) # отклонение 1000 нм
      send()
}
```

Пример 10.

```
match(CAN1, 0x1F51300, 0x1FFFF00)
{
      if (get(DATA,UINT8) & 0x1F == 1) # 2-е сообщение в последовательности?
      {
            set(DATA+1, UINT32, get(DATA+1,UINT32) - 1852000) # смещение на 1000 нм
      }
      send()
}
```

Конечный результат обеих программ будет одинаковым, но в примере 10 подпрограмма `match()` будет выполняться 3 раза для каждого PGN 0x1F513, так как этот PGN отправляется последовательно в 3-х кадрах fast-packet. Время выполнения кода во втором примере больше, так как в нём присутствуют дополнительные операции.

Однако в случае примера 9 устройство сначала примет все три сообщения CAN и только после этого передаст сформированное сообщение NMEA в программу. Время между получением первого и третьего сообщений составит более 1000 микросекунд — именно столько времени требуется для передачи двух сообщений по сети CAN. При вызове функции `send()` сообщение NMEA 2000 будет разделено на три CAN-сообщения и помещено в очередь отправки. Третье сообщение будет отправлено не раньше, чем через 1000 микросекунд.

Таким образом, в примере 10 все сообщения будут отправляться примерно на 1000 микросекунд раньше, чем в примере 9. На практике разница в 1 миллисекунду может показаться незначительной, но чем длиннее сообщение NMEA 2000, тем больше будет задержка. Кроме того, при высокой нагрузке на любую из сетей CAN программа, представленная в примере 10, имеет дополнительное преимущество.

X. Функции отладки и ведения журнала

Режим отладки не является стандартным режимом работы устройства. Запись данных отладки может привести к дополнительной задержке сообщений и потере некоторых сообщений шины CAN.

Устройство поддерживает запись как необработанных данных шины CAN, так и отладочной информации о выполнении программы.

Чтобы начать запись, добавьте следующую настройку в файл YDNB.CFG:

```
DIAGNOSTICS=x
```

Где x — продолжительность записи журнала в секундах.

Вставьте карту в устройство, убедитесь, что файл YDNB.CFG был принят (устройство ответит тремя короткими миганиями ЗЕЛЕНОГО светодиода) и что запись началась (3-секундное мигание ЗЕЛЕНОГО светодиода). Не извлекайте карту во время записи, так как это повредит файловую систему карты. По истечении заданного времени запись журнала прекратится, и устройство выдаст 3-секундную мигающую красную индикацию, указывая, что все операции записи на карту завершены и карту можно безопасно извлечь.

Формат файла журнала можно выбрать, установив:

`LOG_FORMAT=TEXT` или `LOG_FORMAT=BINARY`

Значение по умолчанию — TEXT. Если настройка LOG_FORMAT опущена или установлена в TEXT, устройство запишет файл YDNBLOG.TXT. Если файл YDNBLOG.TXT уже существует на карте, он будет перезаписан.

Файл TEXT будет записывать все CAN-фреймы, для которых какой-либо фильтр `match()` дает совпадение, а также все CAN-фреймы, отправленные с помощью `send()`. Кроме того, во время выполнения программы можно записывать значения и типы переменных или выражений с помощью функции `log()`:

`log`(выражение)

Пример содержимого файла YDNBLOG.TXT:

```
00:30.310 RX CAN1 FILTER 01, 0x09FD0205 1A1A018544FAFFFF
00:30.310 LG CAN1 FILTER 01, INT32 19400
00:30.310 TX CAN2 FILTER 01, 0x09FD0205 1A1A018544FAFFF
00:30.319 !! Тайм-аут выполнения FUNC (3000 мс)
```

Записи «RX» и «TX» обозначают принятые и отправленные CAN-фреймы, за которыми следует указание интерфейса, номера фильтра, по которому произошло совпадение, а также идентификатора CAN-фрейма и его полезной нагрузки. Запись «LG» создается функцией `log()`. Запись «!!» создается при возникновении ошибок выполнения.

В ы можете записывать каждое сообщение, полученное устройством, в текстовый журнал, используя следующий универсальный фильтр:

```
match(ANY,0,0)
{
      send()
}
```

Не забудьте удалить или закомментировать все вызовы функции `log()` по окончании сеанса отладки. Даже если параметр DIAGNOSTICS не включен, функция `log()` всё равно будет вызываться, что приведёт к ненужной трате ресурсов ЦП.

Если LOG_FORMAT установлен в BINARY, вместо этого будут записываться файлы .CAN. Эти файлы будут содержать только данные шины CAN. Все CAN-фреймы, полученные на интерфейсах CAN1 и CAN2, будут записываться, независимо от соответствия фильтрам. В отличие от текстового журнала, вы не можете использовать функцию `log()` для сохранения значения в файл журнала BINARY.

Журналы BINARY можно открывать, просматривать, конвертировать или экспортировать в другие форматы с помощью нашей бесплатной программы CAN Log Viewer, которая работает в системах Microsoft Windows, Linux и macOS. Вы можете скачать её с нашего веб-сайта. Формат файлов .CAN является открытым и подробно описан в документации к CAN Log Viewer.

Программный код компилируется в байт-код, а затем выполняется в «виртуальной машине» устройства. Обычно вам не нужно беспокоиться о преобразовании программы в «ассемблерный код», но если вы хотите узнать, что происходит «под капотом», просто добавьте настройку:

```
DECOMPILER=1
```

в свою программу, загрузите её на устройство, и вы получите файл YDNBSAVE.CFG, в котором каждая строка вашего кода сопровождается дизассемблированным байт-кодом.

XI. Обновления прошивки

Чтобы обновить прошивку устройства, скачайте архив .ZIP с нашего сайта и извлеките файл BUPDATE.BIN, возьмите карту microSD, отформатированную в файловую систему FAT или FAT32, скопируйте файл BUPDATE.BIN в корневую папку карты, отключите NMEA 2000, вставьте карту в устройство, включите NMEA 2000.

В течение 5–15 секунд после включения светодиод 5 раз мигнет зеленым светом. Это означает, что обновление прошивки успешно завершено.

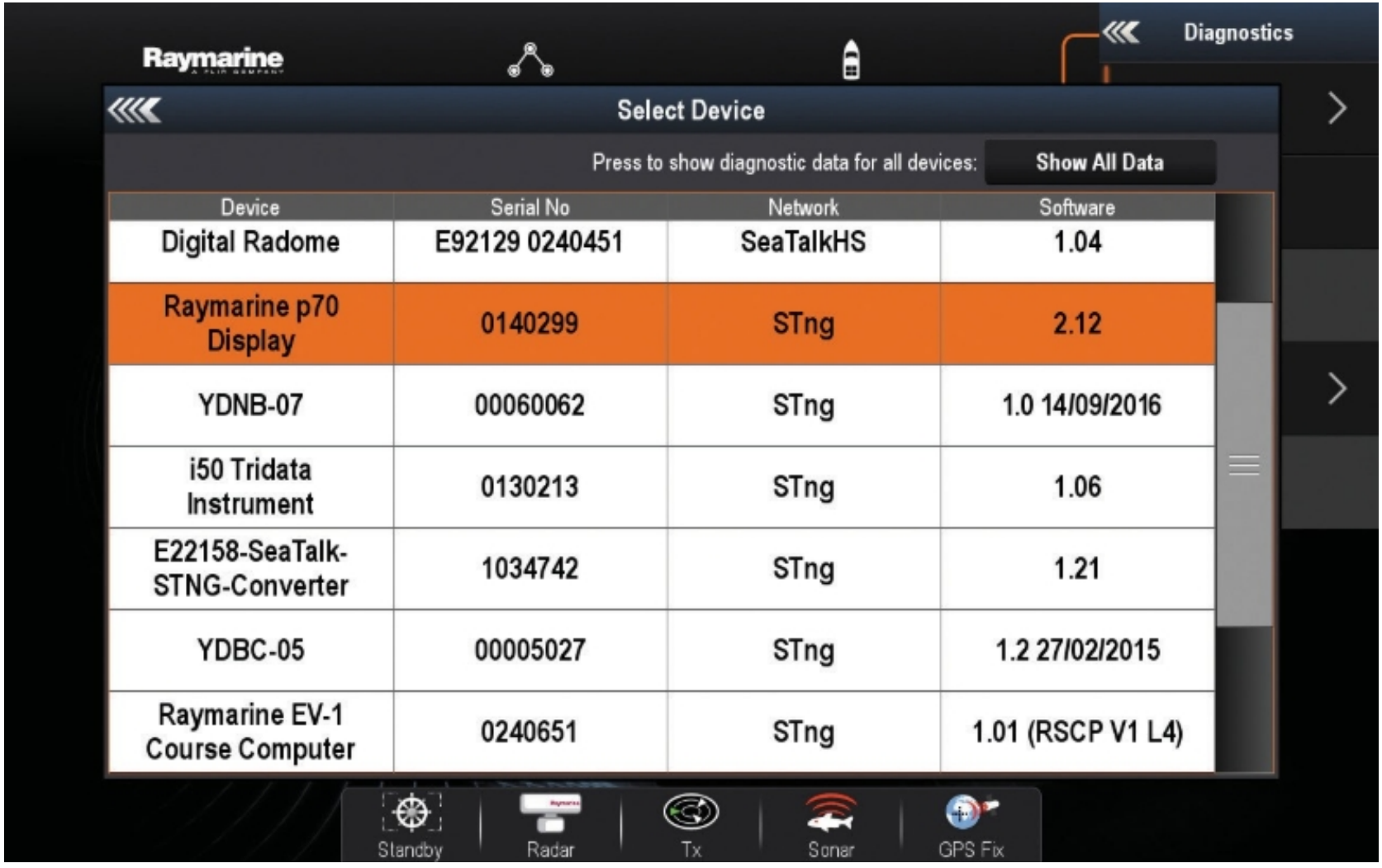

Рисунок 4. Список устройств MFD Raymarine c125 с Bridge (YDNB-07)

Если устройство уже использует данную версию прошивки, или если устройство не может открыть файл, либо файл поврежден, загрузчик немедленно передает управление основной программе. Это происходит без визуальных сигналов.

Информация об устройстве, включая версию прошивки, отображается в списке устройств NMEA 2000 (SeaTalk NG, SimNet, Furuno CAN) или в общем списке внешних устройств на картплоттере (см. третью строку на рисунке 4). Обычно доступ к этому списку осуществляется через меню «Диагностика», «Внешние интерфейсы» или «Внешние устройства» картплоттера

XII. Защита и шифрование программ

Если вы разрабатываете программы для интеграции стороннего оборудования, такое оборудование может иметь проприетарные протоколы, и на вас могут быть наложены обязательства по неразглашению, не позволяющие раскрывать программный код конечным пользователям.

Также возможно поставлять устройство с программой, которую конечный пользователь не может изменять, а может лишь обновлять с помощью зашифрованного файла, или даже полностью заблокировать устройство и исключить любую возможность чтения, удаления или обновления программы конечным пользователем без ввода мастер-пароля, установленного поставщиком решения.

Поэтому мы реализовали несколько механизмов защиты программ. Ознакомьтесь с приложением

YDNB-07-AN-001, доступным на нашем веб-сайте по адресу: http://www.yachtd.com/downloads/

Приложение A. Устранение неисправностей

Ситуация	Возможная причина и устранение
На устройстве не горит светодиод	**1. Интерфейс CAN1 не получает питание. Процессор устройства питается от интерфейса CAN1.** Убедитесь, что шины источника питания подключены к интерфейсу CAN1. **2. На шине отсутствует питание.** Проверьте, подается ли питание на шину (сеть NMEA 2000 требует отдельного подключения к источнику питания и не может питаться от плоттера или другого устройства, подключенного к сети). **3. Ослабленное соединение в цепи питания.** Обработайте разъем устройства спреем для очистки электрических контактов. Подключите устройство к другому разъему.
Один из двух мигающих светодиодов всегда мигает красным	**1. Отсутствует питание на шине.** Проверьте, подается ли питание на шину (сеть NMEA 2000 требует отдельного подключения к источнику питания и не может питаться от плоттера или другого устройства, подключенного к сети). **2. Ослабленное соединение в цепи передачи данных.** Обработайте разъем устройства спреем для очистки электрических контактов. Подключите устройство к другому разъему. **3. Проблема с шиной CAN или сетью NMEA 2000.** Убедитесь, что в обеих сетях установлены оба терминатора (см. раздел II). Подключите другое устройство к выбранному разъему и проверьте его работоспособность. **4. Большинство сообщений отбрасывается аппаратными фильтрами.** Выключите и снова включите любое устройство в сети (см. III.1).
Загруженная программа работает некорректно.	**Отладьте программу.** См. раздел X.

Ситуация	**Возможная причина и устранение**
Устройство не отображается в списке устройств NMEA 2000 на дисплейном устройстве NMEA 2000 (картплоттере/MFD), хотя соответствующий светодиод состояния интерфейса CAN горит ЗЕЛЕНЫМ	**В сети NMEA 2000 возникли проблемы.** Сегмент сети не подключен к плоттеру, в сети отсутствуют терминаторы или сеть слишком длинная. Подключите другое устройство к выбранному разъему и убедитесь, что оно отображается в списке устройств на плоттере.

Приложение B. Список сообщений NMEA 2000 устройства

Сообщения PGN из приведенной ниже таблицы принимаются и отправляются Устройством независимо от каких-либо настроек, аппаратных фильтров (см. раздел VII.3) или программы. Все полученные сообщения, включая адресованные Устройству, передаются в программу и могут быть ею перенаправлены на другой интерфейс CAN. Устройство обрабатывает адресованные ему сообщения после их обработки программой (но независимо от результатов работы программы). Ответы Устройства на такие сообщения не передаются в программу, а отправляются напрямую на оба интерфейса CAN. См. также раздел VII.4.

Таблица 1. Поддерживаемые сообщения NMEA 2000

PGN	Передача	Прием	Описание
59392	Да	Да	Подтверждение ISO
59904	—	Да	Запрос ISO
60928	Да	Да	Заявка на адрес ISO
126464	Да	—	Список групп PGN
126996	Да	—	Информация о продукте

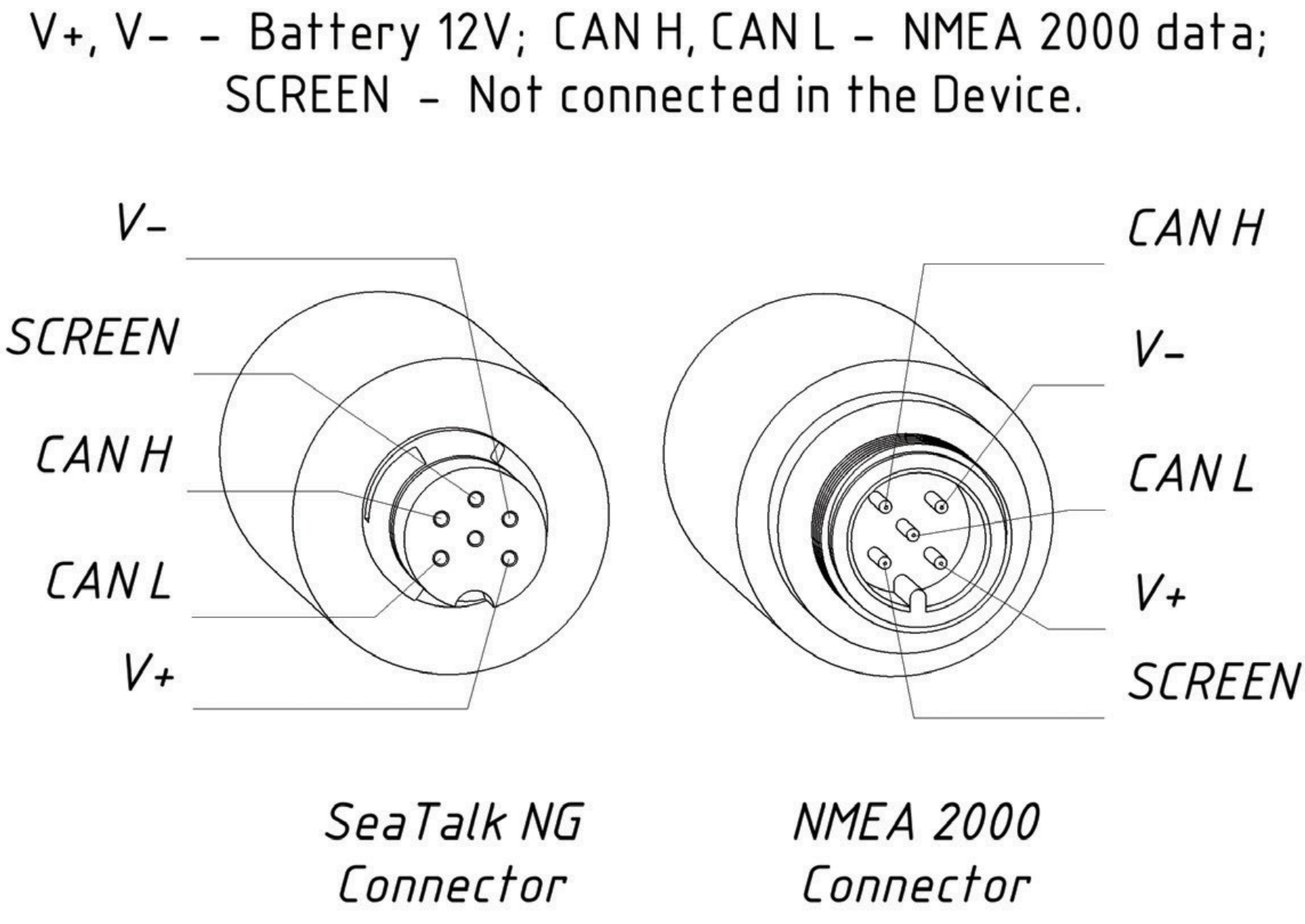

Рисунок 1. Разъемы NMEA 2000 моделей YDNB-07R (слева) и YDNB-07N (справа)