Руководство (v3.0)

По работе с библиотекой модулей “mPCIe – CAN”, “PCIe – CAN”

Интерфейс ISO-11898
(CAN Bus)

Для библиотек версии 3.x

ОС WINDOWS

22.01.2021

ООО “НОВОМАР”

Оглавление

1. Назначение программы

Программное обеспечение «Библиотека взаимодействия CAN» (далее – библиотека) обеспечивает вспомогательный сервисный функционал при взаимодействии с PCI-устройством CAN.

Библиотека обеспечивает выполнение следующих основных задач:
- поиск присутствующих в системе устройств
- реализация сервисных функций поканально.

2. Использование библиотеки

Для получения доступа к функциям библиотеки в разрабатываемый проект необходимо подключить заголовочный файл "*libnmcan.h*". Далее, в настройках сборщика указать путь к файлу библиотеки "*libnmcan.lib*"

3. Список доступных функций

CAN_enumerate	Получение списка устройств
CAN_enumerate_free	Освобождение памяти занятой списком устройств
CAN_open_device	Открытие устройства
CAN_read_bar	Запись в адресное пространство основного контроллера
CAN_write_bar	Чтение из адресного пространства основного контроллера
CAN_read_can	Запись в адресное пространство контроллера CAN
CAN_write_can	Чтение из адресного пространства контроллера CAN
CAN_modify_can	Модификация бит в адресном пространстве контроллера CAN
CAN_d_enable	Инициализация платы в отладочном режиме
CAN_device_version	Получение версии устройства
CAN_driver_version	Получение версии драйвера
CAN_read_dma	Чтение из буфера DMA
CAN_write_txbuf	Запись в буфер отправки
CAN_send_data_now	Отправка данных из буфера
CAN_check_tx	Проверка состояния отправки
CAN_wait_tx	Ожидание отправки
CAN_remove_txreq	Очистка буфера отправки
CAN_abat	Отмена всех ожидающих транзакций
CAN_reset_device	Сброс основного контроллера
CAN_reset_channel	Сброс контроллера CAN
CAN_set_mode	Установка режима работы контроллера CAN
CAN_get_mode	Получение режима работы контроллера CAN
CAN_set_speed	Установка скорости работы
CAN_set_speed_params	Установка произвольной скорости работы
CAN_get_speed	Получение скорости работы

CAN_dma_enable	Включение DMA
CAN_dma_disable	Отключение DMA
CAN_set_oneshot	Установка режима однократной отправки
CAN_get_errors	Получение данных об ошибках
CAN_set_masks	Установка масок для получения
CAN_set_send_mode	Установка режима работы контроллера CAN (Native/FIFO)
CAN_get_fifo_count	Получение количества пакетов в буфере FIFO
CAN_write_data_to_fifo	Запись данных в буфер FIFO
CAN_set_tx_pause	Управление флагом TXPAUSE для режима FIFO
CAN_decode_buf	Декодирование пакета DMA

4. Описание методов библиотеки

Все методы, если не указано иное, возвращают код ошибки обращения к драйверу или 0 в случае успеха.

4.1. CAN_enumerate

Получение списка устройств CAN

UINT8 CAN_enumerate(CAN_device_info** devices)

CAN_device_info** devices - указатель, в который будет записан список устройств
Возвращает количество доступных устройств CAN

4.2. CAN_enumerate_free

Освобождение памяти занятой списком устройств

VOID CAN_enumerate_free(CAN_device_info** devices)

CAN_device_info** devices - указатель на список устройств

4.3. CAN_open_device

Открытие устройства

HANDLE CAN_open_device(CAN_device_info* can)

CAN_device_info* can - указатель на структуру данных об устройстве
Возвращает хэндл устройства или NULL

4.4. CAN_d_enable

Инициализация устройства в режиме отладки

VOID CAN_d_enable(HANDLE* can)

HANDLE* can - хэндл устройства

4.5. CAN_reset_device

Сброс основного контроллера

UINT32 CAN_reset_device(HANDLE* can)

HANDLE* can - хэндл устройства

4.6. CAN_reset_channel

Сброс одного из контроллеров CAN

UINT32 CAN_reset_channel(HANDLE* can, UINT8 channel)

HANDLE* can - хэндл устройства
UINT8 channel - номер канала

4.7. CAN_read_bar

Чтение данных в адресном пространства основного контроллера

UINT32 CAN_read_bar(HANDLE* can, UINT32 bar_addr, UINT32* pbuf)

HANDLE* can - хэндл устройства
UINT32 bar_addr - адрес в памяти контроллера
UINT32* pbuf - указатель по которому будет записано считанное значение

4.8. CAN_write_bar

Запись данных в адресном пространства основного контроллера

UINT32 CAN_write_bar(HANDLE* can, UINT32 bar_addr, UINT32* pbuf)

HANDLE* can - хэндл устройства
UINT32 bar_addr - адрес в памяти контроллера
UINT32* pbuf - указатель на значение для записи

4.9. CAN_read_can

Чтение данных из адресного пространства контроллера CAN

UINT32 CAN_read_can(HANDLE* can, UINT32 can_addr, UINT8 channel, UINT8* pbuf, UINT8 rsize)

HANDLE* can - хэндл устройства
UINT32 can_addr - адрес первого байта в памяти контроллера
UINT8 channel - номер контроллера
UINT8* pbuf - указатель на область памяти, в которую будут прочитаны данные
UINT8 rsize - количество байт для чтения

Память под pbuf должна быть выделена заранее

4.10. CAN_write_can

Запись данных в адресное пространство контроллера CAN

UINT32 CAN_write_can(HANDLE* can, UINT32 can_addr, UINT8 channel, UINT8* pbuf, UINT8 rsize)

HANDLE* can - хэндл устройства
UINT32 can_addr - адрес первого байта в памяти контроллера
UINT8 channel - номер контроллера
UINT8* pbuf - указатель на массив данных для записи
UINT8 rsize - количество байт для чтения

Память под pbuf должна быть выделена заранее

4.11. CAN_modify_can

Изменение бит в памяти контроллера CAN

UINT32 CAN_modify_can(HANDLE* can, UINT32 can_addr, UINT8 channel, UINT8 data, UINT8 mask)

HANDLE* can - хэндл устройства
UINT32 can_addr - адрес в памяти контроллера
UINT8 channel - номер контроллера
UINT8 data - данные для записи
UINT8 mask - маска

4.12. CAN_read_dma

Чтение из буфера DMA

UINT32 CAN_read_dma(HANDLE* can, UINT8 channel, UINT32 count, DMA_SLOT_CAN* dmabuf, UINT32 timeout)

HANDLE* can - хэндл устройства
UINT8 channel - номер контроллера
UINT32 count - количество блоков для чтения
DMA_SLOT_CAN* dmabuf - указатель на структуру данных, в которую будут записаны блоки данных
UINT32 timeout - таймаут ожидания

Память под dmabuf должна быть выделена заранее

4.13. CAN_device_version

Получение версии устройства

UINT32 CAN_device_version(HANDLE* can, CAN_DEVINFO* pdevinfo)

HANDLE* can - хэндл устройства
CAN_DEVINFO* pdevinfo - структура данных с версией устройства

Память под pdevinfo должна быть выделена заранее

4.14. CAN_driver_version

Получение версии драйвера

UINT32 CAN_driver_version(HANDLE* can, CAN_DRVINFO* pdrvinfo)

HANDLE* can - хэндл устройства
CAN_DRVINFO* pdrvinfo - структура данных с версией драйвера

Память под pdrvinfo должна быть выделена заранее

4.15. CAN_write_txbuf

Запись в буфер отправки

UINT32 CAN_write_txbuf(HANDLE* can, UINT8 channel, UINT8 nbuf, UINT8 prio, UINT32 timeout, UINT32 sid, UINT32 eid, UINT8* pdata, UINT32 datasize, BOOLEAN autorts)

HANDLE* can - хэндл устройства
UINT8 channel - номер контроллера
UINT8 nbuf - номер буфера
UINT8 prio - приоритет
UINT32 timeout - таймаут отправки
UINT32 sid - SID
UINT32 eid - EID
UINT8* pdata - указатель на буфер с данными для отправки
UINT32 datasize - размер буфера с данными
BOOLEAN autorts - автоматическая отправка после записи

4.16. CAN_send_data_now

Отправка записанных в буфер данных

UINT32 CAN_send_data_now(HANDLE* can, UINT8 channel, UINT8 nbuf)

HANDLE* can - хэндл устройства
UINT8 channel - номер контроллера
UINT8 nbuf - номер буфера

4.17. CAN_check_tx

Проверка состояния отправки

UINT32 CAN_check_tx(HANDLE* can, UINT8 channel, UINT8 nbuf, UINT8 *txbnctrl)

HANDLE* can - хэндл устройства
UINT8 channel - номер контроллера
UINT8 nbuf - номер буфера
UINT8 *txbnctrl - указатель на адрес куда будет записан регистр txbnctrl (см. документацию по программированию платы)

4.18. CAN_wait_tx

Ожидание отправки данных

UINT32 CAN_wait_tx(HANDLE* can, UINT8 channel, UINT8 nbuf, UINT32 timeout)

HANDLE* can - хэндл устройства
UINT8 channel - номер контроллера
UINT8 nbuf - номер буфера
UINT32 timeout - таймаут ожидания

4.19. CAN_remove_txreq

Отмена запроса на отправку

UINT32 CAN_remove_txreq(HANDLE* can, UINT8 channel, UINT8 nbuf)

HANDLE* can - хэндл устройства
UINT8 channel - номер контроллера
UINT8 nbuf - номер буфера

4.20. CAN_abat

Отмена всех транзакций

UINT32 CAN_abat(HANDLE* can, UINT8 channel)

HANDLE* can - хэндл устройства
UINT8 channel - номер контроллера

4.21. CAN_send_by_trigger

Программирование отправки по срабатыванию триггера

UINT32 CAN_send_by_trigger(HANDLE* can, UINT8 channel, UINT8 buffer, UINT32 epoch, UINT32 epoch_val, UINT32 ntu, UINT32 div, BOOL loop)

HANDLE* can - хэндл устройства
UINT8 channel - номер контроллера
UINT8 buffer - номер буфера
UINT32 epoch - флаг использования epoch
UINT32 epoch_val - счётчик циклов
UINT32 ntu
UINT32 div
BOOLEAN loop - программирование многократной передачи

4.22. CAN_check_trigger

Получение состояния триггера

UINT32 CAN_check_trigger(HANDLE* can, UINT8 channel, UINT8 buffer)

HANDLE* can - хэндл устройства
UINT8 channel - номер контроллера
UINT8 buffer - номер буфера

Функция читает регистр CANn_TRIG_CTRL и проверяет биты TX_TRIGm_EN.
Если биты равны 01, то триггер занят, функция вернёт значение 1.
Если биты равны 00, то триггер свободен и функция вернёт значение 0

4.23. CAN_set_mode

Установка режима работы контроллера

UINT32 CAN_set_mode(HANDLE* can, UINT8 channel, CAN_MODE mode)

HANDLE* can - хэндл устройства
UINT8 channel - номер контроллера
CAN_MODE mode - режим работы

4.24. CAN_get_mode

Получение режима работы контроллера

UINT32 CAN_get_mode(HANDLE* can, UINT8 channel, CAN_MODE *mode)

HANDLE* can - хэндл устройства
UINT8 channel - номер контроллера
CAN_MODE *mode - указатель для записи CAN_MODE

Память под mode должна быть выделена заранее

4.25. CAN_set_speed

Установка стандартной скорости работы контроллера CAN

UINT32 CAN_set_speed(HANDLE* can, UINT8 channel, UINT32 speed)

HANDLE* can - хэндл устройства
UINT8 channel - номер контроллера
UINT32 speed - скорость

4.26. CAN_set_speed_params

Установка произвольной скорости работы контроллера CAN

UINT32 CAN_set_speed_params(HANDLE* can, UINT8 channel, UINT32 *speed, UINT8 *params)

HANDLE* can - хэндл устройства
UINT8 channel - номер контроллера
UINT32 *speed - не используется
UINT8 *params - параметры скорости контроллера (см. документацию по программированию модуля)

4.27. CAN_get_speed

Получение скорости работы контроллера

UINT32 CAN_get_speed(HANDLE* can, UINT8 channel, UINT32 *speed, UINT8 *params)

HANDLE* can - хэндл устройства
UINT8 channel - номер контроллера
UINT32 *speed - указатель для записи значения стандартной скорости
UINT8 *params - указатель для записи значения параметров скороси

4.28. CAN_dma_enable

Включение DMA

UINT32 CAN_dma_enable(HANDLE* can)

HANDLE* can - хэндл устройства

4.29. CAN_dma_enable

Выключение DMA

UINT32 CAN_dma_disable (HANDLE* can)

HANDLE* can - хэндл устройства

4.30. CAN_set_oneshot

Установка режима однократной передачи

UINT32 CAN_set_oneshot(HANDLE* can, UINT8 channel, UINT8 oneshot)

HANDLE* can - хэндл устройства
UINT8 channel - номер контроллера
UINT8 oneshot - 1 для включения или 0 для выключения режима однократной передачи

4.31. CAN_get_errors

Чтение регистров ошибок CAN контроллера

UINT32 CAN_get_errors(HANDLE* can, UINT8 channel, CAN_ERROR_INFO *errorinfo)

HANDLE* can - хэндл устройства
UINT8 channel - номер контроллера
CAN_ERROR_INFO *errorinfo - указатель на структуру, в которую будут записаны регистры ошибок

Память под errorinfo должна быть выделена заранее

4.32. CAN_set_masks

Установка масок и фильтров приёма CAN контроллера.

UINT32 CAN_set_masks(HANDLE* can, UINT8 channel, UINT8 filter, UINT8 ident, UINT32 id_filter, UINT32 id_mask, UINT8 rxb)

HANDLE* can - хэндл устройства
UINT8 channel - номер контроллера
UINT8 filter -
UINT8 ident -
UINT32 id_filter -
UINT32 id_mask -
UINT8 rxb -

В зависимости от выбранного канала (**channel**) и номера фильтра (**filter**) функция записывает значение **rxb_mode** в биты RXM регистра **RXBn*CTRL****.

* n – номер буфера.
** См. раздел 6.8.1 и 6.8.2 документа "*Руководство по программированию модуля "mPCIe-CAN*".

Далее в зависимости от выбранного идентификатора (**ident**) и номера фильтра (**filter**) функция записывает значения масок и фильтров в соответствующие регистры*.

* См. раздел 6.9 документа "*Руководство по программированию модуля "mPCIe-CAN*".

4.33. CAN_set_timer_trsh

Запуск таймера с указанным периодом.

UINT32 CAN_set_timer_trsh(HANDLE* can, UINT8 channel, UINT32 nValue, UINT8 bEpoch, UINT32 nEpoch)

HANDLE* can - хэндл устройства
UINT8 channel - номер контроллера
UINT32 nValue - значение таймера
UINT8 bEpoch - битовая маска
UINT32 nEpoch - не используется

См. раздел 5.2.2 документа "*Руководство по программированию модуля "mPCIe-CAN*".
См. раздел 5.2.4 документа "*Руководство по программированию модуля "mPCIe-CAN*".

4.34. CAN_set_timer_ceed

Установка режима и значений корректировки начального значения таймера.

Значение поля **nValue** записывается в регистр **CANn*_TIMER_TRSH**** (значение записывается целиком с нулевого бита).

Биты RST, ENABLE и NTU_MODE регистра **CANn*_TIMER_CTRL***** устанавливаются в единицу, а битовая маска EPOCH_MASK устанавливается в соответствии с параметром **bEpoch**. Значения остальных битов данного регистра не изменяются.

UINT32 CAN_set_timer_ceed(HANDLE* can, UINT8 channel, UINT32 nValue, UINT8 bEpoch, UINT32 nEpoch)

HANDLE* can - хэндл устройства
UINT8 channel - номер контроллера
UINT32 nValue
UINT8 bEpoch
UINT32 nEpoch

4.35. CAN_set_timer_free

Запуск таймера в режиме свободного счёта.

UINT32 CAN_set_timer_free(HANDLE* can, UINT8 channel, UINT8 epochBits)

HANDLE* can - хэндл устройства
UINT8 channel - номер контроллера
UINT8epochBits - битовая маска

4.36. CAN_set_timer_rst_rxb

Установка/снятие режима сброса таймера по приёму сообщения в буфер CAN контроллера.

UINT32 CAN_set_timer_rst_rxb(HANDLE* can, UINT8 channel, UINT32 nValue)

HANDLE* can - хэндл устройства
UINT8 channel - номер контроллера
UINT32 nValue

В регистре **CANn*_TIMER_CTRL**** бит RST_ON_RXB0 устанавливается в соответствии со значением бита 0 параметра **nValue**, а бит RST_ON_RXB1 устанавливается в соответствии со значением бита 1 параметра **nValue**. Значения остальных битов данного регистра не изменяются.

* n – номер канала.
** См. раздел 5.2.4 документа "*Руководство по программированию модуля "mPCIe-CAN*".

4.37. CAN_stop_timer

Остановка таймера

UINT32 CAN_stop_timer(HANDLE* can, UINT8 channel)

HANDLE* can - хэндл устройства
UINT8 channel - номер контроллера

4.38. CAN_get_timer

Получение значения таймера

UINT32 CAN_get_timer(HANDLE* can, UINT8 channel, UINT32 *value, UINT32 *epoch)

HANDLE* can - хэндл устройства
UINT8 channel - номер контроллера
UINT32 *value - значение регистра CANn_TIMER
UINT32 *epoch - значение регистра CANn_TIMER_EPOCH

Память под value и epoch должна быть выделена заранее

4.39. CAN_start_timer_int

Запуск таймера прерываний
см. описание IOCTL **4.3.18 IOCTL_START_TIMER_INT** в руководстве по работе с драйвером модулей CAN

UINT32 CAN_start_timer_int(HANDLE* can, UINT8 channel, UINT8 bEpoch, UINT32 nValue)

HANDLE* can - хэндл устройства
UINT8 channel - номер контроллера
UINT8 bEpoch
UINT32 nValue

4.40. CAN_stop_timer_int

Остановка таймера прерываний

UINT32 CAN_stop_timer_int(HANDLE* can, UINT8 channel)

HANDLE* can - хэндл устройства
UINT8 channel - номер контроллера

4.41 CAN_wait_timer_int

Ожидание таймера прерываний

UINT32 CAN_wait_timer_int(HANDLE* can, UINT8 channel)

HANDLE* can - хэндл устройства
UINT8 channel - номер контроллера

4.42 CAN_set_timeouts

Установка таймаутов

UINT32 CAN_set_timeouts(HANDLE* can, UINT8 channel, UINT32 absolute, UINT32 interval)

HANDLE* can - хэндл устройства
UINT8 channel - номер контроллера
UINT32 absolute
UINT32 interval

Поле **absolute** пишется в регистр **CANn*_TIMEOUT_ABSOLUTE****, а поле **interval** пишется в регистр **CANn*_TIMEOUT_INTERVAL*****.
Значение обоих полей в микросекундах, допустимые значения - 0x0 .. 0x3FFFFFF

* n – номер канала.
** См. раздел 5.1.5 документа “*Руководство по программированию модуля “mPCIe-CAN*”.
*** См. раздел 5.1.6 документа “*Руководство по программированию модуля “mPCIe-CAN*”.

4.43 CAN_set_send_mode

Установка режима контроллера CAN

UINT32 CAN_set_send_mode(HANDLE* can, UINT8 channel, CTRL_MODE mode)

HANDLE* can - хэндл устройства
UINT8 channel - номер контроллера
CTRL_MODE mode - режим контроллера

Возможны 4 режима - CAN_NATIVE (по умолчанию), CAN_FIFO, CAN_TIMEPLAN, CAN_RSRV

В текущей ревизии драйвера поддерживаются режимы CAN_NATIVE, CAN_FIFO

4.44 CAN_get_fifo_count

Получение количества пакетов в буфере FIFO

UINT32 CAN_get_fifo_count(HANDLE* can, UINT8 channel, UINT8 isHPFIFO, UINT8 *count)

HANDLE* can - хэндл устройства
UINT8 channel - номер контроллера
UINT8 isHPFIFO - 1 - использовать буфер HPFIFO (иначе - FIFO)
UINT8* count - указатель на переменную куда будет записано количество пакетов

4.45 CAN_write_data_to_fifo

Запись данных в буфер FIFO

UINT32 CAN_write_data_to_fifo(HANDLE* can, UINT8 channel, UINT32 sid, UINT32 eid, UINT8 dlc, UINT8 *data, UINT32 dsize, UINT8 msgid, UINT8 isHPFIFO)

HANDLE* can - хэндл устройства
UINT8 channel - номер контроллера
UINT32 sid - стандартный идентификатор
UINT32 eid - расширенный идентификатор
UINT8 *data - указатель на массив с данными
UINT32 size - размер массива данных
UINT8 msgid - id сообщения
UINT8 isHPFIFO - 1 - использовать буфер HPFIFO (иначе - FIFO)

4.46 TTACN_set_tx_pause

Управление флагом TXPAUSE для FIFO

UINT32 CAN_set_tx_pause(HANDLE* can, UINT8 channel, UINT32 txpause, UINT8 isHPFIFO)

HANDLE* can - хэндл устройства
UINT8 channel - номер контроллера
UINT32 txpause - новое содержимое регистра
UINT8 isHPFIFO - 1 - использовать буфер HPFIFO (иначе - FIFO)

4.47 CAN_write_tg_fifo

Управление триггерами FIFO

UINT32 CAN_write_tg_fifo(HANDLE* can, UINT8 channel, UINT8 epoch, UINT32 div_trig, UINT32 ntu_trig)

HANDLE* can - хэндл устройства
UINT8 channel - номер контроллера
UINT8 epoch
UINT32 div_trig
UINT32 ntu_trig

4.48 CAN_decode_buf

Декодирование буфера с данными

void CAN_decode_buf(DMA_SLOT_CAN* dma, UINT32* nbuf, UINT8* channel, UINT32* tmr_epoch, UINT32* tmr_ntu, UINT32* tmr_div, UINT32* sid, UINT32* eid, UINT8* data, UINT8* datasize)

4.49 CAN_get_stats

Получение данных счётчиков

UINT32 CAN_get_stats(HANDLE* tan, UINT8 channel, UINT8 reset, CAN_CANXSTAT* canstat)

Данные счётчиков будут записаны в структуру CAN_CANXSTAT. Ненулевое значение флага reset сбросит счётчики после чтения данных.

5. Обновление руководства.

Версия документа	Дата	Изменение
1.0	12.02.2013	Документ создан
2.0	31.05.2018	Обновление до версии драйвера 2.0.
3.0	22.01.2021	Добавлен режим FIFO