Меркурий-PLC-II

Техническое описание

Содержание

Данный документ представляет собой техническое описание сети передачи информации по распределительной сети переменного тока 220/380В 50Гц, далее по тексту − PLC-сеть второго поколения, или просто PLC-сеть.

1 Введение

PLC-сеть является средством связи с устройствами, подключенными к распределительной сети переменного тока. Её основными особенностями являются: высокая надёжность связи и простота эксплуатации.

PLC-сеть имеет централизованную организацию ('дерево'). Центром сети является концентратор (M), он же − узел доступа к сети. Обмен данными с любым из подчинённых узлов (S) возможен только через концентратор, см. рисунок 1.

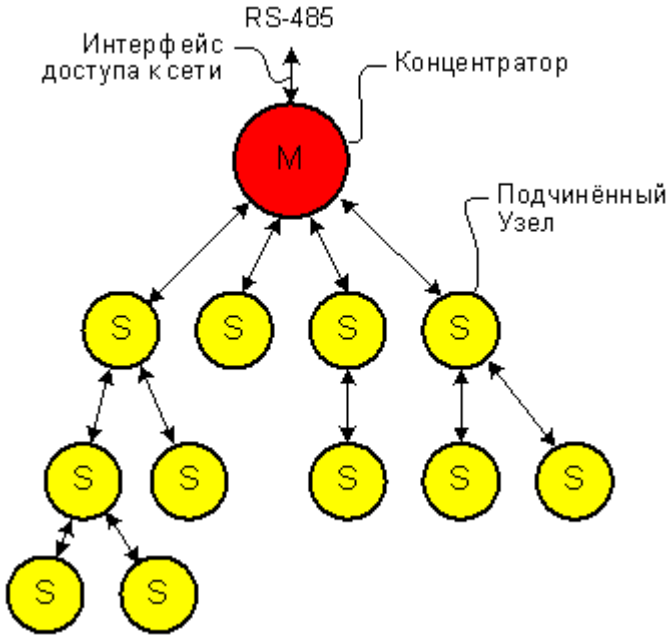


Рисунок 1 Схема PLC сети

Каждый узел сети имеет индивидуальный 32-х битный адрес, − таким образом, полная адресная ёмкость PLC-сети составляет 4 294 967 296 уникальных узлов.

Подчинённый узел сети, как правило, состоит из PLC-модема и "интеллектуального" хост-устройства.

Сегменты адресного пространства сети

Диапазон адресов	Размер сегмента	НАЗНАЧЕНИЕ
DDDDDDDD	100 000 000	Сегмент резервируется за изделиями ООО "Инкотекс" по принципу: одно изделие − один адрес подчинённого узла
ZZZ.XXXXX	216 x 1M	Сегменты (за исключением сегмента FFF.XXXXX), предназначенные для лицензирования третьим фирмам
FFF.XXXXX	1M	'Верхний' сегмент

Распределение адресов в 'верхнем' сегменте:

Диапазон адресов	Размер сегмента	НАЗНАЧЕНИЕ
FFF.F.3XXX	4096	Концентраторы PLC-II
FFF.F.4XXX	4096	Прототипы подчинённых узлов PLC-II
FFF.F.5XXX	4096	Полуконцентраторы PLC-II
FFF.F.6XXX	4096	PLC-шлюзы
FFF.F.7XXX	4096	RF-шлюзы
FFF.F.8XXX	4096	USB-RF конвертеры

Диапазон адресов	Размер сегмента	НАЗНАЧЕНИЕ
FFF.A.XXXX	65536	Аварийные устройства (модемы PLC-II, которым не удалось прочитать серийный номер своего хост-устройства и т.д.)

ПРИМЕЧАНИЯ:

D − любой десятичный символ в интервале 0..9.

X − любой шестнадцатеричный символ в интервале 0..F.

Z − любой шестнадцатеричный символ в интервале A..F.

В отличие от подчинённых узлов, уникальная адресация концентраторов не является обязательной, т.к., по определению, на одном PLC-II объекте (подстанция, дом, предприятие) может быть только один концентратор, и, соответственно, конфликты адресации здесь исключены. Младшая половина PLC-адреса концентратора используется в качестве его UART-идентификатора при работе с ним по его локальному интерфейсу RS-485 (интерфейс доступа к сети, см. рисунок 1). Присвоение двум концентраторам разных адресов может потребоваться лишь в случае, когда оба они окажутся подключены к одному и тому же сегменту линии RS-485.

PLC-сеть использует для передачи данных процедуру 'запрос-ответ', объединяющую в себе свойства как канальной, так и пакетной коммутации. В тех случаях, когда узел-приёмник находится вне поля зрения главного узла сети, используется техника передачи данных через ряд промежуточных узлов.

2 Определения

Цикл	Интервал между последовательными пересечениями нулевого уровня напряжением распределительной сети 220/380В 50Гц.
Псевдоцикл	Интервал между двумя последовательными моментами начала приёма/передачи узлами сети.
Микропакет	Пакет канального уровня (см. Процедуры канального уровня). Размер пакета сокращён до минимума в целях увеличения помехоустойчивости по отношению к импульсным помехам.
Уровень видимости (уровень ретрансляции)	Условное название совокупности узлов, связь с которыми осуществляется через заданное число промежутков ретрансляции. Узлы, с которыми концентратор связывается непосредственно, находятся на первом уровне видимости.
Регистрация	Создание почтового ящика для вновь обнаруженного либо ранее не зарегистрированного подчинённого узла.

3 Сокращения и аббревиатуры

PLC	Power Line Communications − проводная технология, использующая для передачи данных распределительные сети переменного тока 220/380В 50Гц.
M или MASTER	Концентратор PLC-II
S или SLAVE	Подчинённый узел PLC-II
MBOX	"Почтовый ящик" подчинённого узла сети PLC-II
MOD	Модем PLC-II подчинённого узла
UART-ID	Уникальный 2-х байтовый идентификатор концентратора, который используется для адресации к нему при обмене по последовательному интерфейсу.
PLC-ID	Уникальный 4-х байтовый PLC-идентификатор сетевого узла.
W-24	Стандартный 3-х байтовый эхо-микропакет = 4C71CDh
M-32	4-х байтовый M-код с хорошими автокорреляционными свойствами = 26C5610Fh
CRC320	Контрольный код CRC32, вычисленный с начальным значением CRC32_INIT = 0x00000000
CRC24	Контрольный код CRC24

4 Физический уровень

Данная глава описывает физический уровень протокольного стека PLC-сети. Физический уровень отвечает за следующие задачи:

– синхронизация приёмника и передатчика;
– переход узла из режима приёма в режим передачи и обратно;
– приём данных на нескольких символьных скоростях;
– передача данных на заданной символьной скорости.

4.1 Синхронизация

Синхронизация приёмника и передатчика осуществляется по моментам перехода нуля сетевым напряжением. Передатчик передаёт сигнал в виде широкополосных посылок (импульсов) длительностью 3.2 мсек. Передача импульсов начинается через 8.4 мсек после того момента, когда абсолютное значение напряжения в точке подключения узла достигнет нулевого значения. При частоте сети 50 Гц центр импульса совпадает с "нулевым моментом", см. рисунок 2.

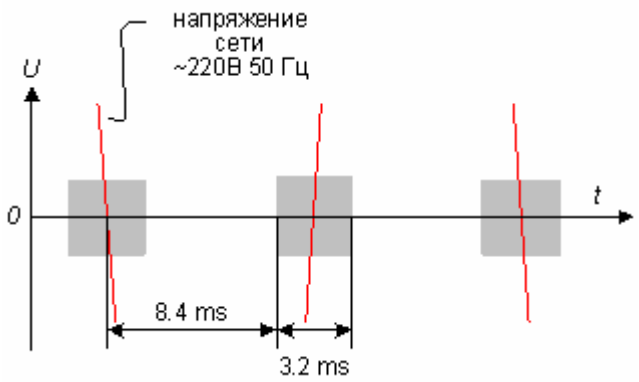


Рисунок 2 Привязка импульсов передатчика к полупериодам сетевого напряжения.

Здесь и далее интервал между двумя точками пересечения нуля сетевым напряжением называется 'цикл'. Интервал между двумя моментами начала передачи называется 'псевдоцикл'. Длительности цикла и псевдоцикла совпадают. Начало псевдоцикла запаздывает относительно начала цикла на 8.4 мс

5 Канальный уровень

Данная глава описывает канальный уровень протокольного стека PLC-системы. Канальный уровень отвечает за следующие задачи:

– реализация надёжной связи между двумя узлами в пределах 'прямой видимости';
– управление скоростью передачи;
– извещение контроллера сети о неудачной попытке связи между двумя узлами.

Под 'прямой видимостью' понимается такое состояние линии передачи информации, когда два узла сети могут устойчиво обмениваются пакетами непосредственно друг с другом.

5.1 Концепция запрос-ответ

В любой сети, узлы которой используют для передачи информации единый ресурс, возможны коллизии − конфликты единовременного доступа к среде передачи двух и более узлов. В целях предотвращения коллизий в описываемой PLC-сети используется процедура обмена информацией в режиме запрос-ответ.

Концепция запрос-ответ реализована как на канальном, так и на транспортном уровнях. Порядок формирования и передачи транспортных пакетов рассмотрен в разделе Сетевой уровень.

В каждый момент времени допускается передача только одним узлом. Узел, инициировавший передачу, выступает в качестве источника информации и в дальнейшем будет называться 'узел-передатчик'. Узел, которому адресованы пакеты узла-передатчика, далее будет называться 'узел-приёмник'.

5.2 Процедуры канального уровня

Канальный уровень PLC-сети поддерживает две базовые процедуры:

– сканирование узлов;
– передача пакетов транспортного уровня.

Процедура сканирования имеет целью обнаружение наличия и определения значений адресов PLC-узлов, находящихся в пределах прямой видимости. При проведении сканирования сканирующий узел передаёт микропакет-запрос специального формата − микропакет сканирования uP-scan. Микропакет сканирования содержит информацию о диапазоне адресов, в пределах которого все узлы должны немедленно откликнуться эхо-микропакетом стандартного вида W-24.

Приём эхо-микропакета позволяет узлу-передатчику сделать вывод о наличии хотя бы одного узла в заданном диапазоне адресов в пределах прямой видимости, что является поводом для дальнейшего уточнения адресного положения обнаруженных узлов методом дихотомии (см. раздел Поиск SLAVE-узлов). В случае непоступления ответного эхо-микропакета, считается, что в заданном диапазоне адресов нет узлов, с которыми можно установить надёжную связь, либо узлов с такими адресами не существует.

Другой процедурой канального уровня является собственно передача транспортных пакетов. Она реализуется в два этапа:

– открытие канала
– передача транспортного пакета

Открытие канала производится стартовым микропакетом-запросом uP-start, содержащим адрес узла-приёмника. В ответ на стартовый микропакет узел-приёмник должен немедленно передать 3-х байтовый стандартный эхо-микропакет W-24. В целях увеличения надёжности доставки данных, процедура открытия канала при необходимости использует перебор имеющихся в наличии битовых потоков, при этом передающий узел пытается передать стартовый микропакет до тех пор, пока не будет получено эхо, либо не будет принято решение об отсутствии связи с узлом-приёмником.

Немедленно после приёма эхо-микропакета передающий узел приступает к последовательной передаче байтов транспортного пакета, защищенных кодом Хемминга [12,8] (см. Приложение В. Код Хемминга [12,8]: кодирование и декодирование). С целью контроля целостности транспортный пакет сопровождается контрольным кодом CRC24, вычисленным по всем байтам транспортного пакета. В случае успешного приёма транспортного пакета, узел-приёмник немедленно передаёт стандартный эхо-микропакет W-24, дополнительно см. Структура транспортного пакета.

5.3 Форматы микропакетов

Для реализации процедур канального уровня, описанных в предыдущем разделе, используются следующие типы микропакетов:

1. Микропакеты-запросы сканирования uP-scan
2. Стартовые микропакеты-запросы uP-start
3. Стандартные эхо-микропакеты W-24

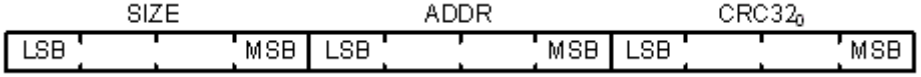
Микропакеты любого типа передаются начиная с младшего байта младшим битом вперёд, аналогичным образом передаются также и отдельные поля, входящие в состав микропакета (если таковые имеются).

Все микропакеты-запросы содержат информационную часть длиной 4 или 8 байт и заканчиваются 4-х байтовым контрольным кодом CRC320, вычисляемым от информационной части.

Узел-приёмник подтверждает получение микропакета-запроса излучением эхо-микропакета W-24. Эхо передаётся в том же битовом потоке, что и запрос, начиная с первого же свободного псевдоцикла после микропакета-запроса.

5.3.1 Микропакеты сканирования uP-scan

Микропакет сканирования типа uP-scan имеет длину 12 байт и его информационная часть содержит два 4-х байтовых поля: 'SIZE' и 'ADDR'.



Поля 'SIZE' и 'ADDR' определяют область сетевых адресов, которая подвергается сканированию. В область сканирования входят все адреса с [ADDR] по [ADDR+SIZE] включительно.

Любой узел, адрес которого попадает в указанный диапазон, должен немедленно (начиная со следующего же псевдоцикла) вернуть эхо-микропакет типа W-24.

5.3.2 Стартовые микропакеты uP-start

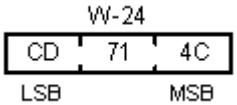
Передача транспортных пакетов между узлами PLC-сети другому носит характер сеанса и начинается с передачи стартового микропакета uP-start длиной 12 байт.

Информационная часть микропакета uP-start содержит единственное 4-х байтовое поле 'ADDR' − адрес PLC-узла-приёмника, с которым узел-передатчик пытается установить соединение. Контрольный код CRC320 рассчитывается от поля 'ADDR'.

Каждый PLC-узел ведёт непрерывный приём в каждом из поддерживаемых битовых потоков. Узел, обнаруживший микропакет uP-start со своим адресом, немедленно отправляет в ответ стандартный эхо-микропакет W-24, подтверждая тем самым свою готовность к приёму данных в этом битовом потоке.

5.3.3 Эхо-микропакеты W-24

Эхо-микропакет W-24 состоит из 3-х фиксированных байт.



6 Транспортный уровень

Данная глава описывает транспортный уровень протокольного стека PLC-системы, который отвечает за следующие задачи:

– передача транспортных пакетов между узлами PLC-системы;

– определение невозможности установления соединения на канальном уровне.

Транспортные пакеты могут осуществлять транспортировку запросов/ответов произвольного формата, либо выступать в качестве самостоятельной информационной единицы.

6.1 Структура транспортного пакета

Структура транспортного пакета изображена на рисунке 3.

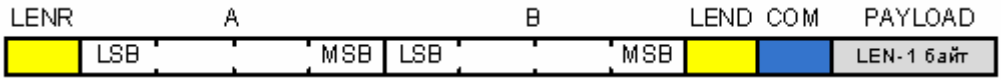


Рисунок 3 Структура транспортного пакета в случае прямой видимости 'A' <-> 'B'.

Максимальный размер транспортного пакета равен 256 байтам.

Первый байт пакета 'LENR' равен длине маршрутного поля пакета в байтах .

Далее следует само маршрутное поле, описывающее маршрут доступа к данному узлу, которое состоит из последовательности 4-х байтовых адресов, содержащей адреса: начального, нескольких промежуточных (при использовании ретрансляции) и оконечного узлов, через которые данному пакету предстоит пройти (на рис.4.1.1 маршрут следования пакета состоит из двух элементов: A и B).

За маршрутным полем следует байт 'LEND' который равен длине следующей за ним области данных.

Область данных, в свою очередь, состоит из двух полей: 'COM' (1 байт) и 'PAYLOAD' (остальное). Содержимое байта 'COM' определяет назначение поля 'PAYLOAD', − это может быть либо команда для PLC-модема оконечного узла, либо команда для хост-устройства оконечного узла, либо их ответы концентратору.

Направление следования данного транспортного пакета определяется старшим битом байта COM. Если этот бит равен 0, то пакет представляет собой запрос к подчинённому узлу, 1 − ответ от подчинённого узла. Оконечный узел обрабатывает предназначенный ему транспортный пакет только в том случае, если старший бит байта COM равен 0, в противном случае пакет игнорируется.

При проведении транзакции по PLC-интерфейсу целостность транспортного пакета подтверждается контрольным кодом CRC24, который передаётся сразу же вслед за последним битом поля PAYLOAD, а правильность приёма транспортного пакета подтверждается принимающим узлом с помощью стандартного эхо-микропакета, см. рисунок 4.

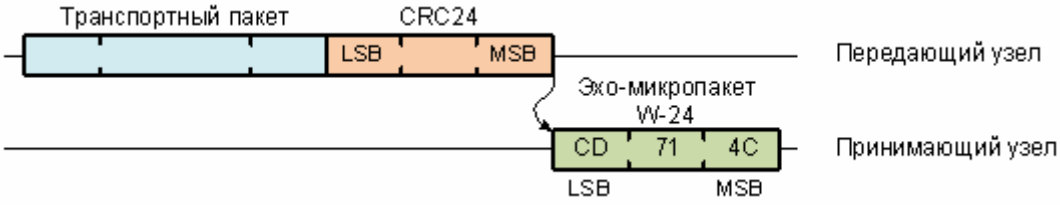


Рисунок 4 Передача транспортного пакета

6.2 Типы транспортных пакетов

Тип пакета (поле COM)	Мнемоническое обозначение и функциональное назначение транспортного пакета
00h	COMM_REMOTE_SCAN

Тип пакета (поле COM)	Мнемоническое обозначение и функциональное назначение транспортного пакета
	Запрос PLC-модему: "Произвести удалённое сканирование".
01h	COMM_GET_CONFIG
	Запрос PLC-модему: "Чтение конфигурации модема SLAVE-узла".
02h	COMM_SET_TIME
	Запрос PLC-модему: "Установка времени в хост-устройстве".
03h	COMM_CLOSE_PLC
	Запрос PLC-модему: "Закрыть PLC-канал".
04h	COMM_OPEN_PLC
	Запрос PLC-модему: "Открыть PLC-канал".
05h	COMM_GET_ACCESS_CNT
	Запрос PLC-модему: "Чтение счетчиков доступа".
06h	COMM_REMOTE_DETECT
	Запрос PLC-модему: "Произвести удалённое обнаружение".
06h...3Fh	Не используется.
40h	COMM_HOST_PAYLOAD
	Запрос хост-устройству: "Передача произвольных данных".
41h...7Fh	Не используется.
80h...86h	Ответы PLC-модемов на запросы типов 00h...06h.
86h...BFh	Не используется.
C0h	Ответ хост-устройства на запрос типа 40h.
C1h...FDh	Не используется.
FDh	RELAY_CONNECT_FAILURE
	Ответ PLC-модема: "Неудачная попытка установить соединение с узлом" (отправляется узлом, которому не удалось установить соединение со следующим узлом в цепочке).
FEh	FORMAT_FAILURE
	Ответ PLC-модема: "Формат транспортного пакета не соответствует спецификации".
FFh	RELAY_TRANSPORT_FAILURE
	Ответ PLC-модема: "Неудачная попытка передать транспортный пакет" (отправляется узлом, которому не удалось передать транспортный пакет по маршруту).

6.2.1 00h – Произвести удалённое сканирование

Полезная нагрузка пакета "произвести удалённое сканирование" содержит базовый адрес и размер области, в которой необходимо произвести сканирование, см. Микропакеты сканирования uP-scan. Подчинённый узел, получивший пакет этого типа, производит излучение микропакета uP-scan соответствующего содержания и обнаружение ответного эхо-микропакета. Результат обнаружения возвращается как 0 или 1 в следующем байте после поля 'COM'.

Пример:

LENR	MASTER				SLAVE				LEND	COM	SIZE				OFFSET			
08	45	30	FF	FF	83	39	29	31	09	00	D2	44	00	00	00	09	3D	71

Запрос:

4. Адрес концентратора: MASTER = FFFF3045h.
5. Адрес подчинённого узла: SLAVE = 31293983h.
6. Ретрансляция: нет.
7. База сканирования: 713D0900h.
8. Размер области сканирования: 44D2h.

LENR	SLAVE				MASTER				LEND	COM	RES
08	83	39	29	31	45	30	FF	FF	02	80	XX

Ответ:

ПРИМЕЧАНИЯ:

Поле RES может принимать следующее множество значений:

– 00 − ответный эхо-микропакет принят
– 01 − ответный эхо-микропакет не принят
– FE − формат запроса не соответствует спецификации
– FF − данный тип запроса не поддерживается данной версией PLC-модема

6.2.2 01h − Чтение конфигурации модема SLAVE-узла

Пример:

LENR	MASTER				SLAVE				LEND	COM
08	45	30	FF	FF	83	39	29	31	01	01

Запрос:

1. Адрес концентратора: MASTER = FFFF3045h.
2. Адрес подчинённого узла: SLAVE = 31293983h.

LENR	SLAVE				MASTER				LEND	COM	RES	VER	HP	BCNF
08	83	39	29	31	45	30	FF	FF	05	81	XX	23	02	01

Ответ:

VER = 2.3, номер прошивки PLC-модема подчинённого узла

HP = 02, условный номер протокола хост-устройства подчинённого узла, в данном случае (hpM230) − счётчик серии "Меркурий-230", см. документ "Меркурий-PLC-II.Интерфейс модема"

BCNF = 01, байт битовых флажков конфигурации модема:

– BCNF.0 − 'PlcProtected', PLC-канал модема "закрыт", см. 03h − Закрыть PLC-канал, 04h − Открыть PLC-канал
– BCNF.1 − 'TimerSet', секундомер модема установлен
– BCNF.2...BCNF.7 − не используются

ПРИМЕЧАНИЯ:

Поле RES может принимать следующее множество значений:

– 00 − запрос выполнен успешно
– FE − формат запроса не соответствует спецификации
– FF − данный тип запроса не поддерживается данной версией PLC-модема

6.2.3 02h − Установка часов/календаря хост-устройства

Получив эту команду, PLC-модем конвертирует принятое время в секундах в командный пакет установки времени в соответствии с известным ему протоколом данного хост-устройства.

Пример:

LENR	MASTER				SLAVE				LEND	COM	SECONDS			
08	45	30	FF	FF	83	39	29	31	05	02	12	34	56	00

Запрос:

1. Адрес концентратора: MASTER = FFFF3045h.

2. Адрес подчинённого узла: SLAVE = 31293983h.

3. SECONDS = 00563412h − количество секунд, прошедших с 00:00, 01 января 2000г. до настоящего времени (в шестнадцатеричном формате).

LENR	SLAVE				MASTER				LEND	COM	RES
08	83	39	29	31	45	30	FF	FF	02	82	XX

Ответ:

ПРИМЕЧАНИЯ:

Поле RES может принимать следующее множество значений:

– 00 − встроенные часы хост-устройства установлены
– 01 − встроенные часы хост-устройства установить не удалось
– 02 − хост-устройство отсутствует, внутренний секундомер модема установлен успешно
– FE − формат запроса не соответствует спецификации
– FF − данный тип запроса не поддерживается данной версией PLC-модема

6.2.4 03h − Закрыть PLC-канал

По команде "Закрыть PLC-канал" подчинённый узел сохраняет в своей энергонезависимой памяти адрес концентратора, от которого он эту команду получил, и переходит в т.н. "залоченное" состояние. В том состоянии модем не отвечает на микропакеты сканирования и пропускает через себя только те транспортные пакеты, которые исходят от или направлены к данному концентратору.

Пример:

LENR	MASTER				SLAVE				LEND	COM
08	45	30	FF	FF	83	39	29	31	01	03

Запрос:

1. Адрес концентратора: MASTER = FFFF3045h.
2. Адрес подчинённого узла: SLAVE = 31293983h.

LENR	SLAVE				MASTER				LEND	COM	RES
08	83	39	29	31	45	30	FF	FF	02	83	XX

Ответ:

ПРИМЕЧАНИЯ:

Поле 'RES' может принимать следующее множество значений:

– 00 − закрытие PLC-канала прошло успешно
– 01 − закрыть PLC-канал модему не удалось
– FE − формат запроса не соответствует спецификации
– FF − данный тип запроса не поддерживается данной версией PLC-модема.

6.2.5 04h − Открыть PLC-канал

Команда "Открыть PLC-канал" является дополнительной по отношению к команде "Закрыть PLC-канал".

Пример:

LENR	MASTER				SLAVE				LEND	COM
08	45	30	FF	FF	83	39	29	31	01	04

Запрос:

1. Адрес концентратора: MASTER = FFFF3045h.
2. Адрес подчинённого узла: SLAVE = 31293983h.

LENR	SLAVE				MASTER				LEND	COM	RES
08	83	39	29	31	45	30	FF	FF	02	84	XX

Ответ:

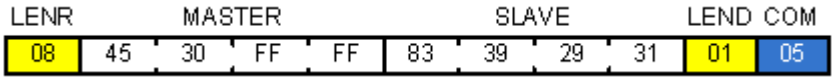
ПРИМЕЧАНИЯ:

Поле RES может принимать следующее множество значений:

– 00 − открытие PLC-канала прошло успешно
– 01 − открыть PLC-канал модему не удалось
– FE − формат запроса не соответствует спецификации
– FF − данный тип запроса не поддерживается данной версией PLC-модема.

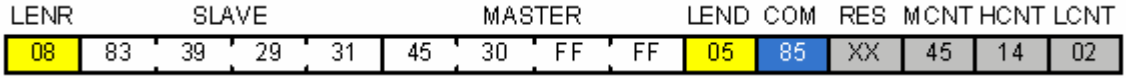
6.2.6 05h − Чтение счётчиков доступа к SLAVE-узлу

Пример:



Запрос:

1. Адрес концентратора: MASTER = FFFF3045h.
2. Адрес подчинённого узла: SLAVE = 31293983h.



Ответ:

– MCNT = 45h, счётчик обращений к модему
– HCNT = 14h, счётчик обращений к хост-устройству
– LCNT = 02h, счётчик команд на закрытие/открытие PLC-канала, см. 03h − Закрыть PLC-канал, 04h − Открыть PLC-канал.

ПРИМЕЧАНИЯ:

MCNT − счётчик принятых модемом транспортных пакетов, в пункте назначения которых указан адрес данного модема.

Все вышеописанные счётчики доступа хранятся в оперативной памяти модема и обнуляются при его перезагрузке.

Поле RES может принимать следующее множество значений:

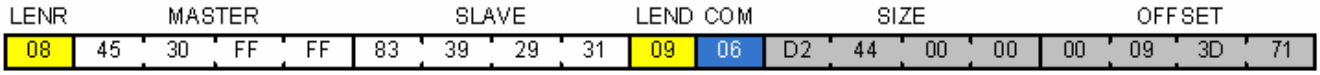
– 00 − запрос выполнен успешно
– 01 − запрос не выполнен
– FE − формат запроса не соответствует спецификации
– FF − данный тип запроса не поддерживается данной версией PLC-модема.

6.2.7 06h − Произвести удалённое обнаружение

Полезная нагрузка пакета "произвести удалённое обнаружение" содержит базовый адрес и размер области, в которой следует произвести обнаружение, см. Микропакеты сканирования uP-scan. Подчинённый узел, получивший пакет данного типа, производит попытку обнаружения другого узла, адрес которого принадлежит указанной области.

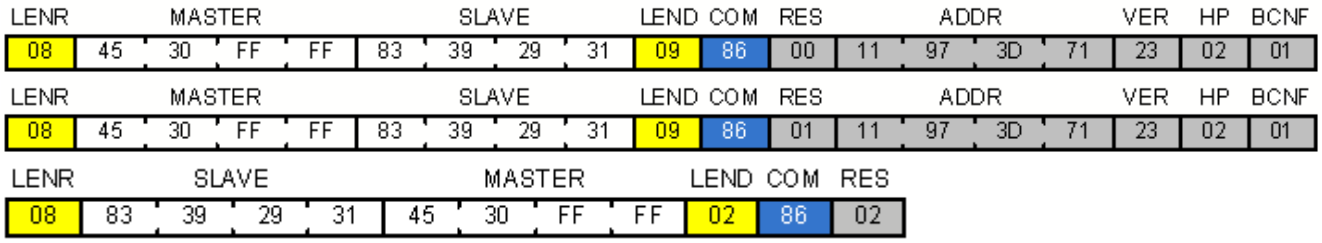
Результат обнаружения возвращается в поле 'RES'.

Пример:



Запрос:

1. Адрес концентратора: MASTER = FFFF3045h.
2. Адрес подчинённого узла: SLAVE = 31293983h.
3. Ретрансляция: нет.
4. База обнаружения: 713D0900h.
5. Размер области обнаружения: 44D2h.



Ответ:

1. Адрес обнаруженного узла: ADDR = 713D9711h.
2. VER, HP, BCNF − см. 01h − Чтение конфигурации модема SLAVE-узла.

ПРИМЕЧАНИЯ:

Поле RES может принимать следующее множество значений:

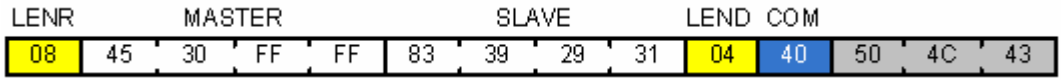
– 00 − обнаружен узел ADDR, сканирование завершено
– 01 − обнаружен узел ADDR, сканирование приостановлено
– 02 − сканирование завершено, узлы не обнаружены
– FE − формат запроса не соответствует спецификации
– FF − данный тип запроса не поддерживается данной версией PLC-модема

Если 'RES' = 01, то повторная посылка данного запроса с теми же параметрами приводит к продолжению сканирования указанной области

6.2.8 40h − Передача произвольных данных

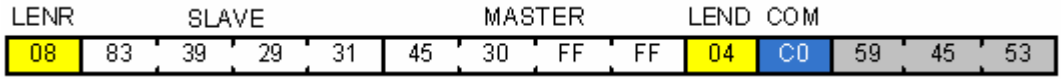
Транспортные пакеты этого типа предназначены для доставки пакетов с произвольными данными от концентратора к хост-устройствам подчинённых узлов PLC-сети и обратно.

Пример:



Запрос:

1. Адрес концентратора: 'MASTER' = FFFF3045h.
2. Адрес подчинённого узла: 'SLAVE' = 31293983h.
3. Полезная нагрузка от концентратора: ASCII-коды строки "PLC".



Ответ: Полезная нагрузка от подчинённого узла: ASCII-коды строки "YES"

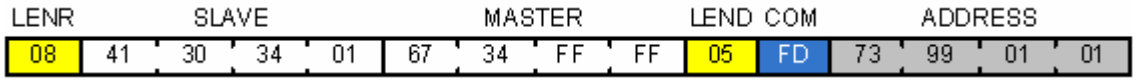
ПРИМЕЧАНИЯ:

Пакеты, у которых поле 'LEND' = 1 (т.е. не несущие другой полезной нагрузки кроме кода команды), считаются тестовыми, и, без дополнительной обработки возвращаются модемом назад, в направлении концентратора.

6.2.9 FDh − Неудачная попытка установить соединение с узлом

Пакет данного типа отправляется узлу-источнику сообщения в случае, когда подчинённому узлу не удалось принять эхо-микропакет во время попытки установить соединение со следующим узлом в цепочке. Пакет отправляется 'от имени' последнего активного узла. В качестве полезной нагрузки отправляется адрес узла, с которым нет связи.

Пример:



1. Адрес концентратора: 'MASTER' = FFFF3467h.
2. Адрес подчинённого узла в маршрутной цепочке, которому не удалось связаться со следующим узлом: 'SLAVE' = 01343041h.
3. Адрес подчинённого узла, с которым нет связи: 01019973h.

6.2.10 FEh − Формат транспортного пакета не соответствует спецификации (COPY)

В случае, если полученный модемом транспортный пакет с запросом, не соответствует спецификации (например, в случае, когда поле LEN=0), модем сигнализирует об этой ситуации тем, что отправляет в сторону концентратора ответный транспортный пакет специального вида:

Пример:

LENR	SLAVE				MASTER				LEND	COM
08	83	39	29	31	45	30	FF	FF	01	FE

1. Адрес концентратора: 'MASTER' = FFFF3045h.
2. Адрес подчинённого узла: 'SLAVE' = 31293983h.

6.2.11 FFh − Неудачная попытка передать транспортный пакет

Пакет данного типа отправляется узлу-источнику сообщения в случае, когда подчинённому узлу не удалось принять эхо-микропакет во время попытки передать транспортный пакет следующему узлу в цепочке. Пакет отправляется 'от имени' последнего активного узла. В качестве полезной нагрузки отправляется адрес узла, с которым нет связи.

Пример:

LENR	SLAVE				MASTER				LEND	COM	ADDRESS			
08	41	30	34	01	67	34	FF	FF	05	FF	73	99	01	01

1. Адрес концентратора: 'MASTER' = FFFF3467h.
2. Адрес подчинённого узла в маршрутной цепочке, которому не удалось связаться со следующим узлом: 'SLAVE' = 01343041h.
3. Адрес подчинённого узла, с которым нет связи: 01019973h.

6.3 Передача транспортных пакетов с ретрансляцией

Данная глава описывает порядок формирования транспортных пакетов в случае их ретрансляции промежуточными звеньями системы.

В качестве примера рассматривается частный случай сети, образованной четырьмя узлами: одним главным и тремя подчинёнными. Предполагается:

– концентратор 'MASTER' поддерживает устойчивую связь только с узлом 'A';
– узел 'A' поддерживает устойчивую связь с узлами 'MASTER' и 'B';
– узел 'B' поддерживает устойчивую связь с узлами 'A' и 'SLAVE';
– узел 'SLAVE' поддерживает устойчивую связь только с узлом 'B'.

Адреса узлов приведены в следующей таблице:

Узел	Адрес
MASTER	FFFF3001h
A	01000002h
B	01000003h
SLAVE	01000004h

На рисунке 5 изображена схема, иллюстрирующая вышеописанную взаимосвязь.

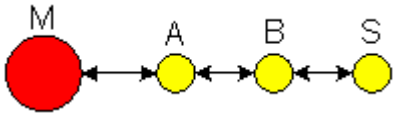
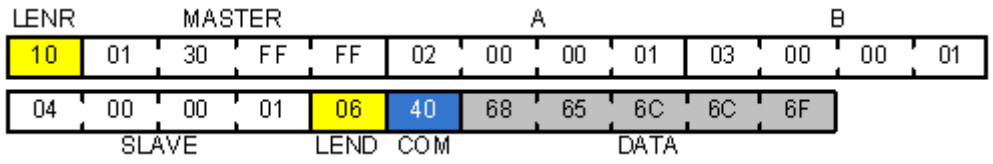
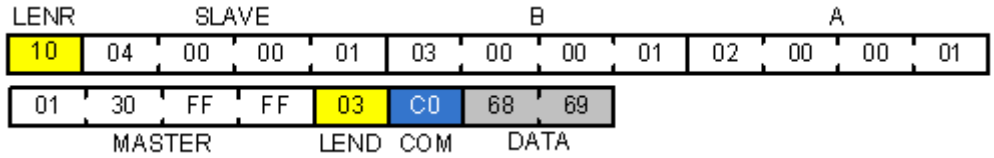


Рисунок 5 Схема взаимной видимости узлов сети.

При данной структуре взаимной видимости узлов обмен данными между концентратором 'MASTER' и подчинённым узлом 'SLAVE' выглядит следующим образом:



Транспортный пакет с запросом "MASTER → A → B → SLAVE"



Транспортный пакет с ответом "SLAVE → B → A → MASTER"

7 Сетевой уровень

Данная глава описывает сетевой уровень протокольного стека PLC-системы. Сетевой уровень отвечает за адресацию узлов сети и маршрутизацию транспортных пакетов.

7.1 Структура PLC-сети

PLC-сеть имеет централизованную структуру типа 'дерево', корнем которой является концентратор, он же − узел доступа к сети, см. Введение.

В соответствии с протоколом доступа к среде, инициировать передачу данных может только концентратор. Он же накапливает и хранит информацию, необходимую для построения маршрутов передачи транспортных пакетов сети.

Топология сети формируется по мере накопления концентратором информации о взаимной видимости подчинённых узлов.

В случае, если между концентратором и узлом, с которым необходимо связаться, отсутствует прямая видимость, используется механизм ретрансляции транспортных пакетов. Запрос в виде транспортного пакета передаётся от узла к узлу до тех пор, пока не достигнет точки назначения. Ответ на запрос передаётся по тому же маршруту в обратном направлении.

В силу описанной специфики PLC-сети, сетевой уровень протокольного стека объединён с транспортным уровнем. Информация о маршруте передачи помещается в транспортный пакет в виде специального поля. Формат транспортных пакетов описан в разделе Транспортный уровень.

Вся информация о структуре сети в целом и о каждом из подчинённых узлов в отдельности сохраняется в энергонезависимой памяти концентратора и при отключении питания не теряется.

7.2 Маршрутизация

В памяти концентратора с каждым из зарегистрированных подчинённых узлов ассоциирована таблица из максимум 16-ти родительских узлов, т.е. других узлов сети (включая, в случае прямой видимости, сам концентратор), через которые когда-либо с данным узлом удавалось установить связь. Каждому из родительских узлов концентратор присваивает рейтинг. Маршрут к данному узлу прокладывается концентратором динамически, путём обратного просмотра таблиц родительских узлов, при этом на каждом этапе используется родительский узел с наибольшим рейтингом.

7.3 Задачи концентратора

Исполнение каждой системной задачи разрешается специальным флажком, входящим в состав регистра конфигурации концентратора, см. MASTER: Запись/чтение конфигурации. Концентратор может квазиодновременно выполнять следующие системные задачи:

'SCAN' − поиск новых подчинённых узлов

'CHECK' − проверка связи, повторное подключение "отпавших" узлов

'MAIL' − автоматическое исполнение запросов

7.3.1 SCAN

Действия, выполняемые концентратором при поиске новых подчинённых узлов описаны в нижеследующих разделах.

7.3.1.1 Поиск SLAVE-узлов

Вследствие большого размера адресного пространства, концентратор PLC-сети не в состоянии обнаруживать подчинённые узлы сети методом прямого перебора адресов. Только лишь обнаружение узлов первого уровня ретрансляции потребовало бы повторения процедуры "запрос-ответ" 4 294 967 296 раз (2 в 32-й степени).

В целях существенного сокращения количества циклов обмена, необходимых для осуществления исчерпывающего поиска подчинённых узлов, используется дихотомический алгоритм поиска.

Механизм алгоритма основан на идентичности ответов всех узлов, попадающих в адресную область пакетов сканирования. Таким образом, появление эхо-пакета при сканировании заданной области сигнализирует о наличии в указанной адресной области как минимум одного узла, находящегося в пределах прямой видимости.

Обнаружение подчинённых узлов начинается с их поиска в пределах прямой видимости самого концентратора. Для этого концентратор разбивает всё адресное пространство пополам и отправляет микропакет uP-scan, охватывающий его младшую половину. Если в ответ приходит эхо-пакет W-24, то это значит, что в данной области имеется как минимум один узел, способный напрямую устойчиво поддерживать связь с концентратором. В противном случае, опрошенная область считается "пустой", а сканированию подвергается верхняя половина адресного пространства. Далее процедура поиска продолжается итеративным дроблением адресного пространства на всё более и более мелкие сегменты и заканчивается сокращением до единицы размеров всех тех сегментов, из которых поступают эхо-пакеты W-24. Базовые адреса таких сегментов и являются адресами обнаруженных узлов. При этом на последнем этапе обнаружения узла, вместо заключительного сканирования области размером в один адрес, используется транспортный пакет с запросом конфигурации PLC-модема, см. 01h − Чтение конфигурации модема SLAVE-узла. Сразу же после обнаружения концентратор проводит регистрацию вновь обнаруженного узла.

Для целей учёта обнаруженных подчинённых узлов концентратор создаёт и поддерживает в актуальном состоянии внутреннюю базу данных, содержащую перечень обнаруженных подчинённых узлов.

Во избежание повторного обнаружения, поиск узлов во втором уровне ретрансляции осуществляется исключительно в тех сегментах адресного пространства, которые были ранее помечены концентратором как "пустые".

По завершению первичного сканирования адресного пространства, концентратор повторяет процедуру сканирования, используя уже обнаруженные узлы в качестве ретрансляторов.

При реализации алгоритма обнаружения узлов следует учитывать большие временные затраты, необходимые для проверки всего лишь 3-4 уровней ретрансляции.

Вся информация о структуре сети в целом и о каждом из подчинённых узлов в отдельности сохраняется в энергонезависимой памяти концентратора и при отключении питания не теряется.

7.3.1.2 Система виртуальных почтовых ящиков

При всяком обнаружении нового SLAVE-узла, концентратор регистрирует его, т.е. заводит для него в своей памяти виртуальный сегментированный почтовый ящик, имеющий в каждом из сегментов "отделения" для запросов, направляемых к данному узлу, ответов, полученных на данный запрос и регистр статуса. Доступ к сегментам почтовых ящиков SLAVE-узлов реализован через UART-интерфейс концентратора, см. Интерфейс концентратора.

7.3.2 CHECK

Концентратор проверяет качество связи с подчинёнными узлами сети путём периодического обмена с ними транспортными пакетами с запросом конфигурации PLC-модема, см. 01h − Чтение конфигурации модема SLAVE-узла. Если 5 подряд таких попыток оказываются неудачными, узел помечается как временно "отпавший".

Повторное подключение временно "отпавших" узлов концентратор осуществляет путём попыток связаться с ними либо напрямую, либо через те узлы, с которыми в данный момент имеется стабильная связь.

7.3.3 MAIL

Концентратор автоматически, на циклической или однократной основе, выполняет запросы, хранящиеся в сегментах почтовых ящиков, см. MBOX+SEG: Запись/чтение отделения запросов указанного сегмента.

7.3.4 SLAVE

Концентратор функционирует в режиме точки ретрансляции транспортных пакетов, не проявляя самостоятельной активности.

8 Интерфейс концентратора

Протокол доступа к концентратору PLC-сети второго поколения совместим с протоколом доступа к концентраторам PLC-сети первого поколения.

Для связи с концентратором используется физический интерфейс RS-485 (38400-8b-NP-1S) и логическая процедура обмена пакетами по формуле "запрос/ответ".

Для предоставления контроллеру сети возможности общения с обнаруженными им подчинёнными узлами, концентратор поддерживает интерфейс, основанный на абстракции виртуального сегментированного "почтового ящика".

Почтовый ящик с соответствующим 4-х байтовым адресом создаётся концентратором сразу же после обнаружения им очередного подчинённого узла PLC-сети. Каждый почтовый ящик имеет несколько (от 4 до 64) функционально идентичных сегментов. Каждый сегмент включает в себя отделение для запроса, направляемого к данному подчинённому узлу, отделение для ответа, полученного на данный запрос, а также регистр статуса. В данном случае запросы и ответы рассматриваются концентратором как неделимые группы байт которые должны быть доставлены к подчинённому узлу и обратно.

Каждый концентратор имеет свой индивидуальный (в пределах данной локальной RS-485 сети) UART-идентификатор, представляющий собой 2-х байтовое число в диапазоне 3001h...3FFEh.

Сетевой PLC-адрес концентратор получается путём добавления к его UART-идентификатору двух байтов FFh слева.

Общий формат как запросов, так и ответов представлен в таблице.

ПАКЕТНЫЙ ОБМЕН ДАННЫМИ ХОСТ ↔ МОДЕМ (общий формат запросов/ответов)

Поле пакета (в порядке следования)	Длина поля (байта)	ПРИМЕЧАНИЯ
CRC24	3	Контрольный код заголовочной части пакета, рассчитанный по содержимому полей SRC+DST+LEN. Служит для обнаружения заголовка пакета в потоке байтов.
SRC	2	UART-ID источника пакета.
DST	2	UART-ID получателя пакета.
LEN	1	Длина полезной нагрузки пакета (0..255) в байтах.
PAYLOAD	LEN	Полезная нагрузка пакета переменной длины.
CHECKSUM	1	Младший байт суммы всех байтов поля PAYLOAD минус 1.

ПРИМЕЧАНИЯ:

- Обязательная группа полей CRC24+SRC+DST+LEN, представляет собой 8-ми байтовый заголовок пакета.
- Алгоритмы вычисления CRC24 даны в "Приложении C".
- UART-идентификаторы концентраторов назначаются в индивидуальном порядке из диапазона 3001h..3FFEh, идентификатор 3FFFh является широковещательным и действителен для всех концентраторов.
- При взаимодействии с концентраторами, контроллер сети использует в качестве своего UART-идентификатора значение FFFFh.
- Сетевой PLC-адрес концентратора получается путём добавления к его UART-идентификатору двух байтов FFh слева, т.е., например, концентратор с UART-идентификатором 3123h будет иметь PLC-адрес FFFF3123h.
- Первый байт поля 'PAYLOAD' во всех случаях определяет тип пакета с запросом/ответом.

– Все поля пакетов, длина которых больше одного байта, передаются младшими байтами вперёд.
– Большую часть команд концентратор исполняет в течении 100 мс с момента поступления запроса, однако на некоторые команды (например, на полную очистку файловой системы) он может потратить до 2.5 сек.

8.1 Типы пакетов

Интерфейс концентратора включает в себя обмен пакетами нескольких предопределённых типов и обеспечивает следующую функциональность:

Тип пакета (HEX)	Мнемоническое обозначение и функциональное назначение пакета
00	MASTER: SET_CONFIG
	Запись конфигурации концентратора
01	MASTER: SET_TIMEDATE
	Запись часов/календаря концентратора
08	MASTER: SET_BAUDRATE
	Переключение скорости работы UART
06h	MASTER: SET_ADDR
	Смена UART-идентификатора концентратора
14h	SLAVE-NODE: SET_P_LIST
	Запись списка "родительских" узлов
15h	MASTER: SET_ADD_NODE
	Добавить узел к списку подчинённых узлов
16h	MASTER: SET_DEL_NODE
	Удалить узел из списка подчинённых узлов
18h	SLAVE-NODE: SET_SL_CONFIG
	Запись/чтение конфигурации подчинённого узла
19h	MASTER: SET_SEG_NUM
	Запись количества сегментов, составляющих почтовый ящик
1Ah	MBOX+SEG: SET_SEG_STATUS
	Запись/чтение регистров статуса (произвольный сегмент)
1Bh	MBOX+SEG: SET_SEG_REQ
	Запись отделения для запросов почтового ящика (произвольный сегмент)
1Dh	MASTER: SET_CLR_ALL_SEG
	Очистка отделений для запросов всех сегментов всех почтовых ящиков
1Eh	SLAVE-NODE: SET_REPLACE_ID
	Смена PLC-ID подчинённого узла
1Fh	SLAVE-NODE: SET_PARENT
	Запись/чтение адреса текущего "родительского" узла
7Dh	MASTER: SET_CLEAR_ST
	Полная очистка файловой системы концентратора
80h	MASTER: GET_CONFIG

Тип пакета (HEX)	Мнемоническое обозначение и функциональное назначение пакета
	Чтение конфигурации концентратора
81h	MASTER: GET_TIMEDATE
	Чтение часов/календаря концентратора
83h	MASTER: GET_VERINFO
	Чтение версии "прошивки" концентратора.
86h	MASTER: GET_ADDR
	Чтение UART-идентификатора концентратора (допускается использовать только в конфигурации с одним устройством)
88h	MASTER: GET_BAUDRATE
	Чтение скорости работы UART
90h	MASTER: GET_NL_PAGE
	Постраничное чтение списка адресов "почтовых ящиков"
94h	SLAVE-NODE: GET_P_LIST
	Чтение списка "родительских" узлов
97h	SLAVE-NODE-MODEM: GET_MOD_CONFIG
	Чтение конфигурации модема
98h	SLAVE-NODE: GET_SL_CONFIG
	Чтение конфигурации подчинённого узла
99h	MASTER: GET_SEG_NUM
	Чтение количества сегментов, составляющих почтовый ящик
9Ah	MBOX+SEG: GET_SEG_STATUS
	Чтение регистров статуса (произвольный сегмент)
9Bh	MBOX+SEG: GET_SEG_REQ
	Чтение отделения для запросов почтового ящика (произвольный сегмент)
9Ch	MBOX+SEG: GET_SEG_ANS
	Чтение отделения для ответов почтового ящика (произвольный сегмент)
9Fh	SLAVE-NODE: GET_PARENT
	Чтение адреса текущего "родительского" узла
A0	MASTER: GET_TIMESLOT
	Постраничное чтение промежуточного буфера
A1	MASTER: GET_LST_INFO
	Чтение информации о последней успешной транзакции
F0h	MASTER: GET_LOG_FILE
	Чтение лог-файла концентратора

8.1.1 MASTER: Чтение UART-идентификатора

Пакеты данного типа используются в случае возникновения необходимости записать новый либо прочитать неизвестный UART-идентификатор данного концентратора. Поле 'DST' в заголовке пакета в этом случае должно быть обязательно заполнено широковещательным идентификатором 3FFFh, действительным для любого концентратора.

Запрос UART-идентификатора концентратора возможен только в том случае, если к последовательной шине подключен лишь один концентратор. В противном случае все услышавшие его концентраторы ответят на него с непредсказуемым результатом.

PAYLOAD				ПРИМЕЧАНИЯ
Запрос GET_ADDR		Ответ GET_ ADDR		
поле	значение	поле	значение	
TYPE	86	TYPE	86	тип пакета (1 байт)
X	X	mUARTL	***	текущий UART-идентификатор концентратора (мл. байт)
X	X	mUARTH	***	текущий UART-идентификатор концентратора (ст. байт)

ПРИМЕЧАНИЯ:

Знаком 'X' отмечены неиспользуемые ячейки таблицы, знак *** означает наличие данных в указанном формате.

8.1.2 MASTER: Смена UART-идентификатора

PAYLOAD				ПРИМЕЧАНИЯ
Запрос SET_ADDR		Ответ GET_ ADDR		
поле	значение	поле	значение	
TYPE	06	TYPE	86	тип пакета (1 байт)
mUARTL	***	mUARTL	***	новый UART-идентификатор концентратора (мл. байт)
mUARTH	***	mUARTH	***	новый UART-идентификатор концентратора (ст. байт)

ПРИМЕЧАНИЯ:

Внимание! Перед посылкой этой команды следует перевести концентратор в режим IDLE, см. MASTER: Запись/чтение конфигурации.

Внимание! Выполнение данной команды приводит к разрушению топологической структуры сети.

Знак *** − означает наличие данных в указанном формате.

Новое значение UART-идентификатора концентратора должно лежать в интервале 3001h...3FFEh.

Для безусловной смены UART-идентификатора в поле 'DST' заголовка пакета следует указать широковещательный идентификатор 3FFFh. Такой вариант смены идентификатора можно применять только в том случае, если к последовательной шине подключен один-единственный концентратор. В противном случае, все услышавшие этот пакет концентраторы синхронно поменяют свои идентификаторы на указанный в пакете.

8.1.3 MASTER: Чтение версии прошивки

PAYLOAD				ПРИМЕЧАНИЯ
Запрос GET_VERINFO		Ответ GET_VERINFO		
поле	значение	поле	значение	
TYPE	83	TYPE	83	тип пакета (1 байт)
X	X	VERSTR	***	произвольная последовательность текстовых символов длиной не более 255 байт

ПРИМЕЧАНИЯ:

Старшая тетрада VERSTR соответствует старшей цифре номера версии микропрограммы, младшая тетрада VERSTR − младшей цифре номера версии.

Знаком 'X' отмечены неиспользуемые ячейки таблицы, знак *** означает наличие данных в указанном формате.

8.1.4 MASTER: Запись/чтение конфигурации

PAYLOAD				ПРИМЕЧАНИЯ
Запрос SET_CONFIG		Ответ GET_CONFIG		
поле	значение	поле	значение	
TYPE	00	TYPE	80	тип пакета (1 байт)
FLAGS	***	FLAGS	***	регистр битовых флажков (1 байт)
PAYLOAD				ПРИМЕЧАНИЯ
Запрос GET_CONFIG		Ответ GET_CONFIG		
поле	значение	поле	значение	
TYPE	80	TYPE	80	тип пакета (1 байт)
X	X	FLAGS	***	регистр битовых флажков (1 байт)

ПРИМЕЧАНИЯ:

Поле FLAGS представляет собой коллекцию битовых флажков и имеет следующий формат:

FLAGS.0 − 'SCAN', поиск новых подчинённых узлов

FLAGS.1 − 'SLAVE', "подчинённый" режим работы концентратора

FLAGS.2 − 'CHECK', проверка связи, повторное подключение

FLAGS.3 − 'MAIL', автоматическое исполнение запросов, содержащихся в отделениях для запросов почтовых ящиков

FLAGS.4 − не используется

FLAGS.5 − 'ABC', обрабатывать PLC-сигнал по трём фазам поочерёдно (сбрасывается, если установлен флажок SLAVE)

FLAGS.6 − не используется

FLAGS.7 − 'DLS', разрешить автоматический перевода часов концентратора на летнее время (ReadOnly, возвращаемое значение всегда 0)

Не все комбинации битовых флажков {SCAN, SLAVE, CHECK, MAIL} допустимы, имеются следующие режимы работы концентратора, каждому из которых соответствует определённое сочетание этих флажков:

MODE=IDLE − все флажки сброшены

MODE=MASTER − флажки SCAN+CHECK+MAIL установлены

MODE=SLAVE − флажок SLAVE установлен (в этом режиме концентратор работает только как ретранслятор пакетов)

Подробнее о сканировании адресного пространства см. SCAN.

Знаком 'X' отмечены неиспользуемые ячейки таблицы, знак *** означает наличие данных в указанном формате.

8.1.5 MASTER: Запись/чтение часов/календаря

PAYLOAD				ПРИМЕЧАНИЯ
Запрос SET_TIMEDATE		Ответ GET_TIMEDATE		
поле	значение	поле	значение	
TYPE	01	TYPE	81	тип пакета (1 байт)

PAYLOAD				ПРИМЕЧАНИЯ
Запрос SET_TIMEDATE		Ответ GET_TIMEDATE		
поле	значение	поле	значение	
SEC	***	SEC	***	секунды
MIN	***	MIN	***	минуты
HOUR	***	HOUR	***	часы
WDAY	***	WDAY	***	день недели (0..6; 0-понедельник)
DATE	***	DATE	***	дата (0..30)
MONTH	***	MONTH	***	месяц (0..11)
YEAR	***	YEAR	***	год (0..99)
PAYLOAD				ПРИМЕЧАНИЯ
Запрос GET_TIMEDATE		Ответ GET_TIMEDATE		
поле	значение	поле	значение	
TYPE	81	TYPE	81	тип пакета (1 байт)
X	X	SEC	***	секунды
X	X	MIN	***	минуты
X	X	HOUR	***	часы
X	X	WDAY	***	день недели (0..6; 0-понедельник)
X	X	DATE	***	дата (0..30)
X	X	MONTH	***	месяц (0..11)
X	X	YEAR	***	год (0..99)

Знаком 'X' отмечены неиспользуемые ячейки таблицы, знак *** означает наличие данных в указанном формате.

8.1.6 MASTER: Постраничное чтение списка адресов подчинённых узлов

Этот тип пакетов предназначен для постраничного чтения адресов подчинённых узлов, обнаруженных к данному моменту концентратором PLC-сети. Каждый адрес подчинённого узла имеет длину 4 байта.

Передача адресов производится страницами размером не более 32 адресов каждая. Формат пакета позволяет адресоваться к 256 страницам с номерами от 0 до 255.

Адреса обнаруженных подчинённых узлов располагаются на страницах последовательно, начиная с младшей, имеющей нулевой адрес.

PAYLOAD				ПРИМЕЧАНИЯ
Запрос GET_NL_PAGE		Ответ GET_NL_PAGE		
поле	значение	поле	значение	
TYPE	90	TYPE	90	тип пакета (1 байт)
PAGE	***	PAGE	***	номер страницы (1 байт)
X	X	ADDR0	***	первый по счёту адрес на данной странице (4 байта)
X	X	ADDR1	***	второй по счёту адрес на данной странице (4 байта)
X	X	ADDR2	***	третий по счёту адрес на данной странице (4 байта)

PAYLOAD				ПРИМЕЧАНИЯ
Запрос GET_NL_PAGE		Ответ GET_NL_PAGE		
поле	значение	поле	значение	
X	X	***	***	и т.д., максимум 32 адреса

ПРИМЕЧАНИЯ:

Знаком 'X' отмечены неиспользуемые ячейки таблицы, знак *** − означает наличие данных в указанном формате.

Все адреса подчинённых узлов 4-х байтовые и передаются младшим байтом вперёд.

Если в ответ на очередной запрос этого типа концентратор вернул частично или полностью пустую страницу, − то эта страница − последняя.

8.1.7 MASTER: Регистрация нового подчинённого узла

PAYLOAD				ПРИМЕЧАНИЯ
Запрос SET_ADD_NODE		Ответ SET_ADD_NODE		
поле	значение	поле	значение	
TYPE	15	TYPE	15	тип пакета (1 байт)
ADDRLL	***	ADDRLL	***	адрес подчиненного узла (мл.байт)
ADDRL	***	ADDRL	***	адрес подчиненного узла, байт #2
ADDRH	***	ADDRH	***	адрес подчиненного узла, байт #3
ADDRHH	***	ADDRHH	***	адрес подчиненного узла (ст.байт)

ПРИМЕЧАНИЯ:

Ответ концентратора в случае невозможности выполнить требуемое действие выглядит следующим образом.

PAYLOAD				ПРИМЕЧАНИЯ
Запрос SET_ADD_NODE		Ответ SET_ADD_NODE		
поле	значение	поле	значение	
TYPE	15	TYPE	15	тип пакета (1 байт)
ADDRLL	***	X	X	адрес подчиненного узла (мл.байт)
ADDRL	***	X	X	адрес подчиненного узла, байт #2
ADDRH	***	X	X	адрес подчиненного узла, байт #3
ADDRHH	***	X	X	адрес подчиненного узла (ст.байт)

Знаком 'X' отмечены неиспользуемые ячейки таблицы, знак *** − означает наличие данных в указанном формате.

Сразу после регистрации нового подчинённого узла, его счётчику ошибок (см. MBOX+SEG: Запись/чтение регистров статуса указанного сегмента) присваивается максимальное значение − это означает, что реальное местоположение данного узла в топологии сети концентратору неизвестно.

8.1.8 MASTER: Удаление узла из списка подчинённых узлов

PAYLOAD				ПРИМЕЧАНИЯ
Запрос SET_ADD_NODE		Ответ SET_ADD_NODE		
поле	значение	поле	значение	
TYPE	16	TYPE	16	тип пакета (1 байт)
ADDRLL	***	ADDRLL	***	адрес подчиненного узла (мл.байт)
ADDRL	***	ADDRL	***	адрес подчиненного узла, байт #2
ADDRH	***	ADDRH	***	адрес подчиненного узла, байт #3
ADDRHH	***	ADDRHH	***	адрес подчиненного узла (ст.байт)

ПРИМЕЧАНИЯ:

Ответ концентратора в случае невозможности выполнить требуемое действие выглядит следующим образом.

PAYLOAD				ПРИМЕЧАНИЯ
Запрос SET_ADD_NODE		Ответ SET_ADD_NODE		
поле	значение	поле	значение	
TYPE	16	TYPE	16	тип пакета (1 байт)
ADDRLL	***	X	X	адрес подчиненного узла (мл.байт)
ADDRL	***	X	X	адрес подчиненного узла, байт #2
ADDRH	***	X	X	адрес подчиненного узла, байт #3
ADDRHH	***	X	X	адрес подчиненного узла (ст.байт)

Знаком 'X' отмечены неиспользуемые ячейки таблицы, знак *** − означает наличие данных в указанном формате.

8.1.9 MASTER: Полная очистка файловой системы концентратора

Очистка файловой системы концентратора приводит к полной утрате информации о подчинённых узлах.

PAYLOAD				ПРИМЕЧАНИЯ
Запрос SET_CLEAR_ST		Ответ GET_CLEAR_ST		
поле	значение	поле	значение	
TYPE	7D	TYPE	7D	тип пакета (1 байт)

ПРИМЕЧАНИЯ:

Одновременно с очисткой файловой системы концентратор переводится в режим IDLE, см. MASTER: Запись/чтение конфигурации.

8.1.10 MASTER: Чтение лог-файла

PAYLOAD				ПРИМЕЧАНИЯ
Запрос GET_LOG_FILE		Ответ GET_LOG_FILE		
поле	значение	поле	значение	
TYPE	F0	TYPE	F0	тип пакета (1 байт)
X	X	LOG	***	Лог-файл концентратора (240 байт)

ПРИМЕЧАНИЯ:

Знаком 'X' отмечены неиспользуемые ячейки таблицы, знак *** − означает наличие данных в указанном формате.

8.1.11 MASTER: Запись/чтение количества сегментов

PAYLOAD				ПРИМЕЧАНИЯ
Запрос SET_SEG_NUM		Ответ GET_SEG_NUM		
поле	значение	поле	значение	
TYPE	19	TYPE	99	тип пакета (1 байт)
SNUM	***	SNUM	***	количество сегментов в каждом почтовом ящике (4..64)
PAYLOAD				ПРИМЕЧАНИЯ
Запрос GET_SEG_NUM		Ответ GET_SEG_NUM		
поле	значение	поле	значение	
TYPE	99	TYPE	99	тип пакета (1 байт)
X	X	SNUM	***	количество сегментов в каждом почтовом ящике

ПРИМЕЧАНИЯ:

При изменении количества сегментов (поле SNUM) концентратор производит операцию полной очистки своей файловой системы, см. MASTER: Полная очистка файловой системы концентратора.

По умолчанию количество сегментов равно 4.

Знаком 'X' отмечены неиспользуемые ячейки таблицы, знак *** − означает наличие данных в указанном формате.

8.1.12 MASTER: Очистка всех запросов

PAYLOAD				ПРИМЕЧАНИЯ
Запрос SET_CLR_ALL_SEG		Ответ SET_CLR_ALL_SEG		
поле	значение	поле	значение	
TYPE	1D	TYPE	1D	тип пакета (1 байт)

8.1.13 MASTER: Чтение промежуточного буфера оцифрованного сигнала линии

Этот тип пакетов предназначен для постраничного чтения промежуточного буфера в котором концентратор сохраняет оцифрованный сигнал с PLC-линии. Полная длина буфера − 1024 шестнадцатиразрядных слова, которые разбиты на 16 страниц по 64 слова в каждом.

PAYLOAD				ПРИМЕЧАНИЯ
Запрос GET_TIMESLOT		Ответ GET_TIMESLOT		
поле	значение	поле	значение	
TYPE	A0	TYPE	A0	тип пакета (1 байт)
PAGE	***	PAGE	***	номер страницы (1 байт, в интервале 0...15)
X	X	D0L	***	мл.байт первого слова данных
X	X	D0H	***	ст.байт первого слова данных

PAYLOAD				ПРИМЕЧАНИЯ
Запрос GET_TIMESLOT		Ответ GET_TIMESLOT		
поле	значение	поле	значение	
X	X	D1L	***	мл.байт второго слова данных
X	X	D1H	***	ст.байт второго слова данных
X	X	***	***	и т.д.
X	X	D63L	***	мл.байт последнего слова данных
X	X	D63H	***	ст.байт последнего слова данных

ПРИМЕЧАНИЯ:

Данный механизм работает только в режимах концентратора IDLE и MASTER, см. MASTER: Запись/чтение конфигурации. Внимание! в режиме MASTER интервал обновления данных в некоторых ситуациях может составлять десятки секунд.

При неготовности данных к считыванию, концентратор возвращает в ответе только поля TYPE и PAGE.

Знаком 'X' отмечены неиспользуемые ячейки таблицы, знак *** − означает наличие данных в указанном формате.

8.1.14 MASTER: Выбор скорости работы UART

PAYLOAD				ПРИМЕЧАНИЯ
Запрос SET_BAUDRATE		Ответ GET_BAUDRATE		
поле	значение	поле	значение	
TYPE	08	TYPE	88	тип пакета
BDR	***	BDR	***	условный номер скорости обмена по UART-интерфейсу концентратора: 0 − 38400 бод 1 − 19200 бод 2 − 9600 бод
PAYLOAD				ПРИМЕЧАНИЯ
Запрос GET_BAUDRATE		Ответ GET_BAUDRATE		
поле	значение	поле	значение	
TYPE	88	TYPE	88	тип пакета
X	X	BDR	***	см. примечания выше

ПРИМЕЧАНИЯ:

Знаком 'X' отмечены неиспользуемые ячейки таблицы, знак *** − означает наличие данных в указанном формате.

Параметр 'BDR' сохраняется в энергонезависимой памяти концентратора.

8.1.15 MASTER: Информация о последней успешной транзакции

PAYLOAD				ПРИМЕЧАНИЯ
Запрос GET_LST_INFO		Ответ GET_LST_INFO		
поле	значение	поле	значение	
TYPE	A1	TYPE	A1	тип пакета (1 байт)

PAYLOAD				ПРИМЕЧАНИЯ
Запрос GET_LST_INFO		Ответ GET_LST_INFO		
поле	значение	поле	значение	
X	X	ADDRLL	***	адрес подчиненного узла (мл.байт)
X	X	ADDRL	***	адрес подчиненного узла, байт #2
X	X	ADDRH	***	адрес подчиненного узла, байт #3
X	X	ADDRHH	***	адрес подчиненного узла (ст.байт)
X	X	SEG	***	номер сегмента (0..63)
X	X	CRC24L	***	контрольный код CRC24 от выполненного запроса (мл.байт)
X	X	CRC24M	***	контрольный код CRC24 от выполненного запроса (ср.байт)
X	X	CRC24H	***	контрольный код CRC24 от выполненного запроса (ст.байт)

ПРИМЕЧАНИЯ:

Ответ концентратора до выполнения первой успешной транзакции:

PAYLOAD				ПРИМЕЧАНИЯ
Запрос GET_LST_INFO		Ответ GET_LST_INFO		
поле	значение	поле	значение	
TYPE	A1	TYPE	A1	тип пакета (1 байт)

Знаком 'X' отмечены неиспользуемые ячейки таблицы, знак *** − означает наличие данных в указанном формате

8.1.16 SLAVE-NODE: Смена PLC-ID

PAYLOAD				ПРИМЕЧАНИЯ
Запрос SET_REPLACE_ID		Ответ SET_REPLACE_ID		
поле	значение	поле	значение	
TYPE	1E	TYPE	1E	тип пакета (1 байт)
OLDLL	***	OLDLL	***	старый адрес подчиненного узла (мл.байт)
OLDL	***	OLDL	***	старый адрес подчиненного узла, байт #2
OLDH	***	OLDH	***	старый адрес подчиненного узла, байт #3
OLDHH	***	OLDHH	***	старый адрес подчиненного узла (ст.байт)
NEWLL	***	NEWLL	***	новый адрес подчиненного узла (мл.байт)
NEWL	***	NEWL	***	новый адрес подчиненного узла, байт #2
NEWH	***	NEWH	***	новый адрес подчиненного узла, байт #3
NEWHH	***	NEWHH	***	новый адрес подчиненного узла (ст.байт)

ПРИМЕЧАНИЯ:

Ответ концентратора в случае невозможности выполнить требуемое действие выглядит следующим образом.

PAYLOAD				ПРИМЕЧАНИЯ
Запрос SET_REPLACE_ID		Ответ SET_REPLACE_ID		
поле	значение	поле	значение	
TYPE	1E	TYPE	1E	тип пакета (1 байт)
OLDLL	***	X	X	старый адрес подчиненного узла (мл.байт)
OLDL	***	X	X	старый адрес подчиненного узла, байт #2
OLDH	***	X	X	старый адрес подчиненного узла, байт #3
OLDHH	***	X	X	старый адрес подчиненного узла (ст.байт)
NEWLL	***	X	X	новый адрес подчиненного узла (мл.байт)
NEWL	***	X	X	новый адрес подчиненного узла, байт #2
NEWH	***	X	X	новый адрес подчиненного узла, байт #3
NEWHH	***	X	X	новый адрес подчиненного узла (ст.байт)

Знаком 'X' отмечены неиспользуемые ячейки таблицы, знак *** − означает наличие данных в указанном формате

8.1.17 SLAVE-NODE: Запись/чтение списка родительских узлов

PAYLOAD				ПРИМЕЧАНИЯ
Запрос SET_P_LIST		Ответ GET_P_LIST		
поле	значение	поле	значение	
TYPE	14	TYPE	94	тип пакета (1 байт)
ADDRLL	***	ADDRLL	***	адрес подчиненного узла (мл.байт)
ADDRL	***	ADDRL	***	адрес подчиненного узла, байт #2
ADDRH	***	ADDRH	***	адрес подчиненного узла, байт #3
ADDRHH	***	ADDRHH	***	адрес подчиненного узла (ст.байт)
P-LIST	***	P-LIST	***	список "родительских" узлов в виде последовательности 4-х байтовых PLC-адресов (не более 8-ми)
PAYLOAD				ПРИМЕЧАНИЯ
Запрос GET_P_LIST		Ответ GET_P_LIST		
поле	значение	поле	значение	
TYPE	94	TYPE	94	тип пакета (1 байт)
ADDRLL	***	ADDRLL	***	адрес подчиненного узла (мл.байт)
ADDRL	***	ADDRL	***	адрес подчиненного узла, байт #2
ADDRH	***	ADDRH	***	адрес подчиненного узла, байт #3
ADDRHH	***	ADDRHH	***	адрес подчиненного узла (ст.байт)
X	X	P-LIST	***	список "родительских" узлов в виде последовательности 4-х байтовых PLC-адресов (не более 8-ми)

ПРИМЕЧАНИЯ:

Ответ концентратора в случае неверно указанного адреса подчинённого узла выглядит следующим образом.

PAYLOAD				ПРИМЕЧАНИЯ
Запрос GET_P_LIST		Ответ GET_P_LIST		
поле	значение	поле	значение	
TYPE	94	TYPE	94	тип пакета (1 байт)
ADDRLL	***	X	X	адрес подчиненного узла (мл.байт)
ADDRL	***	X	X	адрес подчиненного узла, байт #2
ADDRH	***	X	X	адрес подчиненного узла, байт #3
ADDRHH	***	X	X	адрес подчиненного узла (ст.байт)

Ответ концентратора в случае, когда список родительских узлов пуст выглядит следующим образом.

PAYLOAD				ПРИМЕЧАНИЯ
Запрос GET_P_LIST		Ответ GET_P_LIST		
поле	значение	поле	значение	
TYPE	94	TYPE	94	тип пакета (1 байт)
ADDRLL	***	ADDRLL	***	адрес подчиненного узла (мл.байт)
ADDRL	***	ADDRL	***	адрес подчиненного узла, байт #2
ADDRH	***	ADDRH	***	адрес подчиненного узла, байт #3
ADDRHH	***	ADDRHH	***	адрес подчиненного узла (ст.байт)

Знаком 'X' отмечены неиспользуемые ячейки таблицы, знак *** − означает наличие данных в указанном формате

8.1.18 SLAVE-NODE: Запись/чтение конфигурации

PAYLOAD				ПРИМЕЧАНИЯ
Запрос SET_SL_CONFIG		Ответ GET_SL_CONFIG		
поле	значение	поле	значение	
TYPE	18	TYPE	98	тип пакета (1 байт)
ADDRLL	***	ADDRLL	***	адрес подчиненного узла (мл.байт)
ADDRL	***	ADDRL	***	адрес подчиненного узла, байт #2
ADDRH	***	ADDRH	***	адрес подчиненного узла, байт #3
ADDRHH	***	ADDRHH	***	адрес подчиненного узла (ст.байт)
FLAGS	***	FLAGS	***	регистр битовых флажков (1 байт)
PAYLOAD				ПРИМЕЧАНИЯ
Запрос GET_SL_CONFIG		Ответ GET_SL_CONFIG		
поле	значение	поле	значение	
TYPE	98	TYPE	98	тип пакета (1 байт)
ADDRLL	***	ADDRLL	***	адрес подчиненного узла (мл.байт)
ADDRL	***	ADDRL	***	адрес подчиненного узла, байт #2
ADDRH	***	ADDRH	***	адрес подчиненного узла, байт #3
ADDRHH	***	ADDRHH	***	адрес подчиненного узла (ст.байт)
X	X	FLAGS	***	регистр битовых флажков (1 байт)

ПРИМЕЧАНИЯ:

Поле 'FLAGS' представляет собой коллекцию битовых флажков и имеет следующий формат:

FLAGS.0 − 'LOCK_PARENT', зафиксировать "родительский" узел

FLAGS.1 ... FLAGS.7 − не используются

Ответ концентратора в случае неверно указанного адреса подчинённого узла выглядит следующим образом.

PAYLOAD				ПРИМЕЧАНИЯ
Запрос GET_SL_CONFIG		Ответ GET_SL_CONFIG		
поле	значение	поле	значение	
TYPE	98	TYPE	98	тип пакета (1 байт)
ADDRLL	***	X	X	адрес подчиненного узла (мл.байт)
ADDRL	***	X	X	адрес подчиненного узла, байт #2
ADDRH	***	X	X	адрес подчиненного узла, байт #3
ADDRHH	***	X	X	адрес подчиненного узла (ст.байт)

Знаком 'X' отмечены неиспользуемые ячейки таблицы, знак *** − означает наличие данных в указанном формате

8.1.19 SLAVE-NODE: Запись/чтение адреса текущего родительского узла

PAYLOAD				ПРИМЕЧАНИЯ
Запрос SET_PARENT		Ответ GET_PARENT		
поле	значение	поле	значение	
TYPE	1F	TYPE	9F	тип пакета (1 байт)
ADDRLL	***	ADDRLL	***	адрес подчиненного узла (мл.байт)
ADDRL	***	ADDRL	***	адрес подчиненного узла, байт #2
ADDRH	***	ADDRH	***	адрес подчиненного узла, байт #3
ADDRHH	***	ADDRHH	***	адрес подчиненного узла (ст.байт)
PRNTLL	***	PRNTLL	***	адрес родительского узла (мл.байт)
PRNTL	***	PRNTL	***	адрес родительского узла, байт #2
PRNTH	***	PRNTH	***	адрес родительского узла, байт #3
PRNTHH	***	PRNTHH	***	адрес родительского узла (ст.байт)

PAYLOAD				ПРИМЕЧАНИЯ
Запрос GET_PARENT		Ответ GET_PARENT		
поле	значение	поле	значение	
TYPE	9F	TYPE	9F	тип пакета (1 байт)
ADDRLL	***	ADDRLL	***	адрес подчиненного узла (мл.байт)
ADDRL	***	ADDRL	***	адрес подчиненного узла, байт #2
ADDRH	***	ADDRH	***	адрес подчиненного узла, байт #3
ADDRHH	***	ADDRHH	***	адрес подчиненного узла (ст.байт)

PAYLOAD				ПРИМЕЧАНИЯ
Запрос GET_PARENT		Ответ GET_PARENT		
поле	значение	поле	значение	
X	X	PRNTLL	***	адрес родительского узла (мл.байт)
X	X	PRNTL	***	адрес родительского узла, байт #2
X	X	PRNTH	***	адрес родительского узла, байт #3
X	X	PRNTHH	***	адрес родительского узла (ст.байт)

ПРИМЕЧАНИЯ:

Ответ концентратора в случае неверно указанного адреса подчинённого узла выглядит следующим образом.

PAYLOAD				ПРИМЕЧАНИЯ
Запрос GET_PARENT		Ответ GET_PARENT		
поле	значение	поле	значение	
TYPE	9F	TYPE	9F	тип пакета (1 байт)
ADDRLL	***	X	X	адрес подчиненного узла (мл.байт)
ADDRL	***	X	X	адрес подчиненного узла, байт #2
ADDRH	***	X	X	адрес подчиненного узла, байт #3
ADDRHH	***	X	X	адрес подчиненного узла (ст.байт)

Знаком 'X' отмечены неиспользуемые ячейки таблицы, знак *** − означает наличие данных в указанном формате

8.1.20 MBOX+SEG: Запись/чтение регистров статуса указанного сегмента

Для каждого из вновь обнаруженных подчинённых узлов концентратор открывает индивидуальный "почтовый ящик", включающий в себя несколько сегментов со следующей структурой:

– "отделение" для запросов, предназначенных для данного подчинённого узла PLC-сети;
– "отделение" для ответов, полученных от данного подчинённого узла PLC-сети;
– регистры статуса.

PAYLOAD				ПРИМЕЧАНИЯ
Запрос SET_SEG_STATUS		Ответ GET_SEG_STATUS		
поле	значение	поле	значение	
TYPE	1A	TYPE	9A	тип пакета (1 байт)
ADDRLL	***	ADDRLL	***	адрес почтового ящика (мл.байт)
ADDRL	***	ADDRL	***	адрес почтового ящика, байт #2
ADDRH	***	ADDRH	***	адрес почтового ящика, байт #3
ADDRHH	***	ADDRHH	***	адрес почтового ящика (ст.байт)
SEG	***	SEG	***	номер сегмента (0..63)
STATUS	***	STATUS	***	статус сегмента почтового ящика (1 байт)
X	X	TIME	***	таймер продолжительности цикла обмена (2 байта)

PAYLOAD				ПРИМЕЧАНИЯ
Запрос SET_SEG_STATUS		Ответ GET_SEG_STATUS		
поле	значение	поле	значение	
X	X	errCNT	***	счётчик неудачных попыток обращения к узлу (1 байт)
PAYLOAD				ПРИМЕЧАНИЯ
Запрос GET_SEG_STATUS		Ответ GET_SEG_STATUS		
поле	значение	поле	значение	
TYPE	9A	TYPE	9A	тип пакета (1 байт)
ADDRLL	***	ADDRLL	***	адрес почтового ящика (мл.байт)
ADDRL	***	ADDRL	***	адрес почтового ящика, байт #2
ADDRH	***	ADDRH	***	адрес почтового ящика, байт #3
ADDRHH	***	ADDRHH	***	адрес почтового ящика (ст.байт)
SEG	***	SEG	***	номер сегмента (0..63)
X	X	STATUS	***	статус сегмента почтового ящика (1 байт)
X	X	TIME	***	таймер продолжительности цикла обмена (2 байта)
X	X	errCNT	***	счётчик неудачных попыток обращения к узлу (1 байт)

ПРИМЕЧАНИЯ:

Структура поля 'STATUS':

STATUS.0 ... STATUS.3 (ReadOnly) − нумератор состояний данного сегмента:

0 − 'EMPTY', − "отделение" для запросов свободно (пусто)

1 − 'PENDING', − "отделение" для запросов занято запросом, который ожидает своей очереди на отправку к узлу назначения

2 − 'TRANSACTION', − данный запрос находится в процессе передачи подчинённому узлу

3 − 'OK', − на данный запрос получен ответ от удалённого узла, который в данный момент находится в "отделении" для ответов

4 − 'ERROR', − транзакция с данным узлом закончилась неудачно

5 − 'T.ERROR', − истёк таймаут ожидания ответа от подчинённого узла

6..7 − не используются

8..15 − 'RX.ERROR', − ошибка ретрансляции в уровнях с 1 до 8+ ('R2.ERROR' − ошибка ретрансляции во втором уровне)

STATUS.4 − 'ANS_READY' (ReadOnly), − битовый флажок готовности ответа

STATUS.5 − 'TRANS_TYPE' (ReadWrite), − битовый флажок типа запроса:

0 − запрос предназначен для однократного исполнения

1 − запрос предназначен для периодического исполнения

Начальное значение этого флажка − 0, при перезаписи запроса его состояние не изменяется.

STATUS.6..7 − 'PCODE' (ReadWrite), − код временного интервала для периодических запросов (STATUS.5 = 1)

0 − минимально возможный (PERIOD=0)

1 − раз в сутки (PERIOD=DAY)

2 − раз в неделю (PERIOD=WEEK)

3 − раз в месяц (PERIOD=MONTH)

Поле 'TIME' содержит показания таймера продолжительности текущего цикла обмена с данным подчинённым узлом (время прошедшее между посылкой запроса и получением ответа в десятках миллисекунд). В ходе цикла обмена с узлом это поле инкрементируется 100 раз в секунду.

Поле 'errCNT' содержит счётчик ошибок обмена с узлом. Его содержимое увеличивается на единицу всякий раз после неуспешной попытки обращения к данному подчинённому узлу.

Ответ концентратора в случае неверно указанного номера почтового ящика или его сегмента:

PAYLOAD				ПРИМЕЧАНИЯ
Запрос GET_SEG_STATUS		Ответ GET_SEG_STATUS		
поле	значение	поле	значение	
TYPE	9A	TYPE	9A	тип пакета (1 байт)
ADDRLL	***	X	X	адрес почтового ящика (мл.байт)
ADDRL	***	X	X	адрес почтового ящика, байт #2
ADDRH	***	X	X	адрес почтового ящика, байт #3
ADDRHH	***	X	X	адрес почтового ящика (ст.байт)
SEG	***	X	X	номер сегмента (0..63)

Знаком 'X' отмечены неиспользуемые ячейки таблицы, знак *** − означает наличие данных в указанном формате

8.1.21 MBOX+SEG: Запись/чтение отделения запросов указанного сегмента

Запись в отделение для запросов произвольного сегмента:

PAYLOAD				ПРИМЕЧАНИЯ
Запрос SET_SEG_REQ		Ответ GET_SEG_REQ		
поле	значение	поле	значение	
TYPE	1B	TYPE	9B	тип пакета (1 байт)
ADDRLL	***	ADDRLL	***	адрес почтового ящика (мл.байт)
ADDRL	***	ADDRL	***	адрес почтового ящика, байт #2
ADDRH	***	ADDRH	***	адрес почтового ящика, байт #3
ADDRHH	***	ADDRHH	***	адрес почтового ящика (ст.байт)
SEG	***	SEG	***	номер сегмента (0..63)
REQ	***	X	X	Запрос к удалённому узлу с указанным адресом

Чтение отделения для запросов произвольного сегмента:

PAYLOAD				ПРИМЕЧАНИЯ
Запрос GET_SEG_REQ		Ответ GET_SEG_REQ		
поле	значение	поле	значение	
TYPE	9B	TYPE	9B	тип пакета (1 байт)
ADDRLL	***	ADDRLL	***	адрес почтового ящика (мл.байт)
ADDRL	***	ADDRL	***	адрес почтового ящика, байт #2
ADDRH	***	ADDRH	***	адрес почтового ящика, байт #3

PAYLOAD				ПРИМЕЧАНИЯ
Запрос GET_SEG_REQ		Ответ GET_SEG_REQ		
поле	значение	поле	значение	
ADDRHH	***	ADDRHH	***	адрес почтового ящика (ст.байт)
SEG	***	SEG	***	номер сегмента (0..63)
X	X	REQ	***	Запрос к удалённому узлу с указанным адресом

ПРИМЕЧАНИЯ:

Поле 'REQ' по формату совпадает с полезной нагрузкой транспортного пакета (см. Структура транспортного пакета) и всегда начинается с поля 'COM', содержание которого определяет кому предназначен данный запрос, − PLC-модему или его хост-устройству, а также тип самого запроса.

При перезаписи запроса отделение для ответов очищается.

При смене запроса "на ходу" (во время его исполнения) возможна ситуация, когда в отделение для ответов попадёт ответ не на новый запрос, а на предыдущий.

Ответ концентратора в случае неверно указанного номера почтового ящика или его сегмента:

PAYLOAD				ПРИМЕЧАНИЯ
Запрос SET_SEG_REQ		Ответ GET_SEG_REQ		
поле	значение	поле	значение	
TYPE	1B	TYPE	9B	тип пакета (1 байт)
ADDRLL	***	X	X	адрес почтового ящика (мл.байт)
ADDRL	***	X	X	адрес почтового ящика, байт #2
ADDRH	***	X	X	адрес почтового ящика, байт #3
ADDRHH	***	X	X	адрес почтового ящика (ст.байт)
SEG	***	X	X	номер сегмента (0..63)
REQ	***	X	X	Запрос к удалённому узлу с указанным адресом
PAYLOAD				ПРИМЕЧАНИЯ
Запрос GET_SEG_REQ		Ответ GET_SEG_REQ		
поле	значение	поле	значение	
TYPE	9B	TYPE	9B	тип пакета (1 байт)
ADDRLL	***	X	X	адрес почтового ящика (мл.байт)
ADDRL	***	X	X	адрес почтового ящика, байт #2
ADDRH	***	X	X	адрес почтового ящика, байт #3
ADDRHH	***	X	X	адрес почтового ящика (ст.байт)
SEG	***	X	X	номер сегмента (0..63)

Знаком 'X' отмечены неиспользуемые ячейки таблицы, знак *** − означает наличие данных в указанном формате

8.1.22 MBOX+SEG: Чтение отделения ответов указанного сегмента

Чтение отделения для ответов произвольного сегмента:

PAYLOAD				ПРИМЕЧАНИЯ
Запрос GET_SEG_ANS		Ответ GET_SEG_ANS		
поле	значение	поле	значение	
TYPE	9C	TYPE	9C	тип пакета (1 байт)
ADDRLL	***	ADDRLL	***	адрес почтового ящика (мл.байт)
ADDRL	***	ADDRL	***	адрес почтового ящика, байт #2
ADDRH	***	ADDRH	***	адрес почтового ящика, байт #3
ADDRHH	***	ADDRHH	***	адрес почтового ящика (ст.байт)
SEG	***	SEG	***	номер сегмента (0..63)
X	X	SEC	***	секунды
X	X	MIN	***	минуты
X	X	HOUR	***	часы
X	X	DATE	***	дата (0..30)
X	X	MONTH	***	месяц (0..11)
X	X	YEAR	***	год (0..99)
X	X	ANS	***	Ответ удалённого узла с указанным адресом

ПРИМЕЧАНИЯ:

Указанные в пакете время/дата соответствуют моменту прихода последнего полученного концентратором ответа от удалённого устройства (по часам концентратора).

Исходное состояние всех полей метки времени/даты − 00h.

Поле 'ANS' по формату совпадает с полезной нагрузкой ответного транспортного пакета и всегда начинается с поля 'COM', см. Структура транспортного пакета.

Данная операция имеет смысл лишь в случае, когда содержимое поля статуса указывает на то, что концентратор уже получил ответ удалённого (STATUS = 3).

Ответ концентратора в случае неверно указанного номера почтового ящика или его сегмента:

PAYLOAD				ПРИМЕЧАНИЯ
Запрос GET_SEG_ANS		Ответ GET_SEG_ANS		
поле	значение	поле	значение	
TYPE	9C	TYPE	9C	тип пакета (1 байт)
ADDRLL	***	X	X	адрес почтового ящика (мл.байт)
ADDRL	***	X	X	адрес почтового ящика, байт #2
ADDRH	***	X	X	адрес почтового ящика, байт #3
ADDRHH	***	X	X	адрес почтового ящика (ст.байт)
SEG	***	X	X	номер сегмента (0..63)

8.1.23 SLAVE-NODE-MODEM: Модем подчинённого узла, чтение конфигурации

PAYLOAD				ПРИМЕЧАНИЯ
Запрос GET_MOD_CONFIG		Ответ GET_MOD_CONFIG		
поле	значение	поле	значение	
TYPE	97	TYPE	97	тип пакета (1 байт)
ADDRLL	***	ADDRLL	***	адрес подчиненного узла (мл.байт)
ADDRL	***	ADDRL	***	адрес подчиненного узла, байт #2
ADDRH	***	ADDRH	***	адрес подчиненного узла, байт #3
ADDRHH	***	ADDRHH	***	адрес подчиненного узла (ст.байт)
X	X	VER	***	номер прошивки PLC-модема подчинённого узла
X	X	HP	***	условный номер протокола хост-устройства подчинённого узла (см. документ "Интерфейс модема")
X	X	BCNF	***	байт битовых флажков конфигурации PLC-модема, см. 01h − Чтение конфигурации модема SLAVE-узла

ПРИМЕЧАНИЯ:

Ответ концентратора в случае отсутствия запрашиваемых данных выглядит следующим образом.

PAYLOAD				ПРИМЕЧАНИЯ
Запрос GET_MOD_CONFIG		Ответ GET_MOD_CONFIG		
поле	значение	поле	значение	
TYPE	97	TYPE	97	тип пакета (1 байт)
ADDRLL	***	ADDRLL	***	адрес подчиненного узла (мл.байт)
ADDRL	***	ADDRL	***	адрес подчиненного узла, байт #2
ADDRH	***	ADDRH	***	адрес подчиненного узла, байт #3
ADDRHH	***	ADDRHH	***	адрес подчиненного узла (ст.байт)

Ответ концентратора в случае неверно указанного адреса подчинённого узла выглядит следующим образом.

PAYLOAD				ПРИМЕЧАНИЯ
Запрос GET_MOD_CONFIG		Ответ GET_MOD_CONFIG		
поле	значение	поле	значение	
TYPE	97	TYPE	97	тип пакета (1 байт)
ADDRLL	***	X	X	адрес подчиненного узла (мл.байт)
ADDRL	***	X	X	адрес подчиненного узла, байт #2
ADDRH	***	X	X	адрес подчиненного узла, байт #3
ADDRHH	***	X	X	адрес подчиненного узла (ст.байт)

8.2 Практическое применение

В данном разделе дано пошаговое описание типового сценария развёртывания, настройки и использования транспортной системы Меркурий PLC-II.

1. Установить счётчики с модемами PLC-II на объекте автоматизации.
2. Комплект из 3-х концентраторов "Меркурий-225.2X" в подходящем для этого месте (в большинстве случаев это питающая подстанция), подключить к трёхфазному фидеру (по одному на каждую фазу) и соединить выходы их интерфейсов RS-485 между собой прилагаемыми кабелями.
3. Перевести все концентраторы в режим IDLE, см. MASTER: Запись/чтение конфигурации.
4. Установить необходимое количество сегментов почтовых ящиков для всех концентраторов, см. MASTER: Запись/чтение количества сегментов.
5. Перевести все концентраторы в режим MASTER, см. MASTER: Запись/чтение конфигурации.
6. В режиме MASTER каждый из концентраторов выполняет следующие действия:

– непрерывно сканирует сеть (см. SCAN) в поисках ещё не обнаруженных счётчиков
– самостоятельно регистрирует их, см. MASTER: Регистрация нового подчинённого узла.
– в первые три сегмента почтового ящика только что зарегистрированного счётчика однократно загружает (см. MBOX+SEG: Запись/чтение отделения запросов указанного сегмента и MBOX+SEG: Запись/чтение регистров статуса указанного сегмента) следующий типовой комплект запросов:

- SEGMENT.0 − установка часов счётчика (PERIOD=MONTH)
- SEGMENT.1 − чтение показаний часов счётчика (PERIOD=WEEK)
- SEGMENT.2 − чтение данных о накопленной энергии по тарифам и сумме тарифов (PERIOD=DAY)

При этом конкретное содержание запроса зависит от типа зарегистрированного счётчика.

7. Если этой функциональности достаточно, то далее, при штатной работе системы, остаётся только периодически выгружать полученные ответы счётчиков из сегментов 1 и 2 (см. MBOX+SEG: Чтение отделения ответов указанного сегмента) с целью их последующего использования. Для дистанционного считывания данных можно использовать GSM-шлюз "Меркурий-228" и свободно распространяемую программу BQuark.
8. При необходимости, запросы в сегментах могут быть далее перезагружены любыми другими командами, потребность в исполнении которых возникнет в ходе работы системы.
9. При наличии на подстанции ещё одного трансформатора и, соответственного, второго питающего фидера, к нему следует подключить либо блок конденсаторных вставок для организации сигнальной связи между одноимёнными фазами обоих фидеров, либо, если этого окажется недостаточно, дополнительно второй комплект концентраторов в режиме SLAVE (см. MASTER: Запись/чтение конфигурации). При этом какие-либо изменения в режим работы MASTER-концентраторов вносить не требуется.

Приложение А. Процедура вычисления 32-разрядного циклического кода (IEEE 802.3)

```
    #define CRC32_POLY  0xEDB88320

    unsigned long crc32_bit_by_bit (unsigned char *octets, int len)
    {
    int i, j, cbit;
    unsigned long crc = CRC32_INIT;

    for (i = 0; i < len; i++)
    {
       crc ^= octets[i];

       for (j = 0; j < 8; j++)
       {
         cbit = crc & 1;                    // a carry bit
         crc >>= 1;                          // arithmetical shift to
the right

         if (cbit)
         {
           crc ^= CRC32_POLY;
         }
       }    // end of for(j)
    }    // end of for(i)

    return (crc ^ 0xFFFFFFFF);
    }    // end of crc32_bit_by_bit ()
```

Приложение Б. Процедура вычисления 24-разрядного циклического кода (RFC-2440)

```
#define CRC24_INIT  0x0B704CE
#define CRC24_POLY  0x1864CFB

unsigned long crc_octets (unsigned char *octets, int len)
{
int i;
unsigned long crc = CRC24_INIT;

while (len--)  {
   crc ^= (*octets++) << 16;
   for (i = 0; i < 8; i++)  {
     crc <<= 1;
     if (crc & 0x1000000)  crc ^= CRC24_POLY;
   }
}

return (crc & 0x00FFFFFF);
   }    // end of crc_octets()
```

Приложение В. Код Хемминга [12,8]: кодирование и декодирование

```
//============================================================
==============
// Помехоустойчивый код с минимальным кодовым расстоянием 3.
//
// ООО "Фирма ИНКОТЕКС",
// Последнее обновление:  2007-07-12
//
// Откомпилируйте проект и запустите его на выполнение из
командной строки:
// >hamming124.exe
//
// Вывод исполняемого файла в стандартный поток должен выглядеть
следующим
// образом:
//
// >Алгоритмический кодер
// >
// >должно быть: C01   получено: C01
// >должно быть: 503   получено: 503
// >должно быть: 607   получено: 607
// >должно быть: 40E   получено: 40E
// >должно быть: 549   получено: 549
// >должно быть: 0D3   получено: 0D3
// >
// >Алгоритмический декодер
// >
// >Отсутствие отшибок
// >было закодировано: 01   должно быть: 001   получено: 001
// >было закодировано: 03   должно быть: 003   получено: 003
// >
// >Одна ошибка
// >было закодировано: 07   должно быть: 007   получено: 007
// >было закодировано: 0E   должно быть: 00E   получено: 00E
// >
// >Две ошибки
// >было закодировано: 49   должно быть: 100   получено: 100
// >было закодировано: D3   должно быть: 100   получено: 100
// >
// >Табличный декодер
// >
// >Отсутствие отшибок
// >было закодировано: 01   должно быть: 001   получено: 001
// >было закодировано: 03   должно быть: 003   получено: 003
// >
// >Одна ошибка
// >было закодировано: 07   должно быть: 007   получено: 007
// >было закодировано: 0E   должно быть: 00E   получено: 00E
// >
// >Две ошибки
// >было закодировано: 49   должно быть: 100   получено: 100
```

```
// >было закодировано: D3   должно быть: 100   получено: 100
// >
//=============================================================
==============

#include <stdio.h>

//-------------------------------------------------------------
--------------
// Таблица быстрого табличного кодирования.
// Требуемый размер ПЗУ - 256 16-разрядных (или 12-разрядных)
слов.
//-------------------------------------------------------------
--------------
const int hamming_codes[256] = {
0x000, 0xC01, 0x902, 0x503, 0x304, 0xF05, 0xA06, 0x607,
0xE08, 0x209, 0x70A, 0xB0B, 0xD0C, 0x10D, 0x40E, 0x80F,
0xD10, 0x111, 0x412, 0x813, 0xE14, 0x215, 0x716, 0xB17,
0x318, 0xF19, 0xA1A, 0x61B, 0x01C, 0xC1D, 0x91E, 0x51F,
0xB20, 0x721, 0x222, 0xE23, 0x824, 0x425, 0x126, 0xD27,
0x528, 0x929, 0xC2A, 0x02B, 0x62C, 0xA2D, 0xF2E, 0x32F,
0x630, 0xA31, 0xF32, 0x333, 0x534, 0x935, 0xC36, 0x037,
0x838, 0x439, 0x13A, 0xD3B, 0xB3C, 0x73D, 0x23E, 0xE3F,
0x740, 0xB41, 0xE42, 0x243, 0x444, 0x845, 0xD46, 0x147,
0x948, 0x549, 0x04A, 0xC4B, 0xA4C, 0x64D, 0x34E, 0xF4F,
0xA50, 0x651, 0x352, 0xF53, 0x954, 0x555, 0x056, 0xC57,
0x458, 0x859, 0xD5A, 0x15B, 0x75C, 0xB5D, 0xE5E, 0x25F,
0xC60, 0x061, 0x562, 0x963, 0xF64, 0x365, 0x666, 0xA67,
0x268, 0xE69, 0xB6A, 0x76B, 0x16C, 0xD6D, 0x86E, 0x46F,
0x170, 0xD71, 0x872, 0x473, 0x274, 0xE75, 0xB76, 0x777,
0xF78, 0x379, 0x67A, 0xA7B, 0xC7C, 0x07D, 0x57E, 0x97F,
0xF80, 0x381, 0x682, 0xA83, 0xC84, 0x085, 0x586, 0x987,
0x188, 0xD89, 0x88A, 0x48B, 0x28C, 0xE8D, 0xB8E, 0x78F,
0x290, 0xE91, 0xB92, 0x793, 0x194, 0xD95, 0x896, 0x497,
0xC98, 0x099, 0x59A, 0x99B, 0xF9C, 0x39D, 0x69E, 0xA9F,
0x4A0, 0x8A1, 0xDA2, 0x1A3, 0x7A4, 0xBA5, 0xEA6, 0x2A7,
0xAA8, 0x6A9, 0x3AA, 0xFAB, 0x9AC, 0x5AD, 0x0AE, 0xCAF,
0x9B0, 0x5B1, 0x0B2, 0xCB3, 0xAB4, 0x6B5, 0x3B6, 0xFB7,
0x7B8, 0xBB9, 0xEBA, 0x2BB, 0x4BC, 0x8BD, 0xDBE, 0x1BF,
0x8C0, 0x4C1, 0x1C2, 0xDC3, 0xBC4, 0x7C5, 0x2C6, 0xEC7,
0x6C8, 0xAC9, 0xFCA, 0x3CB, 0x5CC, 0x9CD, 0xCCE, 0x0CF,
0x5D0, 0x9D1, 0xCD2, 0x0D3, 0x6D4, 0xAD5, 0xFD6, 0x3D7,
0xBD8, 0x7D9, 0x2DA, 0xEDB, 0x8DC, 0x4DD, 0x1DE, 0xDDF,
0x3E0, 0xFE1, 0xAE2, 0x6E3, 0x0E4, 0xCE5, 0x9E6, 0x5E7,
0xDE8, 0x1E9, 0x4EA, 0x8EB, 0xEEC, 0x2ED, 0x7EE, 0xBEF,
0xEF0, 0x2F1, 0x7F2, 0xBF3, 0xDF4, 0x1F5, 0x4F6, 0x8F7,
0x0F8, 0xCF9, 0x9FA, 0x5FB, 0x3FC, 0xFFD, 0xAFE, 0x6FF
};    // end of hamming_codes[]

//-------------------------------------------------------------
--------------
// Формирующий массив малоразмерного кодера.
//-------------------------------------------------------------
--------------
```

```
const int generator[8] = { 0xC, 0x9, 0x3, 0xE, 0xD, 0xB, 0x7, 0xF
};

//---------------------------------------------------------------
---------------
// Исправляющие коды по синдрому.
//---------------------------------------------------------------
---------------
const int refiner[16] = {
0x000, 0x000, 0x000, 0x004, 0x000, 0x100, 0x100, 0x040,
0x000, 0x002, 0x100, 0x020, 0x001, 0x010, 0x008, 0x080
};

//---------------------------------------------------------------
---------------
// Подпрограммы и таблицы, следующие за функцией main().
//---------------------------------------------------------------
---------------
int hamming_coder (int byte);
int hamming_decoder (int code);
const int decoder_table[4096];

//---------------------------------------------------------------
---------------
// Тестирование кодера и декодера.
//---------------------------------------------------------------
---------------
void main (void)
{
int byte, code;

// проверка алгоритмического кодера
printf ("Алгоритмический кодер\n\n");
printf ("должно быть: %03X   получено: %03X\n",
         hamming_codes[  1], hamming_coder (  1));
printf ("должно быть: %03X   получено: %03X\n",
         hamming_codes[  3], hamming_coder (  3));
printf ("должно быть: %03X   получено: %03X\n",
         hamming_codes[  7], hamming_coder (  7));
printf ("должно быть: %03X   получено: %03X\n",
         hamming_codes[ 14], hamming_coder ( 14));
printf ("должно быть: %03X   получено: %03X\n",
         hamming_codes[ 73], hamming_coder ( 73));
printf ("должно быть: %03X   получено: %03X\n",
         hamming_codes[211], hamming_coder (211));

// проверка алгоритмического декодера
printf ("\nАлгоритмический декодер\n");

// Отсутствие отшибок
printf ("\nОтсутствие отшибок\n");
printf ("было закодировано: %02X   должно быть: %03X   получено:
%03X\n",
```

```
        0x01, 0x01, hamming_decoder (0xC01));
printf ("было закодировано: %02X   должно быть: %03X   получено:
%03X\n",
        0x03, 0x03, hamming_decoder (0x503));

// Одна ошибка
printf ("\nОдна ошибка\n");
printf ("было закодировано: %02X   должно быть: %03X   получено:
%03X\n",
        0x07, 0x07, hamming_decoder (0x607 ^ 0x001));
printf ("было закодировано: %02X   должно быть: %03X   получено:
%03X\n",
        0x0E, 0x0E, hamming_decoder (0x40E ^ 0x100));

// Две ошибки
printf ("\nДве ошибки\n");
printf ("было закодировано: %02X   должно быть: %03X   получено:
%03X\n",
        0x49, 0x100, hamming_decoder (0x549 ^ 0x050));
printf ("было закодировано: %02X   должно быть: %03X   получено:
%03X\n",
        0xD3, 0x100, hamming_decoder (0x0D3 ^ 0x500));

// Проверка табличного декодера
printf ("\nТабличный декодер\n");

// Отсутствие отшибок
printf ("\nОтсутствие отшибок\n");
printf ("было закодировано: %02X   должно быть: %03X   получено:
%03X\n",
        0x01, 0x01, decoder_table[0xC01]);
printf ("было закодировано: %02X   должно быть: %03X   получено:
%03X\n",
        0x03, 0x03, decoder_table[0x503]);

// Одна ошибка
printf ("\nОдна ошибка\n");
printf ("было закодировано: %02X   должно быть: %03X   получено:
%03X\n",
        0x07, 0x07, decoder_table[0x607 ^ 0x001]);
printf ("было закодировано: %02X   должно быть: %03X   получено:
%03X\n",
        0x0E, 0x0E, decoder_table[0x40E ^ 0x100]);

// Две ошибки
printf ("\nДве ошибки\n");
printf ("было закодировано: %02X   должно быть: %03X   получено:
%03X\n",
        0x49, 0x100, decoder_table[0x549 ^ 0x050]);
printf ("было закодировано: %02X   должно быть: %03X   получено:
%03X\n",
        0xD3, 0x100, decoder_table[0x0D3 ^ 0x500]);
}    // end of main()
```

```
//---------------------------------------------------------------
---------------
// Подпрограмма кодирования.
//---------------------------------------------------------------
---------------
int hamming_coder (int byte)
{
int i, sft, vrf;

sft = byte;
vrf = 0;

for (i = 0; i < 8; i++)
{
   if (sft & 1)  vrf ^= generator[i];
   sft >>= 1;
}    // end of for(j)

return (vrf << 8) + (byte & 0x0FF);
}    // end of hamming_coder()

//---------------------------------------------------------------
---------------
// Подпрограмма синдромного декодирования.
//---------------------------------------------------------------
---------------
int hamming_decoder (int code)
{
int syndrome, mask;

syndrome = (code ^ hamming_coder (code)) >> 8;
mask = refiner[syndrome];

if (mask & 0x100)  return mask;
else  return (code ^ mask) & 0x0FF;
}    // end of hamming_decoder()

//---------------------------------------------------------------
---------------
// Таблица быстрого табличного декодирования.
// Требуемый размер ПЗУ - 4096 16-разрядных (или 9-разрядных)
слов.
//---------------------------------------------------------------
---------------
const int decoder_table[4096] = {
0x000, 0x000, 0x000, 0x100, 0x000, 0x085, 0x100, 0x100,
0x000, 0x009, 0x04A, 0x02B, 0x01C, 0x00D, 0x00E, 0x00F,
0x000, 0x011, 0x012, 0x013, 0x01C, 0x015, 0x056, 0x037,
0x01C, 0x099, 0x100, 0x100, 0x01C, 0x01C, 0x01C, 0x100,
0x000, 0x061, 0x022, 0x02B, 0x024, 0x025, 0x026, 0x037,
0x100, 0x02B, 0x02B, 0x02B, 0x100, 0x100, 0x0AE, 0x02B,
0x100, 0x100, 0x0B2, 0x037, 0x100, 0x037, 0x037, 0x037,
0x038, 0x039, 0x03A, 0x02B, 0x01C, 0x07D, 0x03E, 0x037,
```

```
0x000, 0x061, 0x04A, 0x043, 0x044, 0x045, 0x056, 0x047,
0x04A, 0x100, 0x04A, 0x04A, 0x100, 0x100, 0x04A, 0x0CF,
0x100, 0x100, 0x056, 0x0D3, 0x056, 0x100, 0x056, 0x056,
0x058, 0x059, 0x04A, 0x05B, 0x01C, 0x07D, 0x056, 0x05F,
0x061, 0x061, 0x100, 0x061, 0x0E4, 0x061, 0x100, 0x100,
0x068, 0x061, 0x04A, 0x02B, 0x06C, 0x07D, 0x06E, 0x06F,
0x070, 0x061, 0x072, 0x073, 0x074, 0x07D, 0x056, 0x037,
0x0F8, 0x07D, 0x100, 0x100, 0x07D, 0x07D, 0x100, 0x07D,
0x000, 0x085, 0x100, 0x100, 0x085, 0x085, 0x100, 0x085,
0x088, 0x099, 0x08A, 0x08B, 0x08C, 0x085, 0x0AE, 0x0CF,
0x090, 0x099, 0x0B2, 0x0D3, 0x094, 0x085, 0x096, 0x097,
0x099, 0x099, 0x100, 0x099, 0x01C, 0x099, 0x100, 0x100,
0x0A0, 0x0A1, 0x0B2, 0x0A3, 0x0E4, 0x085, 0x0AE, 0x0A7,
0x100, 0x100, 0x0AE, 0x02B, 0x0AE, 0x100, 0x0AE, 0x0AE,
0x0B2, 0x100, 0x0B2, 0x0B2, 0x100, 0x100, 0x0B2, 0x037,
0x0F8, 0x099, 0x0B2, 0x0BB, 0x0BC, 0x0BD, 0x0AE, 0x0BF,
0x0C0, 0x0C1, 0x0C2, 0x0D3, 0x0E4, 0x085, 0x0C6, 0x0CF,
0x100, 0x100, 0x04A, 0x0CF, 0x100, 0x0CF, 0x0CF, 0x0CF,
0x100, 0x0D3, 0x0D3, 0x0D3, 0x100, 0x100, 0x056, 0x0D3,
0x0F8, 0x099, 0x0DA, 0x0D3, 0x0DC, 0x0DD, 0x0DE, 0x0CF,
0x0E4, 0x061, 0x100, 0x100, 0x0E4, 0x0E4, 0x0E4, 0x100,
0x0F8, 0x0E9, 0x0EA, 0x0EB, 0x0E4, 0x0ED, 0x0AE, 0x0CF,
0x0F8, 0x0F1, 0x0B2, 0x0D3, 0x0E4, 0x0F5, 0x0F6, 0x0F7,
0x0F8, 0x0F8, 0x0F8, 0x100, 0x0F8, 0x07D, 0x100, 0x100,
0x000, 0x011, 0x002, 0x003, 0x004, 0x00D, 0x026, 0x047,
0x088, 0x00D, 0x100, 0x100, 0x00D, 0x00D, 0x100, 0x00D,
0x011, 0x011, 0x100, 0x011, 0x094, 0x011, 0x100, 0x100,
0x018, 0x011, 0x03A, 0x05B, 0x01C, 0x00D, 0x01E, 0x01F,
0x100, 0x100, 0x026, 0x0A3, 0x026, 0x100, 0x026, 0x026,
0x028, 0x029, 0x03A, 0x02B, 0x06C, 0x00D, 0x026, 0x02F,
0x070, 0x011, 0x03A, 0x033, 0x034, 0x035, 0x026, 0x037,
0x03A, 0x100, 0x03A, 0x03A, 0x100, 0x100, 0x03A, 0x0BF,
0x100, 0x100, 0x0C2, 0x047, 0x100, 0x047, 0x047, 0x047,
0x048, 0x049, 0x04A, 0x05B, 0x06C, 0x00D, 0x04E, 0x047,
0x070, 0x011, 0x052, 0x05B, 0x054, 0x055, 0x056, 0x047,
0x100, 0x05B, 0x05B, 0x05B, 0x100, 0x100, 0x0DE, 0x05B,
0x070, 0x061, 0x062, 0x063, 0x06C, 0x065, 0x026, 0x047,
0x06C, 0x0E9, 0x100, 0x100, 0x06C, 0x06C, 0x06C, 0x100,
0x070, 0x070, 0x070, 0x100, 0x070, 0x0F5, 0x100, 0x100,
0x070, 0x079, 0x03A, 0x05B, 0x06C, 0x07D, 0x07E, 0x07F,
0x088, 0x081, 0x0C2, 0x0A3, 0x094, 0x085, 0x086, 0x087,
0x088, 0x088, 0x088, 0x100, 0x088, 0x00D, 0x100, 0x100,
0x094, 0x011, 0x100, 0x100, 0x094, 0x094, 0x094, 0x100,
0x088, 0x099, 0x09A, 0x09B, 0x094, 0x09D, 0x0DE, 0x0BF,
0x100, 0x0A3, 0x0A3, 0x0A3, 0x100, 0x100, 0x026, 0x0A3,
0x088, 0x0E9, 0x0AA, 0x0A3, 0x0AC, 0x0AD, 0x0AE, 0x0BF,
0x0B0, 0x0B1, 0x0B2, 0x0A3, 0x094, 0x0F5, 0x0B6, 0x0BF,
0x100, 0x100, 0x03A, 0x0BF, 0x100, 0x0BF, 0x0BF, 0x0BF,
0x0C2, 0x100, 0x0C2, 0x0C2, 0x100, 0x100, 0x0C2, 0x047,
0x088, 0x0E9, 0x0C2, 0x0CB, 0x0CC, 0x0CD, 0x0DE, 0x0CF,
0x0D0, 0x0D1, 0x0C2, 0x0D3, 0x094, 0x0F5, 0x0DE, 0x0D7,
0x100, 0x100, 0x0DE, 0x05B, 0x0DE, 0x100, 0x0DE, 0x0DE,
0x0E0, 0x0E9, 0x0C2, 0x0A3, 0x0E4, 0x0F5, 0x0E6, 0x0E7,
0x0E9, 0x0E9, 0x100, 0x0E9, 0x06C, 0x0E9, 0x100, 0x100,
```

```
0x070, 0x0F5, 0x100, 0x100, 0x0F5, 0x0F5, 0x100, 0x0F5,
0x0F8, 0x0E9, 0x0FA, 0x0FB, 0x0FC, 0x0F5, 0x0DE, 0x0BF,
0x000, 0x009, 0x022, 0x043, 0x004, 0x015, 0x006, 0x007,
0x009, 0x009, 0x100, 0x009, 0x08C, 0x009, 0x100, 0x100,
0x090, 0x015, 0x100, 0x100, 0x015, 0x015, 0x100, 0x015,
0x018, 0x009, 0x01A, 0x01B, 0x01C, 0x015, 0x03E, 0x05F,
0x022, 0x100, 0x022, 0x022, 0x100, 0x100, 0x022, 0x0A7,
0x068, 0x009, 0x022, 0x02B, 0x02C, 0x02D, 0x03E, 0x02F,
0x030, 0x031, 0x022, 0x033, 0x074, 0x015, 0x03E, 0x037,
0x100, 0x100, 0x03E, 0x0BB, 0x03E, 0x100, 0x03E, 0x03E,
0x100, 0x043, 0x043, 0x043, 0x100, 0x100, 0x0C6, 0x043,
0x068, 0x009, 0x04A, 0x043, 0x04C, 0x04D, 0x04E, 0x05F,
0x050, 0x051, 0x052, 0x043, 0x074, 0x015, 0x056, 0x05F,
0x100, 0x100, 0x0DA, 0x05F, 0x100, 0x05F, 0x05F, 0x05F,
0x068, 0x061, 0x022, 0x043, 0x074, 0x065, 0x066, 0x067,
0x068, 0x068, 0x068, 0x100, 0x068, 0x0ED, 0x100, 0x100,
0x074, 0x0F1, 0x100, 0x100, 0x074, 0x074, 0x074, 0x100,
0x068, 0x079, 0x07A, 0x07B, 0x074, 0x07D, 0x03E, 0x05F,
0x090, 0x081, 0x082, 0x083, 0x08C, 0x085, 0x0C6, 0x0A7,
0x08C, 0x009, 0x100, 0x100, 0x08C, 0x08C, 0x08C, 0x100,
0x090, 0x090, 0x090, 0x100, 0x090, 0x015, 0x100, 0x100,
0x090, 0x099, 0x0DA, 0x0BB, 0x08C, 0x09D, 0x09E, 0x09F,
0x100, 0x100, 0x022, 0x0A7, 0x100, 0x0A7, 0x0A7, 0x0A7,
0x0A8, 0x0A9, 0x0AA, 0x0BB, 0x08C, 0x0ED, 0x0AE, 0x0A7,
0x090, 0x0F1, 0x0B2, 0x0BB, 0x0B4, 0x0B5, 0x0B6, 0x0A7,
0x100, 0x0BB, 0x0BB, 0x0BB, 0x100, 0x100, 0x03E, 0x0BB,
0x100, 0x100, 0x0C6, 0x043, 0x0C6, 0x100, 0x0C6, 0x0C6,
0x0C8, 0x0C9, 0x0DA, 0x0CB, 0x08C, 0x0ED, 0x0C6, 0x0CF,
0x090, 0x0F1, 0x0DA, 0x0D3, 0x0D4, 0x0D5, 0x0C6, 0x0D7,
0x0DA, 0x100, 0x0DA, 0x0DA, 0x100, 0x100, 0x0DA, 0x05F,
0x0E0, 0x0F1, 0x0E2, 0x0E3, 0x0E4, 0x0ED, 0x0C6, 0x0A7,
0x068, 0x0ED, 0x100, 0x100, 0x0ED, 0x0ED, 0x100, 0x0ED,
0x0F1, 0x0F1, 0x100, 0x0F1, 0x074, 0x0F1, 0x100, 0x100,
0x0F8, 0x0F1, 0x0DA, 0x0BB, 0x0FC, 0x0ED, 0x0FE, 0x0FF,
0x004, 0x081, 0x100, 0x100, 0x004, 0x004, 0x004, 0x100,
0x018, 0x009, 0x00A, 0x00B, 0x004, 0x00D, 0x04E, 0x02F,
0x018, 0x011, 0x052, 0x033, 0x004, 0x015, 0x016, 0x017,
0x018, 0x018, 0x018, 0x100, 0x018, 0x09D, 0x100, 0x100,
0x020, 0x021, 0x022, 0x033, 0x004, 0x065, 0x026, 0x02F,
0x100, 0x100, 0x0AA, 0x02F, 0x100, 0x02F, 0x02F, 0x02F,
0x100, 0x033, 0x033, 0x033, 0x100, 0x100, 0x0B6, 0x033,
0x018, 0x079, 0x03A, 0x033, 0x03C, 0x03D, 0x03E, 0x02F,
0x040, 0x041, 0x052, 0x043, 0x004, 0x065, 0x04E, 0x047,
0x100, 0x100, 0x04E, 0x0CB, 0x04E, 0x100, 0x04E, 0x04E,
0x052, 0x100, 0x052, 0x052, 0x100, 0x100, 0x052, 0x0D7,
0x018, 0x079, 0x052, 0x05B, 0x05C, 0x05D, 0x04E, 0x05F,
0x0E0, 0x065, 0x100, 0x100, 0x065, 0x065, 0x100, 0x065,
0x068, 0x079, 0x06A, 0x06B, 0x06C, 0x065, 0x04E, 0x02F,
0x070, 0x079, 0x052, 0x033, 0x074, 0x065, 0x076, 0x077,
0x079, 0x079, 0x100, 0x079, 0x0FC, 0x079, 0x100, 0x100,
0x081, 0x081, 0x100, 0x081, 0x004, 0x081, 0x100, 0x100,
0x088, 0x081, 0x0AA, 0x0CB, 0x08C, 0x09D, 0x08E, 0x08F,
0x090, 0x081, 0x092, 0x093, 0x094, 0x09D, 0x0B6, 0x0D7,
```

```
0x018, 0x09D, 0x100, 0x100, 0x09D, 0x09D, 0x100, 0x09D,
0x0E0, 0x081, 0x0AA, 0x0A3, 0x0A4, 0x0A5, 0x0B6, 0x0A7,
0x0AA, 0x100, 0x0AA, 0x0AA, 0x100, 0x100, 0x0AA, 0x02F,
0x100, 0x100, 0x0B6, 0x033, 0x0B6, 0x100, 0x0B6, 0x0B6,
0x0B8, 0x0B9, 0x0AA, 0x0BB, 0x0FC, 0x09D, 0x0B6, 0x0BF,
0x0E0, 0x081, 0x0C2, 0x0CB, 0x0C4, 0x0C5, 0x0C6, 0x0D7,
0x100, 0x0CB, 0x0CB, 0x0CB, 0x100, 0x100, 0x04E, 0x0CB,
0x100, 0x100, 0x052, 0x0D7, 0x100, 0x0D7, 0x0D7, 0x0D7,
0x0D8, 0x0D9, 0x0DA, 0x0CB, 0x0FC, 0x09D, 0x0DE, 0x0D7,
0x0E0, 0x0E0, 0x0E0, 0x100, 0x0E0, 0x065, 0x100, 0x100,
0x0E0, 0x0E9, 0x0AA, 0x0CB, 0x0FC, 0x0ED, 0x0EE, 0x0EF,
0x0E0, 0x0F1, 0x0F2, 0x0F3, 0x0FC, 0x0F5, 0x0B6, 0x0D7,
0x0FC, 0x079, 0x100, 0x100, 0x0FC, 0x0FC, 0x0FC, 0x100,
0x000, 0x001, 0x012, 0x003, 0x044, 0x025, 0x00E, 0x007,
0x100, 0x100, 0x00E, 0x08B, 0x00E, 0x100, 0x00E, 0x00E,
0x012, 0x100, 0x012, 0x012, 0x100, 0x100, 0x012, 0x097,
0x058, 0x039, 0x012, 0x01B, 0x01C, 0x01D, 0x00E, 0x01F,
0x0A0, 0x025, 0x100, 0x100, 0x025, 0x025, 0x100, 0x025,
0x028, 0x039, 0x02A, 0x02B, 0x02C, 0x025, 0x00E, 0x06F,
0x030, 0x039, 0x012, 0x073, 0x034, 0x025, 0x036, 0x037,
0x039, 0x039, 0x100, 0x039, 0x0BC, 0x039, 0x100, 0x100,
0x044, 0x0C1, 0x100, 0x100, 0x044, 0x044, 0x044, 0x100,
0x058, 0x049, 0x04A, 0x04B, 0x044, 0x04D, 0x00E, 0x06F,
0x058, 0x051, 0x012, 0x073, 0x044, 0x055, 0x056, 0x057,
0x058, 0x058, 0x058, 0x100, 0x058, 0x0DD, 0x100, 0x100,
0x060, 0x061, 0x062, 0x073, 0x044, 0x025, 0x066, 0x06F,
0x100, 0x100, 0x0EA, 0x06F, 0x100, 0x06F, 0x06F, 0x06F,
0x100, 0x073, 0x073, 0x073, 0x100, 0x100, 0x0F6, 0x073,
0x058, 0x039, 0x07A, 0x073, 0x07C, 0x07D, 0x07E, 0x06F,
0x0A0, 0x0C1, 0x082, 0x08B, 0x084, 0x085, 0x086, 0x097,
0x100, 0x08B, 0x08B, 0x08B, 0x100, 0x100, 0x00E, 0x08B,
0x100, 0x100, 0x012, 0x097, 0x100, 0x097, 0x097, 0x097,
0x098, 0x099, 0x09A, 0x08B, 0x0BC, 0x0DD, 0x09E, 0x097,
0x0A0, 0x0A0, 0x0A0, 0x100, 0x0A0, 0x025, 0x100, 0x100,
0x0A0, 0x0A9, 0x0EA, 0x08B, 0x0BC, 0x0AD, 0x0AE, 0x0AF,
0x0A0, 0x0B1, 0x0B2, 0x0B3, 0x0BC, 0x0B5, 0x0F6, 0x097,
0x0BC, 0x039, 0x100, 0x100, 0x0BC, 0x0BC, 0x0BC, 0x100,
0x0C1, 0x0C1, 0x100, 0x0C1, 0x044, 0x0C1, 0x100, 0x100,
0x0C8, 0x0C1, 0x0EA, 0x08B, 0x0CC, 0x0DD, 0x0CE, 0x0CF,
0x0D0, 0x0C1, 0x0D2, 0x0D3, 0x0D4, 0x0DD, 0x0F6, 0x097,
0x058, 0x0DD, 0x100, 0x100, 0x0DD, 0x0DD, 0x100, 0x0DD,
0x0A0, 0x0C1, 0x0EA, 0x0E3, 0x0E4, 0x0E5, 0x0F6, 0x0E7,
0x0EA, 0x100, 0x0EA, 0x0EA, 0x100, 0x100, 0x0EA, 0x06F,
0x100, 0x100, 0x0F6, 0x073, 0x0F6, 0x100, 0x0F6, 0x0F6,
0x0F8, 0x0F9, 0x0EA, 0x0FB, 0x0BC, 0x0DD, 0x0F6, 0x0FF,
0x100, 0x003, 0x003, 0x003, 0x100, 0x100, 0x086, 0x003,
0x028, 0x049, 0x00A, 0x003, 0x00C, 0x00D, 0x00E, 0x01F,
0x010, 0x011, 0x012, 0x003, 0x034, 0x055, 0x016, 0x01F,
0x100, 0x100, 0x09A, 0x01F, 0x100, 0x01F, 0x01F, 0x01F,
0x028, 0x021, 0x062, 0x003, 0x034, 0x025, 0x026, 0x027,
0x028, 0x028, 0x028, 0x100, 0x028, 0x0AD, 0x100, 0x100,
0x034, 0x0B1, 0x100, 0x100, 0x034, 0x034, 0x034, 0x100,
0x028, 0x039, 0x03A, 0x03B, 0x034, 0x03D, 0x07E, 0x01F,
0x040, 0x049, 0x062, 0x003, 0x044, 0x055, 0x046, 0x047,
```

```
0x049, 0x049, 0x100, 0x049, 0x0CC, 0x049, 0x100, 0x100,
0x0D0, 0x055, 0x100, 0x100, 0x055, 0x055, 0x100, 0x055,
0x058, 0x049, 0x05A, 0x05B, 0x05C, 0x055, 0x07E, 0x01F,
0x062, 0x100, 0x062, 0x062, 0x100, 0x100, 0x062, 0x0E7,
0x028, 0x049, 0x062, 0x06B, 0x06C, 0x06D, 0x07E, 0x06F,
0x070, 0x071, 0x062, 0x073, 0x034, 0x055, 0x07E, 0x077,
0x100, 0x100, 0x07E, 0x0FB, 0x07E, 0x100, 0x07E, 0x07E,
0x100, 0x100, 0x086, 0x003, 0x086, 0x100, 0x086, 0x086,
0x088, 0x089, 0x09A, 0x08B, 0x0CC, 0x0AD, 0x086, 0x08F,
0x0D0, 0x0B1, 0x09A, 0x093, 0x094, 0x095, 0x086, 0x097,
0x09A, 0x100, 0x09A, 0x09A, 0x100, 0x100, 0x09A, 0x01F,
0x0A0, 0x0B1, 0x0A2, 0x0A3, 0x0A4, 0x0AD, 0x086, 0x0E7,
0x028, 0x0AD, 0x100, 0x100, 0x0AD, 0x0AD, 0x100, 0x0AD,
0x0B1, 0x0B1, 0x100, 0x0B1, 0x034, 0x0B1, 0x100, 0x100,
0x0B8, 0x0B1, 0x09A, 0x0FB, 0x0BC, 0x0AD, 0x0BE, 0x0BF,
0x0D0, 0x0C1, 0x0C2, 0x0C3, 0x0CC, 0x0C5, 0x086, 0x0E7,
0x0CC, 0x049, 0x100, 0x100, 0x0CC, 0x0CC, 0x0CC, 0x100,
0x0D0, 0x0D0, 0x0D0, 0x100, 0x0D0, 0x055, 0x100, 0x100,
0x0D0, 0x0D9, 0x09A, 0x0FB, 0x0CC, 0x0DD, 0x0DE, 0x0DF,
0x100, 0x100, 0x062, 0x0E7, 0x100, 0x0E7, 0x0E7, 0x0E7,
0x0E8, 0x0E9, 0x0EA, 0x0FB, 0x0CC, 0x0AD, 0x0EE, 0x0E7,
0x0D0, 0x0B1, 0x0F2, 0x0FB, 0x0F4, 0x0F5, 0x0F6, 0x0E7,
0x100, 0x0FB, 0x0FB, 0x0FB, 0x100, 0x100, 0x07E, 0x0FB,
0x100, 0x100, 0x082, 0x007, 0x100, 0x007, 0x007, 0x007,
0x008, 0x009, 0x00A, 0x01B, 0x02C, 0x04D, 0x00E, 0x007,
0x030, 0x051, 0x012, 0x01B, 0x014, 0x015, 0x016, 0x007,
0x100, 0x01B, 0x01B, 0x01B, 0x100, 0x100, 0x09E, 0x01B,
0x030, 0x021, 0x022, 0x023, 0x02C, 0x025, 0x066, 0x007,
0x02C, 0x0A9, 0x100, 0x100, 0x02C, 0x02C, 0x02C, 0x100,
0x030, 0x030, 0x030, 0x100, 0x030, 0x0B5, 0x100, 0x100,
0x030, 0x039, 0x07A, 0x01B, 0x02C, 0x03D, 0x03E, 0x03F,
0x040, 0x051, 0x042, 0x043, 0x044, 0x04D, 0x066, 0x007,
0x0C8, 0x04D, 0x100, 0x100, 0x04D, 0x04D, 0x100, 0x04D,
0x051, 0x051, 0x100, 0x051, 0x0D4, 0x051, 0x100, 0x100,
0x058, 0x051, 0x07A, 0x01B, 0x05C, 0x04D, 0x05E, 0x05F,
0x100, 0x100, 0x066, 0x0E3, 0x066, 0x100, 0x066, 0x066,
0x068, 0x069, 0x07A, 0x06B, 0x02C, 0x04D, 0x066, 0x06F,
0x030, 0x051, 0x07A, 0x073, 0x074, 0x075, 0x066, 0x077,
0x07A, 0x100, 0x07A, 0x07A, 0x100, 0x100, 0x07A, 0x0FF,
0x082, 0x100, 0x082, 0x082, 0x100, 0x100, 0x082, 0x007,
0x0C8, 0x0A9, 0x082, 0x08B, 0x08C, 0x08D, 0x09E, 0x08F,
0x090, 0x091, 0x082, 0x093, 0x0D4, 0x0B5, 0x09E, 0x097,
0x100, 0x100, 0x09E, 0x01B, 0x09E, 0x100, 0x09E, 0x09E,
0x0A0, 0x0A9, 0x082, 0x0E3, 0x0A4, 0x0B5, 0x0A6, 0x0A7,
0x0A9, 0x0A9, 0x100, 0x0A9, 0x02C, 0x0A9, 0x100, 0x100,
0x030, 0x0B5, 0x100, 0x100, 0x0B5, 0x0B5, 0x100, 0x0B5,
0x0B8, 0x0A9, 0x0BA, 0x0BB, 0x0BC, 0x0B5, 0x09E, 0x0FF,
0x0C8, 0x0C1, 0x082, 0x0E3, 0x0D4, 0x0C5, 0x0C6, 0x0C7,
0x0C8, 0x0C8, 0x0C8, 0x100, 0x0C8, 0x04D, 0x100, 0x100,
0x0D4, 0x051, 0x100, 0x100, 0x0D4, 0x0D4, 0x0D4, 0x100,
0x0C8, 0x0D9, 0x0DA, 0x0DB, 0x0D4, 0x0DD, 0x09E, 0x0FF,
0x100, 0x0E3, 0x0E3, 0x0E3, 0x100, 0x100, 0x066, 0x0E3,
0x0C8, 0x0A9, 0x0EA, 0x0E3, 0x0EC, 0x0ED, 0x0EE, 0x0FF,
```

```
0x0F0, 0x0F1, 0x0F2, 0x0E3, 0x0D4, 0x0B5, 0x0F6, 0x0FF,
0x100, 0x100, 0x07A, 0x0FF, 0x100, 0x0FF, 0x0FF, 0x0FF,
0x040, 0x021, 0x00A, 0x003, 0x004, 0x005, 0x016, 0x007,
0x00A, 0x100, 0x00A, 0x00A, 0x100, 0x100, 0x00A, 0x08F,
0x100, 0x100, 0x016, 0x093, 0x016, 0x100, 0x016, 0x016,
0x018, 0x019, 0x00A, 0x01B, 0x05C, 0x03D, 0x016, 0x01F,
0x021, 0x021, 0x100, 0x021, 0x0A4, 0x021, 0x100, 0x100,
0x028, 0x021, 0x00A, 0x06B, 0x02C, 0x03D, 0x02E, 0x02F,
0x030, 0x021, 0x032, 0x033, 0x034, 0x03D, 0x016, 0x077,
0x0B8, 0x03D, 0x100, 0x100, 0x03D, 0x03D, 0x100, 0x03D,
0x040, 0x040, 0x040, 0x100, 0x040, 0x0C5, 0x100, 0x100,
0x040, 0x049, 0x00A, 0x06B, 0x05C, 0x04D, 0x04E, 0x04F,
0x040, 0x051, 0x052, 0x053, 0x05C, 0x055, 0x016, 0x077,
0x05C, 0x0D9, 0x100, 0x100, 0x05C, 0x05C, 0x05C, 0x100,
0x040, 0x021, 0x062, 0x06B, 0x064, 0x065, 0x066, 0x077,
0x100, 0x06B, 0x06B, 0x06B, 0x100, 0x100, 0x0EE, 0x06B,
0x100, 0x100, 0x0F2, 0x077, 0x100, 0x077, 0x077, 0x077,
0x078, 0x079, 0x07A, 0x06B, 0x05C, 0x03D, 0x07E, 0x077,
0x080, 0x081, 0x082, 0x093, 0x0A4, 0x0C5, 0x086, 0x08F,
0x100, 0x100, 0x00A, 0x08F, 0x100, 0x08F, 0x08F, 0x08F,
0x100, 0x093, 0x093, 0x093, 0x100, 0x100, 0x016, 0x093,
0x0B8, 0x0D9, 0x09A, 0x093, 0x09C, 0x09D, 0x09E, 0x08F,
0x0A4, 0x021, 0x100, 0x100, 0x0A4, 0x0A4, 0x0A4, 0x100,
0x0B8, 0x0A9, 0x0AA, 0x0AB, 0x0A4, 0x0AD, 0x0EE, 0x08F,
0x0B8, 0x0B1, 0x0F2, 0x093, 0x0A4, 0x0B5, 0x0B6, 0x0B7,
0x0B8, 0x0B8, 0x0B8, 0x100, 0x0B8, 0x03D, 0x100, 0x100,
0x040, 0x0C5, 0x100, 0x100, 0x0C5, 0x0C5, 0x100, 0x0C5,
0x0C8, 0x0D9, 0x0CA, 0x0CB, 0x0CC, 0x0C5, 0x0EE, 0x08F,
0x0D0, 0x0D9, 0x0F2, 0x093, 0x0D4, 0x0C5, 0x0D6, 0x0D7,
0x0D9, 0x0D9, 0x100, 0x0D9, 0x05C, 0x0D9, 0x100, 0x100,
0x0E0, 0x0E1, 0x0F2, 0x0E3, 0x0A4, 0x0C5, 0x0EE, 0x0E7,
0x100, 0x100, 0x0EE, 0x06B, 0x0EE, 0x100, 0x0EE, 0x0EE,
0x0F2, 0x100, 0x0F2, 0x0F2, 0x100, 0x100, 0x0F2, 0x077,
0x0B8, 0x0D9, 0x0F2, 0x0FB, 0x0FC, 0x0FD, 0x0EE, 0x0FF,
0x000, 0x001, 0x002, 0x013, 0x024, 0x045, 0x006, 0x00F,
0x100, 0x100, 0x08A, 0x00F, 0x100, 0x00F, 0x00F, 0x00F,
0x100, 0x013, 0x013, 0x013, 0x100, 0x100, 0x096, 0x013,
0x038, 0x059, 0x01A, 0x013, 0x01C, 0x01D, 0x01E, 0x00F,
0x024, 0x0A1, 0x100, 0x100, 0x024, 0x024, 0x024, 0x100,
0x038, 0x029, 0x02A, 0x02B, 0x024, 0x02D, 0x06E, 0x00F,
0x038, 0x031, 0x072, 0x013, 0x024, 0x035, 0x036, 0x037,
0x038, 0x038, 0x038, 0x100, 0x038, 0x0BD, 0x100, 0x100,
0x0C0, 0x045, 0x100, 0x100, 0x045, 0x045, 0x100, 0x045,
0x048, 0x059, 0x04A, 0x04B, 0x04C, 0x045, 0x06E, 0x00F,
0x050, 0x059, 0x072, 0x013, 0x054, 0x045, 0x056, 0x057,
0x059, 0x059, 0x100, 0x059, 0x0DC, 0x059, 0x100, 0x100,
0x060, 0x061, 0x072, 0x063, 0x024, 0x045, 0x06E, 0x067,
0x100, 0x100, 0x06E, 0x0EB, 0x06E, 0x100, 0x06E, 0x06E,
0x072, 0x100, 0x072, 0x072, 0x100, 0x100, 0x072, 0x0F7,
0x038, 0x059, 0x072, 0x07B, 0x07C, 0x07D, 0x06E, 0x07F,
0x0C0, 0x0A1, 0x08A, 0x083, 0x084, 0x085, 0x096, 0x087,
0x08A, 0x100, 0x08A, 0x08A, 0x100, 0x100, 0x08A, 0x00F,
0x100, 0x100, 0x096, 0x013, 0x096, 0x100, 0x096, 0x096,
0x098, 0x099, 0x08A, 0x09B, 0x0DC, 0x0BD, 0x096, 0x09F,
```

```
0x0A1, 0x0A1, 0x100, 0x0A1, 0x024, 0x0A1, 0x100, 0x100,
0x0A8, 0x0A1, 0x08A, 0x0EB, 0x0AC, 0x0BD, 0x0AE, 0x0AF,
0x0B0, 0x0A1, 0x0B2, 0x0B3, 0x0B4, 0x0BD, 0x096, 0x0F7,
0x038, 0x0BD, 0x100, 0x100, 0x0BD, 0x0BD, 0x100, 0x0BD,
0x0C0, 0x0C0, 0x0C0, 0x100, 0x0C0, 0x045, 0x100, 0x100,
0x0C0, 0x0C9, 0x08A, 0x0EB, 0x0DC, 0x0CD, 0x0CE, 0x0CF,
0x0C0, 0x0D1, 0x0D2, 0x0D3, 0x0DC, 0x0D5, 0x096, 0x0F7,
0x0DC, 0x059, 0x100, 0x100, 0x0DC, 0x0DC, 0x0DC, 0x100,
0x0C0, 0x0A1, 0x0E2, 0x0EB, 0x0E4, 0x0E5, 0x0E6, 0x0F7,
0x100, 0x0EB, 0x0EB, 0x0EB, 0x100, 0x100, 0x06E, 0x0EB,
0x100, 0x100, 0x072, 0x0F7, 0x100, 0x0F7, 0x0F7, 0x0F7,
0x0F8, 0x0F9, 0x0FA, 0x0EB, 0x0DC, 0x0BD, 0x0FE, 0x0F7,
0x002, 0x100, 0x002, 0x002, 0x100, 0x100, 0x002, 0x087,
0x048, 0x029, 0x002, 0x00B, 0x00C, 0x00D, 0x01E, 0x00F,
0x010, 0x011, 0x002, 0x013, 0x054, 0x035, 0x01E, 0x017,
0x100, 0x100, 0x01E, 0x09B, 0x01E, 0x100, 0x01E, 0x01E,
0x020, 0x029, 0x002, 0x063, 0x024, 0x035, 0x026, 0x027,
0x029, 0x029, 0x100, 0x029, 0x0AC, 0x029, 0x100, 0x100,
0x0B0, 0x035, 0x100, 0x100, 0x035, 0x035, 0x100, 0x035,
0x038, 0x029, 0x03A, 0x03B, 0x03C, 0x035, 0x01E, 0x07F,
0x048, 0x041, 0x002, 0x063, 0x054, 0x045, 0x046, 0x047,
0x048, 0x048, 0x048, 0x100, 0x048, 0x0CD, 0x100, 0x100,
0x054, 0x0D1, 0x100, 0x100, 0x054, 0x054, 0x054, 0x100,
0x048, 0x059, 0x05A, 0x05B, 0x054, 0x05D, 0x01E, 0x07F,
0x100, 0x063, 0x063, 0x063, 0x100, 0x100, 0x0E6, 0x063,
0x048, 0x029, 0x06A, 0x063, 0x06C, 0x06D, 0x06E, 0x07F,
0x070, 0x071, 0x072, 0x063, 0x054, 0x035, 0x076, 0x07F,
0x100, 0x100, 0x0FA, 0x07F, 0x100, 0x07F, 0x07F, 0x07F,
0x100, 0x100, 0x002, 0x087, 0x100, 0x087, 0x087, 0x087,
0x088, 0x089, 0x08A, 0x09B, 0x0AC, 0x0CD, 0x08E, 0x087,
0x0B0, 0x0D1, 0x092, 0x09B, 0x094, 0x095, 0x096, 0x087,
0x100, 0x09B, 0x09B, 0x09B, 0x100, 0x100, 0x01E, 0x09B,
0x0B0, 0x0A1, 0x0A2, 0x0A3, 0x0AC, 0x0A5, 0x0E6, 0x087,
0x0AC, 0x029, 0x100, 0x100, 0x0AC, 0x0AC, 0x0AC, 0x100,
0x0B0, 0x0B0, 0x0B0, 0x100, 0x0B0, 0x035, 0x100, 0x100,
0x0B0, 0x0B9, 0x0FA, 0x09B, 0x0AC, 0x0BD, 0x0BE, 0x0BF,
0x0C0, 0x0D1, 0x0C2, 0x0C3, 0x0C4, 0x0CD, 0x0E6, 0x087,
0x048, 0x0CD, 0x100, 0x100, 0x0CD, 0x0CD, 0x100, 0x0CD,
0x0D1, 0x0D1, 0x100, 0x0D1, 0x054, 0x0D1, 0x100, 0x100,
0x0D8, 0x0D1, 0x0FA, 0x09B, 0x0DC, 0x0CD, 0x0DE, 0x0DF,
0x100, 0x100, 0x0E6, 0x063, 0x0E6, 0x100, 0x0E6, 0x0E6,
0x0E8, 0x0E9, 0x0FA, 0x0EB, 0x0AC, 0x0CD, 0x0E6, 0x0EF,
0x0B0, 0x0D1, 0x0FA, 0x0F3, 0x0F4, 0x0F5, 0x0E6, 0x0F7,
0x0FA, 0x100, 0x0FA, 0x0FA, 0x100, 0x100, 0x0FA, 0x07F,
0x100, 0x100, 0x006, 0x083, 0x006, 0x100, 0x006, 0x006,
0x008, 0x009, 0x01A, 0x00B, 0x04C, 0x02D, 0x006, 0x00F,
0x050, 0x031, 0x01A, 0x013, 0x014, 0x015, 0x006, 0x017,
0x01A, 0x100, 0x01A, 0x01A, 0x100, 0x100, 0x01A, 0x09F,
0x020, 0x031, 0x022, 0x023, 0x024, 0x02D, 0x006, 0x067,
0x0A8, 0x02D, 0x100, 0x100, 0x02D, 0x02D, 0x100, 0x02D,
0x031, 0x031, 0x100, 0x031, 0x0B4, 0x031, 0x100, 0x100,
0x038, 0x031, 0x01A, 0x07B, 0x03C, 0x02D, 0x03E, 0x03F,
0x050, 0x041, 0x042, 0x043, 0x04C, 0x045, 0x006, 0x067,
```

```
0x04C, 0x0C9, 0x100, 0x100, 0x04C, 0x04C, 0x04C, 0x100,
0x050, 0x050, 0x050, 0x100, 0x050, 0x0D5, 0x100, 0x100,
0x050, 0x059, 0x01A, 0x07B, 0x04C, 0x05D, 0x05E, 0x05F,
0x100, 0x100, 0x0E2, 0x067, 0x100, 0x067, 0x067, 0x067,
0x068, 0x069, 0x06A, 0x07B, 0x04C, 0x02D, 0x06E, 0x067,
0x050, 0x031, 0x072, 0x07B, 0x074, 0x075, 0x076, 0x067,
0x100, 0x07B, 0x07B, 0x07B, 0x100, 0x100, 0x0FE, 0x07B,
0x100, 0x083, 0x083, 0x083, 0x100, 0x100, 0x006, 0x083,
0x0A8, 0x0C9, 0x08A, 0x083, 0x08C, 0x08D, 0x08E, 0x09F,
0x090, 0x091, 0x092, 0x083, 0x0B4, 0x0D5, 0x096, 0x09F,
0x100, 0x100, 0x01A, 0x09F, 0x100, 0x09F, 0x09F, 0x09F,
0x0A8, 0x0A1, 0x0E2, 0x083, 0x0B4, 0x0A5, 0x0A6, 0x0A7,
0x0A8, 0x0A8, 0x0A8, 0x100, 0x0A8, 0x02D, 0x100, 0x100,
0x0B4, 0x031, 0x100, 0x100, 0x0B4, 0x0B4, 0x0B4, 0x100,
0x0A8, 0x0B9, 0x0BA, 0x0BB, 0x0B4, 0x0BD, 0x0FE, 0x09F,
0x0C0, 0x0C9, 0x0E2, 0x083, 0x0C4, 0x0D5, 0x0C6, 0x0C7,
0x0C9, 0x0C9, 0x100, 0x0C9, 0x04C, 0x0C9, 0x100, 0x100,
0x050, 0x0D5, 0x100, 0x100, 0x0D5, 0x0D5, 0x100, 0x0D5,
0x0D8, 0x0C9, 0x0DA, 0x0DB, 0x0DC, 0x0D5, 0x0FE, 0x09F,
0x0E2, 0x100, 0x0E2, 0x0E2, 0x100, 0x100, 0x0E2, 0x067,
0x0A8, 0x0C9, 0x0E2, 0x0EB, 0x0EC, 0x0ED, 0x0FE, 0x0EF,
0x0F0, 0x0F1, 0x0E2, 0x0F3, 0x0B4, 0x0D5, 0x0FE, 0x0F7,
0x100, 0x100, 0x0FE, 0x07B, 0x0FE, 0x100, 0x0FE, 0x0FE,
0x020, 0x041, 0x002, 0x00B, 0x004, 0x005, 0x006, 0x017,
0x100, 0x00B, 0x00B, 0x00B, 0x100, 0x100, 0x08E, 0x00B,
0x100, 0x100, 0x092, 0x017, 0x100, 0x017, 0x017, 0x017,
0x018, 0x019, 0x01A, 0x00B, 0x03C, 0x05D, 0x01E, 0x017,
0x020, 0x020, 0x020, 0x100, 0x020, 0x0A5, 0x100, 0x100,
0x020, 0x029, 0x06A, 0x00B, 0x03C, 0x02D, 0x02E, 0x02F,
0x020, 0x031, 0x032, 0x033, 0x03C, 0x035, 0x076, 0x017,
0x03C, 0x0B9, 0x100, 0x100, 0x03C, 0x03C, 0x03C, 0x100,
0x041, 0x041, 0x100, 0x041, 0x0C4, 0x041, 0x100, 0x100,
0x048, 0x041, 0x06A, 0x00B, 0x04C, 0x05D, 0x04E, 0x04F,
0x050, 0x041, 0x052, 0x053, 0x054, 0x05D, 0x076, 0x017,
0x0D8, 0x05D, 0x100, 0x100, 0x05D, 0x05D, 0x100, 0x05D,
0x020, 0x041, 0x06A, 0x063, 0x064, 0x065, 0x076, 0x067,
0x06A, 0x100, 0x06A, 0x06A, 0x100, 0x100, 0x06A, 0x0EF,
0x100, 0x100, 0x076, 0x0F3, 0x076, 0x100, 0x076, 0x076,
0x078, 0x079, 0x06A, 0x07B, 0x03C, 0x05D, 0x076, 0x07F,
0x080, 0x081, 0x092, 0x083, 0x0C4, 0x0A5, 0x08E, 0x087,
0x100, 0x100, 0x08E, 0x00B, 0x08E, 0x100, 0x08E, 0x08E,
0x092, 0x100, 0x092, 0x092, 0x100, 0x100, 0x092, 0x017,
0x0D8, 0x0B9, 0x092, 0x09B, 0x09C, 0x09D, 0x08E, 0x09F,
0x020, 0x0A5, 0x100, 0x100, 0x0A5, 0x0A5, 0x100, 0x0A5,
0x0A8, 0x0B9, 0x0AA, 0x0AB, 0x0AC, 0x0A5, 0x08E, 0x0EF,
0x0B0, 0x0B9, 0x092, 0x0F3, 0x0B4, 0x0A5, 0x0B6, 0x0B7,
0x0B9, 0x0B9, 0x100, 0x0B9, 0x03C, 0x0B9, 0x100, 0x100,
0x0C4, 0x041, 0x100, 0x100, 0x0C4, 0x0C4, 0x0C4, 0x100,
0x0D8, 0x0C9, 0x0CA, 0x0CB, 0x0C4, 0x0CD, 0x08E, 0x0EF,
0x0D8, 0x0D1, 0x092, 0x0F3, 0x0C4, 0x0D5, 0x0D6, 0x0D7,
0x0D8, 0x0D8, 0x0D8, 0x100, 0x0D8, 0x05D, 0x100, 0x100,
0x0E0, 0x0E1, 0x0E2, 0x0F3, 0x0C4, 0x0A5, 0x0E6, 0x0EF,
0x100, 0x100, 0x06A, 0x0EF, 0x100, 0x0EF, 0x0EF, 0x0EF,
0x100, 0x0F3, 0x0F3, 0x0F3, 0x100, 0x100, 0x076, 0x0F3,
```

```
0x0D8, 0x0B9, 0x0FA, 0x0F3, 0x0FC, 0x0FD, 0x0FE, 0x0EF,
0x001, 0x001, 0x100, 0x001, 0x084, 0x001, 0x100, 0x100,
0x008, 0x001, 0x02A, 0x04B, 0x00C, 0x01D, 0x00E, 0x00F,
0x010, 0x001, 0x012, 0x013, 0x014, 0x01D, 0x036, 0x057,
0x098, 0x01D, 0x100, 0x100, 0x01D, 0x01D, 0x100, 0x01D,
0x060, 0x001, 0x02A, 0x023, 0x024, 0x025, 0x036, 0x027,
0x02A, 0x100, 0x02A, 0x02A, 0x100, 0x100, 0x02A, 0x0AF,
0x100, 0x100, 0x036, 0x0B3, 0x036, 0x100, 0x036, 0x036,
0x038, 0x039, 0x02A, 0x03B, 0x07C, 0x01D, 0x036, 0x03F,
0x060, 0x001, 0x042, 0x04B, 0x044, 0x045, 0x046, 0x057,
0x100, 0x04B, 0x04B, 0x04B, 0x100, 0x100, 0x0CE, 0x04B,
0x100, 0x100, 0x0D2, 0x057, 0x100, 0x057, 0x057, 0x057,
0x058, 0x059, 0x05A, 0x04B, 0x07C, 0x01D, 0x05E, 0x057,
0x060, 0x060, 0x060, 0x100, 0x060, 0x0E5, 0x100, 0x100,
0x060, 0x069, 0x02A, 0x04B, 0x07C, 0x06D, 0x06E, 0x06F,
0x060, 0x071, 0x072, 0x073, 0x07C, 0x075, 0x036, 0x057,
0x07C, 0x0F9, 0x100, 0x100, 0x07C, 0x07C, 0x07C, 0x100,
0x084, 0x001, 0x100, 0x100, 0x084, 0x084, 0x084, 0x100,
0x098, 0x089, 0x08A, 0x08B, 0x084, 0x08D, 0x0CE, 0x0AF,
0x098, 0x091, 0x0D2, 0x0B3, 0x084, 0x095, 0x096, 0x097,
0x098, 0x098, 0x098, 0x100, 0x098, 0x01D, 0x100, 0x100,
0x0A0, 0x0A1, 0x0A2, 0x0B3, 0x084, 0x0E5, 0x0A6, 0x0AF,
0x100, 0x100, 0x02A, 0x0AF, 0x100, 0x0AF, 0x0AF, 0x0AF,
0x100, 0x0B3, 0x0B3, 0x0B3, 0x100, 0x100, 0x036, 0x0B3,
0x098, 0x0F9, 0x0BA, 0x0B3, 0x0BC, 0x0BD, 0x0BE, 0x0AF,
0x0C0, 0x0C1, 0x0D2, 0x0C3, 0x084, 0x0E5, 0x0CE, 0x0C7,
0x100, 0x100, 0x0CE, 0x04B, 0x0CE, 0x100, 0x0CE, 0x0CE,
0x0D2, 0x100, 0x0D2, 0x0D2, 0x100, 0x100, 0x0D2, 0x057,
0x098, 0x0F9, 0x0D2, 0x0DB, 0x0DC, 0x0DD, 0x0CE, 0x0DF,
0x060, 0x0E5, 0x100, 0x100, 0x0E5, 0x0E5, 0x100, 0x0E5,
0x0E8, 0x0F9, 0x0EA, 0x0EB, 0x0EC, 0x0E5, 0x0CE, 0x0AF,
0x0F0, 0x0F9, 0x0D2, 0x0B3, 0x0F4, 0x0E5, 0x0F6, 0x0F7,
0x0F9, 0x0F9, 0x100, 0x0F9, 0x07C, 0x0F9, 0x100, 0x100,
0x010, 0x001, 0x002, 0x003, 0x00C, 0x005, 0x046, 0x027,
0x00C, 0x089, 0x100, 0x100, 0x00C, 0x00C, 0x00C, 0x100,
0x010, 0x010, 0x010, 0x100, 0x010, 0x095, 0x100, 0x100,
0x010, 0x019, 0x05A, 0x03B, 0x00C, 0x01D, 0x01E, 0x01F,
0x100, 0x100, 0x0A2, 0x027, 0x100, 0x027, 0x027, 0x027,
0x028, 0x029, 0x02A, 0x03B, 0x00C, 0x06D, 0x02E, 0x027,
0x010, 0x071, 0x032, 0x03B, 0x034, 0x035, 0x036, 0x027,
0x100, 0x03B, 0x03B, 0x03B, 0x100, 0x100, 0x0BE, 0x03B,
0x100, 0x100, 0x046, 0x0C3, 0x046, 0x100, 0x046, 0x046,
0x048, 0x049, 0x05A, 0x04B, 0x00C, 0x06D, 0x046, 0x04F,
0x010, 0x071, 0x05A, 0x053, 0x054, 0x055, 0x046, 0x057,
0x05A, 0x100, 0x05A, 0x05A, 0x100, 0x100, 0x05A, 0x0DF,
0x060, 0x071, 0x062, 0x063, 0x064, 0x06D, 0x046, 0x027,
0x0E8, 0x06D, 0x100, 0x100, 0x06D, 0x06D, 0x100, 0x06D,
0x071, 0x071, 0x100, 0x071, 0x0F4, 0x071, 0x100, 0x100,
0x078, 0x071, 0x05A, 0x03B, 0x07C, 0x06D, 0x07E, 0x07F,
0x080, 0x089, 0x0A2, 0x0C3, 0x084, 0x095, 0x086, 0x087,
0x089, 0x089, 0x100, 0x089, 0x00C, 0x089, 0x100, 0x100,
0x010, 0x095, 0x100, 0x100, 0x095, 0x095, 0x100, 0x095,
0x098, 0x089, 0x09A, 0x09B, 0x09C, 0x095, 0x0BE, 0x0DF,
```

```
0x0A2, 0x100, 0x0A2, 0x0A2, 0x100, 0x100, 0x0A2, 0x027,
0x0E8, 0x089, 0x0A2, 0x0AB, 0x0AC, 0x0AD, 0x0BE, 0x0AF,
0x0B0, 0x0B1, 0x0A2, 0x0B3, 0x0F4, 0x095, 0x0BE, 0x0B7,
0x100, 0x100, 0x0BE, 0x03B, 0x0BE, 0x100, 0x0BE, 0x0BE,
0x100, 0x0C3, 0x0C3, 0x0C3, 0x100, 0x100, 0x046, 0x0C3,
0x0E8, 0x089, 0x0CA, 0x0C3, 0x0CC, 0x0CD, 0x0CE, 0x0DF,
0x0D0, 0x0D1, 0x0D2, 0x0C3, 0x0F4, 0x095, 0x0D6, 0x0DF,
0x100, 0x100, 0x05A, 0x0DF, 0x100, 0x0DF, 0x0DF, 0x0DF,
0x0E8, 0x0E1, 0x0A2, 0x0C3, 0x0F4, 0x0E5, 0x0E6, 0x0E7,
0x0E8, 0x0E8, 0x0E8, 0x100, 0x0E8, 0x06D, 0x100, 0x100,
0x0F4, 0x071, 0x100, 0x100, 0x0F4, 0x0F4, 0x0F4, 0x100,
0x0E8, 0x0F9, 0x0FA, 0x0FB, 0x0F4, 0x0FD, 0x0BE, 0x0DF,
0x008, 0x001, 0x042, 0x023, 0x014, 0x005, 0x006, 0x007,
0x008, 0x008, 0x008, 0x100, 0x008, 0x08D, 0x100, 0x100,
0x014, 0x091, 0x100, 0x100, 0x014, 0x014, 0x014, 0x100,
0x008, 0x019, 0x01A, 0x01B, 0x014, 0x01D, 0x05E, 0x03F,
0x100, 0x023, 0x023, 0x023, 0x100, 0x100, 0x0A6, 0x023,
0x008, 0x069, 0x02A, 0x023, 0x02C, 0x02D, 0x02E, 0x03F,
0x030, 0x031, 0x032, 0x023, 0x014, 0x075, 0x036, 0x03F,
0x100, 0x100, 0x0BA, 0x03F, 0x100, 0x03F, 0x03F, 0x03F,
0x042, 0x100, 0x042, 0x042, 0x100, 0x100, 0x042, 0x0C7,
0x008, 0x069, 0x042, 0x04B, 0x04C, 0x04D, 0x05E, 0x04F,
0x050, 0x051, 0x042, 0x053, 0x014, 0x075, 0x05E, 0x057,
0x100, 0x100, 0x05E, 0x0DB, 0x05E, 0x100, 0x05E, 0x05E,
0x060, 0x069, 0x042, 0x023, 0x064, 0x075, 0x066, 0x067,
0x069, 0x069, 0x100, 0x069, 0x0EC, 0x069, 0x100, 0x100,
0x0F0, 0x075, 0x100, 0x100, 0x075, 0x075, 0x100, 0x075,
0x078, 0x069, 0x07A, 0x07B, 0x07C, 0x075, 0x05E, 0x03F,
0x080, 0x091, 0x082, 0x083, 0x084, 0x08D, 0x0A6, 0x0C7,
0x008, 0x08D, 0x100, 0x100, 0x08D, 0x08D, 0x100, 0x08D,
0x091, 0x091, 0x100, 0x091, 0x014, 0x091, 0x100, 0x100,
0x098, 0x091, 0x0BA, 0x0DB, 0x09C, 0x08D, 0x09E, 0x09F,
0x100, 0x100, 0x0A6, 0x023, 0x0A6, 0x100, 0x0A6, 0x0A6,
0x0A8, 0x0A9, 0x0BA, 0x0AB, 0x0EC, 0x08D, 0x0A6, 0x0AF,
0x0F0, 0x091, 0x0BA, 0x0B3, 0x0B4, 0x0B5, 0x0A6, 0x0B7,
0x0BA, 0x100, 0x0BA, 0x0BA, 0x100, 0x100, 0x0BA, 0x03F,
0x100, 0x100, 0x042, 0x0C7, 0x100, 0x0C7, 0x0C7, 0x0C7,
0x0C8, 0x0C9, 0x0CA, 0x0DB, 0x0EC, 0x08D, 0x0CE, 0x0C7,
0x0F0, 0x091, 0x0D2, 0x0DB, 0x0D4, 0x0D5, 0x0D6, 0x0C7,
0x100, 0x0DB, 0x0DB, 0x0DB, 0x100, 0x100, 0x05E, 0x0DB,
0x0F0, 0x0E1, 0x0E2, 0x0E3, 0x0EC, 0x0E5, 0x0A6, 0x0C7,
0x0EC, 0x069, 0x100, 0x100, 0x0EC, 0x0EC, 0x0EC, 0x100,
0x0F0, 0x0F0, 0x0F0, 0x100, 0x0F0, 0x075, 0x100, 0x100,
0x0F0, 0x0F9, 0x0BA, 0x0DB, 0x0EC, 0x0FD, 0x0FE, 0x0FF,
0x080, 0x005, 0x100, 0x100, 0x005, 0x005, 0x100, 0x005,
0x008, 0x019, 0x00A, 0x00B, 0x00C, 0x005, 0x02E, 0x04F,
0x010, 0x019, 0x032, 0x053, 0x014, 0x005, 0x016, 0x017,
0x019, 0x019, 0x100, 0x019, 0x09C, 0x019, 0x100, 0x100,
0x020, 0x021, 0x032, 0x023, 0x064, 0x005, 0x02E, 0x027,
0x100, 0x100, 0x02E, 0x0AB, 0x02E, 0x100, 0x02E, 0x02E,
0x032, 0x100, 0x032, 0x032, 0x100, 0x100, 0x032, 0x0B7,
0x078, 0x019, 0x032, 0x03B, 0x03C, 0x03D, 0x02E, 0x03F,
0x040, 0x041, 0x042, 0x053, 0x064, 0x005, 0x046, 0x04F,
0x100, 0x100, 0x0CA, 0x04F, 0x100, 0x04F, 0x04F, 0x04F,
```

```
0x100, 0x053, 0x053, 0x053, 0x100, 0x100, 0x0D6, 0x053,
0x078, 0x019, 0x05A, 0x053, 0x05C, 0x05D, 0x05E, 0x04F,
0x064, 0x0E1, 0x100, 0x100, 0x064, 0x064, 0x064, 0x100,
0x078, 0x069, 0x06A, 0x06B, 0x064, 0x06D, 0x02E, 0x04F,
0x078, 0x071, 0x032, 0x053, 0x064, 0x075, 0x076, 0x077,
0x078, 0x078, 0x078, 0x100, 0x078, 0x0FD, 0x100, 0x100,
0x080, 0x080, 0x080, 0x100, 0x080, 0x005, 0x100, 0x100,
0x080, 0x089, 0x0CA, 0x0AB, 0x09C, 0x08D, 0x08E, 0x08F,
0x080, 0x091, 0x092, 0x093, 0x09C, 0x095, 0x0D6, 0x0B7,
0x09C, 0x019, 0x100, 0x100, 0x09C, 0x09C, 0x09C, 0x100,
0x080, 0x0E1, 0x0A2, 0x0AB, 0x0A4, 0x0A5, 0x0A6, 0x0B7,
0x100, 0x0AB, 0x0AB, 0x0AB, 0x100, 0x100, 0x02E, 0x0AB,
0x100, 0x100, 0x032, 0x0B7, 0x100, 0x0B7, 0x0B7, 0x0B7,
0x0B8, 0x0B9, 0x0BA, 0x0AB, 0x09C, 0x0FD, 0x0BE, 0x0B7,
0x080, 0x0E1, 0x0CA, 0x0C3, 0x0C4, 0x0C5, 0x0D6, 0x0C7,
0x0CA, 0x100, 0x0CA, 0x0CA, 0x100, 0x100, 0x0CA, 0x04F,
0x100, 0x100, 0x0D6, 0x053, 0x0D6, 0x100, 0x0D6, 0x0D6,
0x0D8, 0x0D9, 0x0CA, 0x0DB, 0x09C, 0x0FD, 0x0D6, 0x0DF,
0x0E1, 0x0E1, 0x100, 0x0E1, 0x064, 0x0E1, 0x100, 0x100,
0x0E8, 0x0E1, 0x0CA, 0x0AB, 0x0EC, 0x0FD, 0x0EE, 0x0EF,
0x0F0, 0x0E1, 0x0F2, 0x0F3, 0x0F4, 0x0FD, 0x0D6, 0x0B7,
0x078, 0x0FD, 0x100, 0x100, 0x0FD, 0x0FD, 0x100, 0x0FD
};    // end of decoder_table[]

   // end of file
```