Регистровый состав и основные принципы работы с модулем CAN микроконтроллера PIC18F458

Примечание: В данной статье приводится настройка модуля CAN микроконтроллера PIC18F458 для простого случая использования: фильтры и маски не используются, приём сообщений ведётся только через один приёмный буфер, передача сообщения осуществляется также через один передающий буфер, скорость передачи составляет 1 Мбит/с, тактовая частота микроконтроллера 20 МГц, передаются и принимаются только расширенные сообщения, удалённый запрос данных (RTR) не используется, обработка ошибок не рассматривается.

Введение

Встроенный модуль CAN микроконтроллера PIC18F458 представляет собой набор регистров и логических модулей, в совокупности реализующих канальный уровень протокола CAN, подразумевающий буферизацию и фильтрацию сообщений, управление передачей сообщения (арбитраж, подтверждение, контрольная сумма) и обнаружение ошибок. С точки зрения программирования представляет собой набор регистров, с помощью которых осуществляется передача и приём сообщений, управление работой логики модуля. Рассмотрим подробнее состав модуля и основные принципы работы с ним, а именно: настройку модуля CAN, приём и передачу сообщений. Особое внимание уделим настройке скорости работы на шине.

Структурная схема и состав модуля CAN

Модуль CAN микроконтроллера PIC18F458 состоит из двух частей – *ядра протокола* и *набора буферов*. Структурная схема модуля приведена на **Рис. 1**.

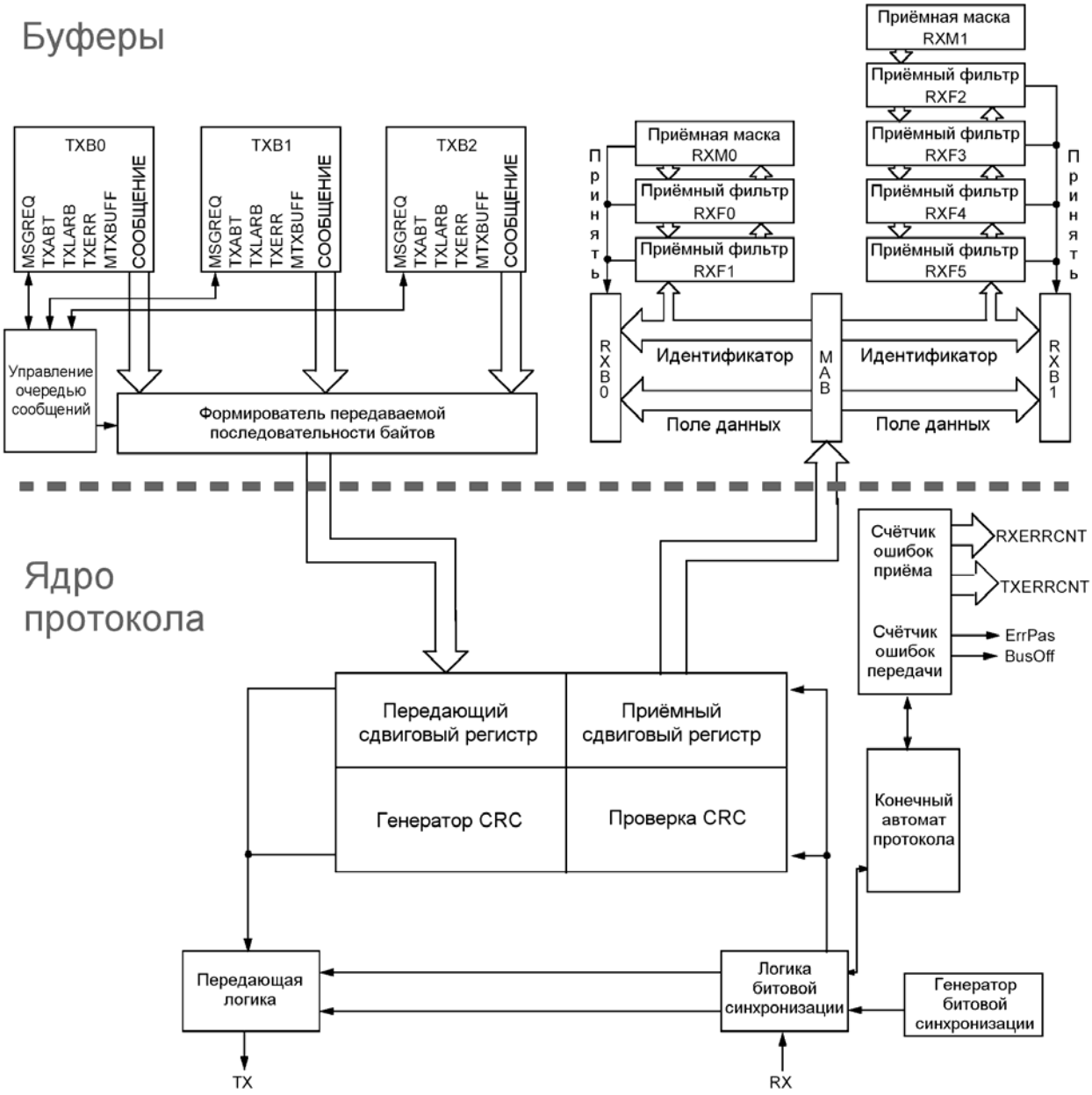

Рис. 1. Структурная схема модуля CAN

Ядро протокола является управляющей частью модуля CAN. Оно состоит из следующих элементов:

1) *конечный автомат протокола* – главная часть ядра протокола. Конечный автомат управляет логикой CRC, логикой управления ошибками, параллельными потоками данных между передающим и приёмным сдвиговыми регистрами и буферами, состоянием линии шины, арбитражем и повторной передачей сообщений.

2) *передающий и приёмный сдвиговые регистры* – выполняют преобразование байтов в последовательность битов для дальнейшей выдачи на шину (передающий регистр), и наоборот – преобразование принятых с шины битов в байты (принимающий регистр).

3) *логика CRC* – содержит генератор CRC (контрольной суммы) и логику проверки CRC. При передаче сообщения генератор CRC создает последовательность CRC, которая передаётся в кадре сообщения. При приёме на основе полученного сообщения тоже генерируется CRC, и вычисленное значение сравнивается с принятым в кадре сообщения. Если обе последовательности CRC совпадают, это означает, что сообщение принято правильно, в ином случае это означает ошибку в принятом сообщении.

4) *передающая логика* – отвечает за передачу битовой последовательности на шину CAN.

5) *счётчики ошибок приёма и передачи* – представляют собой логику управления ошибками. Счетчики инкрементируются и декрементируются согласно протоколу CAN. В зависимости от значений счётчиков, устройство может находиться в состояниях *активной ошибки*, *пассивной ошибки* и *отключения от шины*. Счётчики представляют собой регистры (TXERRCNT – счетчик ошибок передачи, RXERRCNT – счётчик ошибок приёма), программно доступные только для чтения.

6) *логика битовой синхронизации* – отвечает за битовую синхронизацию с шиной CAN. Позволяет настраивать сегменты времени передачи бита и точку выборки для компенсации задержек распространения и смещения фазы.

Набор буферов состоит из следующих элементов:

1) *три полных передающих буфера с возможностью указания приоритетов* – предназначены для хранения сообщения, ожидающего передачи. Сообщение хранится в буфере без полей CRC, ACK, конца кадра и начала кадра (эти поля автоматически генерируются и добавляются в кадр при передаче сообщения). Эти буферы представляют собой наборы регистров: восемь регистров для данных (TXBnD0[*] – TXBnD7) и пять регистров для служебной информации – идентификатора и типа сообщения (TXBnSIDH, TXBnSIDL, TXBnEIDH, TXBnEIDL) и длины данных (TXBnDLC), а также один регистр управления (TXBnCON).

2) *формирователь передаваемой последовательности байтов* – разбивает хранящееся в передающем буфере сообщение на байты и побайтно записывает в передающий сдвиговый регистр.

3) *управление очередью сообщений* – управляет приоритетной передачей сообщений.

4) *буфер сборки принимаемых сообщений (MAB)* – программно недоступный буфер, в нём осуществляется сборка принимаемого сообщения, которое затем передаётся в один из приёмных буферов – RXB0 или RXB1. По сути, является третьим приёмным буфером, предназначенным для хранения текущего принимаемого сообщения.

5) *два приёмных буфера RXB0 и RXB1* – программно доступны для чтения, содержат полностью принятое сообщение. Буфер RXB0 имеет более высокий приоритет, чем RXB1. Приёмные буферы представляют собой наборы регистров, аналогичные передающим буферам, но предназначенные для хранения принятых сообщений:

RXBnD0 – RXBnD7 – регистры данных;

RXBnSIDH, RXBnSIDL, RXBnEIDH, RXBnEIDL – регистры идентификатора и типа сообщения;

RXBnDLC – регистр длины данных;

RXBnCON – регистр управления.

6) *приёмные фильтры и маски* – используются для фильтрации сообщений, доступны для программирования. Маска определяет, какие биты фильтра используются для проверки сообщения, биты фильтров определяют критерий отбора сообщения. Сообщение, принятое в MAB, проверяется на соответствие критерию фильтра, и, если сообщение соответствует критерию отбора, оно загружается в приёмный буфер.

У буфера RXB0, обладающего более высоким приоритетом, всего два приёмных фильтра, то есть фильтрация сообщений осуществляется быстрее и сообщение поступает в буфер с меньшими задержками, чем в RXB1, который имеет четыре приёмных фильтра.

Регистры приёмных масок: RXMnSIDH, RXMnSIDL, RXMnEIDH, RXMnEIDL.

Регистры приёмных фильтров: RXFnSIDH, RXFnSIDL, RXFnEIDH, RXFnEIDL.

Режимы работы и приёма сообщений

Модуль CAN микроконтроллера PIC18F458 может работать в одном из нескольких режимов, которые задаются установкой битов REQOP2:REQOP0 регистра CANCON:

1) *режим настройки* (REQOP2:REQOP0 = '1xx') - в этом режиме модуль не принимает и не передаёт данные, счётчики ошибок сброшены, флаги прерываний остаются неизменными. Регистры управления

[*]Здесь и далее буква 'n' в названии регистра обозначает набор аналогичных регистров, отличающихся только номерами.

модуля доступны для записи. Этот режим необходим для выполнения настройки модуля CAN перед его включением, а также во время работы для изменения текущих настроек.

2) *нормальный режим* (REQOP2:REQOP0 = '000') - в этом режиме модуль может передавать и принимать сообщения, в том числе флаги ошибок и сигналы подтверждения; порты ввода-вывода работают под управлением модуля. Это обычный режим работы модуля CAN.

3) *режим прослушивания* (REQOP2:REQOP0 = '011') - в этом режиме устройство принимает все сообщения, но не передаёт сообщений, включая флаги ошибок и сигналы подтверждения; счётчики ошибок сброшены и выключены. Этот режим можно использовать для наблюдения за шиной или определения скорости передачи опытным путём (для этого необходимы два других узла, обменивающихся информацией).

4) *режим петли (loopback)* (REQOP2:REQOP0 = '010') - позволяет осуществить передачу сообщения по внутренним цепям модуля от передающего буфера в приёмный буфер без выдачи сообщения на шину CAN. Этот режим можно использовать при разработке и тестировании системы. В этом режиме бит ACK игнорируется, и устройство будет признавать входящие сообщения от самого себя, как если бы они поступали от других узлов. При этом сообщения с шины не принимаются и на шину не передаются, включая флаги ошибок и сигналы подтверждения.

5) *режим "отключен от шины"* (REQOP2:REQOP0 = '001') - в этом режиме модуль не принимает и не передаёт данные, счётчики ошибок не изменяют своих значений, флаги прерываний сохраняют свои состояния (кроме флага обнаружения активности на шине WAKIF). При этом порты ввода-вывода модуля переходят в режим цифровых входов/выходов. То есть модуль CAN в этом режиме выключен.

Кроме режимов работы модуля CAN дополнительно можно задать режимы приёма сообщений. Они задаются с помощью битов RXM1:RXM0 управляющих регистров приёмных буферов (RXBnCON). Возможны следующие режимы:

1) *приём всех безошибочных сообщений* (RXM1:RXM0 = '00') – принимаются все сообщения, не содержащие ошибок, независимо от типа сообщения.

2) *приём безошибочных сообщений только со стандартным идентификатором* (RXM1:RXM0 = '01').

3) *приём безошибочных сообщений только с расширенным идентификатором* (RXM1:RXM0 = '10').

4) *приём сообщений с ошибками*[*] (RXM1:RXM0 = '11') – в этом случае в приёмный буфер копируются все сообщения, в том числе и сообщения с ошибками, независимо от типа идентификатора. Этот режим следует использовать только при тестировании системы.

Из перечисленных выше режимов работы в нашей статье мы используем только режим настройки и нормальный режим, а в качестве режима приёма сообщений используется режим приёма безошибочных сообщений только с расширенным идентификатором.

Примечание: в регистре состояния модуля CANSTAT содержатся биты OPMODE2:OPMODE0, которые отражают текущий режим работы. При переходе из одного режима в другой нужно дождаться, пока эти биты не примут то же значение, что было задано в REQOP2:REQOP0. Это будет означать, что модуль переключился в заданный режим.

Включение и настройка модуля CAN

Для включения модуля CAN в работу необходимо выполнить следующие действия:

1) настроить выводы RB2 и RB3 порта PORTB как порты ввода-вывода модуля CAN установкой бита TRISB<3> в единицу (линия RX), а бита TRISB<2> в ноль (линия TX).

2) в регистре CANCON установить биты REQOP2:REQOP0 в '100' (режим настройки), затем ждать, пока не установится этот режим, проверяя биты OPMODE2:OPMODE0 в регистре CANSTAT. Как только эти биты примут значение '100', значит, режим установился и можно перейти к следующему шагу.

[*] В англоязычной версии технического описания на микроконтроллер этот режим был выделен в отдельный режим работы модуля, наравне с такими режимами, как нормальный и режим настройки, и назывался "error recognition mode – режим распознавания ошибок". На самом деле это просто режим приёма сообщений, и правильнее было бы его назвать "режим приёма сообщений с ошибками" или "режим восприимчивости к ошибкам".

3) произвести настройку приёмных фильтров и масок, настройку прерываний (если они используются)[*]. Так как фильтры использоваться не будут, устанавливаем в ноль значения всех регистров масок:

RXMnSIDH – регистры масок, содержащие критерий для старших битов стандартного идентификатора.

RXMnSIDL - регистры масок, содержащие критерий для младших битов стандартного идентификатора и старших битов расширенного идентификатора, а также флага стандартный/расширенный идентификатор (флаг типа сообщения).

RXMnEIDH - регистры масок, содержащие критерий для средних битов расширенного идентификатора.

RXMnEIDL - регистры масок, содержащие критерий для младших битов расширенного идентификатора.

4) произвести настройку приёмных буферов. Так как в нашем случае используется только один приёмный буфер, нужно настроить его для работы в режиме приёма всех действительных расширенных сообщений установкой в '10' битов RXM1:RXM0 регистра RXB0CON и запретить дублирование буфера сбросом бита RX0DBEN регистра RXB0CON.

5) произвести настройку скорости передачи. Для этого необходимо рассчитать значения сегментов времени и ширины фазового перехода и занести в регистры BRGCON1, BRGCON2, BGRCON3. Более подробно процесс настройки скорости передачи описывается ниже.

6) настроить регистр CIOCON – установить бит ENDRHI, чтобы притянуть вывод TX к питанию в рецессивном состоянии, и сбросить бит CANCAP, чтобы отключить режим захвата CAN с помощью модуля CCP1 (этот режим используется для реализации временной метки, которую в нашем простом примере использовать не будем).

7) установить нормальный режим работы (REQOP2:REQOP0 = '000') и дождаться, пока этот режим установится (OPMODE2:OPMODE0 = '000').

Далее модуль функционирует в нормальном режиме, то есть можно принимать и передавать сообщения.

Настройка скорости передачи модуля CAN[**]

Настройка скорости передачи данных выполняется, когда модуль CAN находится в режиме настройки.

Для настройки скорости передачи в модуле CAN имеются следующие регистры: BRGCON1, BRGCON2, BRGCON3. В них настраивается предделитель частоты (BRP) для собственного генератора битовой синхронизации модуля, а также сегменты времени передачи бита: сегмент распространения (PropSeg), фазовые сегменты 1 (PS1) и 2 (PS2), ширина перехода синхронизации (SJW).

Время передачи бита (битовый интервал) состоит из квантов времени (TQ) в количестве от 8 до 25 TQ. Эти кванты времени распределяются между сегментами времени передачи бита по следующим правилам:

$$PropSeg + PS1 \geq PS2;$$

$$PS2 \geq SJW.$$

Для вычисления скорости передачи данных по шине CAN используются следующие формулы:

1) для расчета длительности кванта времени TQ в зависимости от значения предделителя и тактовой частоты генератора:

$$TQ = \frac{2 \cdot (BRP + 1)}{F_{OSC}},$$

где TQ - квант времени (в секундах);

BRP - значение предделителя;

[*] В данной статье прерывания не рассматриваются.

[**] Дополнительно рекомендуем ознакомиться с документом "*AN754 - Понимание битовой синхронизации модуля CAN фирмы Microchip*".

F_{OSC} - тактовая частота генератора (в герцах).

2) для расчёта значения предделителя в зависимости от длительности кванта времени и тактовой частоты генератора:

$$BRP = \frac{TQ \cdot F_{OSC}}{2} - 1.$$

3) для расчёта количества квантов времени при известной скорости передачи и длительности кванта времени:

$$N_{TQ} = \frac{1}{\text{Скорость передачи} \cdot TQ},$$

где N_{TQ} - количество квантов времени в одном битовом интервале;

Скорость передачи - скорость передачи информации (в бит/с).

4) для расчёта длительности кванта времени при заданном количестве квантов времени в битовом интервале и известной скорости передачи:

$$TQ = \frac{1}{\text{Скорость передачи} \cdot N_{TQ}}.$$

5) для расчёта скорости передачи при известной длительности кванта времени и количестве квантов времени в битовом интервале:

$$\text{Скорость передачи} = \frac{1}{TQ \cdot N_{TQ}}.$$

При известной скорости (1 Мбит/с) и частоте тактового генератора (20 МГц) необходимо определить длительность кванта времени и количество квантов времени на один битовый интервал и распределить их между его сегментами. Для этого необходимо выбрать значение предделителя частоты BRP (это значение должно находится в интервале от 0 до 63).

Чтобы найти доступные при таких параметрах значения BRP, найдём интервал, в которых лежат эти значения, используя минимальное количество квантов времени $N_{TQ} = 8$ и максимальное количество $N_{TQ} = 25$.

Для $N_{TQ} = 8$ длительность кванта времени составит:

$$TQ = \frac{1}{1 \cdot 10^6 \cdot 8} = 125 \cdot 10^{-9} \text{ с.}$$

Тогда

$$BRP = \frac{125 \cdot 10^{-9} \cdot 20 \cdot 10^6}{2} - 1 = 0.25.$$

Для $N_{TQ} = 25$ длительность кванта времени составит:

$$TQ = \frac{1}{1 \cdot 10^6 \cdot 25} = 40 \cdot 10^{-9} \text{ с.}$$

Тогда

$$BRP = \frac{40 \cdot 10^{-9} \cdot 20 \cdot 10^6}{2} - 1 = -0.6.$$

Таким образом, в качестве значений BRP мы можем выбрать целое положительное число в интервале от -0.6 до 0.25. То есть нам доступно только одно значение BRP = 0.

Найдём длительность квантов времени и их количество в битовом интервале для BRP = 0.

В этом случае длительность кванта времени составит:

$$TQ = \frac{2 \cdot (0+1)}{20 \cdot 10^6} = 100 \cdot 10^{-9} \text{ с.}$$

Тогда получим количество квантов времени в битовом интервале:

$$N_{TQ} = \frac{1}{1 \cdot 10^{6} \cdot 100 \cdot 10^{-9}} = 10 .$$

Теперь полученное количество квантов времени нужно распределить между сегментами времени передачи бита.

Сегмент синхронизации всегда равен 1TQ. Сегмент распространения выберем равным 2TQ (именно такое значение рекомендовано для линий связи шины CAN "средней протяжённости", в остальных случаях значение подбирается экспериментально). Точка выборки должна находиться в 60 - 70 % от начала битового интервала, следовательно, величину первого фазового сегмента выберем равной 4TQ. Из 10TQ для второго фазового сегмента остаётся 3TQ. Ширина перехода синхронизации для кварцевого генератора, как правило, принимается равной 1TQ.

Эти значения соответствуют указанным выше правилам: $2 + 4 > 3$ и $3 > 1$.

Для программирования сегментов битового интервала необходимо выполнить следующее:

1) Установить BRP = 0: в биты BRP5:BRP0 регистра BRGCON1 занести значение '000000'.

2) Установить PropSeg = 2TQ: в биты PRSEG2:PRSEG0 регистра BRGCON2 занести значение '001'.

3) Установить PS1 = 4TQ: в биты SEG1PH2:SEG1PH0 регистра BRGCON2 занести значение '011'.

4) Установить PS2 = 3TQ: в биты SEG2PH2:SEG2PH0 регистра BRGCON3 занести значение '010'.

5) Установить SJW = 1TQ: в биты SJW1:SJW0 регистра BRGCON1 занести значение '00'

Кроме этого необходимо выполнить следующие настройки:

1) бит SEG2PHTS регистра BRGCON2 установить в единицу (свободное программирование второго фазового сегмента), причём это нужно сделать ещё до настройки сегментов!

2) бит SAM регистра BRGCON2 сбросить в ноль (осуществлять выборку один раз).

3) бит WAKFIL регистра BRGCON3 сбросить в ноль (не использовать пробуждение микроконтроллера при активности на шине – режим SLEEP в данной статье мы не рассматриваем).

Передача данных

Для передачи сообщений CAN будем использовать только один передающий буфер TXB0.

Для передачи сообщения сначала необходимо определить, свободен ли передающий буфер (бит TXREQ регистра TXB0CON сброшен в 0).

Перед тем как передать сообщение, нужно занести его в регистры передающего буфера следующим образом:

1) Занести идентификатор сообщения в следующие регистры:

TXB0SIDH – старший байт стандартной части идентификатора.

TXB0SIDL – содержит младшие три бита стандартной части идентификатора, бит типа сообщения (так как у нас расширенное сообщение, устанавливаем этот бит в единицу), а также два старших бита расширенного идентификатора.

TXB0EIDH – старший байт расширенной части идентификатора.

TXB0EIDL – младший байт расширенной части идентификатора.

2) Занести данные в следующие регистры:

TXB0Dm – m байтов данных (от 0 до 8 байтов).

3) Занести длину данных (биты DLC<3:0>) и сбросить в ноль бит удалённого запроса (TXRTR) (так как удалённый запрос не используется) в следующие регистры:

TXB0DLC – длина данных (количество байтов данных) и бит удалённого запроса.

4) Если необходимо, установить приоритет передачи в битах TXPRI <1:0> регистра TXB0CON. Доступны следующие значения приоритетов:

'11' – уровень приоритета 3 (наивысший приоритет)

'10' – уровень приоритета 2 (высокий приоритет)

'01' – уровень приоритета 1 (низкий приоритет)

'00' – уровень приоритета 0 (низший приоритет)

Чем выше приоритет сообщения, тем раньше оно будет передано. Это имеет смысл при использовании нескольких передающих буферов.

После заполнения всех регистров передающего буфера передача инициируется установкой в единицу бита TXREQ регистра TXB0CON. Этот бит не инициирует саму передачу, а лишь сигнализирует модулю, что сообщение в буфере готово к передаче. Как только сообщение будет передано, соответствующий буфер генерирует прерывание, сигнализируя о возможности загрузки нового пакета данных.

В случае необходимости подготовленные сообщения можно сбросить установкой бита ABAT в регистре CANCON (сбросить все передающие буферы) или сбросом бита TXREQ в регистре TXB0CON (сбросить только один передающий буфер, в данном случае буфер с номером 0). Если сообщение в этот момент уже передаётся, сброс не будет иметь эффекта, и сообщение будет передано полностью.

Приём данных

Для приёма сообщений будем использовать только один буфер, RXB0.

Из буфера сборки сообщения MAB готовое сообщение поступает в приёмный буфер и генерируется соответствующее прерывание. Чтобы прочитать сообщение из приёмного буфера, нужно выполнить следующие действия:

1) Извлечь идентификатор сообщения из следующих регистров:

RXB0SIDH – содержит старший байт стандартной части идентификатора.

RXB0SIDL – содержит младшие три бита стандартной части идентификатора, флаг удалённого запроса для стандартных сообщений (в расширенном сообщении он всегда равен нулю – см. описание регистров CAN), бит типа сообщения (так как принимаем только расширенные сообщения, этот бит будет установлен в единицу), а также два старших бита расширенного идентификатора.

RXB0EIDH – содержит старший байт расширенной части идентификатора.

RXB0EIDL – содержит младший байт расширенной части идентификатора.

2) Из регистра RXB0DLC считать длину данных.

3) Извлечь данные из следующих регистров:

RXB0Dm – m байтов данных (от 0 до 8 байт).

4) сбросить бит RXFUL в регистре RXB0CON, освобождая буфер для приёма нового сообщения.

