МАРАФОН
CAN технология

CANopen библиотека

DLL мастер

для ОС Windows

Приложение для LabVIEW

Руководство программиста

Код проекта: **0004**$_h$

Москва, 2022

Оглавление

1. Общие положения.

Документ является составной частью руководства по CANopen библиотеке (свидетельство о государственной регистрации программы для ЭВМ № 2022610093). В нем приводится описание CANopen мастера для ОС Windows, реализованного в виде DLL модуля. В качестве одного из приложений используется среда программирования LabVIEW.

Если CANopen мастер используется только в виде DLL модуля, достаточно ознакомиться с разделом 4 настоящего руководства. В разделе 5 приведено описание подсистемы взаимодействия со средой программирования LabVIEW.

2. Изменения в версиях программ.

Версия CANopen библиотеки 2.3

В состав библиотеки включен DLL мастер для ОС Windows.

Версия CANopen библиотеки 3.0

CANopen DLL master реализован на основе версии 3.0 CANopen библиотеки.

Версия приложения 1.1

Внесены изменения в алгоритм обработки мастером протоколов загрузки и сердцебиения (раздел «NMT состояние CAN узла»).
Расширен набор программ-эмуляторов и виртуальных инструментов LabVIEW (раздел «Приложения CANopen DLL мастера»).

2.1 Управление версиями модулей.

Каждый библиотечный модуль DLL версии заключается в условный макрос вида:

```
#if CHECK_VERSION_CANLIB(3, 0, 1)
    код DLL модуля CANopen библиотеки
#endif
```

Аргументы макроса фиксируют версию модуля CANopen библиотеки, на основе которого сформирован код DLL версии.

Аналогичные макросы используются для контроля версий приложения:

```
#if CHECK_VERSION_APPL(1, 1, 0)
    код модуля приложения
#endif
```

3. Структура и параметры DLL мастера.

3.1 Проекты DLL мастера и конечного приложения.

Проекты CANopen DLL мастера и конечного приложения размещаются в директории CANopen_WinDLL_Commander:

- \DLL_src – исходные коды DLL модулей мастера.
- \headers – заголовочные файлы определений и прототипов, которые используются совместно DLL мастером и приложением.
- \LabVIEW – приложение для среды LabVIEW. Содержит VI файлы виртуальных приборов (инструментов).
- \x64 – проект для платформы Windows x64 (64-разрядный).
- \x86 – проект для платформы Windows x86 (32-разрядный).
  \DLL_Win – проект для сборки DLL модулей мастера.
  \Lib_DLL– lib файл для сборки конечного приложения; dll файл для использования совместно с конечным приложением.
  \User – проект конечного приложения.

3.1.1 Структура модулей CANopen DLL мастера.

Директория \DLL_src:

- \application – модули, обеспечивающие связь DLL мастера с приложением.
- \CANopen – модули CANopen стека.
- \include – заголовочные файлы определений и прототипов DLL мастера.

3.1.2 Совместные заголовочные файлы.

Директория \headers.

- __application_dll_defines.h – настраиваемые параметры DLL мастера и приложения.
- __logger_dll_defines.h – параметры регистратора.
- __master_dll_appl_defunc.h – прототипы функций, которые экспортируются DLL мастером и могут вызываться конечным приложением (см. раздел 4)
- master_dll_defines.h – определение параметров (констант) CANopen.
- master_dll_macros.h – определение макросов.
- master_dll_structures.h – определение структур данных.
- master_dll_typedefs.h – определение типов данных.

3.1.3 Структура модулей конечного приложения.

Директория \User. Файлы проекта размещены в поддиректории \src:

- \include – заголовочные файлы определений и прототипов конечного приложения.

3.2 Структуры данных DLL мастера и приложения.

3.2.1 Типы данных CANopen DLL мастера.

Обозначение	Тип данных	Описание
canbyte	**unsigned8**	Без-знаковое целое 8 бит.
cannode	**unsigned8**	Без-знаковое целое 8 бит, идентификатор CAN узла.
canindex	**unsigned16**	Без-знаковое целое 16 бит, индекс объектного словаря.
cansubind	**unsigned8**	Без-знаковое целое 8 бит, субиндекс объектного словаря.
canlink	**unsigned16**	Без-знаковое целое 16 бит, CAN идентификатор канального уровня для 11 битового CAN–ID.

3.2.2 Типы данных объектного словаря CANopen устройства.

Обозначение типа данных	Индекс	Описание
CAN_DEFTYPE_INTEGER8	0002_h	Целое со знаком 8 бит.
CAN_DEFTYPE_UNSIGNED8	0005_h	Без-знаковое целое 8 бит.
CAN_DEFTYPE_INTEGER16	0003_h	Целое со знаком 16 бит.
CAN_DEFTYPE_UNSIGNED16	0006_h	Без-знаковое целое 16 бит.
CAN_DEFTYPE_INTEGER32	0004_h	Целое со знаком 32 бита.
CAN_DEFTYPE_UNSIGNED32	0007_h	Без-знаковое целое 32 бита.
CAN_DEFTYPE_INTEGER64	0015_h	Целое со знаком 64 бита.
CAN_DEFTYPE_UNSIGNED64	$001B_h$	Без-знаковое целое 64 бита.
CAN_DEFTYPE_REAL32	0008_h	С плавающей точкой 32 бита (float).
CAN_DEFTYPE_REAL64	0011_h	С плавающей точкой 64 бита (double).

3.2.3 Структуры данных DLL мастера.

```
union cansdob0 {
        struct segm {
    unsigned8 ndata        число байт в сегменте, которые НЕ содержат данных.
    unsigned8 bit_0        бит 0 нулевого байта сегмента.
    unsigned8 bit_1        бит 1 нулевого байта сегмента.
    unsigned8 toggle       значение мерцающего бита.
        } sg;
};
```

Структура **segm** объединения **cansdob0** заполняется по итогам разбора управляющего (нулевого) байта данных ускоренного и сегментированного SDO протоколов.

struct sdoixs {
canindex index индекс прикладного объекта.
cansubind subind суб-индекс прикладного объекта.
};
Структура **sdoixs** определяет индекс и суб-индекс прикладного CANopen объекта для SDO протокола (мультиплексор SDO протокола).

struct cansdo {
unsigned8 cs команда SDO протокола.
struct sdoixs si индекс и суб-индекс словаря прикладного объекта (мультиплексор SDO протокола).
union cansdob0 b0 управляющий байт SDO протокола.
canbyte bd[8] прикладные данные CAN кадра SDO протокола.
};
Структура **cansdo** размещает информацию SDO кадра в разобранном виде.

struct sdostatus {
int16 state статус во время и после завершения SDO транзакции клиента.
unsigned32 abortcode SDO аборт код, если по завершении транзакции **state** принимает значение CAN_TRANSTATE_SDO_SRVABORT.
};
Структура **sdostatus** размещает информацию о статусе SDO транзакции клиента.

struct sdoclttrans {
unsigned8 adjcs команда SDO протокола, которой сервер должен отвечать на запрос клиента.
struct sdostatus ss статус SDO транзакции.
struct cansdo sd информация об SDO кадре в разобранном виде.
unsigned32 rembytes число оставшихся для передачи байт прикладного объекта.
};
Структура **sdoclttrans** обеспечивает поддержку базовой SDO транзакции клиента (запрос от клиента, прием и обработка ответа сервера). Здесь же ведется подсчет числа оставшихся для передачи байт, что обеспечивает управления полным SDO обменом.

struct sdocltbasic {
int16 busy семафор занятия буфера (инкрементный).
unsigned8 capture флаг захвата буфера (обеспечивает двухфазный семафор).
unsigned32 timeout таймаут операции обмена одним сегментом данных в рамках SDO протокола (базовой транзакции).
struct sdoclttrans ct структура поддержки базовой транзакции клиента.
};
Структура **sdocltbasic** размещает данные, необходимые для реализации базовой SDO транзакции на стороне клиента.

struct sdocltappl {
unsigned8 operation базовый режим передачи SDO (upload / download).
unsigned32 datasize размер данных в байтах.
canbyte *datapnt байтовый указатель на локальный буфер.
struct sdoixs si индекс и суб-индекс прикладного CANopen объекта.
struct sdostatus ss статус SDO транзакции.
};
Структура **sdocltappl** служит для взаимодействия с приложением клиента и используется при обмене данными с помощью SDO протокола.

struct canframe {

unsigned32 id	CAN–ID.
unsigned8 data[8]	поле данные CAN кадра.
unsigned8 len	реальная длина данных (от 0 до 8 байт).
unsigned16 flg	битовые флаги CAN кадра. Бит 0 – RTR, бит 2 – EFF.
unsigned32 ts	временн*ая* метка получения CAN кадра в микросекундах.

};

Структура **canframe** размещает CAN кадр канального уровня. Ее определение содержится в заголовочном файле CAN драйвера CHAI (структура **canmsg_t**).

struct cancache {

int16 busy	семафор занятия кэша (инкрементный).
unsigned8 capture	флаг захвата кэша и занесения в него данных.
canframe cf	CAN кадр канального уровня.

};

Структура **cancache** формирует кэш для размещения отсылаемых CAN кадров.

struct eventcache {

int16 busy	семафор занятия кэша (инкрементный).
unsigned8 capture	флаг захвата кэша и занесения в него данных.
struct eventlog ev	событие регистратора.

};

Структура **eventcache** формирует кэш для размещения событий регистратора.

struct canopennode {

unsigned8 nmt_state	NMT состояние узла.
unsigned16 ecpvalue	период сердцебиения потребителя (мастера) в миллисекундах.
unsigned32 ecpcnt	счетчик протокола сердцебиения мастера.

};

В структуре **canopennode** храниться статус каждого узла CANopen сети.

struct obdentry {

canindex index	индекс объека CANopen устройства.
cansubind subind	субиндекс объека CANopen устройства.
cannode node	номер узла CANopen устройства.
unsigned8 size	размер объекта в байтах (1..8).
unsigned8 updated	флаг обновления значения объекта посредством PDO.
unsigned8 bd[8]	значение (данные) объекта.

};

В структуре **obdentry** размещается одна запись объектного словаря CANopen устройства.

3.2.4 Общие структуры данных DLL мастера и приложения.

union numbers {

unsigned64	**init**	служит для инициализации объединения и переноса данных.
int8	**i8**	целое 8 бит со знаком.
unsigned8	**uns8**	без-знаковое целое 8 бит. Либо булево значение false / true.
int16	**i16**	целое 16 бит со знаком.
unsigned16	**uns16**	без-знаковое целое 16 бит.
int32	**i32**	целое 32 бита со знаком.
unsigned32	**uns32**	без-знаковое целое 32 бита.
int64	**i64**	целое 64 бита со знаком.
unsigned64	**uns64**	без-знаковое целое 64 бита.
real32	**re32**	с плавающей точкой одинарной точности (float).
real64	**re64**	с плавающей точкой двойной точности (double).

};

Объединение **numbers** служит для единого представления различных типов численных данных.

Замечание.

Согласованность данных в объединении numbers обеспечивается только для little-endian порядка байт и лишь в случае, когда размер байта составляет 8 бит.

struct eventlog {

time_t ts	временн*а*я метка события.
unsigned8 node	номер CAN узла, в котором было порождено событие. для событий мастера равен нулю.
unsigned8 cls	класс события.
unsigned8 type	тип события (info, warning, error и т.д.)
unsigned8 misc	зарезервировано (выравнивание).
int16 code	код события.
int32 info	дополнительная информация о событии.

};

Структура **eventlog** содержит информацию о событии для регистратора.

3.3 Параметры DLL мастера и конечного приложения.

3.3.1 Параметры DLL мастера.

Параметры DLL мастера определены в файлах: __application_dll_defines.h, __logger_dll_defines.h, __obd_dll_defines.h.

- CAN_TIMERUSEC
  Период CANopen таймера в микросекундах. Значение параметра должно быть не менее 100. Рекомендуемый период таймера от 1 до 10 миллисекунд (значение параметра от 1000 до 10000).
- CAN_TIMEOUT_RETRIEVE
  Таймаут получения данных из CAN сети для базовой SDO транзакции клиента. Значение по умолчанию в микросекундах. В базовой SDO транзакции клиент ожидает ответа от сервера.
- CAN_TIMEOUT_READ
  Таймаут чтения приложением принятых из CAN сети данных для базовой SDO транзакции клиента в микросекундах.
- CAN_NOF_PDO_RECV_SLAVE
  Число принимаемых PDO параметров (RPDO) для каждого CANopen узла.
- CAN_NOF_PDO_TRAN_SLAVE
  Число передаваемых PDO параметров (TPDO) для каждого CANopen узла.
- CAN_NOF_SYNCPDO_MASTER
  Размер каждого FIFO буфера для принимаемых и передаваемых DLL мастером синхронных PDO.
- CAN_RPDO_TRTYPE
  Значение по умолчанию для типа передачи принимаемых PDO DLL мастера. Используется для инициализации субиндекса 2 (transmission type) объектов $1400_h..15FB_h$ – коммуникационные параметры принимаемых PDO.
- CAN_RPDO_ET_MS
  Значение по умолчанию для таймера событий принимаемых PDO DLL мастера. Инициализирует субиндекс 5 (event timer) объектов $1400_h..15FB_h$ – коммуникационные параметры принимаемых PDO.
- CAN_TPDO_TRTYPE
  Значение по умолчанию для типа передачи передаваемых PDO DLL мастера. Используется для инициализации субиндекса 2 (transmission type) объектов $1800_h..19FB_h$ – коммуникационные параметры передаваемых PDO.
- CAN_TPDO_INHIBIT_100MCS
  Значение по умолчанию для времени подавления посылок передаваемых PDO DLL мастера. Инициализирует субиндекс 3 объектов $1800_h..19FB_h$ – коммуникационные параметры передаваемых PDO.
- CAN_TPDO_ET_MS
  Значение по умолчанию для таймера событий передаваемых PDO DLL мастера. Инициализирует субиндекс 5 (event timer) объектов $1800_h..19FB_h$ – коммуникационные параметры передаваемых PDO.
- CAN_TPDO_SYNC_START
  Начальное значение по умолчанию для SYNC счетчика передаваемых PDO DLL мастера. Инициализирует субиндекс 6 (SYNC start value) объектов $1800_h..19FB_h$ – коммуникационные параметры передаваемых PDO.

- EVENT_CACHE_SIZE
  Размер кэша регистратора. Нулевой элемент используется при переполнении кэша.
- EVENT_FIFO_SIZE
  Размер FIFO регистратора. Размещает до (EVENT_FIFO_SIZE-1) событий.
- EVENT_NODE_MASTER
  Для регистрации событий, порожденных в самом CANopen мастере, используется нулевой номер CAN узла.
- EVENT_CLASS_*
  Классы событий регистратора.
- EVENT_TYPE_*
  Типы событий регистратора (info, warning, error и т.д.).
- EVENT_CODE_*
  Коды отдельных событий. Для событий различных классов могут иметь одинаковые значения.
- OBD_NODES_SIZE
  Зарезервированный размер объектного словаря CANopen узлов в DLL мастере. Максимальное число объектов CANopen устройств, которые могут передаваться посредством байт-ориентированного PDO, составляет порядка 8128 (127 узлов * (4 TPDO + 4 RPDO) * 8 объектов).

3.3.2 Параметры конечного приложения.

Определены в заголовочном файле конечного приложения __application_user_defines.h

- CAN_NETWORK_CONTROLLER
  Номер канала контроллера CAN сети. Значение по умолчанию.
- CAN_BITRATE_INDEX
  Индекс битовой скорости CAN сети. Значение по умолчанию.

4. CANopen протоколы и экспортируемые функции DLL мастера.

4.1 Функции запуска и останова DLL мастера.

int16 start_can_master(unsigned8 net, unsigned8 br);

Осуществляет инициализацию и запуск DLL мастера. Вызывается однократно при старте конечного приложения.

Параметры:

- **net** – номер канала контроллера CAN сети.
- **br** – индекс битовой скорости CAN сети.

Возвращаемые значения: нормальное завершение = 0; ошибка < 0.

- CAN_RETOK – запуск CAN мастера выполнен успешно.
- CAN_ERRET_CI_INIT – ошибка инициализации CAN драйвера.
- CAN_ERRET_CI_OPEN – не удалось открыть канал CAN контроллера.
- CAN_ERRET_CI_CLOSE – не удалось закрыть канал CAN контроллера.
- CAN_ERRET_CI_START – не удалось перевести CAN контроллер в рабочее состояние.
- CAN_ERRET_CI_STOP – не удалось перевести CAN контроллер в состояние останова.
- CAN_ERRET_CI_HANDLER – не удалось назначить обработчик сигналов CAN драйвера.
- CAN_ERRET_CI_BITRATE – ошибка установа битовой скорости CAN контроллера.

int16 stop_can_master(void);

Осуществляет останов DLL мастера. Вызывается однократно при останове конечного приложения.

Возвращаемые значения: нормальное завершение = 0; ошибка < 0.

- CAN_RETOK – останов CAN мастера выполнен успешно.
- CAN_ERRET_CI_STOP – не удалось перевести CAN контроллер в состояние останова.
- CAN_ERRET_CI_CLOSE – не удалось закрыть канал CAN контроллера.

void canopen_monitor(void);

CANopen монитор DLL мастера. Вызывается из главного цикла программы.

4.2 Регистратор событий DLL мастера и узлов CAN сети.

Регистратор событий DLL мастера фиксирует как события, порожденные в узлах CAN сети (срочные сообщения Emergency и т.д.), так и события самого мастера.

int16 read_logger_event(struct eventlog *ev);

Читает самое старое из сохраненных событий регистратора. После чтения это событие удаляется из FIFO регистратора.

Параметры:

- ***ev** – информация о зарегистрированном событии.

Возвращаемые значения: нормальное завершение = 0; ошибка < 0.

- GEN_RETOK – событие прочитано и удалено из регистратора.
- GEN_ERRET_LOGGER_BUSY – регистратор занят, прочитать информацию о событии можно будет позже.
- GEN_ERRET_NO_LOGEVENT – нет зарегистрированных событий.

4.3 Объектный словарь узлов CAN сети.

Для обеспечения возможности работы с данными, которые принимаются и передаются посредством PDO протокола, в DLL мастере формируется объектный словарь CANopen устройств. В нем определяются лишь те объекты, значения которых могут обновляться и передаваться асинхронно с использованием PDO.

Словарь формируется в виде упорядоченного списка. Выборка записей словаря осуществляется методом бинарного поиска по ключу node, index, subind.

int16 add_node_object_dictionary(cannode node, canindex index, cansubind subind, canindex type);

Добавляет объект в словарь DLL мастера.

Параметры:

- **node** – номер CAN узла объекта от 1 до 127.
- **index** – индекс объекта CAN узла, который передается посредством PDO протокола.
- **subind** – субиндекс объекта CAN узла, который передается посредством PDO протокола.
- **type** – индекс типа объекта (см. п. 3.2.2).

Возвращаемые значения: нормальное завершение = 0; ошибка < 0.

- GEN_RETOK – нормальное завершение.
- CAN_ERRET_NODEID – ошибочный номер CAN узла.
- GEN_ERRET_DATATYPE – указанный тип данных не существует или не поддерживается.
- GEN_ERRET_OUTOFMEM – нет места для размещения нового объекта.
- GEN_ERRET_DUPLICATED – объект с указанными номером CAN узла, индексом и субиндексом уже занесен в объектный словарь.

int16 read_node_object_dictionary(cannode node, canindex index, cansubind subind, unsigned8 *upd, union numbers *num);

Чтение значения объекта из словаря DLL мастера.

Для выборки объекта используется бинарный поиск.

Параметры:

- **node** – номер CAN узла объекта от 1 до 127.
- **index** – индекс объекта CAN узла, который передается посредством PDO протокола.
- **subind** – субиндекс объекта CAN узла, который передается посредством PDO протокола.
- ***upd** – TRUE - значение объекта было обновлено из принятого PDO, FALSE - значение объекта не обновлялось.
- ***num** – содержит значение объекта, соответствующее его типу.

Возвращаемые значения: нормальное завершение = 0; ошибка < 0.

- CAN_RETOK – нормальное завершение.
- CAN_ERRET_OBD_NOOBJECT – в узле **node** не существует объекта с индексом **index** и субиндексом **subind**.

i**nt16 write_node_object_dictionary(cannode node, canindex index, cansubind subind, union numbers *num);**

Запись значения объекта в словарь DLL мастера.

Для выборки объекта используется бинарный поиск. В дальнейшем этот объект может быть передан посредством PDO. Не запрещена запись значений объектов, которые обновляются из принимаемых мастером PDO. При этом флаг обновления значения не устанавливается.

Параметры:

- **node** – номер CAN узла объекта от 1 до 127.
- **index** – индекс объекта CAN узла, который передается посредством PDO протокола.
- **subind** – субиндекс объекта CAN узла, который передается посредством PDO протокола.

- ***num** – значение объекта, соответствующее его типу.

Возвращаемые значения: нормальное завершение = 0; ошибка < 0.

- CAN_RETOK – нормальное завершение.
- CAN_ERRET_OBD_NOOBJECT – в узле **node** не существует объекта с индексом **index** и субиндексом **subind**.

int16 get_node_updated_object(cannode *node, canindex *index, cansubind *subind, union numbers *num);

Отыскивает и читает значение первого обновленного объекта из словаря DLL мастера. Для выборки объекта используется последовательный просмотр записей словаря.

Параметры:

- ***node** – номер CAN узла обновленного объекта.
- ***index** – индекс обновленного объекта CAN узла.
- ***subind** – субиндекс обновленного объекта CAN узла.
- ***num** – значение обновленного объекта, соответствующее его типу.

Возвращаемые значения: нормальное завершение = 0; ошибка < 0.

- CAN_RETOK – нормальное завершение.
- CAN_ERRET_NO_UPDATED – не обнаружено ни одного обновленного объекта.

4.4 NMT протоколы.

4.4.1 Команды NMT управления.

Обозначение NMT команды	Значение	Описание
CAN_NMT_START_REMOTE_NODE	1	Запуск CANopen устройства.
CAN_NMT_STOP_REMOTE_NODE	2	Останов CANopen устройства.
CAN_NMT_ENTER_PRE_OPERATIONAL	128	Переход в пред-операционное состояние.
CAN_NMT_RESET_NODE	129	Полная инициализация устройства.
CAN_NMT_RESET_COMMUNICATION	130	Инициализация коммуникационной подсистемы устройства.

4.4.2 Формирование NMT команд.

void nmt_master_command(unsigned8 cs, cannode node);

Формирует и отсылает в сеть кадр NMT протокола (с нулевым значением CAN идентификатора). Функция не осуществляет проверку значения NMT команды и номера CAN узла. Нулевое значение номера CAN узла означает, что NMT команда адресована всем NMT responder устройствам.

Параметры:

- **cs** – NMT команда (см. п. 4.4.1).
- **node** – номер CAN узла от 0 до 127.

4.4.3 NMT состояние CAN узла.

DLL мастер принимает кадры протоколов контроля ошибок (boot-up, сердцебиения) от всех узлов CAN сети. Фактическое NMT состояние узла, помимо загрузки boot-up, может быть определено мастером только после активации протокола сердцебиения в CAN узле. Для этого нужно задать период сердцебиения поставщика в миллисекундах (объект 1017_h) в соответствующем узле.

Обозначение NMT состояния узла	Значение	Описание
CAN_NODE_STATE_INITIALISING	0	CANopen устройство активировано (boot-up протокол).
CAN_NODE_STATE_STOPPED	4	Состояние останова CAN узла.
CAN_NODE_STATE_OPERATIONAL	5	Операционное состояние узла.
CAN_NODE_STATE_PRE_OPERATIONAL	127	Пред-операционое состояние узла.
CAN_NODE_STATE_DUMMY	254	Нет данных о NMT состоянии CAN узла.
CAN_NODE_STATE_UNCERTAIN	255	Не определенное состояние CAN узла (произошло событие сердцебиения).

4.4.4 Протокол сердцебиения DLL мастера.

int16 read_master_hbt(cannode node, unsigned16 *hbt);

Чтение периода сердцебиения потребителя DLL мастера.

Параметры:

- **node** – номер CAN узла от 1 до 127.
- ***hbt** – период сердцебиения потребителя в миллисекундах для CAN узла **node**.

Возвращаемые значения: нормальное завершение = 0; ошибка < 0.

- CAN_RETOK – нормальное завершение.
- CAN_ERRET_NODEID – ошибочный номер CAN узла.

int16 write_master_hbt(cannode node, unsigned16 hbt);

Запись периода сердцебиения потребителя.

Параметры:

- **node** – номер CAN узла от 1 до 127.
- **hbt** – период сердцебиения потребителя в миллисекундах для CAN узла **node**.
  Для контроля события сердцебиения период потребителя должен превышать период поставщика (объект 1017_h) соответствующего CANopen узла.
  Нулевое значение периода отключает протокол сердцебиения потребителя и устанавливает NMT состояние CAN узла **node** в CAN_NODE_STATE_DUMMY. Если при этом сердцебиение поставщика остается активным, мастер будет регистрировать NMT состояние CAN узла без отслеживания события сердцебиения.

Возвращаемые значения: нормальное завершение = 0; ошибка < 0.

- CAN_RETOK – нормальное завершение.
- CAN_ERRET_NODEID – ошибочный номер CAN узла.

unsigned8 read_nmt_state(cannode node);

Возвращает NMT состояние узла **node**.

Параметры:

- **node** – номер CAN узла от 1 до 127.

Возвращаемые значения:

- NMT состояние узла.
  При ошибочном номере CAN узла возвращает значение CAN_NODE_STATE_DUMMY.

4.5 SYNC протокол.

4.5.1 Объекты протокола синхронизации.

Индекс	Название объекта
1005_h	COB-ID объекта синхронизации SYNC.
1006_h	Период объекта синхронизации в микросекундах.
1007_h	Длительность окна синхронизации в микросекундах.
1019_h	Значение переполнения SYNC счетчика.

1005_h
COB-ID объекта синхронизации SYNC.
Значение по умолчанию: 80_h.

X	0/1	0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	11-битовый идентификатор
X	0/1	1	29-битовый идентификатор	
31	30	29	28-11	10-0

Биты	Значение	Описание
31	X	Не используется
30	0 1	Устройство НЕ генерирует SYNC Устройство генерирует SYNC
29	0 1	Используется 11-битовый CAN-ID Используется 29-битовый CAN-ID
28 - 0	X	29-битовый CAN-ID расширенного формата кадра
10 - 0	X	11-битовый CAN-ID основного формата кадра

Установ бита 29 в значение 1 запрещен. Первая посылка SYNC кадра производится после установа бита 30 в единицу в течение одного периода CANopen таймера. При записи объекта значение SYNC счетчика сбрасывается в единицу. Изменение бит 0-28 запрещено в случае, когда устройство осуществляет генерацию SYNC (бит 30 = 1).

1006_h
Период объекта синхронизации SYNC в микросекундах.
DLL мастер НЕ генерирует SYNC (бит 30 объекта 1005_h сброшен в 0):
Задает контрольный интервал поступления SYNC посылок. Если в течение контрольного интервала не принято ни одного SYNC кадра любого вида, регистрируется ошибка. Установ нулевого значения прекращает SYNC контроль.
DLL мастер генерирует SYNC (бит 30 объекта 1005_h установлен в 1):
Задает период коммуникационного цикла (SYNC интервал). Установ нулевого значения прекращает генерацию SYNC и сбрасывает значение SYNC счетчика (объект 1019_h) в единицу. При изменении периода синхронизации на значение, отличное от нуля, передача SYNC посылок возобновляется в течение одного периода CANopen таймера.
Фактическое разрешение объекта синхронизации определяется разрешением CANopen таймера. Если период синхронизации задан меньшим, нежели период таймера, но отличен от нуля, генерация SYNC посылок будет осуществляться с частотой таймера. В остальных случаях фактический период генерации будет равен целому числу тиков таймера, но не превышать период объекта синхронизации.

1007_h
Окно синхронизации в микросекундах.
Задает длительность временн*о*го окна для синхронных PDO. Установ нулевого значения

прекращает использование окна синхронизации. Если длительность окна превышает период объекта синхронизации (1006_h), оно также не будет оказывать влияние на обработку синхронных PDO.
При поступлении объекта синхронизации SYNC для синхронных PDO выполняются следующие операции:
1. Запись в объектный словарь (активация) значений объектов, принятых в предшествующем SYNC цикле.
2. Постановка соответствующих синхронных PDO на отправку в CAN сеть.
3. Прием синхронных PDO для активации в последующем SYNC цикле.

Если какие-либо из указанных действий для части PDO не были завершены до истечения окна синхронизации, дальнейшая обработка этих PDO не производится. В п. 2 истечение временн*о*го окна контролируется по моменту размещения PDO в выходном CANopen кэше. Фактическая отправка PDO в CAN сеть может произойти с некоторой задержкой. Длительность временн*о*го окна определяется с точностью до периода CANopen таймера. Поскольку SYNC объект принимается из CAN сети не зависимо от таймерного сигнала, фактическая длительность окна “дрожит” в пределах одного периода таймера.

1019_h
Значение переполнения для SYNC счетчика. Задает его максимальное значение:

Значение	Описание
0	SYNC кадры должны иметь длину поля данных 0 байт. SYNC счетчик не разрешен.
1	Зарезервировано.
2..240	SYNC кадры должны иметь длину поля данных 1 байт. SYNC счетчик активирован. Поле данных содержит значение счетчика.
241..255	Зарезервировано.

Если значение объекта превышает 1, принимаемые и передаваемые SYNC кадры должны иметь длину поля данных 1 байт. В случае, если длина поля данных не соответствует ожидаемой, SYNC кадр не обрабатывается и выдается срочное сообщение (объект EMCY) с кодом ошибки 8240_h (неподходящая длина данных SYNC кадра). Изменение объекта 1019_h запрещено, если значение периода объекта синхронизации 1006_h отлично от нуля.

4.5.2 Функции SYNC протокола.

unsigned32 read_sync_num(void);
Возвращает число обработанных SYNC кадров с момента последнего обращения к данной функции.
Возвращаемые значения:
- число принятых, проверенных и обработанных SYNC кадров.

int16 read_sync_object(canindex index, unsigned32 *data);
Чтение объектов протокола синхронизации DLL мастера.
Параметры:
- **index** – индекс объекта синхронизации.
- ***data** – значение объекта.

Возвращаемые значения: нормальное завершение = 0; ошибка < 0.
- CAN_RETOK – нормальное завершение.
- CAN_ERRET_OBD_NOOBJECT – не существует SYNC объекта с индексом **index**.

int16 write_sync_object(canindex index, unsigned32 data);

Запись объектов протокола синхронизации DLL мастера.

Параметры:

- **index** – индекс объекта синхронизации.
- **data** – значение записываемого объекта.

Возвращаемые значения: нормальное завершение = 0; ошибка < 0.

- CAN_RETOK – нормальное завершение.
- CAN_ERRET_OBD_NOOBJECT – не существует SYNC объекта с индексом **index**.
- CAN_ERRET_OBD_VALRANGE – ошибка диапазона записываемого значения.
- CAN_ERRET_OBD_OBJACCESS – в текущем состоянии объект не может быть изменен.
- CAN_ERRET_OBD_DEVSTATE – состояние других объектов не позволяет изменить значение данного объекта.

4.6 Emergency протокол.

Все срочные сообщения Emergency, которые принимаются от узлов сети, а также возникают в самом DLL мастере, заносятся в регистратор (см. п. 4.2). Номер CAN узла, с которым соотносится EMCY, определяется в соответствии с предопределенным распределением CANopen идентификаторов. Срочным сообщениям, порожденным в DLL мастере, присваивается нулевой номер CAN узла. Класс EMCY события имеет значение EVENT_CLASS_EMCY. Код срочного сообщения записывается в поле **code** структуры **eventlog** в без-знаковом формате. Обработка срочных сообщений является задачей конечного приложения.

4.7 PDO протокол.

В DLL мастере определены 508+508 PDO объектов, по 4 TPDO и 4 RPDO для каждого узла сети. Их инициализация выполняется в соответствии с предопределенным распределением CAN идентификаторов. При этом все идентификаторы PDO определяются как не действительные (invalid). Остальные параметры инициализации PDO приведены в заголовочном файле __application_dll_defines.h. Коммуникационные параметры всех PDO могут быть переконфигурированы конечным приложением.

Для работы с данными PDO необходимо сформировать в DLL мастере объектный словарь CANopen узлов (см. п. 4.3). В нем достаточно задать лишь объекты, значения которых могут обновляться из PDO асинхронно. Затем нужно определить PDO отображение. При этом можно использовать объекты определения типов данных длиной 1, 2 и 4 байта (индексы 0002_h..0007_h). DLL мастер поддерживает только байт-ориентированное отображение, когда в PDO может быть отображено не более восьми объектов.

Замечание.

Принимаемые DLL мастером PDO (RPDO мастера) являются передаваемыми для CANopen узлов (TPDO CAN узла). Соответственно, передаваемые TPDO мастера являются принимаемыми RPDO для CANopen узлов.

4.7.1 Объекты PDO протокола.

Индексы	Название объекта
1400_h..$15FB_h$	Коммуникационные параметры PDO, которые принимаются DLL мастером (TPDO CAN узлов). При инициализации группируются последовательно по 4 индекса для каждого CAN узла.
1600_h..$17FB_h$	Параметры отображения PDO, которые принимаются DLL мастером.

$1800_h..19FB_h$	Коммуникационные параметры PDO, которые передаются DLL мастером (RPDO CAN узлов). При инициализации группируются последовательно по 4 индекса для каждого CAN узла.
$1A00_h..1AFB_h$	Параметры отображения PDO, которые передаются DLL мастером.

4.7.2 Коммуникационные параметры PDO.

Субиндекс 1. PDO COB-ID.

0/1	0/1	0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	11-битовый идентификатор
0/1	0/1	1	29-битовый идентификатор	
31	30	29	28-11	10-0

Биты	Значение	Описание
31	0 1	PDO существует / действителен PDO не существует / не действителен
30	0 1	Удаленный запрос PDO (RTR) разрешен Удаленный запрос PDO (RTR) запрещен
29	0 1	Используется 11-битовый CAN-ID Используется 29-битовый CAN-ID
28 - 0	X	29-битовый CAN-ID расширенного формата кадра
10 - 0	X	11-битовый CAN-ID основного формата кадра

Установ бита 29 в значение 1 запрещен. Изменение бит 0-28 и бита 30 запрещено если PDO действителен (бит 31 = 0).

Субиндекс 2. Тип передачи PDO.

Тип приема/передачи	Прием или передача PDO				
	циклический	а-циклический	синхронный	а-синхронный	только RTR
0		X	X		
1-240	X		X		
241-251	зарезервированы				
252			X		X
253				X	X
254				X	
255				X	

Синхронная передача (типы 0-240 и 252) означает привязку выдачи PDO к объекту синхронизации SYNC. Асинхронная передача такой привязки не предусматривает. Тип передачи 0 означает, что передача PDO не будет периодической, однако остается привязанной к SYNC объекту. Значения 1-240 определяют периодическую передачу, причем тип передачи задает число SYNC посылок, которые должны быть получены для инициализации (выдачи) PDO. Синхронные принимаемые PDO (тип передачи 0-240) активируются (обновляют принятые данные) при получении очередного SYNC объекта после приема самого PDO.
Типы передачи 252 и 253 означают, что PDO передается только при наличии удаленного запроса (RTR). Причем PDO типа 252 будет передан лишь при получении - вслед за RTR - очередного SYNC объекта. Эти два значения типов передачи возможны только для передаваемых PDO.

Тип 254 для передаваемого PDO означает, что асинхронное событие, которое инициирует передачу, определяется производителем. Тип 255 подразумевает, что соответствующее событие задается в прикладном профиле устройства. Принимаемые PDO типа 254 и 255 обновляют принятые данные (активируются) сразу после получения.

Субиндекс 3. Время подавления посылок передаваемых PDO.
Может использоваться для передаваемых PDO типов 254 и 255. Объект задается в виде числа (множителя) 100 мкс временны*х* интервалов. Изменение объекта запрещено если PDO действителен (бит 31 COB-ID = 0). Время подавления определяется с точностью до периода CANopen таймера. Поскольку PDO является асинхронным и может возникать не зависимо от таймерного сигнала, время подавления "дрожит" в пределах одного периода таймера. В случае использования субиндекса для принимаемых PDO запись любого значения завершается успешно без каких-либо последствий, а по чтению всегда возвращается ноль.

Субиндекс 4. Зарезервирован.
Запись любого значения завершается успешно без каких-либо последствий, а по чтению всегда возвращается ноль.

Субиндекс 5. Таймер события в миллисекундах.
Может использоваться для передаваемых PDO типов 254 и 255. Задает максимальный интервал времени между передачей PDO при отсутствии в системе других событий, вызывающих передачу этого PDO. Разрешение таймера события определяется разрешением CANopen таймера. Поскольку передача PDO осуществляется асинхронно, интервал до первого таймерного PDO "дрожит" в пределах одного периода таймера. Если длительность таймера события задана меньшей, нежели период таймера, но отлична от нуля, генерация PDO будет осуществляться с частотой CANopen таймера. В остальных случаях фактический период генерации будет равен целому числу тиков CANopen таймера, но не превышать заданного значения таймера события.
В случае использования субиндекса для принимаемых PDO задает контрольный интервал времени приема соответствующего PDO. Если в течение установленного времени не поступило ни одного PDO, регистрируется ошибка истечения контрольного времени. Интервал времени переустанавливается только после успешной записи всех данных из PDO в объектный словарь приложения (активации PDO).
Для синхронных принимаемых PDO при выборе контрольного интервала следует учитывать дополнительные обстоятельства. Во-первых, активация синхронных PDO производится при получении очередного SYNC объекта после приема самих PDO, то есть задержка активации может достигать одного периода SYNC. Во-вторых, установ временно*г*о окна для синхронных PDO (объект 1007_h) может привести к тому, что PDO, поступившие по истечении окна синхронизации, не будет приняты к обработке.
Контрольный интервал времени определяется с точностью до периода CANopen таймера. Поскольку PDO является асинхронным, фактическая длительность интервала "дрожит" в пределах одного периода таймера.

Субиндекс 6. Стартовое значение SYNC счетчика.
Объект определен только для передаваемых PDO. Нулевое значение означает, что SYNC счетчик не используется для данного PDO. Значения от 1 до 240 определяют, что для PDO учитывается значение SYNC счетчика. Если SYNC счетчик не разрешен (объект 1019_h), значение данного субиндекса игнорируется. В случае активного SYNC счетчика первым SYNC кадром считается тот, значение счетчика которого совпадает со стартовым значением. Изменение объекта запрещено если PDO действителен (бит 31 COB-ID = 0).

4.7.3 Параметры PDO отображения.

Суб-индекс 0.
Фиксирует число действительных записей PDO отображения, то есть число прикладных объектов CANopen узла, которые передаются или принимаются соответствующим PDO. Для каждого PDO доступно до восьми записей отображения.

Суб-индексы 1..8.
Содержат описание прикладных объектов PDO отображения в следующем формате:

Индекс объекта	Суб-индекс	Длина объекта (бит)
31 16	15 8	7 0

Ошибки при формировании PDO отображения могут быть вызваны стремлением записать индекс и субиндекс не существующего в словаре объекта, его неверной длине, либо не правильной длине всего PDO. Последняя не должна превышать 8 байт (64 бита). Возможно включение в PDO отображение объектов определения типа $0002_h..0007_h$. Это позволяет при необходимости выравнивать размещения прикладных объектов в PDO.
Изменять параметры PDO отображения можно как для действительных, так и для не действительных PDO. При этом используется следующая процедура:
1. Запретить PDO отображение, установив значение 0 для нулевого субиндекса. При этом PDO будет переведено в не действительное состояние.
2. Изменить PDO отображение, модифицировав соответствующие субиндексы.
3. Разрешить PDO отображение, записав в субиндекс 0 число отображаемых объектов.
4. Перевести PDO в действительное состояние, записав 0 в бит 31 COB-ID соответствующего коммуникационного параметра PDO.

Когда мастер принимает PDO, длина которого превышает записанную в соответствующем PDO отображении, используется необходимое число первых байт PDO. Если же число байт принятого PDO оказывается меньшим, нежели количество байт отображения, данные не обрабатываются и регистрируется ошибка.

4.7.4 Функции PDO протокола.

void set_all_pdos_state(unsigned8 state);

Переводит все определенные в мастере PDO в действительное или не действительное состояние. PDO переводится в действительное состояние только если для него задано состоятельное PDO отображение.
Параметры:
- **state** – новое состояние PDO (VALID / NOT_VALID).

int16 get_pdo_node(canindex index, cannode *node);

Чтение номера CAN узла для выбранного PDO.
Параметры:
- ***node** – номер CAN узла, которому соответствует PDO **index**.
- **index** – индекс коммуникационного параметра PDO.

Возвращаемые значения: нормальное завершение = 0; ошибка < 0:
- CAN_RETOK – нормальное завершение.
- CAN_ERRET_OBD_NOOBJECT – не существует коммуникационного PDO объекта с индексом **index**.

int16 put_pdo_node(canindex index, cannode node);

Запись номера CAN узла для выбранного PDO.

Параметры:
- **node** – номер CAN узла, которому должен соответствовать PDO **index**.
- **index** – индекс коммуникационного параметра PDO.

Возвращаемые значения: нормальное завершение = 0; ошибка < 0:
- CAN_RETOK – нормальное завершение.
- CAN_ERRET_OBD_NOOBJECT – не существует коммуникационного PDO объекта с индексом **index**.

int16 read_pdo_communication(canindex index, cansubind subind, unsigned32 *data);

Чтение коммуникационного PDO объекта DLL мастера.

Параметры:
- **index** – индекс коммуникационного PDO объекта.
- **subind** – субиндекс коммуникационного PDO объекта.
- ***data** – значение объекта.

Возвращаемые значения: нормальное завершение = 0; ошибка < 0.
- CAN_RETOK – нормальное завершение.
- CAN_ERRET_OBD_NOOBJECT – не существует коммуникационного PDO объекта с индексом **index**.
- CAN_ERRET_OBD_NOSUBIND – несуществующий субиндекс объекта.

int16 write_pdo_communication(canindex index, cansubind subind, unsigned32 data);

Запись значения коммуникационного PDO объекта DLL мастера.

Параметры:
- **index** – индекс коммуникационного PDO объекта.
- **subind** – субиндекс коммуникационного PDO объекта.
- **data** – значение объекта.

Возвращаемые значения: нормальное завершение = 0; ошибка < 0.
- CAN_RETOK – нормальное завершение.
- CAN_ERRET_OBD_NOOBJECT – не существует коммуникационного PDO объекта с индексом **index**.
- CAN_ERRET_OBD_NOSUBIND – несуществующий субиндекс объекта.
- CAN_ERRET_OBD_READONLY – попытка записи объекта с доступом только по чтению (субиндекс 0).
- CAN_ERRET_OBD_VALRANGE – ошибка диапазона записываемого значения.
- CAN_ERRET_OBD_OBJACCESS – в текущем состоянии объект не может быть изменен.

int16 read_pdo_mapping(canindex index, cansubind subind, unsigned32 *data);

Чтение объекта PDO отображения DLL мастера.

Параметры:
- **index** – индекс объекта PDO отображения.
- **subind** – субиндекс объекта PDO отображения.
- ***data** – значение объекта.

Возвращаемые значения: нормальное завершение = 0; ошибка < 0.
- CAN_RETOK – нормальное завершение.
- CAN_ERRET_OBD_NOOBJECT – не существует объекта PDO отображения с индексом **index**.
- CAN_ERRET_OBD_NOSUBIND – несуществующий субиндекс объекта.

int16 write_pdo_mapping(canindex index, cansubind subind, unsigned32 data);

Запись объекта PDO отображения DLL мастера.

Параметры:
- **index** – индекс объекта PDO отображения.
- **subind** – субиндекс объекта PDO отображения.

- **data** – значение объекта.

Возвращаемые значения: нормальное завершение = 0; ошибка < 0.

- CAN_RETOK – нормальное завершение.
- CAN_ERRET_OBD_NOOBJECT – не существует объекта PDO отображения с индексом **index**.
- CAN_ERRET_OBD_NOSUBIND – несуществующий субиндекс объекта.
- CAN_ERRET_OBD_OBJACCESS – в текущем состоянии объект не может быть изменен.
- CAN_ERRET_PDO_NOMAP – объект не может быть отображен в PDO.
- CAN_ERRET_PDO_ERRMAP – ошибка размера объекта либо превышена максимальная длина PDO (64 бита).

int16 transmit_can_pdo(canindex index);

Формирует и отсылает PDO мастера с типами передачи:

- 0 – ациклическое синхронное (заносится в FIFO для синхронной отправки);
- 254, 255 – асинхронное (отсылается немедленно);

Параметры:

- **index** – индекс коммуникационного PDO объекта.

Возвращаемые значения: нормальное завершение = 0; ошибка < 0:

- CAN_RETOK – PDO успешно отправлено CAN драйверу (типы 254, 255) или занесено в очередь на синхронную отправку (тип 0).
- CAN_ERRET_OBD_NOOBJECT – не существует PDO с индексом **index**.
- CAN_ERRET_PDO_INVALID – PDO не действителен.
- CAN_ERRET_PDO_TRTYPE – неподходящий тип передачи PDO.
- CAN_ERRET_PDO_INHIBIT – PDO находится в состоянии подавления.
- CAN_ERRET_PDO_MAP_DEACT – PDO отображение не активировано.
- CAN_ERRET_PDO_ERRMAP – ошибка длины данных PDO отображения.
- CAN_ERRET_COMM_SEND – не удалось разместить кадр в CANopen кэше.

4.8 SDO протокол.

Для обмена данными с использованием SDO протокола не требуется создавать объектный словарь CANopen устройства.

int16 read_device_object_sdo(cannode node, canindex index, cansubind subind, canbyte *data, unsigned32 *datasize);

Чтение данных прикладного объекта из CAN узла с использованием SDO upload протокола.

Параметры:

- **node** – номер CAN узла от 1 до 127.
- **index** – индекс объекта CAN узла.
- **subind** – субиндекс объекта CAN узла.
- ***data** – байтовый указатель на размещаемые данные.
- ***datasize** – максимальный размер принимаемых данных в байтах. После успешного выполнения операции чтения содержит фактическое число принятых байт данных.

Возвращаемые значения: статус SDO транзакции.

- CAN_TRANSTATE_SDO_NODE – ошибочный номер CAN узла.
- см. функцию can_sdo_client_transfer(...) модуля master_dll_client.c

int16 write_device_object_sdo(cannode node, canindex index, cansubind subind, canbyte *data, unsigned32 datasize);

Запись данных прикладного объекта в CAN узел с использованием SDO download протокола.

Параметры:
- **node** – номер CAN узла от 1 до 127.
- **index** – индекс объекта CAN узла.
- **subind** – субиндекс объекта CAN узла.
- ***data** – байтовый указатель на передаваемые данные.
- **datasize** – фактический размер передаваемых данных в байтах.

Возвращаемые значения: статус SDO транзакции.
- CAN_TRANSTATE_SDO_NODE – ошибочный номер CAN узла.
- см. функцию can_sdo_client_transfer(...) модуля master_dll_client.c

void set_sdo_timeout(unsigned32 microseconds);

Устанавливает таймаут приема данных из CAN сети для базовой SDO транзакции клиента. Значение по умолчанию задается параметром CAN_TIMEOUT_RETRIEVE.

Параметры:
- **microseconds** – таймаут приема SDO данных в микросекундах.

unsigned32 get_sdo_timeout(void);

Возвращает значение таймаута приема SDO данных из CAN сети.

Возвращаемое значение:
- таймаут приема SDO данных в микросекундах.

5. Приложения CANopen DLL мастера.

5.1 Программные эмуляторы устройств IO Remote.

Набор программ для ОС Windows осуществляет эмуляцию трех типов устройств IO Remote:

- IOremote_R2DIO_8in_8counters_8out.exe
  CAN сеть 0, скорость 500 Кбит/С, номер CAN узла 1.
  Эмулятор устройства IO Remote R2DIO-8I/8O на 8 цифровых входов и 8 выходов. Значения цифровых входов программно формируются с использованием таймера. Состояние цифровых выходов выводится на терминал одной строкой, первый бит слева.
- IOremote_R2AIO_16bit_8in.exe
  CAN сеть 0, скорость 500 Кбит/С, номер CAN узла 2.
  Эмулятор устройства IO Remote R2AIO-8I на 8 аналоговых входов. Значения аналоговых входов программно формируются с использованием таймера.
- IOremote_R2AIO_16bit_8out.exe
  CAN сеть 0, скорость 500 Кбит/С, номер CAN узла 3.
  Эмулятор устройства IO Remote R2AIO-8O на 8 аналоговых выходов. Уставки аналоговых выходов выводится на терминал одной строкой, первый канал слева.

CANopen функциональность программ полностью идентична поведению устройств IO Remote в качестве узла CAN сети. Соответствующий коммуникационный и прикладной профили описаны в документации IOremote.pdf. Программы используют нулевой канал CAN контроллера производства “Марафон”. CANopen мастер может запускаться на первом канале. Для этого должна быть сформирована CAN сеть (соединены соответствующие шины) на основе двух-канального CAN контроллера, например CAN-bus-USBnp. В качестве мастера может использоваться программа CANwise с соответствующими подгружаемыми модулями. Виртуальные инструменты LabVIEW также осуществляют поддержку устройств IO Remote.

5.2 Проект конечного приложения.

Проект конечного приложения размещается в поддиректории \User и предназначен для обслуживания устройств IO Remote R2DIO-8I/8O. Прикладная программа обеспечивает следующую функциональность:

- Осуществляет инициализацию и запуск DLL мастера с использованием параметров конечного приложения.
- Инициализирует устройство R2DIO (номер CAN узла 1) NMT командой Reset Node.
- Конфигурирует протокол сердцебиения в DLL мастере и в узле R2DIO.
- Формирует в DLL мастере объектный словарь для поддержки PDO устройства R2DIO.
- Формирует в DLL мастере PDO отображение цифровых входов и выходов устройства R2DIO. При этом выходы R2DIO отображаются на его входы.
- Переводит все PDO мастера в действительное состояние.
- Переводит CANopen узел устройства R2DIO в операционное состояние NMT командой Start Remote Node.
- Осуществляет циклическую передачу в CAN сеть PDO для формирования состояния выходов устройства R2DIO. Тип передачи PDO устанавливается при инициализации DLL мастера и по умолчанию равен 255.
- Осуществляет циклический контроль и вывод на терминал сообщений регистратора.

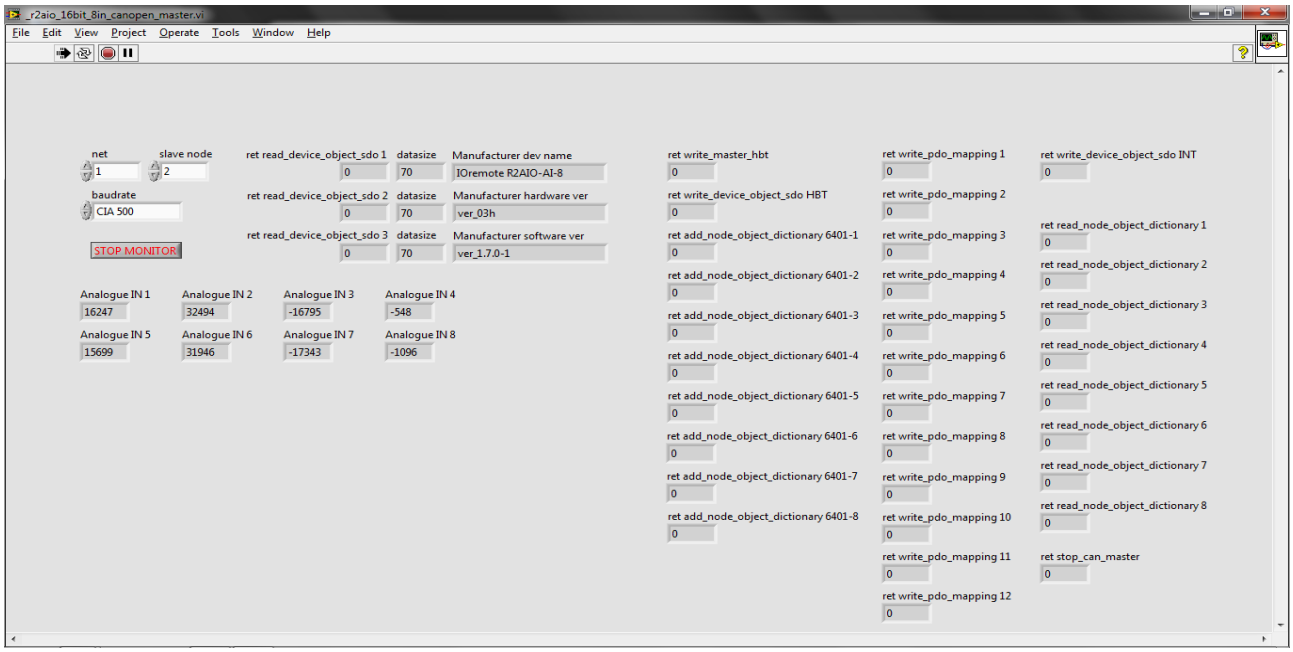
5.3 Приложение для среды LabVIEW.

Для запуска приложений LabVIEW нужно записать DLL файл CANopen мастера canopen_dll_commander_x%%.dll в директорию \resource среды LabVIEW.

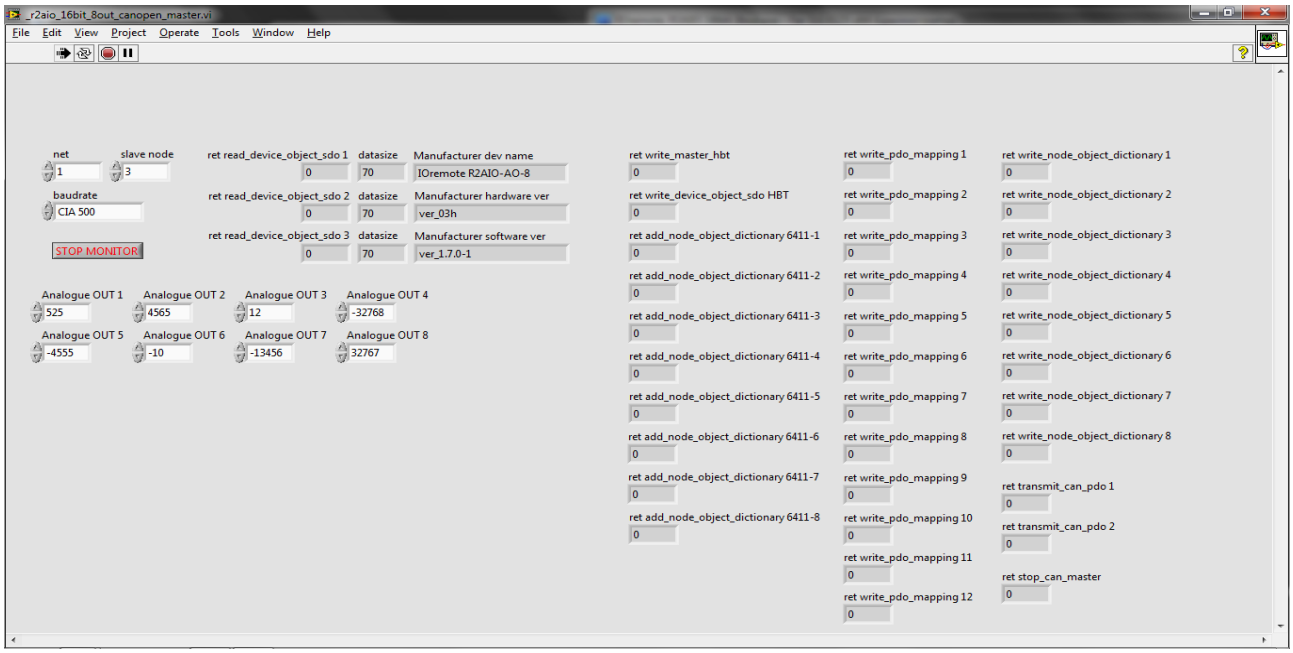
Приложение для среды программирования LabVIEW выполнено в виде набора файлов виртуальных инструментов (VI файлы). В директории \LabVIEW\CANopen_VI размещены VI файлы, которые реализуют отдельные функции, экспортируемые DLL мастером (см. раздел 4). Названия VI файлов совпадают с именами соответствующих функций DLL мастера.

Виртуальный прибор _r2dio_8in_8out_canopen_master.vi осуществляет поддержку устройства IO Remote R2DIO-8I/8O на 8 цифровых входов и 8 выходов:

Виртуальный прибор _r2aio_16bit_8in_canopen_master.vi осуществляет поддержку устройства IO Remote R2AIO-8I на 8 аналоговых входов:



Виртуальный прибор _r2aio_16bit_8out_canopen_master.vi осуществляет поддержку устройства IO Remote R2AIO-8O на 8 аналоговых выходов:



Для запуска виртуальных приборов нужно выбрать канал CAN контроллера (поле net), задать скорость CAN сети (поле baudrate) и определить номер CAN узла устройства (поле slave node). Далее инструмент может быть запущен в работу, навигация Operate→Run. Останов виртуальных приборов следует производить кнопкой STOP MONITOR, которая вызывает функцию DLL мастера stop_can_master().

6. CANopen модули DLL мастера.

Модули CANopen DLL мастера размещаются в корневой директории CANopen_WinDLL_Commander\DLL_src и поддиректориях DLL_src\CANopen и DLL_src\application. В разделе приведен API всех видимых функций мастера, за исключением экспортируемых, которые описаны в разделе 4.

6.1 Модуль can_master_system_winlib.c.

Размещается в корневой директории CANopen_WinDLL_Commander\DLL_src. Содержит системно-зависимые функции.

void can_sleep(int32 microseconds);

Функция временн*о*й задержки.

Параметры:

- **microseconds** – временн*а*я задержка в микросекундах. Точное время задержки определяется разрешением соответствующего таймера системы. Любое положительное значение аргумента функции должно обеспечивать не нулевую задержку.

void can_init_system_timer(void (*handler)(void));

Инициализация CANopen таймера.

Период таймера в микросекундах задается константой CAN_TIMERUSEC. Поток CANopen таймера должен обладать высоким приоритетом. Поскольку метод работы диспетчера ОС может не гарантировать непрерывного выполнения этого потока, код обработчика таймера формируется как единая критическая секция.

Параметры:

- **handler** – функция обработчика таймера, имеет прототип void canopen_timer(void).

void can_cancel_system_timer(void);

Отмена CANopen таймера. Прекращает либо завершает работу таймера.

void init_critical(void);

Функция инициализации критической секции.

Внедряется в код с помощью макроса CAN_CRITICAL_INIT, определенного в модуле master_dll_macros.h.

void enter_critical(void);
void leave_critical(void);

Функции входа и выхода из критической секции.

Служат для обеспечения атомарности семафорных операций и непрерывности сегментов кода при использовании мастера в многопоточной среде, когда CANopen таймер и обработчик CAN кадров запускаются как отдельные потоки (нити). Функции должны обеспечивать многократный (вложенный) вход и выход из критической секции. Функции внедряются в код с помощью макросов CAN_CRITICAL_BEGIN и CAN_CRITICAL_END, определенных в модуле master_dll_macros.h.

void enable_can_transmitter(void);
void disable_can_transmitter(void);

Функции разрешения работы и блокировки передающего CAN трансивера.

Служат для исключения выдачи CAN контроллером в сеть ложных сигналов при включении питания.

6.2 Модуль master_dll_backinit.c

Реализует функции инициализации CANopen мастера. Формирует и поддерживает диспетчер таймера и CANopen монитор. Функции модуля описаны в разделе 4.1.

6.3 Модуль master_dll_canid.c

Формирует таблицу соответствия индексов объектного словаря и CAN идентификаторов (CAN-ID). В DLL мастере эта таблица реализована в виде массива, в котором размещаются все возможные значения CAN идентификаторов. Осуществляет проверку идентификаторов ограниченного использования.

int16 correct_recv_canid(canindex index, canlink canid);

Формирует и поддерживает таблицу соответствия индексов коммуникационных CANopen объектов и CAN идентификаторов канального уровня.

Параметры:

- **index** – индекс коммуникационного объекта.
- **canid** – CAN идентификатор канального уровня.

Возвращаемые значения: нормальное завершение = 0; ошибка < 0.

- CAN_RETOK – успешное завершение.
- CAN_ERRET_OUTOFMEM – нет места для размещения записи таблицы.

canindex find_comm_recv_index(canlink canid);

Возвращает индекс коммуникационного объекта CANopen, который соответствует CAN идентификатору канального уровня. При отсутствии соответствующего объекта возвращается значение CAN_INDEX_DUMMY.

Параметры:

- **canid** – CAN идентификатор канального уровня.

Возвращаемые значения:

- индекс коммуникационного CANopen объекта.

unsigned8 check_canid_restricted(canlink canid);

Определяет принадлежность CAN идентификатора канального уровня к идентификаторам ограниченного использования.

Параметры:

- **canid** – проверяемый CAN идентификатор канального уровня.

Возвращаемые значения:

- RESTRICTED – **canid** является идентификатором ограниченного использования.
- UN_RESTRICTED – **canid** не относится к идентификаторам ограниченного использования.

void can_init_recv_canids(void);

Инициализирует таблицу соответствия индексов коммуникационных CANopen объектов и CAN идентификаторов канального уровня.

6.4 Модуль master_dll_client.c

Реализует функции SDO протокола клиента.

void can_sdo_client_transfer(struct sdocltappl *ca);

Выполняет полную транзакцию передачи данных между клиентом и сервером с использованием SDO протокола. Поддерживает возможность передачи данных переменного размера. Режимы проведения транзакции, ее условия и результаты содержатся в структуре

***ca**.

Параметры:

- **ca.operation** – определяет базовый режим передачи SDO. Задается пользователем и модифицируется функцией.
  CAN_SDOPER_DOWNLOAD – передача данных от клиента серверу (download),
  CAN_SDOPER_UPLOAD – передача данных от сервера клиенту (upload).
  Если размер данных не превышает 4 байта, используется ускоренный (expedited) режим передачи. При размере данных более 4 байт, применяется сегментированный (segmented) SDO протокол. После выполнения функции параметр **ca.operation** содержит код режима, фактически использованного при SDO обмене:
  CAN_SDOPER_(UP/DOWN)_EXPEDITED – ускоренный режим или
  CAN_SDOPER_(UP/DOWN)_SEGMENTED – сегментированный режим.
- **ca.datasize** – размер в байтах прикладных данных, передаваемых посредством SDO. Задается пользователем и модифицируется функцией для upload протокола. При передаче данных серверу (download) определяет фактический размер прикладных данных. В случае передачи данных от сервера клиенту (upload) задает максимально возможный размер данных. При успешном завершении upload операции содержит фактическое число принятых клиентом байт данных. Нулевое значение данного параметра не допустимо.
- **ca.datapnt** – указатель на локальный буфер прикладных данных. Задается пользователем. Значение NULL для данного параметра не допустимо.
- **ca.si** – определяет индекс и суб-индекс прикладного CANopen объекта для SDO протокола (мультиплексор SDO протокола). Задается пользователем.
- **ca.ss** – структура статуса транзакции. Устанавливается функцией и содержит код завершения SDO транзакции клиента **ca.ss.state**:
  CAN_TRANSTATE_OK – успешное завершение SDO транзакции.
  CAN_TRANSTATE_SDO_TOGGLE – ошибка мерцающего бита (toggle) в протоколе сегментированной передачи;
  CAN_TRANSTATE_SDO_DATASIZE – неверное значение размера данных;
  CAN_TRANSTATE_SDO_MPX – несоответствие мультиплексоров клиента и сервера;
  CAN_TRANSTATE_SDO_SRVABORT – от сервера получен аборт SDO протокола. При этим поле **ca.ss.abortcode** содержит значение аборт кода.
  CAN_TRANSTATE_SDO_WRITERR – ошибка отправки SDO кадра;
  CAN_TRANSTATE_SDO_SCSERR – SDO клиент получил от сервера неверную или неизвестную команду;
  CAN_TRANSTATE_SDO_TRANS_TIMEOUT – внутренний таймаут базовой транзакции SDO клиента;
  CAN_TRANSTATE_SDO_NET_TIMEOUT – сетевой таймаут базовой транзакции SDO клиента;
  CAN_TRANSTATE_SDO_READ_TIMEOUT – таймаут чтения данных приложением;
  CAN_TRANSTATE_SDO_NOWORKB – переполнение рабочего буфера базовых транзакций SDO клиента;
  CAN_TRANSTATE_ERROR – прочая ошибка.

6.5 Модуль master_dll_cltrans.c

Обеспечивает базовый обмен данными SDO протокола: запрос клиента, прием и обработка ответа сервера.

void can_client_sdo(canframe *cf);

Производит обработку принятого из CAN сети SDO ответа сервера.

Параметры:

- ***cf** – принятый кадр SDO протокола (ответ сервера на запрос клиента).

void can_client_basic(struct sdoclttrans *ct);

Формирует и проводит базовую SDO транзакцию клиента (запрос клиента, прием и обработка ответа сервера).

Параметры:

- ***ct** – структура поддержки базовой SDO транзакции клиента.

void can_client_control(void);

Контролирует таймаут базовой SDO транзакции клиента (запрос клиента и ответ сервера). Вызывается из CANopen таймера.

void can_init_client(void);

Инициализирует данные модуля.

6.6 Модуль master_dll_globals.c

В модуле определены внешние (глобальные) переменные и структуры данных CANopen DLL мастера.

6.7 Модуль master_dll_inout.c

Осуществляет прием и передачу CAN кадров канального уровня. Производит первичный разбор идентификаторов принимаемых кадров.

void push_all_can_data(void);

Пересылает CAN драйверу накопленные в CANopen кэше кадры канального уровня с целью дальнейшего вывода в CAN сеть. Для гарантированного вывода всех данных из кэша также вызывается из CANopen таймера. Функция является сигналобезопасной.

Замечание.

Использование кэша может привести к тому, что кадры будут выводиться в CAN сеть в последовательности, отличной от очередности их записи со стороны приложения.

int16 send_can_data(canframe *cf, unsigned16 priority);

Размещает в CANopen кэше кадр канального уровня. Осуществляет пересылку CAN драйверу всех накопленных в кэше кадров. Функция является сигналобезопасной.

Параметры:

- ***cf** – кадр, предназначенный для пересылки CAN драйверу.
- **priority** – программный приоритет CAN кадра.

Возвращаемые значения: нормальное завершение = 0; ошибка < 0.

- CAN_RETOK – данные размещены в CANopen кэше. Для вызывающих функций означает успешное завершение отправки кадра в CAN сеть.
- CAN_ERRET_COMM_SEND – не удалось разместить кадр в CANopen кэше.

void can_read_handler(canev ev);

Обработчик сигналов приема кадров канального уровня из CAN сети. Функция является сигналобезопасной и обеспечивает чтение кадров, поступивших в буфер драйвера как до выдачи сигнала, так и принятых в процессе его обработки.

void can_init_io(void);

Инициализирует семафоры и другие данные модуля.

6.8 Модуль master_dll_lib.c

Функции общего применения.

int16 check_node_id(cannode node);

Проверка номера CAN узла.

Параметры:

- **node** – проверяемый номер CAN узла.

Возвращаемые значения: нормальное завершение = 0; ошибка < 0.

- CAN_RETOK – допустимый номер CAN узла от 1 до 127.
- CAN_ERRET_NODEID – ошибочный номер CAN узла.

int16 check_bitrate_index(unsigned8 br);

Проверка индекса битовой скорости CAN сети.

Параметры:

- **br** – проверяемый индекс битовой скорости.

Возвращаемые значения: нормальное завершение = 0; ошибка < 0.

- CAN_RETOK – допустимый индекс битовой скорости.
- CAN_ERRET_BITRATE – ошибочный индекс битовой скорости.

void clear_can_data(canbyte *data);

Очистка поля данных CAN кадра (8 байт).

Параметры:

- **data** – поле данных CAN кадра.

void u16_to_canframe(unsigned16 ud, canbyte *data);

Преобразование данных типа unsigned16 в байтовый (сетевой) формат.

Параметры:

- **ud** – преобразуемое данное.
- ***data** – байтовый указатель на преобразованные данные.

unsigned16 canframe_to_u16(canbyte *data);

Преобразование данных из байтового (сетевого) формата в тип unsigned16.

Параметры:

- ***data** – байтовый указатель на преобразуемые данные.

Возвращаемое значение:

- данные типа unsigned16.

void u32_to_canframe(unsigned32 ud, canbyte *data);

Преобразование данных типа unsigned32 в байтовый (сетевой) формат.

Параметры:

- **ud** – преобразуемое данное.
- ***data** – байтовый указатель на преобразованные данные.

unsigned32 canframe_to_u32(canbyte *data);

Преобразование данных из байтового (сетевого) формата в тип unsigned32.

Параметры:

- ***data** – байтовый указатель на преобразуемые данные.

Возвращаемое значение:

- данные типа unsigned32.

6.9 Модуль master_dll_obdsdo_client.c

Поддерживает объектный словарь SDO параметров клиента, который используется для динамического формирования CAN идентификаторов SDO протокола.

int16 find_sdo_client_recv_canid(canlink *canid);

Выдает идентификатор принимаемого (от сервера клиенту) CAN кадра SDO протокола.

Параметры:

- ***canid** – CAN идентификатор канального уровня SDO протокола. Его значение определено только если SDO действителен.

Возвращаемые значения: нормальное завершение = 0; ошибка < 0.

- CAN_RETOK – SDO действителен.
- CAN_ERRET_SDO_INVALID – SDO не действителен.

int16 find_sdo_client_send_canid(canlink *canid);

Выдает идентификатор передаваемого (от клиента серверу) CAN кадра SDO протокола.

Параметры:

- ***canid** – CAN идентификатор канального уровня SDO протокола. Его значение определено только если SDO действителен.

Возвращаемые значения: нормальное завершение = 0; ошибка < 0.

- CAN_RETOK – SDO действителен.
- CAN_ERRET_SDO_INVALID – SDO не действителен.

int16 read_sdo_client_data(cansubind subind, unsigned32 *data);

Осуществляет чтение из объектного словаря идентификаторов коммуникационных SDO объектов.

Параметры:

- **subind** – субиндекс SDO объекта.
- ***data** – идентификатор коммуникационного SDO объекта.

Возвращаемые значения: нормальное завершение = 0; ошибка < 0.

- CAN_RETOK – нормальное завершение.
- CAN_ERRET_OBD_NOSUBIND – несуществующий субиндекс объекта.

int16 write_sdo_client_data(cansubind subind, unsigned32 data);

Осуществляет запись в объектный словарь идентификаторов коммуникационных SDO объектов.

Параметры:

- **subind** – субиндекс SDO объекта.
- **data** – идентификатор коммуникационного SDO объекта.

Возвращаемые значения: нормальное завершение = 0; ошибка < 0.

- CAN_RETOK – нормальное завершение.
- CAN_ERRET_OBD_NOSUBIND – несуществующий субиндекс объекта.
- CAN_ERRET_OBD_READONLY – попытка записи объекта с доступом только по чтению (субиндекс 0).
- CAN_ERRET_OBD_VALRANGE – ошибка диапазона записываемого значения.
- CAN_ERRET_OBD_OBJACCESS – в текущем состоянии объект не может быть изменен.

void can_init_sdo_client(void);

Инициализирует данные модуля.

6.10 Модуль master_dll_obj_deftype .c

Формирует словарь объектов определения типов данных. Поддерживают объекты, заданные индексами:
0002_h - INTEGER8; 0005_h - UNSIGNED8.
0003_h - INTEGER16; 0006_h - UNSIGNED16.
0004_h - INTEGER32; 0007_h - UNSIGNED32.

int16 get_deftype_bytes_objsize(canindex index, cansubind subind);

Возвращает размера объекта в байтах.

Параметры:

- **index** – индекс объекта.
- **subind** – субиндекс объекта.

Возвращаемые значения: размер объекта > 0; ошибка < 0:

- > 0 – размер объекта в байтах. Определяется его типом и составляет 1, 2 или 4 байта.
- CAN_ERRET_OBD_NOOBJECT – не существует объекта с индексом **index**.
- CAN_ERRET_OBD_NOSUBIND – несуществующий субиндекс объекта.

int16 see_deftype_access(canindex index, cansubind subind);

Запрос маски доступа к записи объекта.

Параметры:

- **index** – индекс объекта.
- **subind** – субиндекс объекта.

Возвращаемые значения: маска доступа > 0; ошибка <= 0:

- CAN_MASK_ACCESS_PDO – флаг допустимости PDO отображения объекта (LSB = 1).
- CAN_MASK_ACCESS_RW – доступ по чтению и записи (LSB+1 = 1 и LSB+2 = 1).
- CAN_ERRET_OBD_NOOBJECT – не существует объекта с индексом **index**.
- CAN_ERRET_OBD_NOSUBIND – несуществующий субиндекс объекта.

int16 get_deftype_objtype(canindex index, cansubind subind);

Запрос типа объекта.

Параметры:

- **index** – индекс объекта.
- **subind** – субиндекс объекта.

Возвращаемые значения: индекс типа объекта > 0; ошибка <= 0:

- > 0 – индекс типа объекта (0002_h..0007_h).
- CAN_ERRET_OBD_NOOBJECT – не существует объекта с индексом **index**.
- CAN_ERRET_OBD_NOSUBIND – несуществующий субиндекс объекта.

int16 read_deftype_objdict(canindex index, cansubind subind, canbyte *data);

Чтение значения объекта типа данных.

Параметры:

- **index** – индекс объекта.
- **subind** – субиндекс объекта.
- ***data** – байтовый указатель на размещаемые данные.

Возвращаемые значения: нормальное завершение = 0; ошибка < 0.

- CAN_RETOK – нормальное завершение (возвращает значение ноль).
- CAN_ERRET_OBD_NOOBJECT – не существует объекта с индексом **index**.
- CAN_ERRET_OBD_NOSUBIND – несуществующий субиндекс объекта.

int16 write_deftype_objdict(canindex index, cansubind subind, canbyte *data);

Запись значения объекта типа данных.

- **index** – индекс объекта.

- **subind** – субиндекс объекта.
- ***data** – байтовый указатель на размещенные данные.

Возвращаемые значения: нормальное завершение = 0; ошибка < 0.

- CAN_RETOK – нормальное завершение (без каких-либо последствий).
- CAN_ERRET_OBD_NOOBJECT – не существует объекта с индексом **index**.
- CAN_ERRET_OBD_NOSUBIND – несуществующий субиндекс объекта.

6.11 Модуль master_dll_obj_sync.c

Формирует и поддерживает объекты синхронизации SYNC.

int16 find_sync_recv_canid(canlink *canid);

Выдает идентификатор CAN кадра SYNC протокола.

Параметры:

- ***canid** – CAN идентификатор канального уровня SYNC протокола.

Возвращаемые значения: нормальное завершение = 0; ошибка < 0.

- CAN_RETOK – нормальное завершение.

unsigned8 sync_window_expired(void);

Определяет состояние (истечение времени) окна синхронизации.

Возвращаемые значения:

- FALSE – окно синхронизации открыто, можно проводить синхронные операции.
- TRUE – окно синхронизации истекло, синхронные операции запрещены.

void sync_received(canframe *cf);

Производит обработку принятого из CAN сети SYNC объекта. Если устройство является источником SYNC, функция вызывается при каждой передаче SYNC кадра.

Параметры:

- ***cf** – принятый или переданный CAN кадр, содержащий объект синхронизации SYNC.

void control_sync(void);

Осуществляет управление объектом синхронизации SYNC. Вызывается из CANopen таймера.

void can_init_sync(void);

Инициализирует данные модуля.

6.12 Модуль master_dll_pdo_map.c

Формирует и поддерживает динамическое байт-ориентированное PDO отображение DLL мастера.

int16 check_pdo_map_object(canindex index);

Осуществляет проверку наличия и состояния объекта PDO отображения.

Параметры:

- **index** – индекс объекта PDO отображения.

Возвращаемые значения: нормальное завершение = 0; ошибка < 0.

- CAN_RETOK – объект PDO отображения существует и активирован.
- CAN_ERRET_OBD_NOOBJECT – не существует PDO отображения с индексом **index**.
- CAN_ERRET_PDO_MAP_DEACT – PDO отображение не активировано.

int16 get_pdo_map_bytes_objsize(canindex index, cansubind subind);

Возвращает размер объекта PDO отображения в байтах.

Параметры:

- **index** – индекс объекта PDO отображения.
- **subind** – субиндекс объекта.

Возвращаемые значения: размер объекта > 0; ошибка < 0:

- > 0 – размер объекта в байтах.
- CAN_ERRET_OBD_NOOBJECT – не существует объекта с индексом **index**.
- CAN_ERRET_OBD_NOSUBIND – несуществующий субиндекс объекта.

int16 see_pdo_map_access(canindex index, cansubind subind);

Возвращает маску доступа к записи объекта PDO отображения.

Параметры:

- **index** – индекс объекта PDO отображения.
- **subind** – субиндекс объекта.

Возвращаемые значения: маска доступа > 0; ошибка <= 0:

- CAN_MASK_ACCESS_RW – доступ по чтению и записи (LSB+1 = 1 и LSB+2 = 1).
- CAN_ERRET_OBD_NOOBJECT – не существует объекта с индексом **index**.
- CAN_ERRET_OBD_NOSUBIND – несуществующий субиндекс объекта.

int16 get_pdo_map_objtype(canindex index, cansubind subind);

Возвращает индекс типа для объекта PDO отображения.

Параметры:

- **index** – индекс объекта PDO отображения.
- **subind** – субиндекс объекта.

Возвращаемые значения: индекс типа объекта > 0; ошибка <= 0:

- > 0 – индекс типа объекта.
- CAN_ERRET_OBD_NOOBJECT – не существует объекта с индексом **index**.
- CAN_ERRET_OBD_NOSUBIND – несуществующий субиндекс объекта.

int16 map_pdo(canindex index, canframe *cf);

Формирует PDO отображение, соответствующее коммуникационному PDO объекту и заносит его в поле данных CAN кадра. Определяет и устанавливает длину поля данных.

Параметры:

- **index** – индекс коммуникационного PDO объекта (как правило, передаваемого PDO).
- ***cf** – CAN кадр, в который заносятся объекты PDO отображения.

Возвращаемые значения: нормальное завершение = 0; ошибка < 0.

- CAN_RETOK – PDO сформировано успешно.
- CAN_ERRET_OBD_NOOBJECT – не существует коммуникационного PDO объекта с индексом **index**, либо соответствующего ему PDO отображения.
- CAN_ERRET_PDO_INVALID – PDO объект не действителен.
- CAN_ERRET_PDO_MAP_DEACT – PDO отображение не активировано.
- CAN_ERRET_PDO_ERRMAP – ошибка длины данных PDO отображения.
- Функция также может возвращать ошибки чтения отображаемых прикладных объектов.

int16 activate_pdo(canindex index, canframe *cf);

Осуществляет разбор CAN кадра, руководствуясь PDO отображением, которое соответствует коммуникационному PDO объекту. Заносит извлеченные из кадра данные в объектный словарь.

Параметры:

- **index** – индекс коммуникационного PDO объекта (как правило, принимаемого PDO).
- ***cf** – CAN кадр, из которого извлекаются отображенные объекты.

Возвращаемые значения: нормальное завершение = 0; ошибка < 0.

- CAN_RETOK – Все отображенные в PDO объекты успешно занесены в словарь.
- CAN_ERRET_OBD_NOOBJECT – не существует коммуникационного PDO объекта с индексом **index**, либо соответствующего ему PDO отображения.

- CAN_ERRET_PDO_INVALID – PDO объект не действителен.
- CAN_ERRET_PDO_MAP_DEACT – PDO отображение не активировано.
- CAN_ERRET_PDO_ERRMAP – ошибка длины данных PDO отображения или принятого CAN кадра.
- Функция также может возвращать ошибки записи отображаемых прикладных объектов.

void can_init_pdo_map(void);

Инициализирует данные модуля.

6.13 Модуль master_dll_pdo_obd.c

Формирует и поддерживает коммуникационные PDO объекты DLL мастера.

int16 check_pdo_comm_object(canindex index);

Определяет наличие и состояние коммуникационного PDO объекта.

Параметры:

- **index** – индекс коммуникационного PDO объекта.

Возвращаемые значения: нормальное завершение = 0; ошибка < 0.

- CAN_RETOK – PDO существует и действителен.
- CAN_ERRET_OBD_NOOBJECT – PDO с индексом **index** не существует.
- CAN_ERRET_PDO_INVALID – PDO объект не действителен.

int16 find_pdo_recv_trantype(canindex index, unsigned8 *trtype);

Выдает тип передачи принимаемого мастером PDO (субиндекс 2 коммуникационного объекта).

Параметры:

- **index** – индекс коммуникационного объекта PDO.
- ***trtype** – тип передачи PDO.

Возвращаемые значения: нормальное завершение = 0; ошибка < 0.

- CAN_RETOK – PDO существует и действителен.
- CAN_ERRET_OBD_NOOBJECT – PDO с индексом **index** не существует.
- CAN_ERRET_PDO_INVALID – PDO не действителен.

int16 find_pdo_tran_trantype(canindex index, unsigned8 *trtype);

Выдает тип передачи передаваемого мастером PDO (субиндекс 2 коммуникационного объекта).

Параметры:

- **index** – индекс коммуникационного объекта PDO.
- ***trtype** – тип передачи PDO.

Возвращаемые значения: нормальное завершение = 0; ошибка < 0.

- CAN_RETOK – PDO существует и действителен.
- CAN_ERRET_OBD_NOOBJECT – PDO с индексом **index** не существует.
- CAN_ERRET_PDO_INVALID – PDO не действителен.

int16 find_pdo_recv_canid(canindex index, canlink *canid);

Выдает идентификатор CAN кадра принимаемого мастером PDO.
Используется для формирования удаленного запроса PDO.

Параметры:

- **index** – индекс коммуникационного объекта PDO.
- ***canid** – CAN идентификатор канального уровня PDO.

Возвращаемые значения: нормальное завершение = 0; ошибка < 0.

- CAN_RETOK – PDO существует и действителен.
- CAN_ERRET_OBD_NOOBJECT – PDO с индексом **index** не существует.

- CAN_ERRET_PDO_INVALID – PDO не действителен.
- CAN_ERRET_PDO_NORTR – удаленный запрос для PDO запрещен.
  Замечание.
  В последних версиях стандарта CiA 301 (4.2.0.xx) бит удаленного запроса для принимаемых PDO не поддерживается (зарезервирован). Это не позволяет локально определить возможность формирования удаленного запроса для таких PDO.

int16 find_pdo_tran_canid(canindex index, canlink *canid);

Выдает идентификатор CAN кадра передаваемого мастером PDO.

Параметры:

- **index** – индекс коммуникационного объекта PDO.
- ***canid** – CAN идентификатор канального уровня PDO.

Возвращаемые значения: нормальное завершение = 0; ошибка < 0.

- CAN_RETOK – PDO существует и действителен.
- CAN_ERRET_OBD_NOOBJECT – PDO с индексом **index** не существует.
- CAN_ERRET_PDO_INVALID – PDO не действителен.

void find_pdo_rtr_tran_index(canlink canid, canindex *index);

Выдает индекс коммуникационного PDO объекта для CAN идентификатора **canid**. Используется для поиска передаваемого PDO, соответствующего удаленному запросу.

Параметры:

- **canid** – CAN идентификатор канального уровня PDO.
- ***index** – индекс коммуникационного объекта PDO.
  CAN_INDEX_DUMMY, если соответствующий PDO не обнаружен или не действителен или удаленный запрос для него запрещен.

int16 set_pdo_state(canindex index, unsigned8 state);

Устанавливает состояние PDO действителен / не действителен.

Параметры:

- **index** – индекс коммуникационного объекта PDO.
- **state** – новое состояние PDO (VALID / NOT_VALID).

Возвращаемые значения: нормальное завершение = 0; ошибка < 0.

- CAN_RETOK – установлено новое состояние PDO.
- CAN_ERRET_OBD_NOOBJECT – PDO с индексом **index** не существует.
- CAN_ERRET_PDO_MAP_DEACT – PDO отображение не активировано (если **state** = VALID).
- CAN_ERRET_OUTOFMEM – нет места для размещения записи в таблице CAN идентификаторов канального уровня.

void set_pdo_recv_event_timer(canindex index);

Установ таймера события принимаемого мастером PDO.

Параметры:

- **index** – индекс коммуникационного объекта PDO.

void set_pdo_tran_event_timer(canindex index);

Установ таймера события передаваемого мастером PDO.

Параметры:

- **index** – индекс коммуникационного объекта PDO.

int16 test_cyclic_tpdo(canindex index, unsigned8 sc);

Определяет состояние циклических синхронных PDO, которые передаются мастером.

Параметры:

- **index** – индекс коммуникационного объекта PDO.
- **sc** – текущее значение SYNC счетчика.

Возвращаемые значения: нормальное завершение = 0; ошибка < 0.

- CAN_RETOK – циклическое синхронное PDO готово к передаче.
- CAN_ERRET_OBD_NOOBJECT – PDO с индексом **index** не существует.
- CAN_ERRET_PDO_INVALID – PDO не действителен.
- CAN_ERRET_PDO_TRTYPE – неподходящий тип передачи PDO.
- CAN_ERRET_PDO_TRIGGER – момент передачи PDO не наступил.

int16 test_tpdo_inhibit(canindex index);

Определяет состояние подавления передаваемого мастером PDO.

Параметры:

- **index** – индекс коммуникационного объекта PDO.

Возвращаемые значения: нормальное завершение = 0; ошибка < 0:

- CAN_RETOK – время подавления PDO истекло.
- CAN_ERRET_OBD_NOOBJECT – PDO с индексом **index** не существует.
- CAN_ERRET_PDO_INVALID – PDO не действителен.
- CAN_ERRET_PDO_INHIBIT – PDO находится в состоянии подавления.

void control_pdo(void);

Осуществляет контроль таймеров события принимаемых и передаваемых PDO, а также времени подавления передаваемых PDO. Вызывается из CANopen таймера.

int16 get_pdo_comm_bytes_objsize(canindex index, cansubind subind);

Возвращает размер коммуникационного PDO объекта в байтах.

Параметры:

- **index** – индекс коммуникационного объекта PDO.
- **subind** – субиндекс объекта.

Возвращаемые значения: размер объекта > 0; ошибка < 0:

- > 0 – размер объекта в байтах.
- CAN_ERRET_OBD_NOOBJECT – не существует объекта с индексом **index**.
- CAN_ERRET_OBD_NOSUBIND – несуществующий субиндекс объекта.

int16 see_pdo_comm_access(canindex index, cansubind subind);

Возвращает маску доступа к записи коммуникационного PDO объекта.

Параметры:

- **index** – индекс коммуникационного объекта PDO.
- **subind** – субиндекс объекта.

Возвращаемые значения: маска доступа > 0; ошибка <= 0:

- CAN_MASK_ACCESS_RO – флаг доступа к объекту по чтению (LSB+1 = 1).
- CAN_MASK_ACCESS_RW – доступ по чтению и записи (LSB+1 = 1 и LSB+2 = 1).
- CAN_ERRET_OBD_NOOBJECT – не существует объекта с индексом **index**.
- CAN_ERRET_OBD_NOSUBIND – несуществующий субиндекс объекта.

int16 get_pdo_comm_objtype(canindex index, cansubind subind);

Возвращает индекс типа для коммуникационного PDO объекта.

Параметры:

- **index** – индекс коммуникационного объекта PDO.
- **subind** – субиндекс объекта.

Возвращаемые значения: индекс типа объекта > 0; ошибка <= 0:

- > 0 – индекс типа объекта.
- CAN_ERRET_OBD_NOOBJECT – не существует объекта с индексом **index**.
- CAN_ERRET_OBD_NOSUBIND – несуществующий субиндекс объекта.

void can_init_pdo(void);

Инициализирует данные модулей master_dll_pdo_obd.c, master_dll_pdo_map.c и master_dll_pdo_proc.c.

6.14 Модуль master_dll_pdo_proc.c

Содержит функции обработки PDO.

int16 pdo_remote_transmit_request(canindex index);

Формирует и отсылает удаленный запрос для PDO, заданного коммуникационным параметром **index**.

Параметры:

- **index** – индекс коммуникационного объекта PDO.

Возвращаемые значения: нормальное завершение = 0; ошибка < 0:

- CAN_RETOK – нормальное завершение.
- CAN_ERRET_OBD_NOOBJECT – не существует PDO объекта с индексом **index**.
- CAN_ERRET_PDO_INVALID – PDO не действителен.
- CAN_ERRET_PDO_NORTR – для данного PDO удаленный запрос запрещен.
- CAN_ERRET_COMM_SEND – не удалось разместить кадр в CANopen кэше.

void receive_can_pdo(canindex index, canframe *cf);

Принимает и обрабатывает PDO. Активирует асинхронные PDO (записывает данные в объектный словарь). Синхронные PDO заносятся в FIFO для последующей активации при поступлении объекта синхронизации SYNC.

Параметры:

- **index** – индекс коммуникационного объекта PDO.
- ***cf** – принятый CAN кадр, содержащий PDO.

void transmit_rtr_pdo(canindex index);

Формирует PDO, для которого получен удаленный запрос.

В зависимости от типа передачи:

- <= 252 – синхронное (заносится в FIFO для синхронной отправки);
- >= 253 – асинхронное (отсылается немедленно);

Параметры:

- **index** – индекс коммуникационного объекта PDO.

void process_sync_pdo(unsigned8 sc);

Обрабатывает синхронные PDO.

Вызывается из функции обработчика объекта синхронизации sync_received(...).

Параметры:

- **sc** – текущее значение SYNC счетчика.

void can_init_pdo_proc(void);

Инициализирует данные модуля.

6.15 Модуль master_dll_sdo_proc.c

Осуществляет прием и разборку, а также сборку и отправку SDO кадров.

void parse_sdo(struct cansdo *sd, canbyte *data);

Производит разборку поля данных CAN кадра SDO протокола.

Параметры:

- ***sd** – информация о принятом SDO кадре в разобранном виде.
- ***data** – указатель на поле данных принятого CAN кадра SDO протокола.

int16 send_can_sdo(struct cansdo *sd);

Осуществляет сборку и отсылку CAN кадра SDO протокола.

Параметры:

- ***sd** – информация об отсылаемом SDO кадре в разобранном виде.

Возвращаемые значения: нормальное завершение = 0; ошибка < 0:

- CAN_RETOK – SDO кадр успешно отправлен CAN драйверу.
- CAN_ERRET_SDO_INVALID – SDO не действительно.
- CAN_ERRET_COMM_SEND – не удалось разместить кадр в CANopen кэше.

void abort_can_sdo(struct sdoixs *si, unsigned32 abortcode);

Производит отправку кадра Abort SDO Transfer протокола.

Параметры:

- ***si** – индекс и суб-индекс словаря прикладного объекта (мультиплексор SDO протокола).
- **abortcode** – значение Abort кода.

6.16 Модуль master_dll_events.c

Содержит обработчики CAN и CANopen событий DLL мастера.

void consume_sync(unsigned8 sc);

Вызывается при получении объекта синхронизации после проверки состоятельности SYNC кадра.

Параметры:

- **sc** – текущее значение SYNC счетчика (диапазон от 1 до 240).

void no_sync_event(void);

Вызывается в случае, если потребитель SYNC не получил объекта синхронизации в течение промежутка времени, который задается объектом 1006_h (период SYNC).

void consume_controller_error(canev ev);

Обрабатывает сигналы ошибок от CAN контроллера.

Коды ошибок определены в заголовочном файле CAN драйвера канального уровня.

Параметры:

- **ev** (тип int16) – код ошибки:
  CIEV_BOFF – bus off,
  CIEV_EWL – error warning limit,
  CIEV_HOVR – hardware overrun,
  CIEV_SOVR – software overrun.
  CIEV_WTOUT – CAN write timeout.

void master_emcy(unsigned16 errorcode);

Вызывается при возникновении в мастере срочного сообщения Emergency.

EMCY объект только регистрируется, но не передается в CAN сеть.

Параметры:

- **errorcode** – код ошибки.

void master_emcy_index(unsigned16 errorcode, canindex index);

Вызывается при возникновении в мастере срочного сообщения Emergency.

Дополнительный параметр **index** записывается в поле **info** структуры **eventlog**.

Параметры:

- **errorcode** – код ошибки.
- **index** – индекс прикладного объекта.

void consume_emcy(canframe *cf);

Обрабатывает принятые из CAN сети объекты срочного сообщения Emergency.

Параметры:

- ***cf** – CAN кадр объекта срочного сообщения EMCY.

void no_pdo_event(canindex index);

Не получено PDO до истечения его таймера события.

Параметры:

- **index** – индекс коммуникационного объекта PDO.

void can_cache_overflow(void);

Вызывается при переполнении выходного CANopen кэша.

6.17 Модуль master_dll_logger.c

Реализует асинхронный регистратор DLL мастера.

void flush_events_cache(void);

Пересылает в FIFO регистратора накопленные в кэше события. Для гарантированного вывода всех данных из кэша регистратора функция дополнительно вызывается из CANopen таймера. Функция является сигналобезопасной.

void log_event(struct eventlog *ev);

Размещает событие в кэше регистратора и осуществляет попытку его вывода в FIFO. Функция является сигналобезопасной.

Параметры:

- ***ev** – зарегистрированное событие.

void master_event(unsigned8 cls, unsigned8 type, int16 code, int32 info);

Функция прикладного интерфейса для записи событий мастера.

Параметры:

- **cls** – класс зарегистрированного события.
- **type** – тип события (info, warning, error и т.д.).
- **code** – код события.
- **info** – дополнительная информация о событии.

void node_event(cannode node, unsigned8 cls, unsigned8 type, int16 code, int32 info);

Функция прикладного интерфейса для записи событий CANopen узла.

Параметры:

- **node** – номер CAN узла, в котором было порождено событие.
- **cls** – класс зарегистрированного события.
- **type** – тип события (info, warning, error и т.д.).
- **code** – код события.
- **info** – дополнительная информация о событии.

void init_logger(void);

Инициализирует данные регистратора..

6.18 Модуль master_dll_nmt_commander.c

Поддерживает сетевой менеджер (NMT протоколы).

void consume_nmt(canframe *cf);

Производит обработку принятого из CAN сети кадра NMT протокола.

Параметры:

- ***cf** – принятый NMT кадр (протоколы загрузки boot-up или сердцебиения heartbeat).

void manage_master_ecp(void);

Осуществляет контроль за прохождением сердцебиения для всех активных узлов сети.

Вызывается из CANopen таймера.

void init_node_status(void);

Инициализирует данные сетевого менеджера.

6.19 Модуль master_dll_node_obd.c

Формирует объектный словарь CANopen устройств в DLL мастере.

int32 node_get_manstan_objsize(cannode node, canindex index, cansubind subind);

Возвращает размер объекта в байтах. Он определяется при добавлении объекта в словарь DLL мастера, исходя из типа данных объекта.

Параметры:

- **node** – номер CAN узла объекта.
- **index** – индекс объекта CAN узла.
- **subind** – субиндекс объекта CAN узла.

Возвращаемые значения: нормальное завершение = 0; ошибка < 0.

- CAN_RETOK – нормальное завершение.
- CAN_ERRET_OBD_NOOBJECT – не существует объекта с параметрами **node**, **index**, **subind**.
- CAN_ERRET_OBD_NOSUBIND – несуществующий субиндекс объекта.

int16 node_read_manstan_objdict(cannode node, canindex index, cansubind subind, canbyte *data);

Чтение значения объекта из словаря DLL мастера с целью формирования PDO для последующей передачи. Для выборки объекта используется бинарный поиск.

Параметры:

- **node** – номер CAN узла объекта.
- **index** – индекс объекта CAN узла.
- **subind** – субиндекс объекта CAN узла.
- ***data** – байтовый указатель на размещаемые в PDO данные.

Возвращаемые значения: нормальное завершение = 0; ошибка < 0.

- CAN_RETOK – нормальное завершение.
- CAN_ERRET_OBD_NOOBJECT – не существует объекта с параметрами **node**, **index**, **subind**.
- CAN_ERRET_OBD_NOSUBIND – несуществующий субиндекс объекта (для объектов определения типов данных).

int16 node_write_manstan_objdict(cannode node, canindex index, cansubind subind, canbyte *data);

Запись значения объекта в словарь DLL мастера на основе данных принятого PDO. Для выборки объекта используется бинарный поиск.

Параметры:

- **node** – номер CAN узла объекта.
- **index** – индекс объекта CAN узла.
- **subind** – субиндекс объекта CAN узла.
- ***data** – байтовый указатель на данные PDO, которые заносятся в словарь.

Возвращаемые значения: нормальное завершение = 0; ошибка < 0.

- CAN_RETOK – нормальное завершение.
- CAN_ERRET_OBD_NOOBJECT – не существует объекта с параметрами **node**, **index**, **subind**.
- CAN_ERRET_OBD_NOSUBIND – несуществующий субиндекс объекта (для объектов определения типов данных).

void init_node_obd(void);

Инициализирует структуры данных объектного словаря CANopen устройств.