МАРАФОН
CAN технология

Драйвер CHAI
для
микроконтроллеров
STM32F1*

Руководство программиста

Москва, 2022

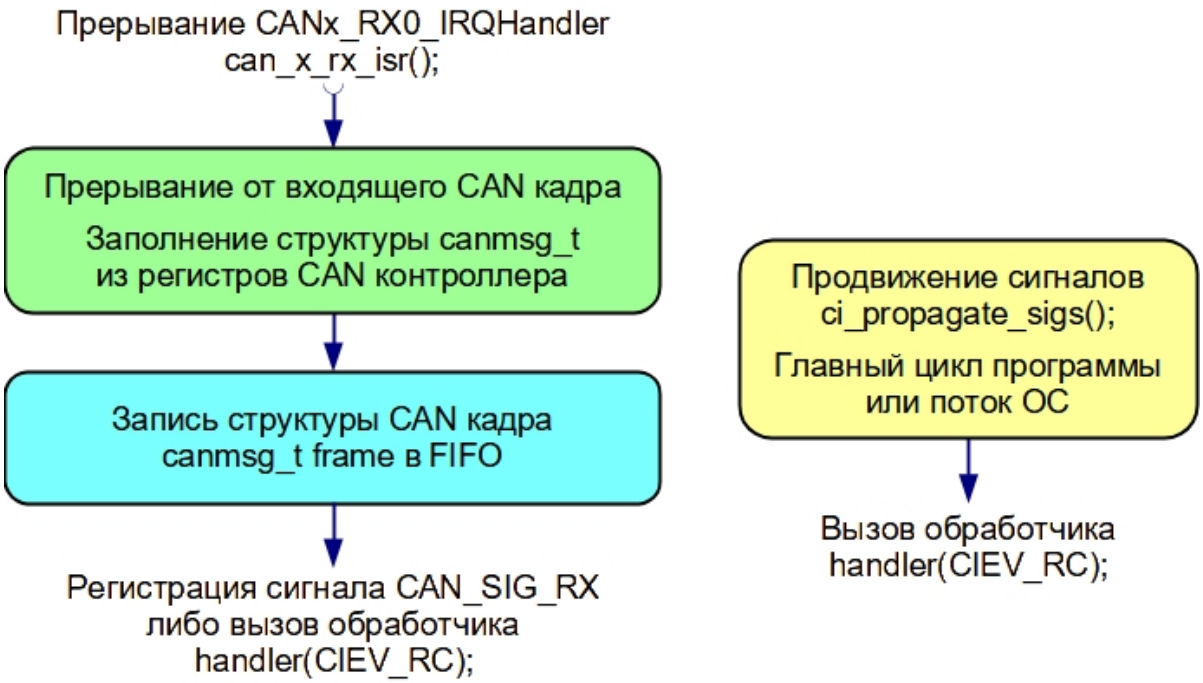
Оглавление

1. Алгоритмы работы драйвера

1.1 Обработка событий и прерываний

Драйвер поддерживает два метода обработки входящих CAN кадров.

Прямой метод осуществляет пересылку CAN кадра приложению из обработчика аппаратных прерываний микроконтроллера без дополнительной буферизации.



Метод сигналов предусматривает буферизацию принятых CAN кадров в FIFO, а также использование счетчика. Сигнал может быть передан приложению как из обработчика прерываний (для этого должен быть определен параметр #define CI_SIGS_CALLFROMISR), так и прочитан в дальнейшем из некоторого потока или главного цикла программы. Обработка ошибок CAN контроллера и самого драйвера осуществляется только методом сигналов.

Выбор метода обработки входящих CAN кадров осуществляется функциями CiSetCB(…) для метода сигналов и CiSetCB_direct_RC(…) для прямого метода.

1.2 Приемные фильтры CAN контроллера

Для фильтрации входящих сообщений по значению CAN идентификатора драйвер использует масочные приемные фильтры. Каждому каналу CAN контроллера по умолчанию сопоставляется 14 фильтров.
Настройку приемных фильтров рекомендуется проводить в состоянии останова канала CAN контроллера, которое обеспечивается вызовом функции CiStop(chan). Соответственно, после перестройки фильтров нужно будет вызвать функцию CiStart(chan).

2. Типы и структуры данных драйвера

2.1 Типы данных драйвера

Обозначение	Описание
_u8	Без-знаковое целое 8 бит.
_s8	Целое 8 бит со знаком.
_u16	Без-знаковое целое 16 бит.
_s16	Целое 16 бит со знаком.
_u32	Без-знаковое целое 32 бита.
_s32	Целое 32 бита со знаком.

2.2 Структуры данных драйвера

struct canmsg_t {
- **_u32 id** — идентификатор CAN кадра канального уровня
- **_u8 data[8]** — данные CAN кадра
- **_u8 len** — фактический размер данных (от 0 до 8)
- **_u16 flg** — битовые флаги CAN кадра. Бит 0 – RTR, бит 2 – EFF
- **_u32 ts** — временн*а*я метка получения CAN кадра в микросекундах

};

Структура **canmsg_t** или **canframe** размещает CAN кадр канального уровня.

struct canque_t {
- **canmsg_t msg[CIQUE_DEFSIZE_RC]** — FIFO CAN кадров канального уровня
- **_u16 head** — элемент FIFO записанный последним
- **_u16 tail** — самый ранний элемент FIFO

};

Структура **canque_t** определяет FIFO входящих CAN кадров канального уровня.

struct ssl_t {
- **_u16 sigcnt** — счетчик числа принятых сигналов одного типа
- **_s16 ciev** — событие CIEV_*, которое порождается данным сигналом
- **void (*hdl)(_s16 ciev)** — указатель на функцию – обработчик события

};

Структура **ssl_t** используется для поддержки сигналов одного типа.

struct slots_t {
- **ssl_t ssl[CI_SIGNUM]** — массив сигналов всех типов

};

Структура **slots_t** обеспечивает поддержку сигналов всех типов.

struct can_dev {
- **_u8 opened** — флаг занятости CAN канала FALSE или TRUE
- **_u8 state** — состояние канала CAN_STOPPED или CAN_RUNNING
- **_u8 flags** — маска обслуживания 11 и/или 29 битовых CAN идентификаторов
- **_u8 priority** — приоритет прерываний контроллера для входящих CAN кадров

_u16 tr_tout	таймаут передачи CAN кадров в миллисекундах
canque_t rcq	FIFO входящих CAN кадров
slots_t slots	используется для поддержки всех возможных сигналов драйвера
void (*dirhdl)(...)	указатель на функцию обработки входящего CAN кадра прямым методом. Если этот указатель не NULL, используется прямой метод.

};

Структура **can_dev** обеспечивает работу одного CAN канала микроконтроллера.

3. Прикладной API драйвера CHAI

_s16 CiInit(void);
Осуществляет начальную инициализацию CAN контроллера на аппаратном уровне.
Инициализирует структуры данных драйвера для всех CAN каналов.
Функция вызывается однократно при запуске CAN устройства.
Возвращаемые значения:
нормальное завершение = 0; ошибка < 0.

_s16 CiOpen(_u8 chan, _u8 flags);
Открывает канал CAN контроллера **chan**. Устанавливает аппаратные режимы канала.
Инициализирует структуры данных этого канала.
Открытый канал остается в остановленном состоянии и для его запуска нужно вызвать функцию CiStart(chan).
Параметры:
- **chan** – номер канала CAN контроллера (считаются с 0).
- **flags** – битовая маска, задает типы обрабатываемых CAN идентификаторов (11-битовые и/или 29-битовые).

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

_s16 CiClose(_u8 chan);
Закрывает канал **chan**. Запрещает прерывания, сбрасывает регистры, удаляет обработчики сигналов. Сбрасывает приемные фильтры CAN контроллера.
Последовательность вызова функций CiClose(chan) → CiOpen(chan) выполняет пере-инициализацию канала CAN контроллера.
Параметры:
- **chan** – номер канала CAN контроллера (считаются с 0).

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

_s16 CiStart(_u8 chan);
Переводит канал контроллера **chan** в рабочее состояние. При этом CAN контроллер выводится из режима инициализации и разрешаются аппаратные прерывания.
Вызов данной функции необходим для запуска канала в штатную работу.
Параметры:
- **chan** – номер канала CAN контроллера (считаются с 0).

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

_s16 CiStop(_u8 chan);
Переводит канал **chan** в состояние останова. При этом запрещаются аппаратные прерывания и CAN контроллер переводится в режим инициализации.
Параметры:
- **chan** – номер канала CAN контроллера (считаются с 0).

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

_s16 CiSetFilter(_u8 chan, _u32 acode, _u32 amask);
Устанавливает одно-уровневый масочный приемный фильтр CAN контроллера.
_s16 CiSetDualFilter(_u8 chan, _u32 acode0, _u32 amask0, _u32 acode1, _u32 amask1);

Устанавливает двух-уровневый масочный приемный фильтр CAN контроллера.
_s16 CiSetFilter_zero(_u8 chan, _u32 acode, _u32 amask);
_s16 CiSetFilter_gfs(_u8 chan, _u32 acode, _u32 amask);
_s16 CiSetFilter_odd(_u8 chan, _u32 acode, _u32 amask);
_s16 CiSetFilter_even(_u8 chan, _u32 acode, _u32 amask);
_s16 CiSetFilter_node(_u8 chan, _u32 acode, _u32 amask);
Устанавливают совокупность масочных фильтров для поддержки протокола EN50325-5.
Параметры:
- **chan** – номер канала CAN контроллера (считаются с 0).
- **acode, acode0, acode1** – требуемые значения бит для фильтров.
- **amask, amask0, amask1** – битовая маска фильтров (=1 — значение соответствующего бита **acode** учитывается, =0 — игнорируется, то есть бит может принимать любое значение).

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

_s16 CiSetBaud(_u8 chan, _u8 bt0, _u8 bt1);
Устанавливает битовую скорость CAN сети для канала контроллера **chan**,
Параметры:
- **chan** – номер канала CAN контроллера (считаются с 0).
- **bt0, bt1** – коды скорости, значения которых зависит от типа CAN контроллера и его тактовой частоты.

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

_s16 CiWrite(_u8 chan, canmsg_t *mbuf);
Записывает в буфер контроллера **chan** один кадр канального уровня.
Параметры:
- **chan** – номер канала CAN контроллера (считаются с 0).
- ***mbuf** – указатель на структуру CAN кадра канального уровня.

Возвращаемые значения:
нормальное завершение = 1 (число записанных CAN кадров);
ошибка <= 0.

_s16 CiRead(_u8 chan, canmsg_t *mbuf);
Считывает из очереди драйвера один кадр канального уровня.
Параметры:
- **chan** – номер канала CAN контроллера (считаются с 0).
- ***mbuf** – указатель на структуру CAN кадра канального уровня.

Возвращаемые значения:
нормальное завершение = 1 (число прочитанных CAN кадров);
ошибка <= 0.

_s16 CiSetCB(_u8 chan, _u8 ev, void (*ci_handler) (_s16 ciev));
Регистрирует обработчик событий для канала **chan**.
Для события приема CAN кадра CIEV_RC устанавливает метод сигналов с буферизацией.
Прямой метод обработки входящих CAN кадров при этом отключается.
Параметры:
- **chan** – номер канала CAN контроллера (считаются с 0).
- **ev** – событие для регистрации обработчика CIEV_RC, CIEV_TR или CIEV_CANERR.
- ***ci_handler** – указатель на функцию обработчика с параметром.

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

_s16 CiSetCB_direct_RC(_u8 chan, void (*ci_handler)(_u8 chan, canmsg_t *frame), _u8 priority);
Регистрирует прямой метод обработки входящих CAN кадров для канала **chan**.
Метод сигналов с буферизацией входящих CAN кадров при этом отключается.
Параметры:
- **chan** – номер канала CAN контроллера (считаются с 0).
- ***ci_handler** – указатель на функцию обработчика с параметрами.
- **priority** – приоритет прерывания CAN контроллера для входящего кадра CAN_PRIORITY_[TOP или HIGH или NORMAL].

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

void ci_propagate_sigs(void);
Пропагатор (функция распространения) сигналов драйвера.
В случае, когда асинхронная доставка входящих сигналов не предусмотрена, вызов пропагатора должен быть включен в некоторый поток либо главный цикл программы.

4. API внутренних функция драйвера

static void fifo_init(canque_t *q);
Инициализация FIFO для размещения входящих CAN кадров.
Параметры:
- ***q** – указатель на FIFO входящих кадров CAN контроллера.

static _s16 fifo_read(canque_t *q, canmsg_t *msg);
Чтение из FIFO одного CAN кадра канального уровня.
Параметры:
- ***q** – указатель на FIFO входящих кадров CAN контроллера.
- ***msg** – указатель на CAN кадр канального уровня.

Возвращаемые значения:
- 1 – прочитан самый ранний CAN кадр.
- 0 – очередь FIFO входящих CAN кадров пуста.

static _s16 fifo_write(canque_t *q, canmsg_t *msg);
Запись в FIFO одного CAN кадра канального уровня.
Параметры:
- ***q** – указатель на FIFO входящих кадров CAN контроллера.
- ***msg** – указатель на CAN кадр канального уровня.

Возвращаемые значения:
- 0 – кадр размещен в FIFO.
- -1 – размещение кадра не возможно, FIFO полон.

static void __canchip_ei(_u8 chan);
static void __canchip_di(_u8 chan);
Запрет и разрешение аппаратных прерываний в регистрах CAN контроллера.
Параметры:
- **chan** – номер канала CAN контроллера (считаются с 0).

static void __canvic_ei(_u8 chan);
static void __canvic_di(_u8 chan);
Запрет и разрешение аппаратных прерываний в контроллере прерывания NVIC.
Параметры:
- **chan** – номер канала CAN контроллера (считаются с 0).

static void ci_sigs_init(slots_t *s);
Инициализация сигналов всех типов для одного канала CAN контроллера.
Параметры:
- ***s** – указатель на структуру массива сигналов всех типов.

static void ci_signal_set(slots_t *s, _u8 sig, _s16 ciev, void (*hdl)(_s16));
Установ обработчика сигналов и соответствующих им кодов событий.
Параметры:
- ***s** – указатель на структуру массива сигналов всех типов.
- **sig** – значение (номер) сигнала CAN_SIG_*.
- **ciev** – событие, для которого устанавливается обработчик сигнала.
- ***hdl** – указатель на функцию обработчика с параметром.

static void ci_send_signal(slots_t *s, _u8 sig);
Отправка сигнала обработчику.
Параметры:
- ***s** – указатель на структуру массива сигналов всех типов.
- **sig** – значение (номер) сигнала CAN_SIG_*.

static void chai_data_init(_u8 chan);
Инициализация структур данных канала CAN контроллера.
Параметры:
- **chan** – номер канала CAN контроллера (считаются с 0).

static void can_filters_init(_u8 chan);
Инициализация приемных фильтров канала CAN контроллера. Все фильтры остаются в отключенном состоянии.
Параметры:
- **chan** – номер канала CAN контроллера (считаются с 0).

static _s16 SetFilter(_u8 chan, _u8 fnum, _u32 acode, _u32 amask);
Установ и включение приемных фильтров канала CAN контроллера. Все фильтры активируются в масочном режиме.
Параметры:
- **chan** – номер канала CAN контроллера (считаются с 0).
- **fnum** – номер приемного фильтра контроллера.
- **acode** – требуемые значения бит для фильтров.
- **amask** – битовая маска фильтров (=1 — значение соответствующего бита **acode** учитывается, =0 — игнорируется, то есть бит может принимать любое значение).

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

static void __direct_receive_chan_0(void);
Непосредственный прием, сборка и передача приложению входящих CAN кадров для канала 0 (CAN1).

static void __direct_receive_chan_1(void);
Непосредственный прием, сборка и передача приложению входящих CAN кадров для канала 1 (CAN2).

static void __chai_receive(_u8 chan);
Прием, сборка и передача приложению входящих CAN кадров методом сигналов.
Параметры:
- **chan** – номер канала CAN контроллера (считаются с 0).

void can_1_rx_isr(void);
Обработчик прерывания по приему CAN кадра первого контроллера CAN1_RX0_IRQHandler.

void can_2_rx_isr(void);
Обработчик прерывания по приему CAN кадра второго контроллера CAN2_RX0_IRQHandler.

static void can_errs_isr(_u8 chan);
Прием и передача приложению сигналов ошибок.
Параметры:
- **chan** – номер канала CAN контроллера (считаются с 0).

void can_1_errs_isr(void);
Обработчик прерывания по ошибке первого контроллера CAN1_SCE_IRQHandler.

void can_2_errs_isr(void);
Обработчик прерывания по ошибке второго контроллера CAN2_SCE_IRQHandler.