http://www.microchip.com/

2004 Microchip Technology DS51297D

MPLAB® C18

C COMPILER

БИБЛИОТЕКИ

Содержание

http://www.microchip.com/

ВВЕДЕНИЕ

Цель этого документа заключается в предоставлении подробной информации о библиотеках и скомпилированные объектные файлы, которые могут быть использованы с фирмы Microchip MPLAB® C18 C Compiler.

макет документа

Компоновка такова:

• Глава 1: Обзор - описывает библиотеки и скомпилированные объектные файлы имеется.

• Глава 2: Аппаратные периферийные функции - описывает каждую оборудование периферическая функция библиотеки.

http://www.microchip.com/

• Глава 3: Программное обеспечение Периферийные Библиотека - описывает каждый программного обеспечения периферийных Библиотека функций.

• Глава 4: Генеральный Software Библиотека - описывает каждую общую библиотеку программного обеспечения Функция.

• Глава 5: Math Library - рассматриваются функции математической библиотеки.

• Глоссарий - словарь терминов, использованных в данном руководстве.

• Главная - Перекрестная ссылка список терминов, функций и разделов этого документа.

//////////////////

Условные

Описание Представляет Примеры

Arial шрифт:

Руководство курсивом Справочные книги MPLAB IDE пользователя

Курьер шрифта:

Обычная Образец Курьер исходный код #define START

Имена файлов autoexec.bat

Пути к файлам C: \ mcc18 \ ч

Ключевые слова _asm, _endasm, статическая

Параметры командной строки -Opa +, -Opa-

Курсив Курьер переменная аргумент file.o, где файл может быть любым допустимым

файла

0bnnnn двоичного числа, где п

двоичный разряд

0b00100, 0b10

0xNNNN шестнадцатеричное число, где

п шестнадцатеричная цифра

0xFFFF, 0x007A

Квадратные скобки [] Необязательные аргументы mcc18 [параметры] файл [варианты]

http://www.microchip.com/

Фигурные скобки и

труба характер: {|}

Выбор взаимоисключающими

аргументы;Выбор ИЛИ

код ошибки {0 | 1}

Многоточие ... Заменяет повторил текст var_name [, var_name ...]

Представляет код поставляется

пользователь

Основными недействительным (недействительными)

{...

}

//////////////////

РЕКОМЕНДУЕМАЯ ЛИТЕРАТУРА

Для получения дополнительной информации о подключенных библиотек и скомпилированных объектных файлов для компиляторы, эксплуатация MPLAB IDE и использование других инструментов, следующие Рекомендуем к прочтению.

readme.c18

Для получения последней информации об использовании MPLAB C18 C Compiler, читать readme.c18 файл (ASCII текст) в комплекте с программным обеспечением. Данный файл Файл содержит обновленную информацию что не могут быть включены в этот документ.

readme.xxx

Для получения последней информации о других инструментов Microchip (MPLAB IDE, линкер MPLINK ™ и т.д.), читать связанные файлы Readme (ASCII текстовый файл) входит в программное обеспечение.

MPLAB® C18 C Compiler Руководство по началу работы (DS51295)

Описывает, как установить компилятор MPLAB C18, как писать простые программы и как использовать MPLAB IDE с компилятором.

Руководство MPLAB® C18 C Compiler пользователя (DS51288)

Полное руководство, которое описывает функции, а также MPLAB компании Microchip Компилятор C18 C для PIC18 устройств.

MPLAB® IDE V6.XX Краткое руководство (DS51281)

http://www.microchip.com/

Описывает, как настроить программное обеспечение MPLAB IDE и использовать его для создания проектов и Программа устройства.

Руководство MPASM ™ пользователя с MPLINK ™ Linker и техподдержка на русском языке ™ Библиотекарь (DS33014)

Описывает, как использовать ассемблер Microchip PICmicro MCU (MPASM), компоновщик (MPLINK) и библиотекарь (техподдержка на русском языке).

PICmicro® 18C MCU Family Reference Manual (DS39500)

Ориентирован на Расширенной MCU семейства устройств. Работа Enhanced MCUсемья архитектура и периферийные модули объясняется, но не распространяется на Специфика каждого устройства.

PIC18 устройств Паспорта и применению

Технические описания описания работы и электрические характеристики устройств PIC18. Указания по применению описывают, как использовать PIC18 устройств. Для получения любой из перечисленных выше документов, посетите веб-сайт Microchip (www.microchip.com), чтобы получить эти документы в формате Adobe Acrobat (.pdf).

//////////////////////////////////

Условные

Описание Представляет Примеры

Arial шрифт:

Руководство курсивом Справочные книги MPLAB IDE пользователя

Курьер шрифта:

Обычная Образец Курьер исходный код #define START

Имена файлов autoexec.bat

Пути к файлам C: \ mcc18 \ ч

Ключевые слова _asm, _endasm, статическая

Параметры командной строки -Opa +, -Opa-

Курсив Курьер переменная аргумент file.o, где файл может быть любым допустимым

файла

0bnnnn двоичного числа, где п

двоичный разряд

http://www.microchip.com/

0b00100, 0b10

0xNNNN шестнадцатеричное число, где

п шестнадцатеричная цифра

0xFFFF, 0x007A

Квадратные скобки [] Необязательные аргументы mcc18 [параметры] файл [варианты]

Фигурные скобки и

труба характер: {|}

Выбор взаимоисключающими

аргументы;Выбор ИЛИ

код ошибки {0 | 1}

Многоточие ... Заменяет повторил текст var_name [, var_name ...]

Представляет код поставляется

пользователь

Основными недействительным (недействительными)

{...

}

//////////////////////////////////////

MICROCHIP САЙТ

Microchip предоставляет онлайн помощью нашей WWW-сайте по адресу www.microchip.com. Этот веб-сайт используется как средство, чтобы сделать файлы и информацию легко доступны для клиентов.

Доступность, используя ваш любимый интернет-браузер, веб-сайт содержит следующее информация:

• **поддержка** - Технические описания и опечаток, указания по применению и образец программы, дизайн ресурсы, инструкций и аппаратная поддержка документы, последние версии программного обеспечения и архивных программное обеспечение

• **Техническая поддержка Общие** - Часто задаваемые вопросы (FAQ), техническая поддержка запросов, онлайн дискуссионные группы, Microchip консультант программы список участников

• **Бизнес Microchip** - Выбор продукта, который заставляет гидов, последнее Microchip пресс-релизы, перечень семинаров и других мероприятий, листингов о Microchip офисов продаж, дистрибьюторы и представители завода

http://www.microchip.com/

Системы разработки с клиентами Изменить служебном сообщении

Служба уведомления клиентов компании Microchip помогает сохранить клиентов текущие на Microchip продукции. Подписчики будут получать по электронной почте уведомления всякий раз, когда есть изменения, обновления, изменения или ошибках, связанные с указанной семейства продуктов или развития инструмента интерес.

Чтобы зарегистрироваться, получить доступ к веб-сайт Microchip в www.microchip.com, нажмите на Заказчика Уведомление об изменении и следуйте инструкциям по регистрации.

Систем разработки категории Группа изделий являются:

• Составители - последняя информация на Microchip C компиляторов и другом языке инструменты. К ним относятся MPLAB C17, MPLAB C18 и MPLAB C30 C компиляторы; MPASM ™ и MPLAB ASM30 сборщики; MPLINK ™ и MPLAB LINK30 объект линкеров; и техподдержка на русском языке ™ и MPLAB LIB30 возражать библиотекарей.

• Эмуляторы - последняя информация на Microchip в замыкания emulators.This включает в себя MPLAB ICE 2000 и MPLAB ICE 4000.

• В замыкания Отладчики - последняя информация на микрочипе в цепи отладчик, MPLAB ICD 2.

• MPLAB IDE - последняя информация на Microchip MPLAB IDE, окна интегралы тертый среда разработки для систем разработки инструментов. Этот список сосредоточены на MPLAB IDE, MPLAB SIM и MPLAB SIM30 тренажеры, MPLAB IDE Руководитель проекта и общие редактирования и отладки особенности.

• Программисты - последняя информация на Microchip программистов. К ним относятся MPLAB PM3 и PRO MATE® II программаторы и PICSTART® Плюс развитие программист.

Глава 1 Обзор

1.1 ВВЕДЕНИЕ

В этой главе дается обзор файлов библиотек MPLAB C18 и предкомпилированного объекта файлы, которые могут быть включены в приложение.

1.2 MPLAB C18 БИБЛИОТЕКИ ОБЗОР

Библиотека представляет собой набор функций, сгруппированных ведения и легкости компоновки для. Смотреть Руководство MPASM ™ пользователя с MPLINK ™ и техподдержка на русском языке ™ (DS33014) для получения дополнительной информации о создании и поддержании библиотек.

Библиотеки MPLAB C18 включены в Lib подкаталог установки. Эти могут быть связаны непосредственно в приложения с помощью линкера MPLINK.

http://www.microchip.com/

Эти файлы предварительной компиляции в C: \ mcc18 \ SRC каталога на Microchip. каталог SRC \ традиционный содержатся файлы для Non-расширенном режиме и SRC \ продлен содержатся файлы для расширенном режиме. Если вы выбрали не устанавливать компилятор и связанные с ним файлы в C: \ mcc18 каталог, исходный код из библиотек будет не показывают в файле линкер листинга и не может быть шагнул через при использовании MPLAB IDE.

Чтобы включить код библиотеки в .lst файл и, чтобы иметь возможность один шаг через библиотеки функции, следуйте инструкциям в разделе 1.3.3, раздел 1.4.3 и раздел 1.5.3 для восстановления библиотек, используя прилагаемые командные файлы (bat-), найденные в SRC, SRC \ традиционные и SRC \ продлен каталоги.

1.3 ЗАПУСК КОД

1.3.1 Обзор

Три версии пуске кода предоставляются с MPLAB C18, с различными уровнями инициализации. Файлы C018 * .o объектов предназначены для использования с компилятор работает в Номера для расширенного режима. Файлы C018 * _e.o объектов предназначены для использования с компилятором при работает в расширенном режиме. В порядке возрастания сложности, то они таковы:

c018.o / c018_e.o инициализирует стек C программного обеспечения и прыгает к началу применение функции, основной ().

c018i.o / c018i_e.o выполняет все те же задачи, как c018.o / c018_e.o а также присваивает соответствующие значения инициализированных данных до вызова приложения пользователя.

Инициализации требуется, если глобальные или статические переменные установлены в начение, когда они определяется. Это код запуска, который входит в компоновщик файлов сценариев, которые обеспечены MPLAB C18.

c018iz.o / c018iz_e.o выполняет все те же задачи, как c018i.o / c018i_e.o а также назначает нулю ко всем необъявленные переменные, как это требуется для строгой ANSI соблюдение.

1.3.2 Исходный код

Исходный код для запуска процедуры можно найти в SRC \ традиционный \ ввод в эксплуатацию и SRC \ продлен \ Автозагрузка подкаталоги установки компилятора.

1.3.3 Восстановление

Пакетный файл makestartup.bat могут быть использованы для восстановления запуска кода и копирования сгенерированный объектные файлы в Библиотека каталога.

Перед восстановления запуска кода с makestartup.bat, проверить, что MPLAB C18 (mcc18.exe) находится в вашем пути.

http://www.microchip.com/

1.4 ПРОЦЕССОР-НЕЗАВИСИМАЯ БИБЛИОТЕКА

1.4.1 Обзор

Стандартная библиотека C (clib.lib или clib_e.lib) предоставляет функции, которые поддерживается ядром PIC18 архитектуры: те, которые поддерживаются во всех Процессоры в семье. Эти функции описаны в следующих главах:

• Генеральный Software Библиотека, Глава 4.

• математические библиотеки, Глава 5.

1.4.2 Исходный код

Исходный код для функций в стандартной библиотеке C можно найти в

следующие подкаталоги установки компилятора:

• SRC \ традиционный \ математика

• SRC \ продлен \ математика

• SRC \ традиционные \ задержки

• SRC \ расширенные \ задержки

• SRC \ традиционный \ stdclib

• SRC \ продлен \ stdclib

1.4.3 Восстановление

Пакетный файл makeclib.bat могут быть использованы для восстановления независимой процессор-Библиотека. До запуска этого пакетного файла, убедитесь, что следующие инструменты на вашем пути:

• MPLAB C18 (mcc18.exe)

• MPASM ассемблер (mpasm.exe)

• техподдержка на русском языке библиотекарь (mplib.exe)

Также до восстановления стандартной библиотечной, убедитесь, что переменная среды MCC_INCLUDE установлен на пути MPLAB C18 включают файлы (например, C: \ mcc18 \ ч).

/***/

http://www.microchip.com/

1,5 БИБЛИОТЕКИ конкретного процессора

1.5.1 Обзор

Библиотека файлов конкретного процессора содержат определения, которые могут отличаться по лицу Члены семейства PIC18. Это включает в себя все периферийные процедур и Регистров специального назначения (SFR) определения. Периферийные процедуры, которые предоставляются включают в себя как те, которые предназначены для использования соответствующим оборудованием и тех, которые реализуют периферийный интерфейс с использованием общего назначения линий ввода / вывода. Функции, включенные в библиотеки для конкретного процессора описаны в следующих главах:

• **Глава 2 "Аппаратные периферийные функции"**

• **Глава 3 «Программное обеспечение Периферийные Библиотека"**

Библиотеки для конкретного процессора названы:

p processor.lib - Номера для расширенного режима процессора конкретных библиотека

p processor_e.lib - Расширенная процессор конкретных библиотека режим

Например, файл библиотеки для PIC18F4620 назван p18f4620.lib для Номера для расширенной версией библиотеки и p18f4620_e.lib для расширенной версии библиотеки.

1.5.2 Исходный код

Исходный код для библиотек конкретного процессора можно найти в следующем подкаталоги установки компилятора:

• SRC \ традиционный \ PMC

• SRC \ продлен \ PMC

• SRC \ традиционный \ Труды

• SRC \ продлен \ Труды

1.5.3 Восстановление

Пакетный файл makeplib.bat могут быть использованы для восстановления библиотек для конкретного процессора.

До запуска этого пакетного файла, убедитесь, что следующие инструменты на вашем пути:

• MPLAB C18 (mcc18.exe)

• MPASM ассемблер (mpasm.exe)

• техподдержка на русском языке библиотекарь (mplib.exe)

Также до вызова makeplib.bat, убедитесь, что переменная среды MCC_INCLUDE установлен на пути MPLAB C18 включают файлы (например, C: \ mcc18 \ ч).

http://www.microchip.com/

Глава 2 Аппаратные периферийные функции

2.1 ВВЕДЕНИЕ

В этой главе документы аппаратные периферийные функции, найденные в библиотеки для конкретного процессора. Исходный код для всех этих функций входит MPLAB C18 в SRC \ традиционный \ PMC и SRC \ расширенные \ PMC подкаталоги из установки компилятора.

Смотрите в Руководстве Пользователя MPASM ™ с MPLINK ™ и техподдержка на русском языке ™ (DS33014) для более Информация об управлении библиотеки, используя библиотекаря MPLIB.

Следующие периферийные устройства поддерживаются MPLAB C18 библиотечных подпрограмм:

• A / D конвертер (Раздел 2.2 "A / D конвертер Функции»)

• Входной Capture (Раздел 2.3 "Ввод Захват Функции»)

• I2C ™ (раздел 2.4 "Функции IIC ™")

• Порты ввода / вывода (Раздел 2.5 "I / O Port Функции»)

• Микропровод (Раздел 2.6 "Микропровод Функции»)

• широтно-импульсной модуляции (PWM) (раздел 2.7 "широтно-импульсной модуляции Функции ")

• SPI ™ (Раздел 2.8 "SPI ™ Функции»)

• Таймер (Раздел 2.9 "Функции таймера")

• USART (Раздел 2.10 "USART Функции»)

//**********

2,2 A / D конвертер ФУНКЦИИ

A / D периферической поддерживается со следующими функциями:

Таблица 2-1: A / D конвертер ФУНКЦИИ

Функция Описание

BusyADC ли A / D конвертер в настоящее время выполняет преобразование?

CloseADC Отключение A / D конвертер.

ConvertADC Начать A / D преобразование.

http://www.microchip.com/

OpenADC Настройка A / D конвертер.

ReadADC Читайте результаты A / Ц преобразования.

SetChanADC Выберите A / D канала, которые будут использоваться.

//**********

2.2.1 Описания функций

BusyADC

Функция: Каково A / D конвертер в настоящее время выполняет преобразование?

Включает в себя: adc.h

Прототип: cЬar BusyADC (недействительными);

Примечание: Эта функция указывает на то, A / D периферийное устройство в процессе преобразования значения.

Возвращаемые значения:

1, если A / D периферической выполняет преобразование.

0, если A / Ц периферийное устройство не выполняет преобразование.

Имя файла: adcbusy.c

//**********

CloseADC

Функция: Отключение A / D конвертер.

Включает в себя: adc.h

Прототип: пустота CloseADC (недействительными);

Примечания: Эта функция отключает A / D конвертер и A / D механизм прерывания.

Имя файла: adcclose.c

//**********

http://www.microchip.com/

ConvertADC

Функция: Запускает / D процесса конверсии.

Включает в себя: adc.h

Прототип: пустота ConvertADC (недействительными);

Примечания: Эта функция запускает / цифровое преобразование. Функция BusyADC () может использоваться для обнаружения завершение конверсии.

Имя файла: adcconv.c

//**

OpenADC

PIC18CXX2, PIC18FXX2, PIC18FXX8, PIC18FXX39

Функция: Настройка / D конвертер.

Включает в себя: adc.h

Прототип: пустота OpenADC (неподписанные символ конфигурации,

неподписанные символ Config2);

Аргументы: конфигурации

Битовая маска, которая создается путем выполнения побитовой операции ('&') со значением от каждой из категорий, перечисленных ниже. Эти значения определены в файле adc.h.

A / D источник синхронизации:

ADC_FOSC_2 FOSC / 2

ADC_FOSC_4 FOSC / 4

ADC_FOSC_8 FOSC / 8

ADC_FOSC_16 FOSC / 16

ADC_FOSC_32 FOSC / 32

ADC_FOSC_64 FOSC / 64

ADC_FOSC_RC Внутренний RC-генератор

A / D результат обоснование:

ADC_RIGHT_JUST Результат в младших битах

http://www.microchip.com/

ADC_LEFT_JUST Результат в старших бита

A / D источником опорного напряжения:

ADC_8ANA_0REF VREF + = VDD, VREF- = VSS,

Все аналоговые каналы

ADC_7ANA_1REF AN3 = VREF +, все аналоговые каналы, за исключением AN3

ADC_6ANA_2REF AN3 = VREF +, AN2 = VREF

ADC_6ANA_0REF VREF + = VDD, VREF- = VSS

ADC_5ANA_1REF AN3 = VREF +, VREF- = VSS

ADC_5ANA_0REF VREF + = VDD, VREF- = VSS

ADC_4ANA_2REF AN3 = VREF +, AN2 = VREF-

ADC_4ANA_1REF AN3 = VREF +

ADC_3ANA_2REF AN3 = VREF +, AN2 = VREF-

ADC_3ANA_0REF VREF + = VDD, VREF- = VSS

ADC_2ANA_2REF AN3 = VREF +, AN2 = VREF-

ADC_2ANA_1REF AN3 = VREF +

ADC_1ANA_2REF AN3 = VREF +, AN2 = VREF-,

AN0 = A

ADC_1ANA_0REF AN0 является аналоговый вход

ADC_0ANA_0REF Все цифровые входы / выходы

Config2

Битовая маска, которая создается путем выполнения побитовой операции ('&') со значением от каждой из категорий, перечисленных ниже. Эти значения определены в файле adc.h.

канал:

ADC_CH0 канала 0

ADC_CH1 канал 1

ADC_CH2 Канал 2

ADC_CH3 канал 3

ADC_CH4 Channel 4

http://www.microchip.com/

ADC_CH5 5 канал

ADC_CH6 Канал 6

ADC_CH7 Канал 7

A / D прерываний:

ADC_INT_ON прерывания Включен

ADC_INT_OFF прерывания отключены

Примечания: Эта функция сбрасывает A / D периферийным устройством состояние ПОР и настраивает A / D, связанных Регистры специального назначения (SFR) в соответствии с Варианты указано.

Имя файла: adcopen.c

Пример кода: OpenADC (ADC_FOSC_32 &

ADC_RIGHT_JUST &

ADC_1ANA_0REF,

ADC_CH0 &

ADC_INT_OFF);

//***

OpenADC

PIC18C658 / 858, PIC18C601 / 801,

PIC18F6X20, PIC18F8X20

Функция: Настройка / D конвертер.

Включает в себя: adc.h

Прототип: пустота **OpenADC** (неподписанные символ конфигурации, неподписанные символ Config2);

Аргументы: конфигурации

Битовая маска, которая создается путем выполнения побитовой операции ('&') со значением от каждой из категорий, перечисленных ниже. Эти значения определены в файле adc.h.

A / D источник синхронизации:

ADC_FOSC_2 FOSC / 2

ADC_FOSC_4 FOSC / 4

http://www.microchip.com/

ADC_FOSC_8 FOSC / 8

ADC_FOSC_16 FOSC / 16

ADC_FOSC_32 FOSC / 32

ADC_FOSC_64 FOSC / 64

ADC_FOSC_RC Внутренний RC-генератор

A / D результат обоснование:

ADC_RIGHT_JUST Результат в младших битах

ADC_LEFT_JUST Результат в старших бита

A / D конфигурация порта:

ADC_0ANA Все цифровые

ADC_1ANA аналоговый: AN0 цифровой: AN1-AN15

ADC_2ANA аналоговый: AN0-AN1 цифровой: AN2-AN15

ADC_3ANA аналоговый: AN0-AN2 цифровой: AN3-AN15

ADC_4ANA аналоговый: AN0-AN3 цифровой: AN4-AN15

ADC_5ANA аналоговый: AN0-AN4 цифровой: AN5-AN15

ADC_6ANA аналоговый: AN0-AN5 цифровой: AN6-AN15

ADC_7ANA аналоговый: AN0-AN6 цифровой: AN7-AN15

ADC_8ANA аналоговый: AN0-AN7 цифровой: AN8-AN15

ADC_9ANA аналоговый: AN0-AN8 цифровой: AN9-AN15

ADC_10ANA аналоговый: AN0-AN9 цифровой: AN10-AN15

ADC_11ANA аналоговый: AN0-AN10 цифровой: AN11-AN15

ADC_12ANA аналоговый: AN0-AN11 цифровой: AN12-AN15

ADC_13ANA аналоговый: AN0-AN12 цифровой: AN13-AN15

ADC_14ANA аналоговый: AN0-AN13 цифровой: AN14-AN15

ADC_15ANA Все аналоговые

Config2

Битовая маска, которая создается путем выполнения побитовой операции ('&') со значением от каждой из категорий, перечисленных ниже. Эти значения определены в файле adc.h.

канал:

http://www.microchip.com/

ADC_CH0 канала 0

ADC_CH1 канал 1

ADC_CH2 Канал 2

ADC_CH3 канал 3

ADC_CH4 Channel 4

ADC_CH5 5 канал

ADC_CH6 Канал 6

ADC_CH7 Канал 7

ADC_CH8 канала 8

ADC_CH9 Channel 9

ADC_CH10 Канал 10

ADC_CH11 Канал 11

ADC_CH12 Канал 12

ADC_CH13 канал 13

ADC_CH14 Канал 14

ADC_CH15 Канал 15

A / D прерываний:

ADC_INT_ON прерывания Включен

ADC_INT_OFF прерывания отключены

A / D VREF + конфигурации:

ADC_VREFPLUS_VDD VREF + = AVDD

ADC_VREFPLUS_EXT VREF + = внешний

A / D VREF- конфигурации:

ADC_VREFMINUS_VSS VREF- = AVSS
ADC_VREFMINUS_EXT VREF- = внешний

Примечания: Эта функция сбрасывает / D, связанных регистрирует в состояние ПОР, а затем астраивает часы, формат результата, источник опорного напряжения, порт и канал.

Имя файла: adcopen.c

Пример кода:

http://www.microchip.com/

```
OpenADC (ADC_FOSC_32 &
        ADC_RIGHT_JUST &
        ADC_14ANA,
        ADC_CH0 &
        ADC_INT_OFF);

//*****************************************
```

OpenADC

Все другие процессоры

Функция: Настройка / D конвертер.

Включает в себя: adc.h

Прототип:

```
пустота OpenADC (неподписанные символ конфигурации,
        неподписанные символ Config2,
        неподписанные символ portconfig);
```

Аргументы: конфигурации

Битовая маска, которая создается путем выполнения побитовой операции ('&') со значением от каждой из категорий, перечисленных ниже. Эти значения определены в файле adc.h. A / D источник синхронизации:

ADC_FOSC_2 FOSC / 2

ADC_FOSC_4 FOSC / 4

ADC_FOSC_8 FOSC / 8

ADC_FOSC_16 FOSC / 16

ADC_FOSC_32 FOSC / 32

ADC_FOSC_64 FOSC / 64

ADC_FOSC_RC Внутренний RC-генератор

A / D результат обоснование:

http://www.microchip.com/

ADC_RIGHT_JUST Результат в младших битах

ADC_LEFT_JUST Результат в старших бита

A / D время захвата выберите:

ADC_0_TAD 0 Тэд

ADC_2_TAD 2 Тэд

ADC_4_TAD 4 Тэд

ADC_6_TAD 6 Тэд

ADC_8_TAD 8 Тэд

ADC_12_TAD 12 Тэд

ADC_16_TAD 16 Тэд

ADC_20_TAD 20 Тэд

Config2

Битовая маска, которая создается путем выполнения побитовой операции ('&') со значением от каждой из категорий, перечисленных ниже. Эти значения определены в файле adc.h. канал:

ADC_CH0 канала 0

ADC_CH1 канал 1

ADC_CH2 Канал 2

ADC_CH3 канал 3

ADC_CH4 Channel 4

ADC_CH5 5 канал

ADC_CH6 Канал 6

ADC_CH7 Канал 7

ADC_CH8 канала 8

ADC_CH9 Channel 9

ADC_CH10 Канал 10

ADC_CH11 Канал 11

ADC_CH12 Канал 12

ADC_CH13 канал 13

ADC_CH14 Канал 14

http://www.microchip.com/

ADC_CH15 Канал 15

A / D прерываний:

ADC_INT_ON прерывания Включен

ADC_INT_OFF прерывания отключены

Конфигурация напряжения / D:

ADC_VREFPLUS_VDD VREF + = AVDD

ADC_VREFPLUS_EXT VREF + = внешний

ADC_VREFMINUS_VDD VREF- = AVDD

ADC_VREFMINUS_EXT VREF- = внешний

portconfig

Значение **portconfig** любое значение от 0 до 127 для PIC18F1220 / 1320 и от 0 до 15 для PIC18F2220 / 2320/4220/4320, включительно. Это значение бита 0 до 6, или биты от 0 до 3 ADCON1 регистр, которые являются биты конфигурации порта.

Примечания: Эта функция сбрасывает A/D, связанных регистрирует в состояние POR, а затем настраивает часы, формат результата, источник опорного напряжения, порт и канал.

Имя файла: adcopen.c

Пример кода:

```
OpenADC (ADC_FOSC_32 &
        ADC_RIGHT_JUST &
        ADC_12_TAD,
        ADC_CH0 &
        ADC_INT_OFF, 15);
```

//***

ReadADC

Функция: Прочитайте результат A / Ц преобразования.

Include: adc.h

http://www.microchip.com/

Прототип: int **ReadADC(void);**

Примечания: Эта функция читает 16-битный результат A / Ц преобразования.

Вернуться Значение: Эта функция возвращает 16-разрядное результат преобразования / D. На основе конфигурации A / D конвертер (например, с помощью OpenADC () функция), то результат будет содержаться в наименее Значительные или старших бита 16-разрядного результата.

Имя файла: adcread.c

//***

SetChanADC

Функция: Выберите канал, используемый в качестве вклада в / АЦП.

Include: adc.h

Прототип: void **SetChanADC**(unsigned char channel);

Аргументы: канал

Один из следующих значений (определено в adc.h):

ADC_CH0 канала 0

ADC_CH1 канал 1

ADC_CH2 Канал 2

ADC_CH3 канал 3

ADC_CH4 Channel 4

ADC_CH5 5 канал

ADC_CH6 Канал 6

ADC_CH7 Канал 7

ADC_CH8 канала 8

ADC_CH9 Channel 9

ADC_CH10 Канал 10

ADC_CH11 Канал 11

Примечания: Выбор штифт, который будет использоваться в качестве вклада в / АЦП.

Имя файла: adcsetch.c

http://www.microchip.com/

Пример кода: SetChanADC (ADC_CH0);

2.2.2 Пример Использование A / D конвертер Подпрограммы

```
//*****************************************



#include <p18C452.h>

#include <adc.h>

#include <stdlib.h>

#include <delays.h>

int result;

void main( void )

{

  // Настроить A / D конвертер

 OpenADC( ADC_FOSC_32 & ADC_RIGHT_JUST & ADC_8ANA_0REF,

        ADC_CH0 & ADC_INT_OFF );

 Delay10TCYx( 5 );     // Задержка 50TCY

 ConvertADC();        // Начать преобразование

 while( BusyADC() );   // Подождите завершения


 result = ReadADC();   // Прочтите результат

 CloseADC();          // Отключить A / D конвертер

}

SetChanADC


//*****************************************
```

http://www.microchip.com/

2.3 входа захвата ФУНКЦИИ

Захват периферической поддерживается со следующими функциями:

Таблица 2-2: входа захвата ФУНКЦИИ

Функция Описание

CloseCapturex Отключить захватить периферийные x.

OpenCapturex Настройка захватить периферийные x.

ReadCapturex Чтение значения от захвата периферической x.

CloseECapturex (1) Отключить enhan CED периферийное x.

OpenECapturex (1) Настройка повышенная периферийное x.

ReadECapturex (1) Чтение значения из усиливается захвата периферической x.

Примечание 1: Расширенные функции захвата доступны только на этих устройствах с ECCPxCON зарегистрируйтесь.

//**********

2.3.1 Описания функций

CloseCapture1

CloseCapture2

CloseCapture3

CloseCapture4

CloseCapture5

CloseECapture1

Функция: Отключить входа захвата x.

Включает в себя: capture.h

Прототип:

void CloseCapture1(void);

void CloseCapture2(void);

void CloseCapture3(void);

void CloseCapture4(void);

void CloseCapture5(void);

http://www.microchip.com/

void CloseECapture1(void);

Примечание: Эта функция отключает прерывание, соответствующее указанному входу захватить.

Имя файла:

cp1close.c

cp2close.c

cp3close.c

cp4close.c

cp5close.c

ep1close.c

//**************************************

OpenCapture1

OpenCapture2

OpenCapture3

OpenCapture4

OpenCapture5

OpenECapture1

Функция: Настройка и активизировать входной захвата x.

Включает в себя: capture.h

Прототип:

void OpenCapture1(unsigned char config);

void OpenCapture2(unsigned char config);

void OpenCapture3(unsigned char config);

void OpenCapture4(unsigned char config);

void OpenCapture5(unsigned char config);

void OpenECapture1(unsigned char config);

Аргументы: config

http://www.microchip.com/

Битовая маска, которая создается путем выполнения побитовой операции ('&') со значением от каждой из категорий, перечисленных ниже. Эти значения определены в файле capture.h:

Включить CCP прерываний:

CAPTURE_INT_ON прерывания Включено

CAPTURE_INT_OFF прерывания инвалидов

Прерывание Trigger (заменить x с КПК номер модуля):

Cx_EVERY_FALL_EDGE прерывания на каждом заднему фронту

Cx_EVERY_RISE_EDGE прерывания на каждом переднем фронте

Cx_EVERY_4_RISE_EDGE прерывания на каждом 4-м переднем фронте

Cx_EVERY_16_RISE_EDGE прерывания на каждом 16-м восстания край

EC1_EVERY_FALL_EDGE прерывания на каждом заднему фронту (расширенная)

EC1_EVERY_RISE_EDGE прерывания на каждом переднем фронте (расширенная)

EC1_EVERY_4_RISE_EDGE прерывания на каждом 4-м переднем фронте (расширенная)

EC1_EVERY_16_RISE_EDGE прерывания на каждом 16-м восстания EDGE (повышенная)

Примечания: Эта функция сначала сбрасывает модуль захвата для государства ПОР, а затем настраивает захвата входа для указанного выделения контуров. Функции захвата использовать структуру, определенную в *capture.h,* к показывают состояние переполнения каждого из модулей захвата. Эта структура называется CapStatus и имеет следующие битовые поля:

Cap1OVF

Cap2OVF

Cap3OVF

Cap4OVF

Cap5OVF

ECap1OVF

В дополнение к открытию захват, соответствующий модуль таймера должна быть включена перед любой из захватов будет работать. Смотреть паспорт для КПК и конфигураций соединительных таймера и разделе 2.9 «Таймер Функции "для аргументов, используемых с CCP в OpenTimer3.

Имя файла:

cp1open.c

cp2open.c

http://www.microchip.com/

cp3open.c

cp4open.c

cp5open.c

ep1open.c

Пример кода:

```
 OpenCapture1 (CAPTURE_INT_ON &
          C1_EVERY_4_RISE_EDGE);
```

```
//*****************************************
```

ReadCapture1

ReadCapture2

ReadCapture3

ReadCapture4

ReadCapture5

ReadECapture1

Функция: Прочитайте результат события захвата от заданного захвата входного сигнала.

Включает в себя: capture.h

Прототип:

```
unsigned int ReadCapture1( void );

unsigned int ReadCapture2( void );

unsigned int ReadCapture3( void );

unsigned int ReadCapture4( void );

unsigned int ReadCapture5( void );

unsigned int ReadECapture1( void );
```

Примечания: Эта функция считывает значение SFRs соответствующих захвата входа в.

Возвращаемые значения: Эта функция возвращает результат случае захвата.

Имя файла:

http://www.microchip.com/

cp1read.c

cp2read.c

cp3read.c

cp4read.c

cp5read.c

ep1read.c

```
//************************************
```

2.3.2 Пример использования захвата Подпрограммы

Этот пример демонстрирует использование библиотечных процедур захвата в "опрошенных" (не по прерываниям) среды.

```
#include <p18C452.h>
#include <capture.h>
#include <timers.h>
#include <usart.h>
#include <stdlib.h>
void main(void)
{
  unsigned int result;
  char str[7];
  // Настройка Capture1
  OpenCapture1( C1_EVERY_4_RISE_EDGE &
            CAPTURE_INT_OFF );
  // Настройка Timer3
  OpenTimer3( TIMER_INT_OFF &
           T3_SOURCE_INT );
  // Настройка USART
  OpenUSART( USART_TX_INT_OFF  &
```

```
http://www.microchip.com/
        USART_RX_INT_OFF  &
        USART_ASYNCH_MODE &
        USART_EIGHT_BIT   &
        USART_CONT_RX,
        25 );
 while(!PIR1bits.CCP1IF);  // Подождите события
 result = ReadCapture1(); // Читать результат
 ultoa(result,str);      // Перевести в строку
 // Написать строку, чтобы USART если
 // Переполнение не произошло.
 if(!CapStatus.Cap1OVF)
 {
   putsUSART(str);
 }
 // Clean up
 CloseCapture1();
 CloseTimer3();
 CloseUSART();
}
//************************************
```

2.4 ФУНКЦИИ I2C ™

Следующие процедуры предназначены для устройств с одной I2C периферии:

Таблица 2-3: ОДИН I2C периферийных функций

Следующие функции предназначены для устройств с несколькими периферийными устройствами I2C:

http://www.microchip.com/

Функция Описание

AckI2C Генерация шины I2C Признать состояние.

CloseI2C Отключить модуль SSP.

DataRdyI2C ли имеющиеся данные в буфере I2C?

getcI2C Читайте один байт из шины I2C.

getsI2C Считать строку из I2C шину, работающую в режиме мастер I2C.

IdleI2C Loop до шине I2C простаивает.

NotAckI2C Генерация шину I2C не признают состояние.

OpenI2C Настройка модуля SSP.

putcI2C Написать одного байта к шине I2C.

putsI2C Записать строку в шине I2C, работающего в любом ведущий или ведомый режим.

ReadI2C Читайте один байт из шины I2C.

RestartI2C Генерация состояние шины I2C перезапуска.

StartI2C Генерация состояние Start шины I2C.

StopI2C Генерация состояние шины I2C Stop.

WriteI2C Написать одного байта к шине I2C.

Таблица 2-4: несколько функций I2C ПЕРИФЕРИЙНЫЕ

Функция Описание

AckI2Cx Генерация I2Cx автобус Признать состояние.

CloseI2Cx Отключить x модуль CC.

DataRdyI2Cx ли имеющиеся данные в буфере I2Cx?

getcI2Cx Читайте один байт из автобуса I2Cx.

getsI2Cx Читать строку из автобуса I2Cx, работающего в режиме мастер I2C.

IdleI2Cx Loop до I2Cx автобусе простаивает.

NotAckI2Cx Генерация I2Cx автобус не признают состояние.

OpenI2Cx Настроить модуль SSPX.

putcI2Cx Написать одного байта на шину I2Cx.

putsI2Cx Записать строку в автобусе I2Cx, работающего в любом ведущий или ведомый режим.

http://www.microchip.com/

ReadI2Cx Читайте один байт из автобуса I2Cx.

RestartI2Cx Генерация состояние автобуса перезапуска I2Cx.

StartI2Cx Генерация автобус Пуск состояние I2Cx.

StopI2Cx Генерация автобусная остановка состояние I2Cx.

WriteI2Cx Написать одного байта на шину I2Cx.

//**********

Следующие функции предусмотрены также для сопряжения с устройством EE памяти такого как Microchip 24LC01B помощью интерфейса I2C:

Таблица 2-5: Функции интерфейса ДЛЯ EE памяти USB

Функция Описание

EEAckPollingx Генерация последовательность Acknowledge опроса.

EEByteWritex Написать одного байта.

EECurrentAddReadx Читайте один байт из следующего места.

EEPageWritex Написать строку данных.

EERandomReadx Читайте один байт из произвольного адреса.

EESequentialReadx Читать строку данных.

//**********

2.4.1 Описания функций

AckI2C

AckI2C1

AckI2C2

Функция: Генерация шины I2C Признать состояние.

Включает в себя: i2c.h

Прототип:

Prototype: void AckI2C(void);

void AckI2C1(void);

http://www.microchip.com/

void AckI2C2(void);

Примечания: Эта функция генерирует автобус I2Cx Признать состояние.

Имя файла:

i2c_ack.c

i2c1ack.c

i2c2ack.c

//**********************************

CloseI2C

CloseI2C1

CloseI2C2

Функция: Отключить модуль SSPX.

Включает в себя: i2c.h

Прототип: пустота CloseI2C (недействительными);

недействительными CloseI2C1 (недействительными);

недействительными CloseI2C2 (недействительными);

Примечания: Эта функция отключает модуль SSPX.

Имя файла:

i2c_close.c

i2c1close.c

i2c2close.c

//**********************************

DataRdyI2C

DataRdyI2C1

DataRdyI2C2

Функция: данные доступные в буфере I2Cx?

http://www.microchip.com/

Включает в себя: i2c.h

Прототип: unsigned char DataRdyI2C(void);

unsigned char DataRdyI2C1(void);

unsigned char DataRdyI2C2(void);

Примечания: Определяет, есть ли байт для чтения в буфере SSPX.

Возвращаемые значения:

1, если есть данные в буфере SSPX

0, если нет никаких данных в буфере SSPX

Имя файла: i2c_dtrd.c

i2c1dtrd.c

i2c2dtrd.c

Пример кода:

```
if (DataRdyI2C())
{
  var = getcI2C();
}
```

```
//************************************
```

getcI2C

getcI2C1

getcI2C2

getcI2Cx определяется как ReadI2Cx. Смотреть ReadI2Cx.

getsI2C

getsI2C1

getsI2C2

Функция: Читать фиксированную строку длиной от шины I2Cx работает в мастер I2C Режим.

Включает в себя: i2c.h

Прототип:

http://www.microchip.com/

```
 unsigned char getsI2C(
        unsigned char  * rdptr,
        unsigned char    length );
unsigned char getsI2C1(
        unsigned char  * rdptr,
        unsigned char    length );
unsigned char getsI2C2(
        unsigned char  * rdptr,
        unsigned char    length );
```

Аргументы: rdptr

Тип символов указатель на PICmicro памяти для хранения данных, считанных из I2C устройство.

длина

Количество байт для чтения из I2Cx устройства.

Примечания: Эта процедура читает предопределенный длину строки данных с шины I2Cx.

Возвращаемое значение: 0, если все байты были отправлены -1, Если столкновение автобуса произошло

Имя файла:

i2c_gets.c

i2c1gets.c

i2c2gets.c

Пример кода: без знака строка символ [15];

```
 getsI2C (строка, 15);
//************************************
```

IdleI2C

IdleI2C1

IdleI2C2

http://www.microchip.com/

Функция: Loop, пока I2Cx автобус простаивает.

Включает в себя: i2c.h

Прототип: void IdleI2C(void);

Примечания: Эта функция проверяет состояние I2C периферических и ждет автобус, чтобы стать доступны. Функция IdleI2C требуется, так как аппаратный I2C периферических не позволяет буферизации последовательностей автобусных. I2C устройство должно быть в состоянии ожидания перед операцией I2C может быть начато или столкновение записи будет.

Имя файла:

idlei2c.c

//**********

NotAckI2C

NotAckI2C1

NotAckI2C2

Функция: Генерация I2Cx автобус не признают состояние.

Включает в себя: i2c.h

Прототип:

void NotAckI2C(void);

void NotAckI2C1(void);

void NotAckI2C2(void);

Примечания: Эта функция генерирует автобус I2Cx Не Признать состояние.

Имя файла:

i2c_nack.c

i2c1nack.c

i2c2nack.c

//**********

OpenI2C

OpenI2C1

http://www.microchip.com/

OpenI2C2

Функция: Настройка модуля SSPX.

Включает в себя: i2c.h

Прототип:

void OpenI2C(unsigned char sync_mode,

unsigned char slew);

void OpenI2C1(unsigned char sync_mode,

unsigned char slew);

void OpenI2C2(unsigned char sync_mode,

unsigned char slew);

Аргументы: **Sync_Mode**

Один из следующих значений, определенных в i2c.h:

Режим SLAVE_7 I2C Slave, 7-битный адрес

Режим SLAVE_10 I2C Slave, 10-битный адрес

МАСТЕР I2C режим Master

нарастания

Один из следующих значений, определенных в i2c.h:

SLEW_OFF Скорость нарастания выходного напряжения отключены в режиме 100 кГц

SLEW_ON Скорость нарастания выходного напряжения для режима 400 кГц

Примечания: OpenI2Cx сбрасывает SSPX модуль в состояние ПОР, а затем настраивает модуль для Ведущий / Ведомый режим и выбранный убивание Скорость.

Имя файла:

i2c_open.c

i2c1open.c

i2c2open.c

Пример кода: OpenI2C (MASTER, SLEW_ON);

//**********

http://www.microchip.com/

putcI2C

putcI2C1

putcI2C2

putcI2Cx является определяет как WriteI2Cx. Смотреть WriteI2Cx.

putsI2C

putsI2C1

putsI2C2

Функция: Написать строку данных к шине I2Cx работающего в главное или подчиненное Режим.

Включает в себя: i2c.h

Прототип:

```
unsigned char putsI2C(
        unsigned char *wrptr );
unsigned char putsI2C1(
        unsigned char *wrptr );
unsigned char putsI2C2(
        unsigned char *wrptr );
```

Аргументы: wrptr

Указатель на данные, которые будут записаны в шине I2C.

Примечания: Эта процедура записывает строку данных на шину I2Cx до нулевой символ не является достигли. Сам нулевой символ не передается. Эта процедура может работать как в ведущий или ведомый режим.

Вернуться Значение:

режим Master I2C:

0, если нулевой символ был достигнут в строке данных

-2, Если раб I2Cx устройство ответил NOT ACK

-3, Если произошло столкновение записи

Режим ведомого I2C:

0, если нулевой символ был достигнут в строке данных

-2, Если мастер I2Cx устройство ответил NOT ACK, которые прекращается передачу данных

http://www.microchip.com/

Имя файла:

i2c_puts.c

i2c1puts.c

i2c2puts.c

Пример кода: unsigned char string[] = “data to send”;

//**********************************

ReadI2C

ReadI2C1

ReadI2C2

getcI2C

getcI2C1

getcI2C2

Функция: Прочитайте один байт из автобуса I2Cx.

Включает в себя: i2c.h

Прототип:

unsigned char ReadI2C (void);

unsigned char ReadI2C1 (void);

unsigned char ReadI2C2 (void);

unsigned char getcI2C (void);

unsigned char getcI2C1 (void);

unsigned char getcI2C2 (void);

Примечания: Функция читает один байт из шины I2Cx. **getcI2Cx** является

определены как **ReadI2Cx** в i2c.h.

Возвращаемые значения: Байт данных, считанных из автобуса I2Cx.

Имя файла:

i2c_read.c

i2c1read.c

i2c2read.c

define in i2c.h

http://www.microchip.com/

define in i2c.h

define in i2c.h

Пример кода:

символьное значение без знака;

Значение = ReadI2C ();

//**********************************

RestartI2C

RestartI2C1

RestartI2C2

Функция: Генерация состояние автобуса перезапуска I2Cx.

Включает в себя: i2c.h

Прототип:

void StartI2C(void);

void StartI2C1(void);

void StartI2C2(void);

Примечания: Эта функция генерирует условие автобус перезапуска I2Cx.

Имя файла:

i2c_rstr.c

i2c1rstr.c

i2c2rstr.c

//**********************************

StartI2C

StartI2C1

StartI2C2

Функция: Генерация автобус Пуск состояние I2Cx.

http://www.microchip.com/

Включает в себя: i2c.h

Прототип:

void StartI2C(void);

void StartI2C1(void);

void StartI2C2(void);

Примечания: Эта функция генерирует I2Cx автобус стартовое условие.

Имя файла:

i2c_start.c

i2c1start.c

i2c2start.c

//**********************************

StopI2C

StopI2C1

StopI2C2

Функция: Генерация I2Cx автобусная остановка состояние.

Включает в себя: i2c.h

Прототип:

void StopI2C(void);

void StopI2C1(void);

void StopI2C2(void);

Примечания: Эта функция генерирует автобусная остановка состояние I2Cx.

Имя файла:

i2c_stop.c

i2c1stop.c

i2c2stop.c

//**********************************

http://www.microchip.com/

WriteI2C

WriteI2C1

WriteI2C2

putcI2C

putcI2C1

putcI2C2

Функция: Написать одного байта до автобусной устройства I2Cx.

Включает в себя: i2c.h

Прототип:

```
unsigned char WriteI2C(
    unsigned char data_out );
unsigned char WriteI2C1(
    unsigned char data_out );
unsigned char WriteI2C2(
    unsigned char data_out );
unsigned char putcI2C(
    unsigned char data_out );
unsigned char putcI2C1(
    unsigned char data_out );
unsigned char putcI2C2(
    unsigned char data_out );
```

Аргументы: data_out

Один байт данных для записи на автобусной устройства I2Cx.

Примечания: Эта функция записывает один байт данных к автобусной устройства I2Cx. putcI2Cx определяется для WriteI2Cx в i2c.h.

Возвращаемое значение:

0, если запись прошла успешно

-1, Если произошло столкновение записи

Имя файла:

http://www.microchip.com/

i2c_write.c

i2c1write.c

i2c2write.c

#define в i2c.h

#define в i2c.h

#define в i2c.h

Пример кода:

WriteI2C ('a');

//**********************************

2.4.2 EE памяти Интерфейс устройства Описания функций

EEAckPolling

EEAckPolling1

EEAckPolling2

Функция: генерация признать избирательный последовательность для Microchip EE I2C устройства памяти.

Включает в себя: i2c.h

Прототип:

```
unsigned char EEAckPolling(
    unsigned char control );
unsigned char EEAckPolling1(
    unsigned char control );
unsigned char EEAckPolling2(
    unsigned char control );
```

Аргументы: контроль

http://www.microchip.com/

Контроль EEPROM / автобус устройство выберите адрес байта.

Примечания: Эта функция используется для генерации последовательности Acknowledge опроса для Устройства памяти EE I2C, которые используют Признать опрос.

Возвращаемое значение:

0, если не было никаких ошибок

-1, Если был автобус ошибке столкновения

-3, Если бы была запись об ошибке столкновения

Имя файла:

i2c_ecap.c

i2c1ecap.c

i2c2ecap.c

Пример кода:

temp = **EEAckPolling**(0xA0);

//**********

EEByteWrite

EEByteWrite1

EEByteWrite2

Функция: Написать одного байта на шину I2Cx.

Включает в себя: i2c.h

Прототип:

```
unsigned char EEByteWrite(
    unsigned char control,
    unsigned char address,
    unsigned char data );
unsigned char EEByteWrite1(
    unsigned char control,
    unsigned char address,
```

http://www.microchip.com/

unsigned char data);

unsigned char EEByteWrite2(

unsigned char control,

unsigned char address,

unsigned char data);

Аргументы:

контроль

Контроль EEPROM / автобус устройство выберите адрес байта.

адрес

EEPROM внутренний адрес расположения.

данных

Данные для записи в EEPROM по адресу, указанному в параметре функции адрес.

Примечания: Эта функция записывает один байт данных на шину I2Cx. Эта процедура может быть использован для любого устройства памяти Microchip I2C EE, который требует только 1 байт адресной информации.

Возвращаемое значение:

0, если не было никаких ошибок

-1, Если был автобус ошибке столкновения

-2, Если произошла ошибка НЕ ACK

-3, Если бы была запись об ошибке столкновения

Имя файла:

i2c_ecbw.c

i2c1ecbw.c

i2c2ecbw.c

Пример кода:

temp = EEByteWrite(0xA0, 0x30, 0xA5);\

//**********

http://www.microchip.com/

EECurrentAddRead

EECurrentAddRead1

EECurrentAddRead2

Функция: Прочитайте один байт из автобуса I2Cx.

Включает в себя: i2c.h

Прототип:

```
unsigned int EECurrentAddRead(
    unsigned char control );
unsigned int EECurrentAddRead1(
    unsigned char control );
unsigned int EECurrentAddRead2(
    unsigned char control );
```

Аргументы:

контроль

Контроль EEPROM / автобус устройство выберите адрес байта.

Примечания: Функция читает один байт из автобуса I2Cx.адрес местоположение данных для чтения является то, что из текущего указателя внутри I2C EE устройство. Запоминающее устройство содержит счетчик адреса, который сохраняет адрес последнего слова доступ, увеличивается на единицу.

Возвращаемые значения:

-1, если произошла ошибка шины столкновения

-2, Если произошла ошибка НЕ ACK

-3, Если произошла ошибка столкновение записи

В противном случае, результат возвращается как беззнаковое количество 16-битной. С сам буфер только 8-битный интерфейс, это означает, что наиболее значимыми Байт будет равна нулю и младший байт будет содержать чтения содержимое буфера.

Имя файла:

i2c_eecr.c

i2c1eecr.c

i2c2eecr.c

Пример кода:

http://www.microchip.com/

temp = ECurrentAddRead(0xA1);

//**********

EEPageWrite

EEPageWrite1

EEPageWrite2

Функция: Написать строку данных в ЕЕ прибора и шины I2Cx.

Включает в себя: i2c.h

Прототип:

```
unsigned char EEPageWrite(
    unsigned char control,
    unsigned char address,
    unsigned char * wrptr );
unsigned char EEPageWrite1(
    unsigned char control,
    unsigned char address,
    unsigned char * wrptr );
unsigned char EEPageWrite2(
    unsigned char control,
    unsigned char address,
    unsigned char * wrptr );
```

Аргументы:

контроль

Контроль EEPROM / автобус устройство выберите адрес байта.

адрес

EEPROM внутренний адрес расположения.

wrptr

http://www.microchip.com/

Тип символов указатель в PICmicro памяти. Объекты данных указал на по wrptr будут записаны в EE устройства.

Примечания: Данная функция позволяет записывать, оканчивающихся нулем строку данных в I2C EE запоминающее устройство. Сам нулевой символ не передается.

Возвращаемое значение:

0, если не было никаких ошибок

-1, Если был автобус ошибке столкновения

-2, Если произошла ошибка НЕ ACK

-3, Если бы была запись об ошибке столкновения

Имя файла:

i2c_eepw.c

i2c1eepw.c

i2c2eepw.c

Пример кода:

temp = EEPageWrite(0xA0, 0x70, wrptr);

//**********

EERandomRead

EERandomRead1

EERandomRead2

Функция: Прочитайте один байт из автобуса I2Cx.

Включает в себя: i2c.h

Прототип:

```
unsigned int EERandomRead(
    unsigned char control,
    unsigned char address );
unsigned int EERandomRead1(
    unsigned char control,
```

http://www.microchip.com/

unsigned char address);

unsigned int EERandomRead2(

unsigned char control,

unsigned char address);

Аргументы:

контроль

Контроль EEPROM / автобус устройство выберите адрес байта.

адрес

EEPROM внутренний адрес расположения.

Примечания: Функция читает один байт из автобуса I2Cx. Процедура может использоваться для устройств памяти Microchip I2C EE, которые требуют только 1 байт адресной информации.

Вернуться Значение: Возвращаемое значение содержит значение, считанное в младший байт и состояние ошибки в старший байт. Условие об ошибке является:

-1, Если был автобус ошибке столкновения

-2, Если произошла ошибка НЕ ACK

-3, Если бы была запись об ошибке столкновения

Имя файла:

i2c_eerr.c

i2c1eerr.c

i2c2eerr.c

Пример кода: unsigned int temp;

temp = EERandomRead(0xA0,0x30);

//**********

EESequentialRead

EESequentialRead1

EESequentialRead2

http://www.microchip.com/

Функция: Считать строку данных из автобуса I2Cx.

Включает в себя: i2c.h

Прототип:

```
unsigned char EESequentialRead(
    unsigned char control,
    unsigned char address,
    unsigned char * rdptr,
    unsigned char length );
unsigned char EESequentialRead1(
    unsigned char control,
    unsigned char address,
    unsigned char * rdptr,
    unsigned char length );
unsigned char EESequentialRead2(
    unsigned char control,
    unsigned char address,
    unsigned char * rdptr,
    unsigned char length );
```

Аргументы:

контроль

Контроль EEPROM / автобус устройство выберите адрес байта.

адрес

EEPROM внутренний адрес расположения.

rdptr

Тип символов указатель на PICmicro RAM области для размещения данных читать EEPROM устройства.

длина

Количество байт для чтения из EEPROM устройства.

http://www.microchip.com/

Примечания: Эта функция читает в предопределенной длины строки данных из I2Cx автобус. Процедура может быть использован для устройств памяти Microchip I2C EE которые требуют только 1 байт информации адреса.

Возвращаемое значение:

0, если не было никаких ошибок

-1, Если был автобус ошибке столкновения

-2, Если произошла ошибка НЕ ACK

-3, Если бы была запись об ошибке столкновения

Имя файла:

i2c_eesr.c

i2c1eesr.c

i2c2eesr.c

Пример кода:

```
unsigned char err;

err = EESequentialRead(0xA0,
                       0x70,
                       rdptr,
                       15);
//************************************
```

2.4.3 Пример применения

Ниже приведен простой пример, иллюстрирующий модуль SSP настроенный для I2C мастер связи. Процедура показана I2C коммуникации с Microchip 24LC01B I2C EE Устройство памяти.

```
#include "p18cxx.h"

#include "i2c.h"

unsigned char arraywr[] = {1,2,3,4,5,6,7,8,0};

unsigned char arrayrd[20];

// ********************************************** ***

void main(void)

{
```

```
http://www.microchip.com/
 OpenI2C(MASTER, SLEW_ON);// Initialize I2C module
 SSPADD = 9;            //400kHz Baud clock(9) @16MHz
                  //100kHz Baud clock(39) @16MHz
 while(1)
 {
  EEByteWrite(0xA0, 0x30, 0xA5);
  EEAckPolling(0xA0);
  EECurrentAddRead(0xA0);
  EEPageWrite(0xA0, 0x70, arraywr);
  EEAckPolling(0xA0);
  EESequentialRead(0xA0, 0x70, arrayrd, 20);
  EERandomRead(0xA0,0x30);
 }
}
//************************************
```

2,5 ФУНКЦИИ I / O Port

PORTB поддерживается со следующими функциями:

Таблица 2-6: ФУНКЦИИ I / O Port

Функция	Описание
ClosePORTB	Отключить прерывания и внутренние подтягивающие резисторы для PORTB.
CloseRBxINT	запрещать прерывания для PORTB контактный x.
DisablePullups	Отключить внутренние подтягивающие резисторы на PORTB.
EnablePullups	Включить внутренние подтягивающие резисторы на PORTB.
OpenPORTB	Настройка прерывания и внутренние подтягивающие резисторы на PORTB.
OpenRBxINT	Разрешить прерывания для PORTB контактный x.

http://www.microchip.com/

2.5.1 Описания функций

//***//

ClosePORTB

Функция: Отключить прерывания и внутренние подтягивающие резисторы для PORTB.

Включает в себя: portb.h

Прототип: void **ClosePORTB**(void);

Примечания: Эта функция отключает PORTB прервать по изменению уровня и внутренняя подтягивающие резисторы.

Имя файла: pbclose.c

//***//

CloseRB0INT

CloseRB1INT

CloseRB2INT

Функция: Отключить прерывания для указанного PORTB штифта.

Включает в себя: portb.h

Прототип:

void CloseRB0INT(void);

void CloseRB1INT(void);

void CloseRB2INT(void);

Примечания: Эта функция отключает PORTB прервать по изменению уровня.

Имя файла:

rb0close.c

http://www.microchip.com/

rb1close.c

rb2close.c

//***//

DisablePullups

Функция: Отключить внутренние подтягивающие резисторы на PORTB.

Включает в себя: portb.h

Прототип: void **DisablePullups**(void);

Примечания: Эта функция отключает внутренние подтягивающие резисторы на PORTB.

Имя файла: pulldis.c

//***//

EnablePullups

Функция: Включить внутренние подтягивающие резисторы на PORTB.

Включает в себя: portb.h

Прототип: void **EnablePullups**(void);

Примечания: Эта функция позволяет внутренние подтягивающие резисторы на PORTB.

Имя файла: pullen.c

//***//

OpenPORTB

Функция: Настройка прерываний и внутренние подтягивающие резисторы на PORTB.

Включает в себя: portb.h

Прототип: void **OpenPORTB**(unsigned char config);

Аргументы:

конфигурации

http://www.microchip.com/

Битовая маска, которая создается путем выполнения побитовой операции ('&') со значением от каждой из категорий, перечисленных ниже. Эти значения определены в файле portb.h.

Прерывание по изменению уровня:

Включен PORTB_CHANGE_INT_ON прерывания

PORTB_CHANGE_INT_OFF прерывания отключены

Включить Подтягивания:

Включен PORTB_PULLUPS_ON подтягивающие резисторы

PORTB_PULLUPS_OFF подтягивающие резисторы отключены

Примечания: Эта функция настраивает прерывания и внутренние подтягивающие резисторы на PORTB.

Имя файла: pbopen.c

Пример кода: **OpenPORTB**(PORTB_CHANGE_INT_ON & PORTB_PULLUPS_ON);

//**//

OpenRB0INT

OpenRB1INT

OpenRB2INT

Функция: Разрешить прерывания для указанного PORTB штифта.

Включает в себя: portb.h

Прототип:

void OpenRB0INT(unsigned char config);

void OpenRB1INT(unsigned char config);

void OpenRB2INT(unsigned char config);

Аргументы:

конфигурации

Битовая маска, которая создается путем выполнения побитовой операции ('&') со значением от каждой из категорий, перечисленных ниже. Эти значения определены в файле portb.h.

Прерывание по изменению уровня:

Включен PORTB_CHANGE_INT_ON прерывания

http://www.microchip.com/

PORTB_CHANGE_INT_OFF прерывания отключены

Прерывание-на-краю:

RISING_EDGE_INT прерывания на переднем фронте

FALLING_EDGE_INT прерывания по заднему фронту

Включить Подтягивания:

Включен PORTB_PULLUPS_ON подтягивающие резисторы

PORTB_PULLUPS_OFF подтягивающие резисторы отключены

Примечания: Эта функция настраивает прерывания и внутренний подтягивающий резисторы на PORTB.

Имя файла:

rb0open.c

rb1open.c

rb2open.c

Пример кода:

OpenRB0INT (PORTB_CHANGE_INT_ON &

PORTB_CHANGE_INT_ON & RISING_EDGE_INT &

PORTB_PULLUPS_ON);

2,6 MICROWIRE ФУНКЦИИ

//**//

Следующие функции предназначены для устройств с одной Microwire периферии:

Таблица 2-7: ОДИН MICROWIRE периферийных функций

Функция Описание

CloseMwire Отключить модуль SSP используется для Microwire связи.

DataRdyMwire Укажите завершения внутреннего цикла записи.

http://www.microchip.com/

getcMwire Считывает байт из устройства Microwire.

getsMwire Считать строку из устройства Microwire.

OpenMwire Настроить модуль SSP для Microwire использования.

putcMwire Записывает байт в устройстве Microwire.

ReadMwire Считывает байт из устройства Microwire.

WriteMwire Записывает байт в устройстве Microwire.

Следующие функции предназначены для устройств с несколькими периферийными Микропровод:

Таблица 2-8: несколько функций MICROWIRE ПЕРИФЕРИЙНЫЕ

Функция Описание

CloseMwirex Отключить модуль SSPX используемый для Microwire связи.

DataRdyMwirex Укажите завершения внутреннего цикла записи.

getcMwirex Считывает байт из устройства Microwire.

getsMwirex Считать строку из устройства Microwire.

OpenMwirex Настроить модуль SSPX для Microwire использования.

putcMwirex Записывает байт в устройстве Microwire.

ReadMwirex Считывает байт из устройства Microwire.

WriteMwirex Записывает байт в устройстве Microwire.

2.6.1 Описания функций

//***//

CloseMwire

CloseMwire1

CloseMwire2

Функция: Отключить модуль SSPX.

Включает в себя: mwire.h

http://www.microchip.com/

Прототип:

void CloseMwire(void);

void CloseMwire1(void);

void CloseMwire2(void);

Примечания: возвращает Pin ввода / вывода под контролем TRISC и LATC настроек зарегистрируйтесь.

Имя файла: mw_close.c

mw1close.c

mw2close.c

//**********//

DataRdyMwire

DataRdyMwire1

DataRdyMwire2

Функция: указать, приняло ли устройство Microwirex завершена внутренняя написать цикл.

Включает в себя: mwire.h

Прототип:

unsigned char DataRdyMwire(void);

unsigned char DataRdyMwire1(void);

unsigned char DataRdyMwire2(void);

Примечания: Определяет, Microwirex устройство готово.

Возвращаемые значения:

1, если устройство Microwirex готов

0, если внутренний цикл записи не является полным или произошла ошибка автобус

Имя файла: mw_drdy.c

mw1drdy.c

mw2drdy.c

Пример кода: в то время как (DataRdyMwire ());

http://www.microchip.com/

//***//

getcMwire

getcMwire1

getcMwire2

getcMwirex определяется как ReadMwirex. Смотреть ReadMwirex.

getsMwire

getsMwire1

getsMwire2

Функция: Чтение строки из устройства Microwirex.

Включает в себя: mwire.h

Прототип:

```
void getsMwire( unsigned char * rdptr,
        unsigned char length);
void getsMwire1( unsigned char * rdptr,
        unsigned char length);
void getsMwire2( unsigned char * rdptr,
        unsigned char length);
```

Аргументы:

rdptr

Указатель на PICmicro памяти для размещения данных, считанных из Microwirex Устройство.

длина

Количество байт для чтения из Microwirex устройства.

Примечания: Эта функция используется для чтения заданную длину данных из Microwirex устройство. Перед использованием этой функции, Readx командовать с соответствующий адрес должен быть выдан.

Имя файла:

mw_gets.c

http://www.microchip.com/

mw1gets.c

mw2gets.c

Пример кода:

unsigned char arryrd[LENGTH];

putcMwire(READ);

putcMwire(address);

getsMwire(arrayrd, LENGTH);

//***//

OpenMwire

OpenMwire1

OpenMwire2

Функция: Настройка модуля SSPX.

Включает в себя: mwire.h

Прототип:

unsigned char arryrd[LENGTH];

putcMwire(READ);

putcMwire(address);

getsMwire(arrayrd, LENGTH);

Аргументы:

Sync_Mode

Один из следующих значений, определенных в mwire.h:

MWIRE_FOSC_4 clock = FOSC/4

MWIRE_FOSC_16 clock = FOSC/16

MWIRE_FOSC_64 clock = FOSC/64

MWIRE_FOSC_TMR2 clock = TMR2 output/2

Примечания: **OpenMwirex** сбрасывает SSPX модуль в состояние ПОР, а затем настраивает модуль для Микропровод коммуникаций.

http://www.microchip.com/

Имя файла:

mw_open.c

mw1open.c

mw2open.c

Пример кода: **OpenMwire** (MWIRE_FOSC_16);

//**//

putcMwire

putcMwire1

putcMwire2

putcMwirex определяется как WriteMwirex. Смотреть WriteMwirex.

ReadMwire

ReadMwire1

ReadMwire2

getcMwire

getcMwire1

getcMwire2

Функция: Считывает байт из устройства Microwirex.

Включает в себя: mwire.h

Прототип:

```
unsigned char ReadMwire(
    unsigned char high_byte,
    unsigned char low_byte );
unsigned char ReadMwire1(
    unsigned char high_byte,
    unsigned char low_byte );
unsigned char ReadMwire2(
    unsigned char high_byte,
    unsigned char low_byte );
```

http://www.microchip.com/

unsigned char getcMwire(

unsigned char high_byte,

unsigned char low_byte);

unsigned char getcMwire1(

unsigned char high_byte,

unsigned char low_byte);

unsigned char getcMwire2(

unsigned char high_byte,

unsigned char low_byte);

Аргументы:

high_byte

Первый байт 16-битного слова команды.

low_byte

Второй байт 16-битного слова команды.

Примечания: Функция читает один байт из устройства Microwirex.Начать немного, код операции и адрес сочинять старший и младший байты передаются в эта функция. **getcMwirex** определяется как **ReadMwirex** в mwire.h.

Возвращаемые значения: Возвращает значение байт данных считывается с устройства Microwirex.

Имя файла:

mw_read.c

mw1read.c

mw2read.c

#define в mwire.h

#define в mwire.h

#define в mwire.h

Пример кода: ReadMwire (0x03, 0x00);

//**********//

http://www.microchip.com/

WriteMwire

WriteMwire1

WriteMwire2

putcMwire

putcMwire1

putcMwire2

Функция: Эта функция используется, чтобы выписать один байт данных (один символ).

Включает в себя: mwire.h

Прототип:

```
unsigned char WriteMwire(
    unsigned char data_out );
unsigned char WriteMwire1(
    unsigned char data_out );
unsigned char WriteMwire2(
    unsigned char data_out );
unsigned char putcMwire(
    unsigned char data_out );
unsigned char putcMwire1(
    unsigned char data_out );
unsigned char putcMwire2(
    unsigned char data_out );
```

Аргументы:

data_out

Один байт данных для записи в Microwirex устройства.

Примечания: Эта функция записывает один байт данных к устройству Microwirex использованием модуль SSPX. putcMwirex определяется для WriteMwirex в mwire.h.

Возвращаемое значение:

0, если запись прошла успешно

-1, Если произошло столкновение записи

http://www.microchip.com/

Имя файла: mw_write.c

mw1write.c

mw2write.c

#define в mwire.h

#define в mwire.h

#define в mwire.h

Пример кода:

WriteMwire (0x55);

//**********

2.6.2 Пример применения

Ниже приведен простой пример, иллюстрирующий модуль SSP, общаясь С помощью устройства памяти Microchip 93LC66 Микропровод EE.

```
#include "p18cxxx.h"

#include "mwire.h"

// 93LC66 x 8

// Прототипы функций

void main(void);

void ew_enable(void);

void erase_all(void);

void busy_poll(void);

void write_all(unsigned char data);

void byte_read(unsigned char address);

void read_mult(unsigned char address,

        unsigned char *rdptr,

        unsigned char length);

void write_byte(unsigned char address,
```

```
http://www.microchip.com/
        unsigned char data);
// Определения переменных
unsigned char arrayrd[20];
unsigned char var;
// DEFINE 93LC66 МАКРОКОМАНД - смотри документ для деталей
#define  READ   0x0C
#define  WRITE  0x0A
#define  ERASE  0x0E
#define  EWEN1  0x09
#define  EWEN2  0x80
#define  ERAL1  0x09
#define  ERAL2  0x00
#define  WRAL1  0x08
#define  WRAL2  0x80
#define  EWDS1  0x08
#define  EWDS2  0x00
#define  W_CS   LATCbits.LATC2

//**********************************
void main(void)
{
 TRISCbits.TRISC2 = 0;
 W_CS = 0;            // обеспечивают CS отрицается
 OpenMwire(MWIRE_FOSC_16);    // включить SSP периферийных
 ew_enable();         // отправить стирания / записи позволяют
 write_byte(0x13, 0x34); // байт записи (адрес, данные)
 busy_poll();
 Nop();
 byte_read(0x13);      // читать один байт (адрес)
```

```
http://www.microchip.com/
  read_mult(0x10, arrayrd, 10); // читать несколько байт
  erase_all();              // стереть весь массив
  CloseMwire();             // отключить SSP периферической
}

//-----------------------------------------
void ew_enable(void)
{
   W_CS = 1;       // утверждают чип выберите
   putcMwire(EWEN1); // включить запись байта команды 1
   putcMwire(EWEN2); // включить запись байта команды 2
   W_CS = 0;       // отрицать чип выберите
}

//--------------------------------------
void busy_poll(void)
{
  W_CS = 1;
  while(! DataRdyMwire() );
  W_CS = 0;
}
//-----------------------------------------
void write_byte(unsigned char address,
             unsigned char data)
{
  W_CS = 1;
  putcMwire(WRITE);    // команда записи
  putcMwire(address);  // адрес
  putcMwire(data);     // написать один байт
```

```
http://www.microchip.com/
  W_CS = 0;
}
//-----------------------------------------
void byte_read(unsigned char address)
{
  W_CS = 1;
  getcMwire(READ,address); // читать один байт
  W_CS = 0;
}

//-------------------------------------
void read_mult(unsigned char address,
            unsigned char *rdptr,
            unsigned char length)
{
  W_CS = 1;
  putcMwire(READ);        // команды чтения
  putcMwire(address);     // адрес (A7 - A0)
  getsMwire(rdptr, length); // читать несколько байт
  W_CS = 0;
}

//-----------------------------------------------
void erase_all(void)
{
  W_CS = 1;
  putcMwire(ERAL1); // стереть все байта команды 1
  putcMwire(ERAL2); // стереть все байта команды 2
  W_CS = 0;
```

http://www.microchip.com/

}

//**********************************

2,7 широтно-импульсной модуляции ФУНКЦИИ

ШИМ периферической поддерживается со следующими функциями:

Таблица 2-9: ШИМ ФУНКЦИИ

Функция Описание

ClosePWMx Отключить PWM канала x.

OpenPWMx Настройка ШИМ канала x.

SetDCPWMx Написать новое значение рабочего цикла к ШИМ канала x.

SetOutputPWMx Устанавливает биты конфигурации выходных ШИМ для ECCP x.

CloseEPWMx (1) Отключить усиленный ШИМ канала x.

OpenEPWMx (1) Настройка усиливается PWM канала x.

SetDCEPWMx (1) Написать новое значение рабочего цикла к расширенной ШИМ канала x.

SetOutputEPWMx (1) Устанавливает расширенные биты конфигурации выходных ШИМ для ECCPx.

Примечание 1: Расширенные функции PWM доступны только на этих устройствах с ECCPxCON зарегистрируйтесь.

//**********************************

2.7.1 Описания функций

ClosePWM1

ClosePWM2

ClosePWM3

ClosePWM4

ClosePWM5

http://www.microchip.com/

CloseEPWM1

Функция: Отключить PWM канала.

Включает в себя: pwm.h

Прототип:

void ClosePWM1(void);

void ClosePWM2(void);

void ClosePWM3(void);

void ClosePWM4(void);

void ClosePWM5(void);

void CloseEPWM1(void);

Примечания: Эта функция отключает указанные PWM канал.

Имя файла:

pw1close.c

pw2close.c

pw3close.c

pw4close.c

pw5close.c

ew1close.c

//**********

OpenPWM1

OpenPWM2

OpenPWM3

OpenPWM4

OpenPWM5

OpenEPWM1

Функция: Настройка ШИМ канала.

Включает в себя: pwm.h

http://www.microchip.com/

Прототип:

void OpenPWM1(char period);

void OpenPWM2(char period);

void OpenPWM3(char period);

void OpenPWM4(char period);

void OpenPWM5(char period);

void OpenEPWM1(char period);

Аргументы:

период

Может быть любое значение от 0x00 до 0xFF. Это значение определяет PWM Частота по следующей формуле:

PWM период = [(период) + 1] x 4 x TOSC x TMR2 предделителя

Примечания: Эта функция настраивает указанный ШИМ канала для периода и для временная база. ШИМ использует только ТАЙМЕР2.

В дополнение к открытию PWM, Таймер2 также должны быть открыты с OpenTimer2 (...) заявление перед PWM будет работать.

Имя файла:

pw1open.c

pw2open.c

pw3open.c

pw4open.c

pw5open.c

ew1open.c

Пример кода: OpenPWM1 (0xff);

//**********

SetDCPWM1

SetDCPWM2

SetDCPWM3

http://www.microchip.com/

SetDCPWM4

SetDCPWM5

SetDCEPWM1

Функция: Написать новое значение рабочего цикла на указанный PWM канала рабочим циклом регистры.

Включает в себя: pwm.h

Прототип:

void SetDCPWM1(unsigned int dutycycle);

void SetDCPWM2(unsigned int dutycycle);

void SetDCPWM3(unsigned int dutycycle);

void SetDCPWM4(unsigned int dutycycle);

void SetDCPWM5(unsigned int dutycycle);

void SetDCEPWM1(unsigned int dutycycle);

Аргументы:

dutycycle

Значение dutycycle может быть любой 10-битный номер. Только нижняя 10-бит dutycycle записываются в регистры рабочего цикла. обязанность цикла, или, более конкретно самое время ШИМ сигнала, может быть рассчитывают по следующей формуле:

ШИМ х Рабочий цикл = (DCx <9: 0>) x TOSC

где DCx <9: 0> является 10-битное значение указано в вызове этой функции.

Примечания: Данная функция позволяет записывать новое значение для dutycycle на указанный PWM регистры рабочий цикл канала.

Максимальное разрешение ШИМ сигнала может быть вычислена по Период с использованием следующей формулы:

Разрешение (бит) = Журнал (FOSC / и fШИМ) / журнал (2)

Имя файла:

pw1setdc.c

pw2setdc.c

pw3setdc.c

pw4setdc.c

pw5setdc.c

http://www.microchip.com/

ew1setdc.c

Пример кода: **SetDCPWM1** (0);

//**********************************

SetOutputPWM1

SetOutputPWM2

SetOutputPWM3

SetOutputEPWM1

Функция: Устанавливает биты конфигурации выходных ШИМ для ECCP.

Включает в себя: pwm.h

Прототип:

```
void SetOutputPWM1 (
    unsigned char outputconfig,
    unsigned char outputmode);
void SetOutputPWM2 (
    unsigned char outputconfig,
    unsigned char outputmode);
void SetOutputPWM3 (
    unsigned char outputconfig,
    unsigned char outputmode);
void SetOutputEPWM1 (
    unsigned char outputconfig,
    unsigned char outputmode);
```

Аргументы:

outputconfig

Значение outputconfig может быть любой из следующих величин (определено в pwm.h):

SINGLE_OUT один выход

FULL_OUT_FWD выход полный мост вперед

http://www.microchip.com/

HALF_OUT полумостовой выход

FULL_OUT_REV выхода от обратной полный мост

OutputMode

Значение OutputMode может быть любой из следующих величин (определено в pwm.h):

PWM_MODE_1 P1A и P1C активно-высокий,

P1B и P1D активно-высокий

PWM_MODE_2 P1A и P1C активно-высокий,

P1B и P1D активным низким

PWM_MODE_3 P1A и P1C активным низким,

P1B и P1D активно-высокий

PWM_MODE_4 P1A и P1C активным низким,

P1B и P1D активным низким

Примечания: Это применимо только к тем устройств с расширенными или Enhanced КПК (ЕКПП).

Имя файла:

pw1setoc.c

pw2setoc.c

pw3setoc.c

ew1setoc.c

Пример кода: **SetOutputPWM1** (SINGLE_OUT, PWM_MODE_1);

//**********

2.8 ФУНКЦИИ SPI ™

Следующие процедуры предназначены для устройств с одной SPI периферией:

Таблица 2-10: ОТДЕЛЬНЫЕ ФУНКЦИИ SPI ПЕРИФЕРИЙНЫЕ

http://www.microchip.com/

Функция Описание

CloseSPI Отключить модуль SSP используется для SPI коммуникаций.

DataRdySPI Определите, если новое значение доступно из буфера SPI.

getcSPI Считывает байт из автобуса SPI.

getsSPI Читать строку из автобуса SPI.

OpenSPI Инициализация модуля SSP используется для SPI коммуникаций.

putcSPI Записывает байт в шине в.

putsSPI Записать строку в шине в.

ReadSPI Считывает байт из автобуса SPI.

WriteSPI Записывает байт в шине в.

Следующие функции предназначены для устройств с несколькими периферийными устройствами SPI:

//**********

Таблица 2-11: несколько функций SPI ПЕРИФЕРИЙНЫЕ

Функция Описание

CloseSPIx Отключить модуль SSPX используемый для SPI коммуникаций.

DataRdySPIx Определите, если новое значение доступно из буфера Spix.

getcSPIx Считывает байт из автобуса Spix.

getsSPIx Читать строку из автобуса Spix.

OpenSPIx инициализацию SSPX модуль, используемый для SPI коммуникаций.

putcSPIx Записывает байт в автобусе Spix.

putsSPIx Записать строку в автобусе Spix.

ReadSPIx Считывает байт из автобуса Spix.

WriteSPIx Записывает байт в автобусе Spix.

//**********

2.8.1 Описания функций

http://www.microchip.com/

CloseSPI

CloseSPI1

CloseSPI2

Функция: Отключить модуль SSPX.

Включает в себя: spi.h

Прототип:

void CloseSPI(void);

void CloseSPI1(void);

void CloseSPI2(void);

Примечания: Эта функция отключает модуль SSPx. Возвращается Pin ввода / вывода под контролем соответствующих TRIS и LAT регистров.

Имя файла:

spi_clos.c

spi1clos.c

spi2clos.c

//**********

DataRdySPI

DataRdySPI1

DataRdySPI2

Функция: Определите, SSPBUFx содержит данные.

Включает в себя: spi.h

Прототип:

unsigned char DataRdySPI(void);

unsigned char DataRdySPI1(void);

unsigned char DataRdySPI2(void);

Примечания: Эта функция определяет, имеется ли байт для чтения из SSPBUFx зарегистрируйтесь.

http://www.microchip.com/

Возвращаемое значение: 0, если нет данных в регистре SSPBUFx 1, если есть данные в реестре SSPBUFx

Имя файла:

spi_dtrd.c

spi1dtrd.c

spi2dtrd.c

Пример кода: while (!**DataRdySPI**());

//**********

getcSPI

getcSPI1

getcSPI2

getcSPIx определяется как ReadSPIx. Смотреть ReadSPIx.

getsSPI

getsSPI1

getsSPI2

Функция: Чтение строки из автобуса Spix.

Включает в себя: spi.h

Прототип:

void getsSPI(unsigned char *rdptr,

unsigned char length);

void getsSPI1(unsigned char *rdptr,

unsigned char length);

void getsSPI2(unsigned char *rdptr,

unsigned char length);

Аргументы:

rdptr

http://www.microchip.com/

Указатель на месте для хранения данных, считанных из SPIx устройства.

длина

Количество байт для чтения из SPIx устройства.

Примечания: Эта функция читает в заданной длины строки данных от SPIx автобус.

Имя файла: spi_gets.c

spi1gets.c

spi2gets.c

Пример кода:

```
unsigned char wrptr(10);

getsSPI (wrptr, 10);

//************************************
```

OpenSPI

OpenSPI1

OpenSPI2

Функция: Инициализация модуля SSPX.

Включает в себя: spi.h

Прототип:

```
void OpenSPI( unsigned char sync_mode,
        unsigned char bus_mode,
        unsigned char smp_phase);
void OpenSPI1( unsigned char sync_mode,
        unsigned char bus_mode,
        unsigned char smp_phase);
void OpenSPI2( unsigned char sync_mode,
        unsigned char bus_mode,
        unsigned char smp_phase);
```

Аргументы:

http://www.microchip.com/

Sync_Mode

Один из следующих значений, определенных в spi.h:

Режим SPI_FOSC_4 ведущего SPI, тактовый сигнал = FOSC / 4

Режим SPI_FOSC_16 ведущего SPI, тактовый сигнал = FOSC / 16

Режим SPI_FOSC_64 ведущего SPI, тактовый сигнал = FOSC / 64

Режим SPI_FOSC_TMR2 ведущего SPI, часы = выход TMR2 / 2

Режим SLV_SSON ведомого SPI, / контроль контактный CC включен

Режим SLV_SSOFF ведомого SPI, / контроль контактный CC отключена

SPI_FOSC_4 SPI Master mode, clock = FOSC/4

SPI_FOSC_16 SPI Master mode, clock = FOSC/16

SPI_FOSC_64 SPI Master mode, clock = FOSC/64

SPI_FOSC_TMR2 SPI Master mode, clock = TMR2 output/2

SLV_SSON SPI Slave mode, /SS pin control enabled

SLV_SSOFF SPI Slave mode, /SS pin control disabled

bus_mode

Один из следующих значений, определенных в spi.h:

MODE_00 Setting for SPI bus Mode 0,0

MODE_01 Setting for SPI bus Mode 0,1

MODE_10 Setting for SPI bus Mode 1,0

MODE_11 Setting for SPI bus Mode 1,1

MODE_00 настройка режима автобус SPI 0,0

MODE_01 настройка режима автобус SPI 0,1

MODE_10 настройка режима автобус SPI 1,0

MODE_11 настройка режима автобус SPI 1,1

smp_phase

http://www.microchip.com/

Один из следующих значений, определенных в spi.h:

SMPEND Ввод образца данных в конце данных вне

SMPMID Ввод образца данных на середине данных вне

Примечания: Эта функция настраивает SSPX модуль для использования с автобусной устройства Spix.

Имя файла:

spi_open.c

spi1open.c

spi2open.c

Пример кода: **OpenSPI** (SPI_FOSC_16, MODE_00, SMPEND);

//**********************************

putcSPI

putcSPI1

putcSPI2

putcSPIx определяется как WriteSPIx. Смотреть WriteSPIx.

putsSPI

putsSPI1

putsSPI2

Функция: Записать строку в автобусе Spix.

Включает в себя: spi.h

Прототип:

void putsSPI(unsigned char *wrptr);

void putsSPI1(unsigned char *wrptr);

void putsSPI2(unsigned char *wrptr);

Аргументы:

wrptr

Указатель на значение, которое будет записано в автобусе Spix.

http://www.microchip.com/

Примечания: Эта функция записывает строку данных с автобусной устройства Spix.регулярные; зубцы завершается чтение нулевой символ в строке данных (в нуль символ не записывается в автобусе).

Имя файла:

spi_puts.c

spi1puts.c

spi2puts.c

Пример кода:

unsigned char wrptr[] = “Hello!”;

putsSPI (wrptr);

//**********

ReadSPI

ReadSPI1

ReadSPI2

getcSPI

getcSPI1

getcSPI2

Функция: Считывает байт из автобуса Spix.

Включает в себя: spi.h

Прототип:

unsigned char ReadSPI(void);

unsigned char ReadSPI1(void);

unsigned char ReadSPI2(void);

unsigned char getcSPI(void);

unsigned char getcSPI1(void);

unsigned char getcSPI2(void);

Примечания: Эта функция инициирует автобус цикл Spix для приобретения байт Данные. **getcSPIx** определяется для **ReadSPIx** в spi.h.

Вернуться Значение: Эта функция возвращает байт данных, считанных во время Spix цикла чтения.

http://www.microchip.com/

Имя файла:

spi_read.c

spi1read.c

spi2read.c

#define в spi.h

#define в spi.h

#define в spi.h

Пример кода:

```
char x;

x = ReadSPI();

//************************************
```

WriteSPI

WriteSPI1

WriteSPI2

putcSPI

putcSPI1

putcSPI2

Функция: Написать байт в автобусе Spix.

Включает в себя: spi.h

Прототип:

```
unsigned char WriteSPI(
        unsigned char data_out );
unsigned char WriteSPI1(
        unsigned char data_out );
unsigned char WriteSPI2(
        unsigned char data_out );
unsigned char putcSPI(
        unsigned char data_out );
```

http://www.microchip.com/

unsigned char putcSPI1(

unsigned char data_out);

unsigned char putcSPI2(

unsigned char data_out);

Аргументы:

data_out

Значение должно быть записано в автобусе Spix.

Примечания: Эта функция записывает один байт данных, и затем проверяет наличие записи столкновение. **putcSPIx** определяется для **WriteSPIx** в spi.h.

Возвращаемое значение:

0, если не произошло столкновение записи

-1, Если произошла коллизия записи

Имя файла:

spi_writ.c

spi1writ.c

spi2writ.c

#define в spi.h

#define в spi.h

#define в spi.h

Пример кода:

WriteSPI (");

//**********

2.8.2 Пример применения

Следующий пример демонстрирует использование модуля SSP общаться с Microchip 25C080 SPI EE Устройство памяти.

```
http://www.microchip.com/
#include <p18cxxx.h>
#include <spi.h>
// FUNCTION Prototypes
void main(void);
void set_wren(void);
void busy_polling(void);
unsigned char status_read(void);
void status_write(unsigned char data);
//-------------------------------------------------------------

void byte_write(unsigned char addhigh,
                unsigned char addlow,
                unsigned char data);
//-------------------------------------------------------------

void page_write(unsigned char addhigh,
                unsigned char addlow,
                unsigned char *wrptr);
//-------------------------------------------------------------

void array_read(unsigned char addhigh,
                unsigned char addlow,
                unsigned char *rdptr,
                unsigned char count);
//-------------------------------------------------------------

unsigned char byte_read(unsigned char addhigh,
                        unsigned char addlow);
// Определения переменных
```

```
http://www.microchip.com/
unsigned char arraywr[] = {1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,0};
// 25C040 / 080/160 страница размер записи
unsigned char arrayrd[16];
unsigned char var;
#define SPI_CS  LATCbits.LATC2
// ********************************************** **


void main(void)
{
  TRISCbits.TRISC2 = 0;
  SPI_CS = 1;  // Обеспечить устройство памяти SPI
               // Chip Select сбрасывается
  OpenSPI(SPI_FOSC_16, MODE_00, SMPEND);
  set_wren();
  status_write(0);
  busy_polling();
  set_wren();
  byte_write(0x00, 0x61, 'E');
  busy_polling();
  var = byte_read(0x00, 0x61);
  set_wren();
  page_write(0x00, 0x30, arraywr);
  busy_polling();
  array_read(0x00, 0x30, arrayrd, 16);
  var = status_read();
  CloseSPI();
  while(1);
//-------------------------------------------------------------
void set_wren(void)
```

```
http://www.microchip.com/
{
  SPI_CS = 0;             // утверждают чип выберите
  var = putcSPI(SPI_WREN);  // отправить разрешения записи команду
  SPI_CS = 1;             // отрицать чип выберите
}
//-------------------------------------------------------------
void page_write (unsigned char addhigh,
                 unsigned char addlow,
                 unsigned char *wrptr)
{
  SPI_CS = 0;               // утверждают чип выберите
  var = putcSPI(SPI_WRITE);   // отправить команду записи
  var = putcSPI(addhigh);     // отправить старший байт адреса
  var = putcSPI(addlow);      // отправить младший байт адреса
  putsSPI(wrptr);             // отправить байт данных
  SPI_CS = 1;               // отрицать чип выберите
}
//-------------------------------------------------------------

void array_read (unsigned char addhigh,
                 unsigned char addlow,
                 unsigned char *rdptr,
                 unsigned char count)
{
  SPI_CS = 0;             // утверждают чип выберите
  var = putcSPI(SPI_READ); // отправить команду чтения
  var = putcSPI(addhigh);  // отправить старший байт адреса
  var = putcSPI(addlow);   // отправить младший байт адреса
  getsSPI(rdptr, count);   // читать несколько байт
```

```
http://www.microchip.com/
  SPI_CS = 1;
}
//-------------------------------------------------------------


void byte_write (unsigned char addhigh,
                 unsigned char addlow,
                 unsigned char data)
{
  SPI_CS = 0;              // утверждают чип выберите
  var = putcSPI(SPI_WRITE); // отправить команду записи
  var = putcSPI(addhigh);   // отправить старший байт адреса
  var = putcSPI(addlow);    // отправить младший байт адреса
  var = putcSPI(data);      // отправить байт данных
  SPI_CS = 1;              // отрицать чип выберите
}
//-------------------------------------------------------------


unsigned char byte_read (unsigned char addhigh,
                         unsigned char addlow)
{
  SPI_CS = 0;              // утверждают чип выберите
  var = putcSPI(SPI_READ);  // отправить команду чтения
  var = putcSPI(addhigh);   // отправить старший байт адреса
  var = putcSPI(addlow);    // отправить младший байт адреса
  var = getcSPI();          // читать одного байта
  SPI_CS = 1;
  return (var);
}
//-------------------------------------------------------------
```

```
http://www.microchip.com/
unsigned char status_read (void)
{
  SPI_CS = 0;             // утверждают чип выберите
  var = putcSPI(SPI_RDSR); // отправить команду чтения статус
  var = getcSPI();        // читать байт данных
  SPI_CS = 1;             // отрицать чип выберите
  return (var);
}
//-------------------------------------------------------------


void status_write (unsigned char data)
{
  SPI_CS = 0;
  var = putcSPI(SPI_WRSR); // команда статус записи
  var = putcSPI(data);     // байт состояния писать
  SPI_CS = 1;             // отрицать чип выберите
}
//-------------------------------------------------------------


void busy_polling (void)
{
  do
  {
    SPI_CS = 0;             // утверждают чип выберите
    var = putcSPI(SPI_RDSR); // отправить команду чтения статус
    var = getcSPI();        // читать байт данных
    SPI_CS = 1;             // отрицать чип выберите
  } while (var & 0x01);      // пребывание в петле до! занят
}
```

http://www.microchip.com/

//**********

2.9 Функции таймера

Периферийные устройства таймера поддерживаются со следующими функциями:

Таблица 2-12: Функции таймера

Функция Описание

CloseTimerx Отключить таймер x.

OpenTimerx настроить и включить таймер x.

ReadTimerx Прочитайте значение таймера x.

WriteTimerx Написать значение в таймер x.

//**********

2.9.1 Описания функций

CloseTimer0

CloseTimer1

CloseTimer2

CloseTimer3

CloseTimer4

Функция: Отключить указанный таймер.

Включает в себя: timers.h

Прототип:

void CloseTimer0(void);

void CloseTimer1(void);

void CloseTimer2(void);

void CloseTimer3(void);

void CloseTimer4(void);

Примечания: Эта функция отключает прерывание и указанный таймер.

http://www.microchip.com/

Имя файла:

t0close.c

t1close.c

t2close.c

t3close.c

t4close.c

//**********************************

OpenTimer0

Функция: Настройка и включить Timer0.

Включает в себя: timers.h

Прототип: void **OpenTimer0**(unsigned char config);

Аргументы:

конфигурации

Битовая маска, которая создается путем выполнения побитовой операции ('&') со значением от каждой из категорий, перечисленных ниже. Эти значения определены в файле timers.h.

Включить Timer0 прерывание:

Включен TIMER_INT_ON прерывания

TIMER_INT_OFF прерывания отключены

Таймер Ширина:

Режим T0_8BIT 8-бит

Режим T0_16BIT 16 бит

Часы Источник:

T0_SOURCE_EXT Внешний источник синхронизации (I / O контактный) Источник

T0_SOURCE_INT Внутренние часы (TOSC)

Внешняя синхронизация триггера (для T0_SOURCE_EXT):

T0_EDGE_FALL внешней синхронизации по заднему фронту

T0_EDGE_RISE Внешний часы на переднем фронте

http://www.microchip.com/

Заданное значение:

T0_PS_1_1 1: 1 предделителя

T0_PS_1_2 1: 2 предделителя

T0_PS_1_4 1: 4 предделителя

T0_PS_1_8 1: 8 предделителя

T0_PS_1_16 1:16 предделителя

T0_PS_1_32 1:32 предделителя

T0_PS_1_64 1:64 предделителя

T0_PS_1_128 1: 128 предделителя

T0_PS_1_256 1: 256 предделителя

Примечания: Эта функция настраивает timer0 в соответствии с параметрами, указанными и то позволяет это.

Имя файла: t0open.c

Пример кода:

OpenTimer0 (TIMER_INT_OFF &

T0_8BIT &

T0_SOURCE_INT &

T0_PS_1_32);

//**********

OpenTimer1

Функция: Настройка и включить timer1.

Включает в себя: timers.h

Прототип: void **OpenTimer1**(unsigned char config);

Аргументы:

конфигурации

Битовая маска, которая создается путем выполнения побитовой операции ('&') со значением от каждой из категорий, перечисленных ниже. Эти значения определены в файле timers.h.

http://www.microchip.com/

Включить Timer1 прерывание:

Включен TIMER_INT_ON прерывания

TIMER_INT_OFF прерывания отключены

Таймер Ширина:

Режим T1_8BIT_RW 8-бит

Режим T1_16BIT_RW 16 бит

Часы Источник:

T1_SOURCE_EXT Внешний источник синхронизации (I / O контактный)

Источник T1_SOURCE_INT Внутренние часы (TOSC)

Prescaler:

T1_PS_1_1 1: 1 предделителя

T1_PS_1_2 1: 2 предделителя

T1_PS_1_4 1: 4 предделителя

T1_PS_1_8 1: 8 предделителя

Осциллятор использования:

T1_OSC1EN_ON Включить Timer1 генератор

T1_OSC1EN_OFF Отключить Таймер1 генератор

Синхронизация часов Вход:

T1_SYNC_EXT_ON Синхронизация внешний вход часы

T1_SYNC_EXT_OFF Не синхронизировать внешний вход синхронизации

Примечания: Эта функция настраивает timer1 в соответствии с параметрами, указанными и то позволяет это.

Имя файла: t1open.c

Пример кода:

OpenTimer1 (TIMER_INT_ON &

T1_8BIT_RW &

T1_SOURCE_EXT &

T1_PS_1_1 &

T1_OSC1EN_OFF &

http://www.microchip.com/

T1_SYNC_EXT_OFF &

T1_SOURCE_CCP);

//**********

OpenTimer2

Функция: Настройка и включить ТАЙМЕР2.

Включает в себя: timers.h

Прототип: void **OpenTimer2**(unsigned char config);

Аргументы:

конфигурации

Битовая маска, которая создается путем выполнения побитовой операции ('&') со значением от каждой из категорий, перечисленных ниже. Эти значения определены в файле timers.h.

Включить Timer2 прерывание:

TIMER_INT_ON прерывания Включен

TIMER_INT_OFF прерывания отключены

Заданное значение:

T2_PS_1_1 1: 1 предделителя

T2_PS_1_4 1: 4 предделителя

T2_PS_1_16 1:16 предделителя

Postscale Значение:

T2_POST_1_1 1: 1 postscale

T2_POST_1_2 1: 2 postscale

::

T2_POST_1_15 1:15 postscale

T2_POST_1_16 1:16 postscale

Примечания: Эта функция настраивает Таймер2 в соответствии с параметрами, указанными и то позволяет это.

http://www.microchip.com/

Имя файла: t2open.c

Пример кода:

OpenTimer2 (TIMER_INT_OFF &

T2_PS_1_1 &

T2_POST_1_8);

//**********************************

OpenTimer3

Функция: Настройка и включить timer3.

Включает в себя: timers.h

Прототип: void **OpenTimer3**(unsigned char config);

Аргументы:

конфигурации

Битовая маска, которая создается путем выполнения побитовой операции ('&') со значением от каждой из категорий, перечисленных ниже. Эти значения определены в файле timers.h.

Включить timer3 прерывание:

Включен TIMER_INT_ON прерывания

TIMER_INT_OFF прерывания отключены

Таймер Ширина:

Режим T3_8BIT_RW 8-бит

Режим T3_16BIT_RW 16 бит

Часы Источник:

T3_SOURCE_EXT Внешний источник синхронизации (I / O контактный)

T3_SOURCE_INT Источник Внутренние часы (TOSC)

Заданное значение:

T3_PS_1_1 1: 1 предделителя

T3_PS_1_2 1: 2 предделителя

T3_PS_1_4 1: 4 предделителя

http://www.microchip.com/

T3_PS_1_8 1: 8 предделителя

Синхронизация часов Вход:

T3_SYNC_EXT_ON Синхронизация внешний вход часы

T3_SYNC_EXT_OFF Не синхронизировать внешний вход синхронизации

Использование С CCP:

Для устройств с 1 или 2 CCP

T3_SOURCE_CCP Timer3 источником для обоих CCP

T1_CCP1_T3_CCP2 Таймер1 источником CCP1 и Timer3 источником CCP2

T1_SOURCE_CCP Таймер1 источником для обоих CCP

Для устройств с более чем 2 CCP

T34_SOURCE_CCP Timer3 и Timer4 являются источниками для всех CCP

T12_CCP12_T34_CCP345 Таймер1 ADN Таймер2 являются источниками

CCP1 и CCP2 и Timer3 и Timer4 являются источниками для CCP3 throughCCP5

T12_CCP1_T34_CCP2345 Таймер1 и Таймер2 являются источниками CCP1 и Timer3 и Timer4 являются источники CCP2 через CCP5

T12_SOURCE_CCP Таймер1 и Таймер2 являются источниками для всех CCP

Примечания: Эта функция настраивает timer3 в соответствии с параметрами, указанными и то позволяет это.

Имя файла: t3open.c

Пример кода:

```
OpenTimer3 (TIMER_INT_ON &
        T3_8BIT_RW &
        T3_SOURCE_EXT &
        T3_PS_1_1 &
        T3_OSC1EN_OFF &
        T3_SYNC_EXT_OFF &
        T3_SOURCE_CCP);
//**********************************
```

http://www.microchip.com/

OpenTimer4

Функция: Настройка и включить timer4.

Включает в себя: timers.h

Прототип: void **OpenTimer4**(unsigned char config);

Аргументы:

конфигурации

Битовая маска, которая создается путем выполнения побитовой операции ('&') со значением от каждой из категорий, перечисленных ниже. Эти значения определены в файле timers.h.

Включить Timer4 прерывание:

TIMER_INT_ON прерывания Включен

TIMER_INT_OFF прерывания отключены

Заданное значение:

T4_PS_1_1 1: 1 предделителя

T4_PS_1_4 1: 4 предделителя

T4_PS_1_16 1:16 предделителя

Postscale Значение:

T4_POST_1_1 1: 1 postscale

T4_POST_1_2 1: 2 postscale

::

T4_POST_1_15 1:15 postscale

T4_POST_1_16 1:16 postscale

Примечания: Эта функция настраивает timer4 в соответствии с параметрами, указанными и

то позволяет это.

Имя файла: t4open.c

Пример кода:

OpenTimer4 (TIMER_INT_OFF &

T4_PS_1_1 &

T4_POST_1_8);

http://www.microchip.com/

//**********

ReadTimer0

ReadTimer1

ReadTimer2

ReadTimer3

ReadTimer4

Функция: Прочитайте значение заданного таймером.

Включает в себя: timers.h

Прототип:

```
unsigned int  ReadTimer0( void );
unsigned int  ReadTimer1( void );
unsigned char ReadTimer2( void );
unsigned int  ReadTimer3( void );
unsigned char ReadTimer4( void );
```

Примечания: Эти функции чтения значения соответствующего реестра (ы) таймера.

Timer0: TMR0L, TMR0H

Таймер1: TMR1L, TMR1H

Таймер2: TMR2

Таймер 3: TMR3L, TMR3H

Timer4: TMR4

Примечание: При использовании таймера в 8-битном режиме, что может быть настроен в 16-битный режим (например, timer0), старший байт не гарантируется равным нулю. Пользователь может пожелать привести результат к гольца для правильных результатов. Для пример:

```
 // Пример чтения 16-битный результат
 // От операционной 16-разрядный таймер в
 // 8-битный режим:
unsigned int result;
```

http://www.microchip.com/

result = (unsigned char) ReadTimer0();

Вернуться Значение: текущее значение таймера.

Имя файла:

t0read.c

t1read.c

t2read.c

t3read.c

t4read.c

//**********

WriteTimer0

WriteTimer1

WriteTimer2

WriteTimer3

WriteTimer4

Функция: Написать значение в указанном таймера.

Включает в себя: timers.h

Прототип:

void WriteTimer0(unsigned int timer);

void WriteTimer1(unsigned int timer);

void WriteTimer2(unsigned char timer);

void WriteTimer3(unsigned int timer);

void WriteTimer4(unsigned char timer);

Аргументы:

таймер

Значение, которое будет загружен в указанном таймера.

Примечания: Эти функции записи значения в регистр (ы) соответствующего таймера:

Timer0: TMR0L, TMR0H

Таймер1: TMR1L, TMR1H

http://www.microchip.com/

Таймер2: TMR2

Таймер 3: TMR3L, TMR3H

Timer4: TMR4

Имя файла:

t0write.c

t1write.c

t2write.c

t3write.c

t4write.c

Пример кода: **WriteTimer0** (10000);

//**********************************

2.9.2 Пример применения

```
#include <p18C452.h>
#include <timers.h>
#include <usart.h>
#include <stdlib.h>
void main( void )
{
 int result;
 char str[7];
  // Настроить timer0
 OpenTimer0( TIMER_INT_OFF &
          T0_SOURCE_INT  &
          T0_PS_1_32 );
  // Настроить USART
  OpenUSART (USART_TX_INT_OFF &
```

```
http://www.microchip.com/
        USART_RX_INT_OFF &
        USART_ASYNCH_MODE &
        USART_EIGHT_BIT &
        USART_CONT_RX,
        25);

 while( 1 )
 {
   while( ! PORTBbits.RB3 ); // Ждут RB3 высокой
   result = ReadTimer0();    // Читать таймер
   if( result > 0xc000 )     // выход из цикла, если значение
     break;                  //  Вне диапазона
   WriteTimer0( 0 );         // Перезапустить таймер
   ultoa( result, str );     // Конвертировать таймер в строку
   putsUSART( str );         // Печати строка
 }
 CloseTimer0();              // close modules
 CloseUSART();
}

//************************************
```

2,10 USART ФУНКЦИИ

Следующие функции предназначены для устройств с одной USART периферии:

Таблица 2-13: ОТДЕЛЬНЫЕ ФУНКЦИИ USART ПЕРИФЕРИЙНЫЕ

Функция Описание

http://www.microchip.com/

BusyUSART ли USART передачи?

CloseUSART Отключить USART.

DataRdyUSART Данные предоставляется в USART читать буфер?

getcUSART Считывает байт из УСАПП.

getsUSART Считать строку из USART.

OpenUSART Настройте USART.

putcUSART Написать байт в USART.

putsUSART Написать строки из памяти данных в USART.

putrsUSART Написать строки из памяти программ к USART.

ReadUSART Считывает байт из УСАПП.

WriteUSART Написать байт в USART.

baudUSART указан биты конфигурации скорость передачи данных для повышения USART.

Следующие функции предназначены для устройств с несколькими периферийными USART:

Таблица 2-14: несколько функций USART ПЕРИФЕРИЙНЫЕ

Функция Описание

BusyxUSART ли USART x передающая?

ClosexUSART Отключить USART x.

DataRdyxUSART ли данные в буфере считывания USART x?

getcxUSART Считывает байт из USART x.

getsxUSART Читать строку из USART x.

OpenxUSART Настройка USART x.

putcxUSART Записывает байт в USART x.

putsxUSART Написать строки из памяти данных в USART x.

putrsxUSART Написать строки из памяти программ к USART x.

ReadxUSART Считывает байт из USART x.

WritexUSART Записывает байт в USART x.

baudxUSART указан биты конфигурации скорость передачи данных для повышения USART x.

http://www.microchip.com/

//**********

2.10.1 Описание функций

BusyUSART

Busy1USART

Busy2USART

Функция: Каково USART передачи?

Включает в себя: usart.h

Прототип:

char BusyUSART(void);

char Busy1USART(void);

char Busy2USART(void);

Примечания: Возвращает значение, указывающее, если передатчик USART в настоящее время занят.

Эта функция должна использоваться до начала новую передачу.

BusyUSART должны быть использованы на части с одним USART периферии.

Busy1USART и Busy2USART должны быть использованы на части с несколькими

Периферия USART.

Возвращаемое значение:

0, если передатчик USART простаивает

1, если передатчик USART используется

Имя файла:

ubusy.c

u1busy.c

u2busy.c

http://www.microchip.com/

Пример кода:

```
while (BusyUSART());

//************************************
```

CloseUSART

Close1USART

Close2USART

Функция: Отключить указанный USART.

Включает в себя: usart.h

Прототип:

```
void CloseUSART(  void );

void Close1USART( void );

void Close2USART( void );
```

Примечания: Эта функция запрещает прерывания, передатчик и приемник для указано USART.

CloseUSART должны быть использованы на части с одним USART периферии.

Close1USART и **Close2USART** должны быть использованы на части с несколько периферия USART.

Имя файла:

uclose.c

u1close.c

u2close.c

```
//************************************
```

DataRdyUSART

DataRdy1USART

DataRdy2USART

Функция: данные доступные в буфере чтения?

Включает в себя: usart.h

http://www.microchip.com/

Прототип:

char DataRdyUSART(void);

char DataRdy1USART(void);

char DataRdy2USART(void);

Примечания: Эта функция возвращает статус флага RCIF бит регистра PIR.

DataRdyUSART должны быть использованы на части с одним USART периферическая. **DataRdy1USART** и **DataRdy2USART** должны быть использованы на части с несколькими периферийными USART.

Возвращаемые значения:

1, если есть данные

0, если данные не доступны

Имя файла:

udrdy.c

u1drdy.c

u2drdy.c

Пример кода: while (!DataRdyUSART());

//**********************************

getcUSART

getc1USART

getc2USART

getcxUSART определяется как ReadxUSART. Смотреть ReadUSART

getsUSART

gets1USART

gets2USART

Функция: Читать фиксированной длины строки символов из указанного USART.

Включает в себя: usart.h

Прототип:

http://www.microchip.com/

void getsUSART (char * buffer,

unsigned char len);

void gets1USART (char * buffer,

unsigned char len);

void gets2USART (char * buffer,

unsigned char len);

Аргументы:

буфер

Указатель на месте, где входящие символы должны быть сохранены.

len

Количество символов для чтения из USART.

Примечания: Эта функция ждет и читает Лен количество символов из указано USART. Там нет времени, когда ждет символов приехать.

getsUSART должны быть использованы на части с одним USART периферии.

gets1USART и **gets2USART** должны быть использованы на части с несколькими Периферия USART.

Имя файла:

ugets.c

u1gets.c

u2gets.c

Пример кода:

char inputstr[10];

getsUSART(inputstr, 5);

//**********

OpenUSART

Open1USART

Open2USART

http://www.microchip.com/

Функция: Настройка указанный модуль USART.

Включает в себя: usart.h

Прототип:

```
void OpenUSART( unsigned char config,
                unsigned int spbrg);
void Open1USART( unsigned char config,
                 unsigned int spbrg);
void Open2USART( unsigned char config,
                 unsigned int spbrg);
```

Аргументы:

конфигурации

Битовая маска, которая создается путем выполнения побитовой операции ('&') со значением от каждой из категорий, перечисленных ниже. Эти значения определены в файле usart.h.

Прерывание на передаче:

USART_TX_INT_ON прерывания от передатчика ВКЛ

USART_TX_INT_OFF прерывания от передатчика OFF

Прерывание работы при получении:

USART_RX_INT_ON Получать прерывания ПО

USART_RX_INT_OFF Получите прервать OFF

USART Режим:

Режим USART_ASYNCH_MODE Асинхронный

USART_SYNCH_MODE Режим синхронного

Передача Ширина:

USART_EIGHT_BIT 8-бит передачи / приема

USART_NINE_BIT 9-бит передачи / приема

Slave / Master Select *:

Режим USART_SYNC_SLAVE Синхронный ведомый

Режим USART_SYNC_MASTER Синхронный Мастер

Режим приема:

http://www.microchip.com/

USART_SINGLE_RX Одноместный прием

USART_CONT_RX Непрерывный прием

Скорость передачи данных:

USART_BRGH_HIGH Высокая скорость передачи данных

USART_BRGH_LOW Низкая скорость передачи

* Относится к синхронном режиме только

SPBRG

Это значение, которое записывается в регистр скорости передачи генератора, который определяет скорость передачи данных, при которой USART работает. формулы для скорости передачи данных являются:

Асинхронный режим, высокая скорость:

FOSC / (16 * (SPBRG +1))

Асинхронный режим, низкая скорость:

FOSC / (64 * (SPBRG +1))

Синхронный режим:

FOSC / (4 * (SPBRG +1))

Где FOSC является частота генератора.

Примечания: Эта функция настраивает модуль USART в соответствии с заданными Варианты конфигурации.

OpenUSART должны быть использованы на части с одним USART периферии.

Open1USART и Open2USART должны быть использованы на части с несколькими Периферия USART.

Имя файла:

uopen.c

u1open.c

u2open.c

Пример кода:

OpenUSART1 (USART_TX_INT_OFF &

USART_RX_INT_OFF &

USART_ASYNCH_MODE &

http://www.microchip.com/

```
        USART_EIGHT_BIT &
        USART_CONT_RX &
        USART_BRGH_HIGH,
        25);
//************************************
```

putcUSART

putc1USART

putc2USART

putcxUSART определяется как WritexUSART. Смотреть WriteUSART

putsUSART

puts1USART

puts2USART

putrsUSART

putrs1USART

putrs2USART

Функция: Записывает строку символов в USART включая нулевой символ.

Включает в себя: usart.h

Прототип:

```
void putsUSART(  char *data );
void puts1USART( char *data );
void puts2USART( char *data );
void putrsUSART(  const rom char *data );
void putrs1USART( const rom char *data );
void putrs2USART( const rom char *data );
```

Аргументы:

данные

Указатель на строку с нулевым символом данных.

http://www.microchip.com/

Примечания: Эта функция записывает строку данных в USART включая нулевой характер. Строки, расположенные в памяти данных следует использовать с "пут" версии из этих функций. Строки, расположенные в памяти программ, в том числе строковые литералы, должны быть используется с "putrs" версии этих функций.

putsUSART и **putrsUSART** должны быть использованы на части с одним

USART периферической. Другие функции должны быть использованы на части с несколько периферия USART.

Имя файла:

uputs.c

u1puts.c

u2puts.c

uputrs.c

u1putrs.c

u2putrs.c

Пример кода:

putrsUSART ("Hello World!");

//**********

ReadUSART

Read1USART

Read2USART

getcUSART

getc1USART

getc2USART

Функция: Считывает байт (один символ) из USART приемного буфера, в том числе 9-й бит, если включен.

Включает в себя: usart.h

Прототип:

char ReadUSART(void);

http://www.microchip.com/

```
char Read1USART( void );

char Read2USART( void );

char getcUSART(  void );

char getc1USART( void );

char getc2USART( void );
```

Примечания: Эта функция считывает байт из USART приемного буфера.Статус биты и девятый бит данных сохраняются в союзе со следующим декларация:

```
 union USART
 {
   unsigned char val;
   struct
   {
     unsigned RX_NINE:1;
     unsigned TX_NINE:1;
     unsigned FRAME_ERROR:1;
     unsigned OVERRUN_ERROR:1;
     unsigned fill:4;
   };
 };
```

9-й бит только для чтения, если 9-битный режим включен. Биты состояния являются всегда читаю.

На части с одной USART периферии, в getcUSART и Функции ReadUSART следует использовать и информация о статусе читать в переменной с именем USART_Status который имеет тип USART описано выше.

На части с несколькими периферийными USART, в getcxUSART и Функции ReadxUSART следует использовать и информация о статусе читать в переменной с именем USARTx_Status который типа USART описано выше.

Вернуться Значение: Эта функция возвращает следующий символ в USART приемного буфера.

Имя файла:

uread.c

u1read.c

u2read.c

http://www.microchip.com/

#define in usart.h

#define in usart.h

#define in usart.h

Пример кода:

```
int result;

result = ReadUSART();

result |= (unsigned int)

        USART_Status.RX_NINE << 8;

//************************************
```

WriteUSART

Write1USART

Write2USART

putcUSART

putc1USART

putc2USART

Функция: Написать байт (один символ) в буфер USART передачи, в том числе 9-й бит, если включен.

Включает в себя: usart.h

Прототип:

```
void WriteUSART(  char data );

void Write1USART( char data );

void Write2USART( char data );

void putcUSART(  char data );

void putc1USART( char data );

void putc2USART( char data );
```

Аргументы: данные Значение, которое будет записано в USART.

http://www.microchip.com/

Примечания: Эта функция записывает байт в буфер USART передачи. Если 9-битный режим включен, 9-й бит написано от поля TX_NINE, найти в переменная типа USART:

```
union USART
{
  unsigned char val;
  struct
  {
    unsigned RX_NINE:1;
    unsigned TX_NINE:1;
    unsigned FRAME_ERROR:1;
    unsigned OVERRUN_ERROR:1;
    unsigned fill:4;
  };
};
```

На части с одной USART периферии, в putcUSART и Функции WriteUSART следует использовать и регистр Статус является имени USART_Status которая от типа USART, описанной выше. На части с несколькими периферийными USART, в putcxUSART и Функции WritexUSART следует использовать и регистр Статус является по имени по имени USARTx_Status который имеет тип USART описанной выше.

Имя файла:

uwrite.c

u1write.c

u2write.c

#define в usart.h

#define в usart.h

#define в usart.h

Пример кода:

```
unsigned int outval;
USART1_Status.TX_NINE = (outval & 0x0100)
                >> 8;
WriteUSART( (char) outval );
```

http://www.microchip.com/

//**********************************

baudUSART

baud1USART

baud2USART

Функция: Установка биты конфигурации скорость передачи данных для повышения работы USART.

Включает в себя: usart.h

Прототип:

void baudUSART(unsigned char baudconfig);

void baud1USART(unsigned char baudconfig);

void baud2USART(unsigned char baudconfig);

Аргументы:

baudconfig

Битовая маска, которая создается путем выполнения побитовой операции ('&') со значением от каждой из категорий, перечисленных ниже. Thses значения определены в файле usart.h:

Часы простоя государство:

Состояние покоя BAUD_IDLE_CLK_HIGH Часы является высокий уровень

Состояние покоя BAUD_IDLE_CLK_LOW Часы является низкий уровень

Скорость передачи поколение:

BAUD_16_BIT_RATE 16-бит Скорость передачи поколение

BAUD_8_BIT_RATE 8 бит Скорость передачи поколение

RX Pin Мониторинг:

BAUD_WAKEUP_ON RXpin Мониторинг

BAUD_WAKEUP_OFF RX контактный не контролируется

Измерение скорости передачи данных:

BAUD_AUTO_ON Авто измерения скорости передачи данных включена

BAUD_AUTO_OFF Авто измерения скорости передачи данных отключены

http://www.microchip.com/

Примечания: Эти функции доступны только для процессоров с повышенной Возможность USART.

Имя файла:

ubaud.c

u1baud.c

u2baud.c

Пример кода:

```
baudUSART (BAUD_IDLE_CLK_HIGH &
        BAUD_16_BIT_RATE &
        BAUD_WAKEUP_ON &
        BAUD_AUTO_ON);



//************************************
```

2.10.2 Пример пользования

```
#include <p18C452.h>
#include <usart.h>
void main(void)
{
  // configure USART
  OpenUSART( USART_TX_INT_OFF  &
          USART_RX_INT_OFF  &
          USART_ASYNCH_MODE &
          USART_EIGHT_BIT   &
          USART_CONT_RX     &
          USART_BRGH_HIGH,
          25 );
```

```
http://www.microchip.com/
 while(1)
 {
  while( ! PORTAbits.RA0 );  // ждать RA0 в
  WriteUSART( PORTD );      // запись значения из PORTD
  if(PORTD == 0x80)         // проверить прекращения
    break;                  //  значение
 }
 CloseUSART();
}
```

```
//************************************
```

Глава 3 Программное обеспечение Периферийные Библиотека

3.1 ВВЕДЕНИЕ

В этой главе документы программное обеспечение периферийные функции библиотеки. Исходный код для всех из этих функций входит MPLAB C18 в SRC \ традиционный \ PMC и SRC \ продлен \ PMC подкаталоги установки компилятора.

Смотрите в Руководстве Пользователя MPASM ™ с MPLINK ™ и техподдержка на русском языке ™ (DS33014) для более Информация о строительстве библиотеки.

Следующие периферийные устройства поддерживаются MPLAB C18 библиотечных подпрограмм

• Внешний ЖК-функций (Раздел 3.2 "Внешние ЖК Функции»)

• Внешний CAN2510 функций (Раздел 3.3 "Внешний CAN2510 Функции»)

• Функции программного обеспечения I2C ™ (Раздел 3.4 "Программное обеспечение IIC Функции»)

• Программное обеспечение SPI функций (Раздел 3.5 "Программное обеспечение SPI® Функции»)

• Программное обеспечение UART функций (Раздел 3.6 "Программное обеспечение UART Функции»)

```
//************************************
```

http://www.microchip.com/

3.2 Внешний ЖК ФУНКЦИИ

Эти функции разработаны, чтобы позволить контроль ЖК контроллера Hitachi HD44780 используя ввода / вывода из PIC18 микроконтроллера. Следующие функции:

ТАБЛИЦА 3-1: Внешний ЖК ФУНКЦИИ

Функция Описание

BusyXLCD ли ЖК контроллер занят?

OpenXLCD Настройка линии ввода / вывода, используемые для управления ЖК и инициализировать LCD.

putcXLCD Записывает байт в ЖК контроллера.

putsXLCD Написать строки из памяти данных в ЖК.

putrsXLCD Написать строки из памяти программ в ЖК.

ReadAddrXLCD Читайте байт адреса от ЖК контроллера.

ReadDataXLCD Считывает байт из ЖК контроллера.

SetCGRamAddr указан адрес генератора характер.

SetDDRamAddr указан адрес отображения данных.

WriteCmdXLCD Написать команду ЖК контроллера.

WriteDataXLCD Записывает байт в ЖК контроллера.

Прекомпилированные версии этих функций использовать по умолчанию назначением контактов, которые могут быть изменен переосмысление следующие макросы в в файле xlcd.h, найти в ч подкаталог установки компилятора:

//**********

Таблица 3-2: MACROS выбора ЖК Назначение контактов

ЖК

контроллер

линия

Макросы Значение по умолчанию Использование

E Pin E_PIN

http://www.microchip.com/

TRIS_E

PORTBbits.RB4

DDRBbits.RB4

Штифт для E-линии.

Бит, который контролирует направление контактный связано с E-линии.

RS Pin RS_PIN

TRIS_RS

PORTBbits.RB5

DDRBbits.RB5

Штифт используется для линии RS.

Бит, который контролирует направление контактный связана с линией RS.

RW Pin RW_PIN

TRIS_RW

PORTBbits.RB6

DDRBbits.RB6

Штифт используется для RW линии.

Бит, который контролирует направление контактный связан с RW линии.

Данные линии DATA_PORT

TRIS_DATA_PORT

PORTB

DDRB

Пальцы, используемые для линий данных. Эти процедуры предполагают, что все контакты находятся на один порт.

Направление данных Зарегистрироваться связано с линиями DATA.

http://www.microchip.com/

Макро Значение по умолчанию Использование Бит8 не определен Если это значение определяется при библиотечные функции являются построен, они будут работать в 8-битном режиме передачи.

В противном случае, они будут работать в 4-битном режиме передачи.

ВЕРХНЯЯ не определен Когда бит8 не определен, это значение определяет, какие клев из DATA_PORT используется для передачи данных.

Если ВЕРХНЯЯ определена, верхние 4 бита (4: 7) DATA_PORT используются.

Если ВЕРХНЯЯ не определен, нижние 4 бита (0: 3) из DATA_PORT используются.

////////////////////////////////

Библиотеки, которые предоставляются может работать как в режиме 4-битном режиме или 8-битном режиме. Когда работает в 8-битном режиме, все линии одного порта используются. При работе в 4-битный Режим, либо верхние 4 бита или младшие 4 бита одного порта используются. В таблице ниже перечислены макросы, используемые для выбора между 4 или 8-битном режиме и для выбора, какие биты из порта используются при работе в 4-битном режиме.

//**********

Таблица 3-3: MACROS ВЫБОРА 4 или 8-битном режиме

После эти определения были сделаны, пользователь должен перекомпилировать подпрограммы XLCD а затем включить обновленные файлы в проекте. Это может быть достигнуто путем добавления исходные XLCD файлы в проект или по перекомпиляции библиотечные файлы с помощью предоставляемые пакетные файлы.

В XLCD библиотеки также требуют, чтобы следующие функции быть определены пользователем для предоставить соответствующие задержки:

Таблица 3-4: XLCD Функция задержки

Функция Поведение

DelayFor18TCY задержки в течение 18 циклов.

DelayPORXLCD задержки в течение 15 мс.

DelayXLCD задержки в течение 5 мс.

//**********

http://www.microchip.com/

3.2.1 Описания функций

//**********

BusyXLCD

Функция: Каково ЖК контроллер занят?

Включает в себя: xlcd.h

Прототип: unsigned char **BusyXLCD**(void);

Примечания: Эта функция возвращает статус оживленной флагом Hitachi HD44780 ЖК контроллер.

Возвращаемые значения:

1, если контроллер занят

0 в противном случае.

Имя файла: busyxlcd.c

Пример кода: while(**BusyXLCD**());

//**********

OpenXLCD

Функция: Настройка флажки PIC® ввода / вывода и инициализировать ЖК контроллер.

Включает в себя: xlcd.h

Прототип: void **OpenXLCD**(unsigned char lcdtype);

Аргументы: lcdtype

Битовая маска, которая создается путем выполнения побитовой операции ('&') со значением от каждой из категорий, перечисленных ниже. Эти значения определены в файле xlcd.h.

Интерфейс передачи данных:

FOUR_BIT 4-битный режим Интерфейс передачи данных

EIGHT_BIT 8-битный режим Интерфейс передачи данных

ЖК Конфигурация:

LINE_5X7 символов 5x7, дисплей одна строка

LINE_5X10 5x10 символов дисплей

http://www.microchip.com/

LINES_5X7 символов 5x7, дисплей многократной линия

Примечания: Эта функция настраивает PIC18 I / вывода общего используется для управления Hitachi HD44780 LCD контроллер. Он также инициализирует этот контроллер.

Имя файла: openxlcd.c

Пример кода:

OpenXLCD (EIGHT_BIT & LINES_5X7);

//**********

putcXLCD

Смотреть WriteDataXLCD.

putsXLCD

putrsXLCD

Функция: Записать строку в ЖК контроллера Hitachi HD44780.

Включает в себя: xlcd.h

Прототип:

void **putsXLCD**(char *buffer);

void **putrsXLCD**(const rom char *buffer);

Аргументы:

buffer Указатель на персонажей, которые будут записаны в контроллер ЖК-монитора.

Примечания: Данная функция позволяет записывать строку символов, расположенных в буфере для Hitachi HD44780 LCD контроллер. Это останавливает передачу когда нуль характер встречается. Нулевой символ не передается. Строки, расположенные в памяти данных следует использовать с "пут" версии из этих функций. Строки, расположенные в памяти программ, в том числе строковые литералы, должны быть используется с "putrs" версии этих функций.

Имя файла:

putsxlcd.c

putrxlcd.c

Пример кода:

char mybuff [20];

putrsXLCD(“Hello World”);

http://www.microchip.com/

putsXLCD(mybuff);

//**********************************

ReadAddrXLCD

Функция: Считывание байта адреса от ЖК контроллера Hitachi HD44780.

Включает в себя: xlcd.h

Прототип: unsigned char **ReadAddrXLCD**(void);

Примечания: Эта функция читает байт адреса от Hitachi HD44780 LCD Контроллер. Контроллер ЖК не должны быть заняты, когда эта операция выполняется - это можно проверить с помощью функции **BusyXLCD**. Адрес считываются из контроллера для генератора символов RAM или ОЗУ данных дисплея в зависимости от предыдущего Установите функцию ?? RAMADDR, которая называлась.

Вернуться Значение: Эта функция возвращает 8-битную количество. Адрес содержится в более низкого порядка 7 бит и флаг BUSY статус в самый старший бит.

Имя файла: readaddr.c

Пример кода:

char addr;

while (BusyXLCD());

addr = **ReadAddrXLCD**();

//**********************************

ReadDataXLCD

Функция: Прочитайте байт данных из ЖК контроллера Hitachi HD44780.

Включает в себя: xlcd.h

Прототип:

char **ReadDataXLCD**(void);

Примечания: Эта функция считывает байт данных от LCD Hitachi HD44780 Пульт управления Лер. Контроллер ЖК не должны быть заняты, когда эта операция выполняется - это можно проверить с

http://www.microchip.com/

помощью функции BusyXLCD. Данные считываются из контроллера для оперативной памяти генератора характер или ОЗУ данных дисплея в зависимости от предыдущего набора ?? RAMADDR функция, которая называлась.

Вернуться Значение: Эта функция возвращает значение данных 8-битный.

Имя файла: readdata.c

Пример кода:

```
char data;

while ( BusyXLCD() );

data = ReadAddrXLCD();

//************************************
```

SetCGRamAddr

Функция: Установка адрес генератора характер.

Включает в себя: xlcd.h

Прототип: void **SetCGRamAddr**(unsigned char addr);

Аргументы: АДРЕС Характер генератор адресов.

Примечания: Эта функция устанавливает адрес генератора характер Hitachi HD44780 LCD контроллер. Контроллер ЖК не должны быть заняты, когда эта операция выполняется - это можно проверить с помощью BusyXLCD Функция.

Имя файла: setcgram.c

Пример кода:

```
char cgaddr = 0x1F;

while( BusyXLCD() );

SetCGRamAddr( cgaddr );

//************************************
```

SetDDRamAddr

Функция: Установка адрес отображения данных.

Включает в себя: xlcd.h

Прототип: void **SetDDRamAddr**(unsigned char addr);

http://www.microchip.com/

Аргументы:

АДРЕС

Показать адрес данных.

Примечания: Эта функция устанавливает адрес отображения данных о Hitachi HD44780 ЖК контроллер. Контроллер ЖК не должны быть заняты, когда это Операция выполняется - это можно проверить с помощью **BusyXLCD** Функция.

Имя файла: setddram.c

Пример кода:

```
char ddaddr = 0x10;

while( BusyXLCD() );

SetDDRamAddr( ddaddr );

//************************************
```

WriteCmdXLCD

Функция: Написать команду ЖК контроллера Hitachi HD44780.

Включает в себя: xlcd.h

Прототип: void **WriteCmdXLCD**(unsigned char cmd);

Аргументы:

CMD

Задает команду, которая будет выполнена. Команда может быть одним из следующие значения, определенные в xlcd.h:

DOFF Экран будет выключен

CURSOR_OFF Включить отображение, без курсора

BLINK_ON Включить дисплей с мигающим курсором

BLINK_OFF Включить дисплей с немигающим курсором

SHIFT_CUR_LEFT Курсор смещается влево

SHIFT_CUR_RIGHT Курсор сдвигается вправо

SHIFT_DISP_LEFT Показать смещается влево

SHIFT_DISP_RIGHT Показать сдвигается вправо

http://www.microchip.com/

Кроме того, команда может быть битовую маску, которая создается выполнение побитовой операции ('и') со значением от каждого из категории перечислены ниже. Эти значения пределены в файле xlcd.h.

Режим передачи данных:

4-битный режим Интерфейс передачи данных FOUR_BIT

EIGHT_BIT 8-битный режим Интерфейс передачи данных

Тип дисплея:

LINE_5X7 символов 5x7, одна строка

LINE_5X10 5x10 символов дисплей

LINES_5X7 символов 5x7, несколько строк

Примечания: Данная функция позволяет записывать командный байт в Hitachi HD44780 LCD

Контроллер. Контроллер ЖК не должны быть заняты, когда эта операция

выполняется - это можно проверить с помощью функции BusyXLCD.

Имя файла: wcmdxlcd.c

Пример кода:

```
while( BusyXLCD() );

WriteCmdXLCD( EIGHT_BIT & LINES_5X7 );

WriteCmdXLCD( BLINK_ON );

WriteCmdXLCD( SHIFT_DISP_LEFT );


//************************************
```

putcXLCD

WriteDataXLCD

Функция: Записывает байт в ЖК контроллера Hitachi HD44780.

Включает в себя: xlcd.h

Прототип: void **WriteDataXLCD**(char data);

Аргументы:

данные

http://www.microchip.com/

Ценность данных, может быть любой 8-битное значение, но должно соответствоватьсимвольной пам ти таблица HD44780 LCD контроллера.

Примечания: Эта функция запись байта данных на ЖК-контроллером Hitachi HD44780. Контроллер ЖК не должны быть заняты, когда эта операция выполняется - это можно проверить с помощью функции **BusyXLCD**. Данные считываются из контроллера для оперативной памяти генератора характер или ОЗУ данных дисплея в зависимости от предыдущего набора ?? RAMADDR функция, которая называлась.

Имя файла: writdata.c

//**********************************

3.2.2 Пример применения

```
#include <p18C452.h>

#include <xlcd.h>

#include <delays.h>

#include <usart.h>

//-----------------------------------------------------


void DelayFor18TCY( void )

{

 Nop();

 Nop();

 Nop();

  Nop ();

  Nop ();

  Nop ();

  Nop ();

  Nop ();

  Nop ();

  Nop ();
```

```
http://www.microchip.com/
  Nop ();
  Nop ();
}
//------------------------------------------------
void DelayPORXLCD (void)
{
  Delay1KTCYx (60); // Задержка 15 мс
                // Циклы = (TimeDelay * FOSC) / 4
                // Циклы = (15 мс * 16 МГц) / 4
                // Циклы = 60000
 return;
}
//------------------------------------------------

void DelayXLCD (void)
{
  Delay1KTCYx (20); // Задержка 5 мс
                // Циклы = (TimeDelay * FOSC) / 4
                // Циклы = (5 мс * 16MHz) / 4
                // Циклы = 20000
 return;
}
//------------------------------------------------

void main( void )
{
 char data;
  // Настроить внешний ЖК-дисплей
  OpenXLCD (EIGHT_BIT & LINES_5X7);
```

```
http://www.microchip.com/
  // Настроить USART
  OpenUSART (USART_TX_INT_OFF & USART_RX_INT_OFF &
          USART_ASYNCH_MODE & USART_EIGHT_BIT &
          USART_CONT_RX,
          25);
 while(1)
  {
   while(!DataRdyUSART()); // ждать данных
   data = ReadUSART();      // чтения данных
   WriteDataXLCD(data);     // запись в ЖК
   if(data=='Q')
     break;
  }
 CloseUSART();
}

//************************************
```

3.3 Внешние функции CAN2510

Этот раздел описывает внешние функции периферийных библиотек MCP2510.

предусмотрены следующие функции:

Таблица 3-5: внешние функции CAN2510

Функция Описание

CAN2510BitModify Изменяет указанные биты в регистре к новым ценностям.

CAN2510ByteRead Читает регистр MCP2510 указанный адрес.

CAN2510ByteWrite Записывает значение в реестре MCP2510 указанный адрес.

CAN2510DataRead Читает сообщение из указанного буфер приема.

CAN2510DataReady Определяет, данные ожидают в указанный получать Буфер.

http://www.microchip.com/

CAN2510Disable диски выбранный PIC18CXXX I штифт / вывода высокой отключить Выбор микросхемы из MCP2510. (1)

CAN2510Enable диски выбранный PIC18CXXX I контактный / вывода низкого до выбора микросхемы MCP2510. (1)

CAN2510ErrorState Читает текущее состояние погрешность CAN шине.

CAN2510Init инициализации порта PIC18CXXX SPI для связи в MCP2510 а затем настраивает регистры MCP2510 для взаимодействия с CAN шиной.

CAN2510InterruptEnable Изменяет CAN2510 разрешения прерывания биты (CANINTE зарегистрируйтесь) на новые значения.

CAN2510InterruptStatus Указывает источник CAN2510 прерывания.

CAN2510LoadBufferStd Загружает Стандартная кадр данных в указанном передачи Буфер.

CAN2510LoadBufferXtd Загружает Расширенная кадр данных в указанном передачи Буфер.

CAN2510LoadRTRStd Загружает стандартный удаленный кадр в указанном передачи Буфер.

CAN2510LoadRTRXtd Загружает Extended Remote кадр в указанное Передача буфера.

CAN2510ReadMode Читает текущий режим MCP2510 работы.

CAN2510ReadStatus Читает статус MCP2510 приема и передачи Буферов.

CAN2510Reset Сброс MCP2510.

CAN2510SendBuffer просит передачу сообщений за указанный передачи буфер (ы).

CAN2510SequentialRead Читает указанное количество байтов в MCP2510, начиная с указанного адреса. Эти значения будут хранится в DataArray.

CAN2510SequentialWrite Записывает указанное количество байтов в MCP2510, начиная с указанного адреса. Эти значения будут написана с DataArray.

CAN2510SetBufferPriority загружает определенный приоритет для указанного передачи Буфер.

CAN2510SetMode Настраивает режим MCP2510 работы.

CAN2510SetMsgFilterStd настраивает все фильтра и маски значений конкретных буфер приема для стандартного сообщения.

CAN2510SetMsgFilterXtd настраивает все фильтра и маски значений конкретных буфер приема для длительного сообщении.

CAN2510SetSingleFilterStd Настраивает указано фильтр со значением фильтра для (Std) сообщение Standard.

CAN2510SetSingleFilterXtd Настраивает указано фильтр со значением фильтра для Расширенный (XTD) сообщение.

http://www.microchip.com/

CAN2510SetSingleMaskStd Настраивает указано буфера маску с маской Значение для сообщения формата: Standard (Std).

CAN2510SetSingleMaskXtd Настраивает указано буфера маску с маской Значение в течение длительного (XTD) сообщения.

CAN2510WriteStd Пишет сообщение стандартного формата из к CAN шине с первого доступного буфера передачи.

CAN2510WriteXtd Пишет сообщение расширенном составе из к CAN шине с первого доступного буфера передачи.

Примечание 1: Функции CAN2510Enable и CAN2510Disable будет нужно перекомпилировать, если:

- Назначение PICmicro MCU штифта CS изменяется от RC2

- Файл заголовка устройство должно быть изменено

//**********

3.3.1 Описания функций

CAN2510BitModify

Функция: Изменяет заданные биты в регистре к новым ценностям.

Требуется CAN

Режим (ы): Все

Включает в себя: can2510.h

Прототип:

void **CAN2510BitModify**(

unsigned char addr

unsigned char mask

unsigned char data);

Аргументы:

АДРЕС

Значение указанного адреса определяет адрес регистра MCP2510 к изменить.

маска

Значение маски определяет биты, которые будут модифицированы.

http://www.microchip.com/

данных

Ценность данных, определяет новое состояние битов.

Примечания: Эта функция изменяет содержимое регистра, указанного адреса, маска определяет, какие биты быть изменены и задает данные новое значение для загрузки в этих битов. Только конкретные регистры могут быть изменение с Изменить команды Bit.

Имя файла: canbmod.c

//**********

CAN2510ByteRead

Функция: Читает регистр MCP2510 указанный адрес.

Требуется CAN

Режим (ы): Все

Включает в себя: can2510.h

Прототип: unsigned char **CAN2510ByteRead**(unsigned char address);

Аргументы: адрес

Адрес MCP2510, что следует рассматривать.

Примечания: Эта функция читает один байт из MCP2510 в указанное адрес.

Вернуться Значение: Содержание указанному адресу.

Имя файла: readbyte.c

//**********

CAN2510ByteWrite

Функция: Записывает значения в регистр MCP2510 указанному адресу.

Требуется CAN

Режим (ы): Все

Включает в себя: can2510.h

Прототип: void **CAN2510ByteWrite**(

unsigned char address,

http://www.microchip.com/

unsigned char value);

Аргументы:

адрес

Адрес MCP2510, что должно быть записано.

значение

Значение, которое должно быть записано.

Примечания: Эта функция записывает один байт из MCP2510 в указанное адрес.

Имя файла: wrtbyte.c

//**********

CAN2510DataRead

Функция: Читает сообщение из указанного буфер приема.

Требуется CAN

Режим (ы): Все (кроме режима конфигурации)

Включает в себя: can2510.h

Прототип:

unsigned char **CAN2510DataRead**(

unsigned char bufferNum,

unsigned long *msgId,

unsigned char *numBytes,

unsigned char *data);

Аргументы:

bufferNum

Получите буфер из которого следует читать сообщения. Один из следующих значения:

CAN2510_RXB0 Читать приемного буфера 0

CAN2510_RXB1 Читать приемного буфера 1

MsgID

Указывает на месте, что будет модифицированные функции содержать CAN стандарт идентификатор сообщения.

http://www.microchip.com/

NumBytes

Указывает на месте, что будет модифицированные функции содержать количество байтов в этом сообщении.

данных

Указывает на массив, который будет изменяться функцией содержать Данные сообщения. Этот массив должен быть не менее 8 байт, так как это максимальная длина данных сообщения.

Примечания: Эта функция определяет, является ли сообщение стандартный или расширенный сообщение, декодирует ID и сообщение длину, и заполняет приобретенные пользователем в местах с соответствующей информацией. Функция CAN2510DataReady должны использоваться для определения того, указано буфер имеет данные для чтения.

Возвращаемые значения: Функция возвращает одно из следующих значений:

CAN2510_XTDMSG Расширенная сообщение формате

CAN2510_STDMSG Стандартное сообщение формате

CAN2510_XTDRTR запрос удаленного передачи (XTD сообщение)

CAN2510_STDRTR запрос удаленного передачи (STD сообщение)

Имя файла: canread.c

//**********

CAN2510DataReady

Функция: Определяет, данные ожидают в указанном буфер приема.

Требуется CAN

Режим (ы): Все (кроме режима конфигурации)

Включает в себя: can2510.h

Прототип:

unsigned char **CAN2510DataReady**(

unsigned char bufferNum);

Аргументы:

bufferNum

Получите буфер для проверки сообщение ждет. Один из следующих значения:

CAN2510_RXB0 Проверьте Приемный буфер 0

CAN2510_RXB1 Проверьте Приемный буфер 1

http://www.microchip.com/

CAN2510_RXBX Проверьте Приемный буфер 0 и Приемный буфер 1

Примечания: Эта функция проверяет соответствующий RXnIF бит в регистре CANINTF.

Возвращаемые значения:

Возвращает ноль, если не обнаружено ни одно сообщение или ненулевое значение, если сообщение был обнаружен.

1 = buffer0

2 = buffer1

3 = оба

Имя файла: canready.c

//**********

CAN2510Disable

Функция: Диски выбранный PIC18CXXX I штифт / вывода высокой отключить выбора микросхемы из MCP2510.

Требуется CAN

Режим (ы): Все

Включает в себя: canenabl.h

Примечание: Эта включают файл нужно будет изменить, если сигнал выбора чипа не связано с RC2 штифт PICmicro MCU.

Прототип: void **CAN2510Disable**(void);

Аргументы: Нет

Примечания: Эта функция требует, чтобы пользователь изменяет файл указать PIC18CXXX ввода / вывода штифт (и порт), который будет использоваться для подключения к MCP2510 CS контактный. Контактный умолчанию RC2.

Примечание: исходный файл, содержащий эту функцию (и CAN2510Enable функция) должен иметь определения изменен, чтобы правильно указать порт (A, B, C, ...) и номер контакта (1, 2, 3, ...) , который используется, чтобы управлять штифт MCP2510 CS. После модификации ние, библиотека процессор конкретных должны быть перестроены. Смотрите раздел 1.5.3 "Восстановление" для получения информации о восстановлении.

Имя файла: canenabl.c

//**********

http://www.microchip.com/

CAN2510Enable

Функция: накопители выбранный PIC18CXXX Я контактный / вывода низкого до выбора микросхемы MCP2510.

Требуется CAN

Режим (ы): Все

Включает в себя: canenabl.h

Примечание: Эта включают файл нужно будет изменить, если сигнал выбора чипа не связано с RC2 штифт PICmicro MCU.

Прототип: void **CAN2510Enable**(void);

Примечания: Эта функция требует, чтобы пользователь изменяет файл указать PIC18CXXX ввода / вывода штифт (и порт), который будет использоваться для подключения к MCP2510 CS контактный. Контактный умолчанию RC2.

Примечание: исходный файл, содержащий эту функцию (и CAN2510Disable функция) должен иметь определения изменение правильно указать порт (A, B, C, ...) и ПИН-код (1, 2, 3, ...), который используется, чтобы управлять штифт MCP2510 CS. После модификация, библиотека процессор конкретных должны быть перестроены. Смотреть Раздел 1.5.3 "Восстановление" для получения информации о восстановлении.

Имя файла: canenabl.c

//**********

CAN2510ErrorState

Функция: Читает текущее состояние погрешность CAN шине.

Требуется CAN

Режим (ы):

Нормальный режим, режим Loopback, слушают только режим (Счетчиков ошибок сбрасываются в режиме настройки)

Включает в себя: can2510.h

Прототип: unsigned char **CAN2510ErrorState**(void);

Примечания: Эта функция возвращает состояние ошибки в CAN шину. Государство Ошибка зависит от значений в ТИК и ЗАП регистров.

Возвращаемые значения: Функция возвращает одно из следующих значений:

CAN2510_BUS_OFF TEC> 255

CAN2510_ERROR_PASSIVE_TX TEC> 127

http://www.microchip.com/

CAN2510_ERROR_PASSIVE_RX REC> 127

CAN2510_ERROR_ACTIVE_WITH_TXWARN TEC> 95

CAN2510_ERROR_ACTIVE_WITH_RXWARN REC> 95

CAN2510_ERROR_ACTIVE TEC ≤ 95 и REC ≤ 95

Имя файла: canerrst.c

//**********

CAN2510Init

Функция: инициализацию порта PIC18CXXX SPI для связи в MCP2510 а затем настраивает MCP2510 регистрирует для взаимодействия с CAN автобус.

Требуется CAN

Режим (ы): Режим конфигурации

Включает в себя: can2510.h

Прототип:

unsigned char CAN2510Init(

unsigned short long BufferConfig,

unsigned short long BitTimeConfig,

unsigned char interruptEnables,

unsigned char SPI_syncMode,

unsigned char SPI_busMode,

unsigned char SPI_smpPhase);

Аргументы: Значения следующих параметров определены в инклюднике can2510.h.

BufferConfig

Значение BufferConfig строится через побитового И (&) Операция из следующих вариантов. Только один вариант за функции группового может быть выбран. Опция в жирным шрифтом является значением по умолчанию.

Сброс MCP2510 устройства

Определяет, если команда MCP2510 Сброс должен быть отправлен. Это не соответствуют бита в регистрах MCP2510.

CAN2510_NORESET Не сбрасывайте MCP2510

http://www.microchip.com/

CAN2510_RESET Сброс MCP2510

Буфер 0 Фильтрация

Управляется RXB0M1: RXB0M0 бит (RXB0CTRL регистр)

CAN2510_RXB0_USEFILT Получайте сообщения, использовать фильтры

CAN2510_RXB0_STDMSG Получать только стандартные сообщения

CAN2510_RXB0_XTDMSG Получать только Расширенные сообщения

CAN2510_RXB0_NOFILT Получайте сообщения, никаких фильтров

Буфер 1 Фильтрация

Управляется RXB1M1: RXB1M0 бит (RXB1CTRL регистр)

CAN2510_RXB1_USEFILT Получайте сообщения, использовать фильтры

CAN2510_RXB1_STDMSG Получать только стандартные сообщения

CAN2510_RXB1_XTDMSG Получать только Расширенные сообщения

CAN2510_RXB1_NOFILT Получайте сообщения, никаких фильтров

Приемный буфер 0 до Receive Buffer 1 Rollover

Управляется BUKT бит (RXB0CTRL регистр)

CAN2510_RXB0_ROLL Если приемный буфер 0 полна, сообщение идет в буфер приема 1

CAN2510_RXB0_NOROLL Перевернулся инвалидов

RX1BF Pin Установка

Контролируется B1BFS: B1BFE: B1BFM бит (BFPCTRL регистр)

CAN2510_RX1BF_OFF RX1BF штифт с высоким сопротивлением

CAN2510_RX1BF_INT RX1BF штифт выход, который указывает приемного буфера 1 был загружен. Может быть использован в качестве сигнала прерывания сигналов.

CAN2510_RX1BF_GPOUTH RX1BF контактный является цифровым общего назначения выход, выход высокой

CAN2510_RX1BF_GPOUTL RX1BF контактный является цифровым общего назначения выход, выход Низкие

RX0BF Pin Установка

Контролируется B0BFS: B0BFE: B0BFM бит (BFPCTRL регистр)

CAN2510_RX0BF_OFF RX0BF штифт с высоким сопротивлением

http://www.microchip.com/

CAN2510_RX0BF_INT RX0BF штифт выход, который указывает Приемного буфера 0 был загружен. Может быть используется в качестве сигнала прерывания.

CAN2510_RX0BF_GPOUTH RX0BF контактный является цифровым общего назначения выход, выход высокой

CAN2510_RX0BF_GPOUTL RX0BF контактный является цифровым общего назначения выход, выход Низкие

TX2 Pin Установка

Управляется B2RTSM бит (TXRTSCTRL регистр)

CAN2510_TX2_GPIN TX2RTS контактный цифровой вход

CAN2510_TX2_RTS TX2RTS контактный является входом используется для инициирования Запрос на передачу кадра от TXBUF2

TX1 Pin Установка

Управляется B1RTSM бит (TXRTSCTRL регистр)

CAN2510_TX1_GPIN TX1RTS контактный цифровой вход

CAN2510_TX1_RTS TX1RTS контактный является входом используется для инициирования

Запрос на передачу кадра от TXBUF1

TX0 Pin Установка

Управляется B0RTSM бит (TXRTSCTRL регистр)

CAN2510_TX0_GPIN TX0RTS контактный цифровой вход

CAN2510_TX0_RTS TX0RTS контактный является входом используется для инициирования Запрос на передачу кадра от TXBUF0

Запрос Режим работы

Управляется REQOP2: REQOP0 бит (CANCTRL регистр)

Режим конфигурации CAN2510_REQ_CONFIG

Режим CAN2510_REQ_NORMAL Нормальная работа

Режим CAN2510_REQ_SLEEP сна

Режим CAN2510_REQ_LOOPBACK Loop Back

CAN2510_REQ_LISTEN Слушайте всего режим

CLKOUT Pin Установка

Управляется CLKEN: CLKPRE1: CLKPRE0 бит (CANCTRL регистр)

CAN2510_CLKOUT_8 CLKOUT = FOSC / 8

http://www.microchip.com/

CAN2510_CLKOUT_4 CLKOUT = FOSC / 4

CAN2510_CLKOUT_2 CLKOUT = FOSC / 2

CAN2510_CLKOUT_1 CLKOUT = FOSC

CAN2510_CLKOUT_OFF CLKOUT отключен

BitTimeConfig

Значение BitTimeConfig строится через побитового И (&) Операция из следующих вариантов. Только один вариант за функции группового может быть выбран. Опция в жирным шрифтом является значением по умолчанию.

Скорость передачи Prescaler (BRP)

Управляется BRP5: BRP0 бит (CNF1 регистр)

CAN2510_BRG_1X TQ = 1 x (2TOSC)

::

CAN2510_BRG_64X TQ = 64 x (2TOSC)

Синхронизация Перейти Ширина

Управляется SJW1: SJW0 бит (CNF1 регистр)

Длина CAN2510_SJW_1TQ SJW = 1 TQ

Длина CAN2510_SJW_2TQ SJW = 2 TQ

Длина CAN2510_SJW_3TQ SJW = 3 TQ

Длина CAN2510_SJW_4TQ SJW = 4 TQ

Фаза 2 Ширина сегментов

Управляется PH2SEG2: PH2SEG0 бит (CNF3 регистр)

CAN2510_PH2SEG_2TQ Длина = 2 TQ

CAN2510_PH2SEG_3TQ Длина = 3 TQ

CAN2510_PH2SEG_4TQ Длина = 4 TQ

CAN2510_PH2SEG_5TQ Длина = 5 TQ

CAN2510_PH2SEG_6TQ Длина = 6 TQ

CAN2510_PH2SEG_7TQ Длина = 7 TQ

CAN2510_PH2SEG_8TQ Длина = 8 TQ

Фаза 1 Ширина сегментов

http://www.microchip.com/

Управляется PH1SEG2: PH1SEG0 бит (CNF2 регистр)

CAN2510_PH1SEG_1TQ Длина = 1 TQ

CAN2510_PH1SEG_2TQ Длина = 2 TQ

CAN2510_PH1SEG_3TQ Длина = 3 TQ

CAN2510_PH1SEG_4TQ Длина = 4 TQ

CAN2510_PH1SEG_5TQ Длина = 5 TQ

CAN2510_PH1SEG_6TQ Длина = 6 TQ

CAN2510_PH1SEG_7TQ Длина = 7 TQ

CAN2510_PH1SEG_8TQ Длина = 8 TQ

Распространение Ширина сегментов

Управляется PRSEG2: PRSEG0 бит (CNF2 регистр)

CAN2510_PROPSEG_1TQ Длина = 1 TQ

CAN2510_PROPSEG_2TQ Длина = 2 TQ

CAN2510_PROPSEG_3TQ Длина = 3 TQ

CAN2510_PROPSEG_4TQ Длина = 4 TQ

CAN2510_PROPSEG_5TQ Длина = 5 TQ

CAN2510_PROPSEG_6TQ Длина = 6 TQ

CAN2510_PROPSEG_7TQ Длина = 7 TQ

CAN2510_PROPSEG_8TQ Длина = 8 TQ

Фаза 2 Источник

Управляется BTLMODE бит (CNF2 регистре). Это определяет, является ли Фаза 2 длина определяется PH2SEG2: PH2SEG0 битов или больше длина PH1SEG2: PH1SEG0 бит и (2TQ).

CAN2510_PH2SOURCE_PH2 Длина = PH2SEG2: PH2SEG0

CAN2510_PH2SOURCE_PH1 Длина = больше из

PH1SEG2: PH1SEG0 и 2TQ

Бит Образец Точка Частота

Управляется SAM бит (CNF2 регистре). Это определяет, является ли бит образцы 1 или 3 раза в точке выборки.

CAN2510_SAMPLE_1x Бит оцифровывается раз

http://www.microchip.com/

CAN2510_SAMPLE_3x Бит оцифровывается три раза

RX контактный Noise Filter в режиме сна

Управляется WAKFIL бит (CNF3 регистре). Это определяет, в RX контактный будет использовать фильтр, чтобы подавлять шум, когда устройство находится в спящем режиме.

CAN2510_RX_FILTER фильтры по RX штифта, когда в режиме сна режим

CAN2510_RX_NOFILTER Без фильтрации на RX штифта, когда в режиме сна режим

interruptEnables

Значение interruptEnables может быть комбинацией следующие значения, в сочетании с использованием побитовой операции (&). опция в жирным шрифтом является значением по умолчанию. Управляется все биты в

CANINTE регистр.

CAN2510_NONE_EN не включен Нет прерывания

CAN2510_MSGERR_EN прерывания при ошибке во время сообщения прием или передача

CAN2510_WAKEUP_EN прерывания на автобусной деятельности CAN

CAN2510_ERROR_EN прерывания на EFLG состоянии ошибки изменение

CAN2510_TXB2_EN прерывания на буфере передачи 2 становится пусто

CAN2510_TXB1_EN прерывания на буфере передачи 1 становится пусто

CAN2510_TXB0_EN прерывания на буфере передачи 0 становится пусто

CAN2510_RXB1_EN прерывания, когда сообщение было получено в приемного буфера 1

CAN2510_RXB0_EN прерывания, когда сообщение было получено в приемного буфера 0

SPI_syncMode

Определяет частота синхронизации PIC18CXXX SPI:

CAN2510_SPI_FOSC4 Общается в FOSC / 4

CAN2510_SPI_FOSC16 Общается в FOSC / 16

CAN2510_SPI_FOSC64 Общается в FOSC / 64

CAN2510_SPI_FOSCTMR2 Общается на TMR2 / 2

SPI_busMode

Определяет режим автобус PIC18CXXX SPI:

CAN2510_SPI_MODE00 Связь использовании режима SPI 00

CAN2510_SPI_MODE01 Связь использовании режима SPI 01

http://www.microchip.com/

SPI_smpPhase

Определяет точка выборки PIC18CXXX SPI:

CAN2510_SPI_SMPMID Образцы в середине SPI бит

CAN2510_SPI_SMPEND Образцы в конце SPI бит

Примечания: Эта функция инициализирует модуль PIC18CXXX SPI, сбрасывает

MCP2510 устройство (если требуется), а затем настраивает MCP2510 регистры.

Примечание: Когда эта функция будет завершена, MCP2510 остается в Режим конфигурации.

Вернуться Значение: Указывает на то, MCP2510 удалось инициализировать.

0, если завершена инициализация

-1, Если инициализация не завершил

Имя файла: caninit.c

//**********

CAN2510InterruptEnable

Функция: Изменяет CAN2510 разрешения прерывания бит (CANINTE регистре) на

Новые значения.

Требуется CAN

Режим (ы): Все

Включает в себя:

can2510.h,

spi_can.h

Прототип:

void **CAN2510InterruptEnable**(

unsigned char interruptEnables);

Аргументы: interruptEnables

Значение interruptEnables может быть комбинацией следующие значения, в сочетании с использованием побитовой операции (&). опция в жирным шрифтом является значением по умолчанию. Управляется все биты в

http://www.microchip.com/

CANINTE регистр.

CAN2510_NONE_EN Нет Прерывания (00000000)

CAN2510_MSGERR_EN прерывания при ошибке во время сообщения прием или передача (10000000)

CAN2510_WAKEUP_EN прерывания на автобусной деятельности CAN (01000000)

CAN2510_ERROR_EN прерывания на EFLG изменения состояния ошибке (00100000)

CAN2510_TXB2_EN прерывания на буфере передачи 2 становится пустым (00010000)

CAN2510_TXB1_EN прерывания на буфере передачи 1 становится пустым (00001000)

CAN2510_TXB0_EN прерывания на буфере передачи 0 становится пустым (00000100)

CAN2510_RXB1_EN прерывания, когда сообщение было получено в Буфер приема 1 (00000010)

CAN2510_RXB0_EN прерывания, когда сообщение было получено в Буфер приема 0 (00000001)

Примечания: Эта функция обновляет регистр CANINTE со значением, которое определяется Anding желаемых источников прерываний.

Имя файла: caninte.c

//**********

CAN2510InterruptStatus

Функция: Указывает источник CAN2510 прерывания.

Требуется CAN

Режим (ы): Все

Включает в себя:

can2510.h,

spi_can.h

Прототип: unsigned char **CAN2510InterruptStatus**(void);

Примечания: Эта функция читает регистр CANSTAT и задает код в зависимости от состояния ICODE2: бит ICODE0.

Возвращаемые значения: Функция возвращает одно из следующих значений:

CAN2510_NO_INTS не произошло прерывания

http://www.microchip.com/

CAN2510_WAKEUP_INT прерывания на автобусной деятельности CAN

CAN2510_ERROR_INT прерывания на EFLG изменения состояния ошибке

CAN2510_TXB2_INT прерывания на буфере передачи 2 становится пусто

CAN2510_TXB1_INT прерывания на буфере передачи 1 становится пусто

CAN2510_TXB0_INT прерывания на буфере передачи 0 становится пусто

CAN2510_RXB1_INT прерывания, когда сообщение было получено в приемного буфера 1

CAN2510_RXB0_INT прерывания, когда сообщение было получено в приемного буфера 0

Имя файла: canints.c

//**********

CAN2510LoadBufferStd

Функция: Загружает Стандартная кадр данных в заданный буфер передачи.

Требуется CAN

Режим (ы): Все

Включает в себя: can2510.h

Прототип:

void **CAN2510LoadBufferStd**(

unsigned char bufferNum,

unsigned int msgId,

unsigned char numBytes,

unsigned char *data);

Аргументы: bufferNum

Задает буфер для загрузки сообщение в. Один из следующих значения:

CAN2510_TXB0 передачи буфера 0

CAN2510_TXB1 передачи буфера 1

CAN2510_TXB2 передачи буфера 2

MsgID

CAN идентификатор сообщения, до 11 бит для стандартного сообщения.

NumBytes

http://www.microchip.com/

Количество байтов данных, подлежащих передаче, с 0 по 8 Если значение больше, чем 8, только первые 8 байт данных будет храниться.

данных

Массив значений данных, которые будут загружены. Массив должен быть по крайней мере такого размера, как Значение, указанное в NumBytes.

Примечания: Эта функция загружает информацию сообщения, но не передает сообщение. Используйте функцию CAN2510WriteBuffer (), чтобы написать сообщение на CAN шине. Эта функция не устанавливает приоритет буфера. Используйте Функция CAN2510SetBufferPriority (), чтобы установить буферную приоритет.

Имя файла: canloads.c

//**********

CAN2510LoadBufferXtd

Функция: Загружает Расширенная кадр данных в заданный буфер передачи.

Требуется CAN

Режим (ы): Все

Включает в себя: can2510.h

Прототип:

void **CAN2510LoadBufferXtd**(

unsigned char bufferNum,

unsigned int msgId,

unsigned char numBytes,

unsigned char *data);

Аргументы:

bufferNum

Задает буфер для загрузки сообщение в. Один из следующих значения:

CAN2510_TXB0 передачи буфера 0

CAN2510_TXB1 передачи буфера 1

CAN2510_TXB2 передачи буфера 2

MsgID

http://www.microchip.com/

CAN идентификатор сообщения, до 29 бит для длительного сообщении.

NumBytes

Количество байтов данных, подлежащих передаче, с 0 по 8 Если значение больше, чем 8, только первые 8 байт данных будет храниться.

данных

Массив значений данных, которые будут загружены. Массив должен быть по крайней мере такого размера, как Значение, указанное в NumBytes.

Примечания: Эта функция загружает информацию сообщения, но не передает сообщение. Используйте функцию CAN2510WriteBuffer (), чтобы написать сообщение на CAN шине. Эта функция не устанавливает приоритет буфера. Используйте Функция CAN2510SetBufferPriority (), чтобы установить буферную приоритет.

Имя файла: canloadx.c

//**********

CAN2510LoadRTRStd

Функция: Загружает стандартный удаленный кадр в указанный буфер передачи.

Требуется CAN

Режим (ы): Все

Включает в себя: can2510.h

Прототип:

void **CAN2510LoadBufferStd**(

unsigned char bufferNum,

unsigned int msgId,

unsigned char numBytes,

unsigned char *data);

Аргументы:

bufferNum

Задает буфер для загрузки сообщение в. Один из следующих значения:

CAN2510_TXB0 передачи буфера 0

CAN2510_TXB1 передачи буфера 1

CAN2510_TXB2 передачи буфера 2

http://www.microchip.com/

MsgID

CAN идентификатор сообщения, до 11 бит для стандартного сообщения.

NumBytes

Количество байтов данных, подлежащих передаче, с 0 по 8 Если значение больше, чем 8, только первые 8 байт данных будет храниться.

данных

Массив значений данных, которые будут загружены. Массив должен быть по крайней мере такого размера, как Значение, указанное в NumBytes.

Примечания: Эта функция загружает информацию сообщения, но не передает сообщение. Используйте функцию CAN2510WriteBuffer (), чтобы написать сообщение на CAN шине. Эта функция не устанавливает приоритет буфера. Используйте Функция CAN2510SetBufferPriority (), чтобы установить буферную приоритет.

Имя файла: canlrtrs.c

//**********

CAN2510LoadRTRXtd

Функция: Загружает Extended Remote кадр в указанный буфер передачи.

Требуется CAN

Режим (ы): Все

Включает в себя: can2510.h

Прототип:

void **CAN2510LoadBufferXtd**(

unsigned char bufferNum,

unsigned long msgId,

unsigned char numBytes,

unsigned char *data);

Аргументы:

bufferNum

Задает буфер для загрузки сообщение в. Один из следующих значения:

CAN2510_TXB0 передачи буфера 0

CAN2510_TXB1 передачи буфера 1

http://www.microchip.com/

CAN2510_TXB2 передачи буфера 2

MsgID

CAN идентификатор сообщения, до 29 бит для длительного сообщении.

NumBytes

Количество байтов данных, подлежащих передаче, с 0 по 8 Если значение больше, чем 8, только первые 8 байт данных будет храниться.

данных

Массив значений данных, которые будут загружены. Массив должен быть по крайней мере такого размера, как Значение, указанное в NumBytes.

Примечания: Эта функция загружает информацию сообщения, но не передает сообщение. Используйте функцию CAN2510WriteBuffer (), чтобы написать сообщение на CAN шине. Эта функция не устанавливает приоритет буфера. Используйте Функция CAN2510SetBufferPriority (), чтобы установить буферную приоритет.

Имя файла: canlrtrx.c

//**********

CAN2510ReadMode

Функция: Читает текущий режим MCP2510 работы.

Требуется CAN

Режим (ы): Все

Включает в себя: can2510.h

Прототип: unsigned char **CAN2510ReadMode**(void);

Примечания: Эта функция читает текущий режим работы. Режим может иметь в ожидании запроса на новый режим.

Вернуться Значение:

режим

Значение режима может быть одним из следующих значений (в определенный can2510.h). Указанные в OPMODE2: OPMODE0 бит (CANSTAT зарегистрируйтесь). Один из следующих значений:

Регистры конфигурации CAN2510_MODE_CONFIG может быть изменение

CAN2510_MODE_NORMAL Нормальный (отправлять и получать сообщения)

CAN2510_MODE_SLEEP Подождите прерывания

http://www.microchip.com/

CAN2510_MODE_LISTEN Слушайте только, не отправить

CAN2510_MODE_LOOPBACK использовано для тестирования, сообщения остаются внутренний

Имя файла: canmoder.c

//**********

CAN2510ReadStatus

Функция: Читает статус MCP2510 приема и передачи буферов.

Требуется CAN

Режим (ы): Все

Включает в себя: can2510.h

Прототип: unsigned char **CAN2510ReadStatus**(void);

Примечания: Эта функция читает текущее состояние приема и передачи буферов.

Вернуться Значение: статус Значение статуса (байт без знака) имеет следующий формат:

бит 7 TXB2IF

бит 6 TXB2REQ

бит 5 TXB1IF

бит 4 TXB1REQ

бит 3 TXB0IF

бит 2 TXB0REQ

бит 1 RXB1IF

бит 0 RXB0IF

Имя файла: canstats.c

//**********

CAN2510Reset

Функция: Сброс MCP2510.

Требуется CAN

Режим (ы): Все

Включает в себя:

http://www.microchip.com/

can2510.h

spi_can.h

spi.h

Прототип: void **CAN2510Reset**(void);

Примечания: Эта функция сбрасывает MCP2510.

Имя файла: canreset.c

//**********

CAN2510SendBuffer

Функция: просит передачу сообщений за указанный буфер (ы) передачи.

Требуется CAN

Режим (ы): Нормальный режим

Включает в себя: can2510.h

Прототип:

void **CAN2510WriteBuffer**

(unsigned char bufferNum);

Аргументы:

bufferNum

Задает буфер требовать передачу. Один из следующих значения:

CAN2510_TXB0 передачи буфера 0

CAN2510_TXB1 передачи буфера 1

CAN2510_TXB2 передачи буфера 2

CAN2510_TXB0_B1 буфер передачи 0 и буфера 1

CAN2510_TXB0_B2 буфер передачи 0 и буфера 2

CAN2510_TXB1_B2 передачи буфера 1 и буфер 2

CAN2510_TXB0_B1_B2 передачи буфера 0, буфер 1 и буфер 2

Примечания: Эта функция запрашивает передачу ранее загруженного сообщения хранятся в указанном буфере (ы). Чтобы загрузить сообщение, использовать CAN2510LoadBufferStd () или CAN2510LoadBufferXtd () подпрограммы.

http://www.microchip.com/

Имя файла: cansend.c

//**********

CAN2510SequentialRead

Функция: Читает указанное количество байтов в MCP2510, начиная с Указанный адрес. Эти значения будут сохранены в DataArray.

Требуется CAN

Режим (ы): Все

Включает в себя: can2510.h

Прототип:

void **CAN2510SequentialRead**(

unsigned char *DataArray

unsigned char CAN2510addr

unsigned char numbytes);

Аргументы:

DataArray

Начальный адрес массива данных, в которой хранятся данные последовательного чтения.

CAN2510addr

Адрес MCP2510 где последовательного чтения начать с.

NumBytes

Число байтов, последовательно читать.

Примечания: Эта функция читает последовательные байт от MCP2510 начиная с Указанный адрес. Эти значения загружаются начиная с первого адреса из массива, заданного.

Имя файла: readseq.c

//**********

CAN2510SequentialWrite

Функция: Записывает указанное количество байтов в MCP2510, начиная с Указанный адрес. Эти значения будут записаны от DataArray.

Требуется CAN

http://www.microchip.com/

Режим (ы): Все

Включает в себя: can2510.h

Прототип:

void **CAN2510SequentialWrite**(

unsigned char *DataArray

unsigned char CAN2510addr

unsigned char numbytes);

Аргументы:

DataArray

Начальный адрес массива данных, который содержит последовательной записи

Данные.

CAN2510addr

Адрес MCP2510 где последовательная запись начать.

NumBytes

Число байтов для последовательного запись.

Примечания: Данная функция позволяет записывать последовательные байты в MCP2510 начиная с Указанный адрес. Эти значения содержатся начиная с первого адрес массива, заданного.

Имя файла: wrtseq.c

//**********

CAN2510SetBufferPriority

Функция: Загружает указанный приоритет для указанного буфера передачи.

Требуется CAN

Режим (ы): Все

Включает в себя: can2510.h

Прототип: void **CAN2510SetBufferPriority**(

unsigned char bufferNum,

unsigned char bufferPriority);

http://www.microchip.com/

Аргументы:

bufferNum

Задает буфер для настройки приоритета. Один из следующих значения:

CAN2510_TXB0 передачи буфера 0

CAN2510_TXB1 передачи буфера 1

CAN2510_TXB2 передачи буфера 2

bufferPriority

Приоритет буфера. Один из следующих значений:

CAN2510_PRI_HIGHEST Наивысший приоритет сообщение

CAN2510_PRI_HIGH высокий приоритет сообщение

CAN2510_PRI_LOW Низкий приоритет сообщение

CAN2510_PRI_LOWEST Самый низкий приоритет сообщение

Примечания: Эта функция загружает указанный приоритет индивидуальной буфера.

Имя файла: cansetpr.c

//**********

CAN2510SetMode

Функция: Настраивает режим MCP2510 работы.

Требуется CAN

Режим (ы): Все

Включает в себя: can2510.h

Прототип: void CAN2510SetMode(unsigned char mode);

Аргументы: режим

Значение режима может быть одним из следующих значений (в определенный can2510.h). Контролируется REQOP2: REQOP0 битов (CANCTRL зарегистрируйтесь). Один из следующих значений:

Регистры конфигурации CAN2510_MODE_CONFIG может быть изменение

CAN2510_MODE_NORMAL Нормальный (отправлять и получать сообщения)

CAN2510_MODE_SLEEP Подождите прерывания

http://www.microchip.com/

CAN2510_MODE_LISTEN Слушайте только, не отправить

CAN2510_MODE_LOOPBACK использовано для тестирования, сообщения остаются внутренний

Примечания: Эта функция настраивает указанный режим. Режим не изменится пока все ожидающие передачу сообщений, не будут выполнены.

Имя файла: canmodes.c

//**********

CAN2510SetMsgFilterStd

Функция: Настраивает ВСЕ фильтров и масок значений конкретных получать буфер для стандартного сообщения.

Требуется CAN

Режим (ы): Режим конфигурации

Включает в себя: can2510.h

Прототип:

unsigned char **CAN2510SetMsgFilterStd**(

unsigned char bufferNum,

unsigned int mask,

unsigned int *filters);

Аргументы: bufferNum

Определяет приемного буфера для настройки маски и фильтры для. Один из следующие значения:

CAN2510_RXB0 Настройка RXM0, RXF0 и RXF1

CAN2510_RXB1 Настройка RXM1, Rxf2, Rxf3, Rxf4 и Rxf5

маска

Значение для хранения в соответствующем маске

фильтры

Массив значений фильтра.

Для буфера 0

Сообщения стандартной длины: Массив 2 целых чисел без знака

Для буфера 1

http://www.microchip.com/

Сообщения стандартной длины: Массив 4 целых чисел без знака

Примечания: Эта функция настраивает MCP2510 в режиме настройки, то пишет маски и фильтра значения, чтобы соответствующие реестры. Перед возвращаясь, он настраивает MCP2510 в исходный режим.

Вернуться Значение: Указывает, режимы MCP2510 может быть изменен должным образом.

0, если инициализация и восстановление Режим работы завершены

-1, Если инициализация и восстановление Режим работы не завершить

Имя файла: canfms.c

//**********

CAN2510SetMsgFilterXtd

Функция: Настраивает ВСЕ фильтров и масок значений конкретных получать буфер для длительного сообщении.

Требуется CAN

Режим (ы): Режим конфигурации

Включает в себя: can2510.h

Прототип:

unsigned char **CAN2510SetMsgFilterXtd**(

unsigned char bufferNum,

unsigned long mask,

unsigned long *filters);

Аргументы:

bufferNum

Указывает буфер приема настроить маску и фильтры для одного из следующие значения:

CAN2510_RXB0 Настройка RXM0, RXF0 и RXF1

CAN2510_RXB1 Настройка RXM1, Rxf2, Rxf3, Rxf4 и Rxf5

маска

Значение для хранения в соответствующем маске

фильтры

Массив значений фильтра.

http://www.microchip.com/

Для буфера 0

Сообщения Extened длины: Массив 4 целых чисел без знака

Для буфера 1

Сообщения Extened длины: Массив 8 целых чисел без знака

Примечания: Эта функция настраивает MCP2510 в режиме настройки, то пишет маски и фильтра значения, чтобы соответствующие реестры. Перед возвращаясь, он настраивает MCP2510 в исходный режим.

Вернуться Значение: Указывает, режимы MCP2510 может быть изменен правильно:

0, если инициализация и восстановление Режим работы завершены

-1, Если инициализация и восстановление Режим работы не завершить

Имя файла: canfmx.c

//**********

CAN2510SetSingleFilterStd

Функция: Настраивает указано фильтр со значением фильтра для стандарта (Std) сообщение.

Требуется CAN

Режим (ы): Режим конфигурации

Включает в себя: can2510.h

Прототип:

void **CAN2510SetSingleFilterStd**(

unsigned char filterNum,

unsigned long filter);

Аргументы:

filterNum

Определяет принятие фильтра для настройки. Один из следующих значений:

CAN2510_RXF0 Настройка RXF0 (для RXB0)

CAN2510_RXF1 Настройка RXF1 (для RXB0)

CAN2510_RXF2 Настройка Rxf2 (для RXB1)

CAN2510_RXF3 Настройка Rxf3 (для RXB1)

http://www.microchip.com/

CAN2510_RXF4 Настройка Rxf4 (для RXB1)

CAN2510_RXF5 Настройка Rxf5 (для RXB1)

фильтр

Значение для хранения в соответствующем фильтра

Примечания: Эта функция записывает значение фильтра в соответствующие регистры.

MCP2510 должен находиться в режиме настройки перед выполнением этой функции.

Имя файла: canfilts.c

//**********

CAN2510SetSingleFilterXtd

Функция: Настраивает указано фильтр со значением фильтра для Расширенная (XTD) сообщение.

Требуется CAN

Режим (ы): Режим конфигурации

Включает в себя: can2510.h

Прототип:

void **CAN2510SetSingleFilterXtd**(

unsigned char filterNum,

unsigned int filter);

Аргументы:

filterNum

Определяет принятие фильтра для настройки. Один из следующих значений:

CAN2510_RXF0 Настройка RXF0 (для RXB0)

CAN2510_RXF1 Настройка RXF1 (для RXB0)

CAN2510_RXF2 Настройка Rxf2 (для RXB1)

CAN2510_RXF3 Настройка Rxf3 (для RXB1)

CAN2510_RXF4 Настройка Rxf4 (для RXB1)

CAN2510_RXF5 Настройка Rxf5 (для RXB1)

фильтр

Значение для хранения в соответствующем фильтра

http://www.microchip.com/

Примечания: Эта функция записывает значение фильтра в соответствующие регистры.

MCP2510 должен находиться в режиме настройки перед выполнением этой функции.

Имя файла: canfiltx.c

//**********

CAN2510SetSingleMaskStd

Функция: Настраивает указано буфера маску со значением маски для

Standard (Std) сообщение формат.

Требуется CAN

Режим (ы): Режим конфигурации

Включает в себя: can2510.h

Прототип:

unsigned char **CAN2510SetSingleMaskStd**(

unsigned char maskNum,

unsigned int mask);

Аргументы:

maskNum

Определяет принятие маска для настройки. Один из следующих значения:

CAN2510_RXM0 Настройка RXM0 (для RXB0)

CAN2510_RXM1 Настройка RXM1 (для RXB1)

маска

Значение для хранения в соответствующем маске

Примечания: Эта функция записывает значение маски в соответствующие реестры.

MCP2510 должен находиться в режиме настройки перед выполнением этой функции.

Имя файла: canmasks.c

//**********

CAN2510SetSingleMaskXtd

http://www.microchip.com/

Функция: Настраивает указано буфера маску со значением маски для Расширенный (XTD) сообщение.

Требуется CAN

Режим (ы): Режим конфигурации

Включает в себя: can2510.h

Прототип:

unsigned char **CAN2510SetSingleMaskXtd**(

unsigned char maskNum,

unsigned long mask);

Аргументы: maskNum

Определяет принятие маска для настройки. Один из следующих значения:

CAN2510_RXM0 Настройка RXM0 (для RXB0)

CAN2510_RXM1 Настройка RXM1 (для RXB1)

маска

Значение для хранения в соответствующем маске

Примечания: Эта функция записывает значение маски в соответствующие реестры.

MCP2510 должен находиться в режиме настройки перед выполнением этой функции.

Имя файла: canmaskx.c

//**********

CAN2510WriteStd

Функция: Пишет сообщение стандартного формата из к CAN шине с помощью первой доступны буфер передачи.

Требуется CAN

Режим (ы): Нормальный режим

Включает в себя: can2510.h

Прототип:

unsigned char **CAN2510WriteStd**(

unsigned int msgId,

http://www.microchip.com/

unsigned char msgPriority,

unsigned char numBytes,

unsigned char *data);

Аргументы:

MsgID

CAN идентификатор сообщения, 11 бит для стандартного сообщения. Это 11-битный

Идентификатор хранится в нижних 11 бит MsgID (целое без знака).

msgPriority

Приоритет буфера. Один из следующих значений:

CAN2510_PRI_HIGHEST Наивысший приоритет сообщение

CAN2510_PRI_HIGH высокая промежуточная приоритет сообщения

CAN2510_PRI_LOW Низкий промежуточный приоритет сообщения

CAN2510_PRI_LOWEST Самый низкий приоритет сообщение

NumBytes

Количество байтов данных, подлежащих передаче, с 0 по 8 Если значение больше, чем 8, только первые 8 байт данных будет отправлен.

данных

Массив значений данных для записи. Должно быть не меньше размера стоимости указано в NumBytes.

Примечания: Эта функция будет запрашивать каждый буфер передачи для отложенного сообщения, и вывесим указанное сообщение в первой доступной буфера.

Возвращаемые значения: Значение указывает, какой буфер был использован для передачи сообщения

(0, 1 или 2).

-1 Указывает, что ни одно сообщение не было отправлено.

Имя файла: canwrits.c

//**********

CAN2510WriteXtd

Функция: Пишет сообщение расширенном составе из к CAN шине с помощью первой

http://www.microchip.com/

доступны буфер передачи.

Требуется CAN

Режим (ы): Нормальный режим

Включает в себя: can2510.h

Прототип:

unsigned char **CAN2510WriteXtd**(

unsigned long msgId,

unsigned char msgPriority,

unsigned char numBytes,

unsigned char *data);

Аргументы:

MsgID

CAN идентификатор сообщения, 29 бит для длительного сообщении. Это 29-битный

Идентификатор хранится в нижних 29 бит MsgID (неподписанных долгое).

msgPriority

Приоритет буфера. Один из следующих значений:

CAN2510_PRI_HIGHEST Наивысший приоритет сообщение

CAN2510_PRI_HIGH высокая промежуточная приоритет сообщения

CAN2510_PRI_LOW Низкий промежуточный приоритет сообщения

CAN2510_PRI_LOWEST Самый низкий приоритет сообщение

NumBytes

Количество байтов данных, подлежащих передаче, с 0 по 8 Если значение больше, чем 8, только первые 8 байт данных будет отправлен.

данных

Массив значений данных для записи. Должно быть не меньше размера стоимости указано в NumBytes.

Примечания: Эта функция будет запрашивать каждый буфер передачи для отложенного сообщения, и вывесим указанное сообщение в первой доступной буфера.

Возвращаемые значения: Значение указывает, какой буфер был использован для передачи сообщения

http://www.microchip.com/

(0, 1 или 2).

-1 Указывает, что ни одно сообщение не было отправлено.

Имя файла: canwritx.c

//**********

3.4 ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ IIC ФУНКЦИИ

Эти функции разработаны, чтобы позволить осуществление шине I2C с использованием флажки ввода / вывода от PIC18 микроконтроллера. Следующие функции:

Таблица 3-6: I2C ФУНКЦИИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

Функция Описание

Clock_test Генерация задержку для рабского часы растяжения.

SWAckI2C Генерация шины I2C Признать состояние.

SWGetcI2C Считывает байт из шины I2C.

SWGetsI2C Считать строку данных.

SWNotAckI2C Генерация шины I2C Признать состояние.

SWPutI2C Написать одного байта к шине I2C.

SWPutsI2C Записать строку в шине I2C.

SWReadI2C Считывает байт из шины I2C.

SWRestartI2C Генерация состояние шины I2C перезапуска.

SWStartI2C Генерация состояние Start шины I2C.

SWStopI2C Генерация состояние шины I2C Stop.

SWWriteI2C Написать одного байта к шине I2C.

Прекомпилированные версии этих функций использовать по умолчанию назначением контактов, которые могут быть изменен переосмысление назначения макросов в файле sw_i2c.h, найти в ч подкаталог установки компилятора:

Таблица 3-7: MACROS для выбора I2C Назначение контактов

http://www.microchip.com/

ДАННЫЕ Pin DATA_PIN

DATA_LAT

DATA_LOW

DATA_HI

PORTBbits.RB4

LATBbits.RB4

TRISBbits.TRISB4 = 0;

TRISBbits.TRISB4 = 1;

Штифт используется для линии данных.

Защелка, связанная с выводами DATA.

О себе настроить данные

контактный качестве выхода.

О себе настроить данные

контактный качестве входа.

ЧАСЫ Pin SCLK_PIN

SCLK_LAT

CLOCK_LOW

CLOCK_HI

PORTBbits.RB3

LATBbits.LATB3

TRISBbits.TRISB3 = 0;

TRISBbits.TRISB3 = 1;

Pin используемый для часов линии.

Фиксация связано с

ЧАСЫ контактный.

Satement настроить

ЧАСЫ контактный качестве выхода.

О себе настроить

ЧАСЫ контактный качестве входа.

http://www.microchip.com/

После эти определения были сделаны, пользователь должен перекомпилировать подпрограммы I2C и затем использовать обновленные файлы в проекте. Это может быть достигнуто путем добавления в библиотеку исходные файлы в проект или по перекомпиляции библиотечные файлы с помощью прилагаемого партию файлы.

I2C линия макросы Значение по умолчанию Использование

//************************************

3.4.1 Описания функций

Clock_test

Функция: Генерация задержку для рабского часы растяжения.

Включает в себя: sw_i2c.h

Прототип: unsigned char **Clock_test**(void);

Примечания: Эта функция вызывается для обеспечения ведомые часы растяжения. Время задержки возможно, должны быть скорректированы в соответствии с требованиями приложений. Если в конце период задержки часы линии низкое, значение возвращается с указанием часы Ошибка.

Возвращаемое значение:

0 возвращается, если не произошло никаких часов ошибке

-2 Возвращается, если произошла ошибка часы

Имя файла: swckti2c.c

//************************************

SWAckI2C

SWNotAckI2C

Функция: Генерация шины I2C Признать состояние.

Включает в себя: sw_i2c.h

Прототип:

http://www.microchip.com/

unsigned char **SWAckI2C**(void);

unsigned char **SWNotAckI2C**(void);

Примечания: Эта функция вызывается для генерации шины I2C последовательность Подтверждение.

Возвращаемое значение:

0, если раб признает

-1, Если раб не Подтверждение

Имя файла: swacki2c.c

//**********

SWGetcI2C

Смотреть SWReadI2C.

SWGetsI2C

Функция: Чтение строки из шине I2C.

Включает в себя: sw_i2c.h

Прототип:

unsigned char **SWGetsI2C**(

unsigned char *rdptr,

unsigned char length);

Аргументы:

rdptr

Расположение для хранения данных, считанных из шине I2C.

длина

Количество байт для чтения.

Примечания: Функция читает строку заданной длины.

Возвращаемые значения:

-1, если мастер генерируется состояние автобусов не ACK до всех байтов были получены

0 в противном случае

http://www.microchip.com/

Имя файла: swgtsi2c.c

Пример кода:

char x[10];

SWGetsI2C(x,5);

//**********************************

SWNotAckI2C

Смотреть SWAckI2C.

SWPutcI2C

Смотреть SWWriteI2C.

SWPutsI2C

Функция: Записать строку в шине I2C.

Включает в себя: sw_i2c.h

Прототип:

unsigned char **SWPutsI2C**(

    unsigned char *wrdptr);

Аргументы:

wrdptr

Указатель на данные, которые будут записаны в шине I2C.

Примечания: Эта функция записывает строку данных до (но не включая) нуль характер.

Возвращаемые значения:

-1, если произошла ошибка записи на шине I2C

0 в противном случае

Имя файла: swptsi2c.c

Пример кода:

char mybuff [20];

SWPutsI2C(mybuff);

SWReadI2C

http://www.microchip.com/

SWGetcI2C

Функция: Считывает байт из шины I2C.

Включает в себя: sw_i2c.h

Прототип:

unsigned char **SWReadI2C**(void);

Примечания: Функция читает один байт данных, генерируя соответствующий сигналы на предопределенной I2C тактовой линии. Возвращаемые значения: Эта функция возвращает полученные байт I2C данных.

-1, Если произошла ошибка в этой функции.

Имя файла: swgtci2c.c

//**********

SWRestartI2C

Функция: Генерация состояние шины I2C перезапуска.

Включает в себя: sw_i2c.h

Прототип: void **SWRestartI2C**(void);

Примечания: Эта функция вызывается для генерации состояние автобуса перезапуска I2C.

Имя файла: swrsti2c.c

//**********

SWStartI2C

Функция: Генерация состояние Start шины I2C.

Включает в себя: sw_i2c.h

Прототип: void **SWStartI2C**(void);

Примечания: Эта функция вызывается для генерации состояние Start шины I2C.

Имя файла: swstri2c.c

//**********

http://www.microchip.com/

SWStopI2C

Функция: Генерация состояние шины I2C Stop.

Включает в себя: sw_i2c.h

Прототип: void **SWStopI2C**(void);

Примечания: Эта функция вызывается для генерации состояние шины I2C Stop.

Имя файла: swstpi2c.c

//**********

SWWriteI2C

SWPutcI2C

Функция: Написать байт в шине I2C.

Включает в себя: sw_i2c.h

Прототип:

unsigned char **SWWriteI2C**(

    unsigned char data_out);

Аргументы:

data_out

Одноместный байт данных записывается в устройства I2C.

Примечания: Эта функция записывает один байт данных в предопределенной штифта данных.

Возвращаемое значение:

0, если запись прошла успешно

-1, Если было условие ошибки

Имя файла: swptci2c.c

Пример кода

```
if(SWWriteI2C(0x80))
 {
   errorHandler();
```

```
http://www.microchip.com/
 }
//************************************
```

3.4.2 Пример применения

Ниже приведен простой пример, иллюстрирующий реализацию программного обеспечения I2C связи с устройством памяти Microchip 24LC01B I2C EE.

```
#include <p18cxxx.h>
#include <sw_i2c.h>
#include <delays.h>
// Прототип функции
void main(void);
void byte_write(void);
void page_write(void);
void current_address(void);
void random_read(void);
void sequential_read(void);
void ack_poll(void);
unsigned char warr[] = {8,7,6,5,4,3,2,1,0};
unsigned char rarr[15];
unsigned char far *rdptr = rarr;
unsigned char far *wrptr = warr;
unsigned char var;
#define W_CS  PORTA.2
//************************************************
void main( void )
{
  byte_write();
```

```
http://www.microchip.com/
  ack_poll();
  page_write();
  ack_poll();
  Nop();
  sequential_read();
  Nop();
  while (1);  // Цикл неопределенный срок
}

//---------------------------------------------------------------

void byte_write( void )
{
   SWStartI2C ();
  var = SWPutcI2C (0xA0); // Управляющий байт
   SWAckI2C ();
  var = SWPutcI2C (0x10); // Адрес слова
   SWAckI2C ();
  var = SWPutcI2C (0x66); // данных
   SWAckI2C ();
   SWStopI2C ();
}
//---------------------------------------------------------------
void page_write( void )
{
   SWStartI2C ();
  var = SWPutcI2C (0xA0); // Управляющий байт
   SWAckI2C ();
  var = SWPutcI2C (0x20); // Адрес слова
```

```
http://www.microchip.com/
   SWAckI2C ();
  var = SWPutsI2C (wrptr); // данных
   SWStopI2C ();
}

//---------------------------------------------------------------
void sequential_read( void )
{
   SWStartI2C ();
  var = SWPutcI2C (0xA0); // Управляющий байт
   SWAckI2C ();
  var = SWPutcI2C (0x00); // Адрес для чтения из
   SWAckI2C ();
   SWRestartI2C ();
  var = SWPutcI2C (0xA1);
   SWAckI2C ();
  var = SWGetsI2C (rdptr, 9);
   SWStopI2C ();
}
//---------------------------------------------------------------

void current_address( void )
{
   SWStartI2C ();
   SWPutcI2C (0xA1); // Управляющий байт
   SWAckI2C ();
   SWGetcI2C (); // Адрес слова
   SWNotAckI2C ();
   SWStopI2C ();
```

```
http://www.microchip.com/
}
//-----------------------------------------------------------


void ack_poll( void )
{
  SWStartI2C ();
 var = SWPutcI2C (0xA0); // Управляющий байт
 while( SWAckI2C() )
  {
    SWRestartI2C ();
   var = SWPutcI2C (0xA0); // данных
  }
  SWStopI2C ();
}


//************************************
```

3.5 ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ SPI® ФУНКЦИИ

Эти функции разработаны, чтобы позволить внедрение SPI используя флажки ввода / вывода от PIC18 микроконтроллеров. Следующие функции:

Таблица 3-8: ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ SPI ФУНКЦИИ

Функция Описание

ClearSWCSSPI Ясно выбора микросхемы (CS) контактный.

OpenSWSPI Настройка ввода / вывода общего назначения для применения в качестве SPI.

putcSWSPI Написать байт данных в программное обеспечение SPI.

SetSWCSSPI указан чип выберите (CS) контактный.

WriteSWSPI Написать байт данных в программное обеспечение SPI автобусе.

http://www.microchip.com/

Прекомпилированные версии этих функций использовать по умолчанию назначением контактов, которые могут быть изменен переосмысление назначения макросов в файле sw_spi.h, найти в ч подкаталог установки компилятора:

Таблица 3-9: MACROS ВЫБОРА SPI Назначение контактов

ЖК

контроллер

линия

Макросы Значение по умолчанию Использование

CS Pin SW_CS_PIN

TRIS_SW_CS_PIN

PORTBbits.RB2

TRISBbits.TRISB2

Pin используется для выбора микросхемы (CS)

линия.

Бит, который контролирует направление

контактный связано с CS

линия.

DIN Pin SW_DIN_PIN

TRIS_SW_DIN_PIN

PORTBbits.RB3

TRISBbits.TRISB3

Pin используется для DIN линии.

Бит, который контролирует направление

контактный связан с DIN

линия.

DOUT Pin SW_DOUT_PIN

TRIS_SW_DOUT_PIN

PORTBbits.RB7

http://www.microchip.com/

TRISBbits.TRISB7

Pin используется для линии DOUT.

Бит, который контролирует направление

контактный связаны с

DOUT линия.

СХК Pin SW_SCK_PIN

TRIS_SW_SCK_PIN

PORTBbits.RB6

TRISBbits.TRISB6

Штифт используется для SCK линии.

Бит, который контролирует направление

контактный связан с SCK

линия.

Библиотеки, которые предоставляются может работать в одном из четырех режимов. В таблице ниже приведены макросы используются для выбора между этими режимами. Именно один из них должен быть определяется при перестройке программных SPI библиотеки.

Таблица 3-10: макросы для выбора режимов

Макро Значение по умолчанию Значение

Mode0 определяется СКР = 0

СКЕ = 0

РЕЖИМ1 не определен СКР = 1

СКЕ = 0

MODE2 не определен СКР = 0

СКЕ = 1

MODE3 не определен СКР = 1

СКЕ = 1

http://www.microchip.com/

После эти определения были сделаны, пользователь должен перекомпилировать программный SPI процедуры, а затем включают обновленные файлы в проекте. Это может быть достигнуто путем добавляя программные SPI исходные файлы в проект или перекомпиляцией библиотечные файлы используя предоставленные командные файлы.

//**********

3.5.1 Описания функций

ClearSWCSSPI

Функция: Очистите чип выберите (CS) контактный, указанный в sw_spi.h заголовка подать.

Включает в себя: sw_spi.h

Прототип: void **ClearSWCSSPI**(void);

Примечания: Эта функция удаляет I / O контактный, который указан в sw_spi.h быть Выбор микросхемы (CS) штифт для программного обеспечения SPI.

Имя файла: clrcsspi.c

//**********

OpenSWSPI

Функция: Настройка флажки ввода / вывода для обеспечения SPI.

Включает в себя: sw_spi.h

Прототип: void **OpenSWSPI**(void);

Примечания: Эта функция настраивает флажки ввода / вывода, используемые для программного обеспечения SPI для правильный вход или выход; состояние и логический уровень.

Имя файла: opensspi.c

putcSWSPI

Смотреть WriteSWSPI.

//**********

http://www.microchip.com/

3.5.2 Пример применения

```
#include <p18C452.h>

#include <sw_spi.h>

#include <delays.h>

void main( void )

{

 char address;

  // Настроить программное обеспечение SPI

  OpenSWSPI ();

 for( address=0; address<0x10; address++ )

  {

    ClearCSSWSPI (); // ясно CS контактный

    WriteSWSPI (0x02); // отправить запись ЦМД

    WriteSWSPI (address); // отправить адрес привет

    WriteSWSPI (address); // отправить адрес низкий

    SetCSSWSPI (); // установить CS контактный

    Delay10KTCYx (50); // ждать 5000,000TCY

  }

}


//************************************
```

SetSWCSSPI

Функция: Установка для выбора микросхемы (CS) штифт, который указан в заголовке sw_spi.h подать.

Включает в себя: sw_spi.h

Прототип: void **SetSWCSSPI**(void);

http://www.microchip.com/

Примечания: Эта функция устанавливает I / O контактный, который указан в sw_spi.h быть Выбор микросхемы (CS) штифт для программного обеспечения SPI.

Имя файла: setcsspi.c

//**********

WriteSWSPI

putcSWSPI

Функция: Написать байт к ПО SPI.

Включает в себя: sw_spi.h

Прототип: char **WriteSWSPI**(char data);

Аргументы:

данные

Данные, которые будут записаны в программное обеспечение SPI.

Примечания: Данная функция позволяет записывать указанный байт данных из программного обеспечения SPI и возвращает байт данных, что было читать. Эта функция не обеспечивает любой контроль выбора микросхемы штифта (CS).

Вернуться Значение: Эта функция возвращает байт данных, что было читать из данных (DIN) контактный программного обеспечения SPI.

Имя файла: wrtsspi.c

Пример кода:

```
char addr = 0x10;

char result;

result = WriteSWSPI( addr );
```

//**********

3.6 ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ UART ФУНКЦИИ

Эти функции разработаны, чтобы позволить реализацию UART, используя флажки ввода / вывода

от PIC18 микроконтроллера. Следующие функции:

Таблица 3-11: ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ UART ФУНКЦИИ

http://www.microchip.com/

Функция Описание

getcUART Считывает байт из программного UART.

getsUART Считать строку из программного UART.

OpenUART Контакты / вывода Настройка I для использования в качестве UART.

putcUART Написать байт в программное обеспечение UART.

putsUART Записать строку в программной UART.

ReadUART Считывает байт из программного UART.

WriteUART Написать байт в программное обеспечение UART.

Прекомпилированные версии этих функций использовать по умолчанию назначением контактов, которые могут быть изменен переосмысление приравнивать (фас) заявления в writuart.asm файлов, readuart.asm и openuart.asm, найти в SRC / традиционной / PMC / sw_uart или Скрин / расширенная / PMC sw_uart подкаталог установки компилятора /:

Таблица 3-12: MACROS выбора UART Назначение контактов

ЖК

контроллер

линия

Определение Значение по умолчанию Использование

TX Pin SWTXD

SWTXDpin

TRIS_SWTXD

PORTB

4

TRISB

Порт используется для передачи линии.

Бит в SWTXD порта для линии TX.

Направление данных Зарегистрироваться связано с

порт, используемый для линии TX.

http://www.microchip.com/

RX Pin SWRXD

SWRXDpin

TRIS_SWRXD

PORTB

5

TRISB

Порт используется для получения линии.

Бит в порту SWRXD, используемого для линии RX.

Направление данных Зарегистрироваться связано с

порт, используемый для линии RX.

Если изменения в этих определений сделаны, пользователь должен перекомпилировать программный UART процедуры, а затем включают обновленные файлы в проекте. Это может быть достигнуто путем добавив программное обеспечение UART исходные файлы в проект или перекомпиляцией библиотечные файлы используя командные файлы, поставляемые с установкой MPLAB C18 компилятора. В UART библиотеки также требуют, чтобы следующие функции быть определены пользователем для предоставить соответствующие задержки:

Таблица 3-13: ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ UART Функция задержки

Функция Поведение

DelayTXBitUART Задержка:

((((2 * FOSC) / (4 * бод)) + 1) / 2) - 12 циклов

DelayRXHalfBitUART Задержка:

((((2 * FOSC) / (8 * бод)) + 1) / 2) - 9 циклов

DelayRXBitUART Задержка:

((((2 * FOSC) / (4 * бод)) + 1) / 2) - 14 циклов

//**********

3.6.1 Описания функций

getcUART

Смотреть ReadUART.

http://www.microchip.com/

getsUART

Функция: Чтение строки из программного UART.

Включает в себя: sw_uart.h

Прототип:

void **getsUART**(char * buffer,

unsigned char len);

Аргументы:

буфер

Указатель на строку символов считываются из программного UART.

Лен

Количество символов для чтения из программного UART.

Примечания: Эта функция читает длина символов с UART программного обеспечения и мест их в буфере.

Имя файла: getsuart.c

Пример кода:

char x[10];

getsUART(x, 5);

//**********

OpenUART

Функция: Настройка флажки ввода / вывода для программного UART.

Включает в себя: sw_uart.h

Прототип: void **OpenUART**(void);

Примечания: Эта функция настраивает флажки ввода / вывода, используемые для программного UART к правильный вход или выход; состояние и логический уровень.

Имя файла: openuart.asm

Пример кода:

OpenUART ();

//**********

http://www.microchip.com/

putcUART

Смотреть WriteUART.

putsUART

Функция: Написать строку программного UART.

Включает в себя: sw_uart.h

Прототип: void **putsUART**(char * buffer);

Аргументы:

буфер

Строка, которая будет написана в программной UART.

Примечания: Данная функция позволяет записывать строку символов в программное обеспечение UART. вся строка, включая нулевой отправляется в UART.

Имя файла: putsuart.c

Пример кода:

char mybuff [20];

putsUART(mybuff);

//**********

ReadUART

getcUART

Функция: Считывает байт из программного UART.

Включает в себя: sw_uart.h

Прототип:

char **ReadUART** (void);

Примечания: Эта функция считывает байт данных выход программного UART.

Возвращаемое значение: байт данных, которые были читать от приема данных (RXD) штифта программного UART.

Имя файла: readuart.asm

Пример кода:

http://www.microchip.com/

символ x;

x = ReadUART ();

//**********

WriteUART

putcUART

Функция: Написать байт в программное обеспечение UART.

Включает в себя: sw_uart.h

Прототип: void **WriteUART** (data CHAR);

Аргументы:

данные

Байт данных должны быть записаны в программное обеспечение UART.

Примечания: Данная функция позволяет записывать указанный байт данных из программного обеспечения UART.

Имя файла: writuart.asm

Пример кода:

char mybuff [20];

putsUART(mybuff);

//**********

3.6.2 Пример применения

#include <p18C452.h>

#include <sw_uart.h>

void main(void)

{

```
http://www.microchip.com/
 char data
  // Настроить программное обеспечение UART
 OpenUART();
 while( 1 )
 {
   data = ReadUART();   // прочитать байт
    WriteUART (data); // отказов его обратно
  }
}



//************************************
```

Глава 4 Общие Библиотека программирования

4.1 ВВЕДЕНИЕ

В этой главе документы целом программные библиотечные функции, найденные в скомпилированных Стандарт С файл библиотеки. Исходный код для всех этих функций входит

MPLAB C18 в следующих подкаталогах установки компилятора:

• SRC \ традиционный \ STDLIB

• SRC \ продлен \ STDLIB

• SRC \ традиционные \ задержки

• SRC \ расширенные \ задержки

Следующие категории подпрограмм поддерживаются библиотеки MPLAB C18:

• Характер Классификация Функции

• Функции преобразования данных

• Память и Функции обработки строк

• Функция задержки

• Функции сброса

http://www.microchip.com/

• Характер Выходные Функции

//**********

4.2 классификацию символов ФУНКЦИИ

Эти функции в соответствии с ANSI 1989 стандартной библиотечной функции одно название. Следующие функции:

Таблица 4-1: классификацию символов ФУНКЦИИ

Функция Описание

isalnum Определить, является ли символ алфавитно-цифровой.

ISALPHA Определите, если символ буквенный.

iscntrl Определить, является ли символ управляющим символом.

ISDIGIT Определить, является ли символ десятичной цифры.

isgraph Определить, является ли символ графическим характер.

ISLOWER Определить, является ли символ ниже алфавитный символ.

isprint Определить, является ли символ печатный символ.

ispunct Определить, является ли символ символом пунктуации.

isspace Определить, является ли символ символ пробела.

ISUPPER Определить, является ли символ верхней алфавитный символ.

isxdigit Определить, является ли символ шестнадцатеричной цифрой.

//**********//

4.2.1 Описания функций

isalnum

Функция: Определить, является ли символ алфавитно-цифровой.

Включает в себя: ctype.h

Прототип:

unsigned char **isalnum**(unsigned char ch);

Аргументы: ch

http://www.microchip.com/

Характер быть проверены.

Примечания: Персонаж считается буквенно-цифровой, если он находится в диапазоне от 'A' до 'Z', '' до 'Я' или «0» до «9».

Возвращаемые значения: ненулевых если символ алфавитно-цифровым

Ноль в противном случае

Имя файла: isalnum.c

//**********//

ISALPHA

Функция: Определите, если символ буквенный.

Включает в себя: ctype.h

Прототип:

unsigned char **isalpha**(unsigned char ch);

Аргументы: ch

Характер быть проверены.

Примечания: Персонаж считается алфавитном если он находится в диапазоне от 'A' до 'Я' или '' до 'Я'.

Возвращаемые значения: не нуль, если символ буквенный

Ноль в противном случае

Имя файла: isalpha.c

//**********//

iscntrl

Функция: Определить, является ли символ управляющим символом.

Включает в себя: ctype.h

Прототип:

unsigned char **iscntrl**(unsigned char ch);

Аргументы: ch

http://www.microchip.com/

Характер быть проверены.

Примечания: Персонаж считается управляющий символ, если это не для печати символов, как это определено isprint ().

Возвращаемые значения: ненулевых если символ это символ управления

Ноль в противном случае

Имя файла: iscntrl.c

//***//

ISDIGIT

Функция: Определить, является ли символ десятичной цифры.

Включает в себя: ctype.h

Прототип:

unsigned char **isdigit**(unsigned char ch);

Аргументы: ch

Характер быть проверены.

Примечание: символ считается цифровой символ, если он находится в диапазоне от "0"до «9». Возвращаемые значения: ненулевых если символ является цифрой характер

Ноль в противном случае

Имя файла: isdigit.c

//***//

isgraph

Функция: Определить, является ли символ графическим характер.

Включает в себя: ctype.h

Прототип:

unsigned char **isgraph**(unsigned char ch);

Аргументы: ch

Характер быть проверены.

Примечания: Персонаж считается графический алфавитный символ, если это любой печатный символ, кроме пространства.

http://www.microchip.com/

Возвращаемые значения:

ненулевых, если символ является графическим характер

Ноль в противном случае

Имя файла: isgraph.c

//***//

ISLOWER

Функция: Определить, является ли символ ниже алфавитный символ.

Включает в себя: ctype.h

Прототип: unsigned char **islower**(unsigned char ch);

Аргументы: ch

Характер быть проверены.

Примечания: Персонаж считается ниже алфавитный символ, если это в диапазоне от 'A' до 'Я'.

Возвращаемые значения:

не нуль, если персонаж находится ниже алфавитный символ

Ноль в противном случае

Имя файла: islower.c

//***//

isprint

Функция: Определить, является ли символ печатный символ.

Включает в себя: ctype.h

Прототип:

unsigned char **isprint**(unsigned char ch);

Аргументы: ch

Характер быть проверены.

Примечание: символ считается печатный символ, если он находится в пределах 0x20 до 0x7E, включительно.

http://www.microchip.com/

Возвращаемые значения:

не нуль, если персонаж находится печатный символ

Ноль в противном случае

Имя файла: isprint.c

//***//

ispunct

Функция: Определить, является ли символ символом пунктуации.

Включает в себя: ctype.h

Прототип: unsigned char **ispunct**(unsigned char ch);

Аргументы: ch

Характер быть проверены.

Примечания: Персонаж считается символом пунктуации, если это печатный символ, который не является ни пространства, ни буквенно-цифровой характер.

Возвращаемые значения:

не нуль, если символ является символом пунктуации

Ноль в противном случае

Имя файла: ispunct.c

//***//

isspace

Функция: Определить, является ли символ символ пробела.

Включает в себя: ctype.h

Прототип:

unsigned char **isspace** (unsigned char ch);

Аргументы: ch

Характер быть проверены.

Примечания: Персонаж считается символ пробела, если он является одним из следующее: пространство (''), вкладка ('\ т'), возврат каретки ('\ г'), новая линия ('\ п'), форма подачи ('\ е') или вертикальная табуляция ('\ V').

http://www.microchip.com/

Возвращаемые значения:

не нуль, если символ является символ пробела

Ноль в противном случае

Имя файла: isspace.c

//***//

ISUPPER

Функция: Определить, является ли символ верхней алфавитный символ.

Включает в себя: ctype.h

Прототип:

unsigned char **isupper** (unsigned char ch);

Аргументы: ch

Характер быть проверены.

Примечания: Персонаж считается верхняя алфавитный символ, если он находится в диапазоне от 'А' до 'Я'.

Возвращаемые значения:

ненулевых, если символ является верхним алфавитный символ

Ноль в противном случае

Имя файла: isupper.c

//***//

isxdigit

Функция: Определить, является ли символ шестнадцатеричной цифрой.

Включает в себя: ctype.h

Прототип: unsigned char **isxdigit**(unsigned char ch);

Аргументы: ch

Характер быть проверены.

http://www.microchip.com/

Примечания: Персонаж считается шестнадцатеричное число символов, если он находится в диапазон «0» до «9», 'A' до 'F' или 'A' до 'F'.

Возвращаемые значения:

не нуль, если символ является шестнадцатеричное число символов

Ноль в противном случае

Имя файла: isxdig.c

//***//

4.3 Функции преобразования данных

За исключением указанных в описании функций, эти функции в соответствии с ANSI 1989 стандартной библиотечной функции с тем же именем. Функции, предоставляемые являются:

Таблица 4-2: ФУНКЦИИ для преобразования данных

Функция Описание

atob Преобразование в строку в 8-битового байта.

atof Преобразует строку в значение с плавающей запятой.

atoi Преобразование строки в 16-разрядное целое число.

atol Преобразование строки в длинное целое представление.

btoa Преобразование 8-разрядное байт в строку.

itoa Преобразование 16-разрядное целое число в строку.

ltoa Преобразование длинное целое в строку.

rand Генерация псевдослучайное целое.

srand указан начальную семя для числа генератора псевдослучайных чисел.

ToLower Преобразование символ в нижний регистр алфавитном ASCII символа.

ToUpper Преобразование символ в верхний регистр алфавитном ASCII символа.

ultoa Преобразование длинное целое без знака в строку.

4.3.1 Описания функций

http://www.microchip.com/

//**//

atob

Функция: Преобразование в строку в 8-битового байта.

Включает в себя: stdlib.h

Прототип:

signed char **atob**(const char * s);

Аргументы: s

Указатель на ASCII строки должны быть преобразованы.

Примечания: Эта функция преобразовывает ASCII строку S в 8-битового байта (-128 до 127). Входная строка должна быть в базе 10 (десятичной системе счисления) и может начинаться с символов с указанием знака («+» или «-»). Переполнение результаты не определено. Эта функция является расширением MPLAB C18 с ANSI стандартные библиотеки.

Вернуться Значение: 8-разрядный байт для всех строк в диапазоне (-128 до 127).

Имя файла: atob.asm

//**//

atof

Функция: Преобразование строки в значение с плавающей запятой.

Включает в себя: stdlib.h

Прототип:

double **atof** (const char * s);

Аргументы: s

Указатель на ASCII строки должны быть преобразованы.

Примечания: Эта функция преобразует ASCII строку **S** в значение с плавающей запятой.

Примеры строк с плавающей запятой, которые признаны являются:

-3,1415

1.0E2

1.0E + 2

http://www.microchip.com/

1.0E-2

Возвращаемые значения: Функция возвращает преобразованное значение.

Имя файла: atof.c

//**//

atoi

Функция: Преобразование строки в 16-разрядное целое число.

Включает в себя: stdlib.h

Прототип:

int **atoi**(const char * s);

Аргументы: s

Указатель на ASCII строки должны быть преобразованы.

Примечания: Эта функция преобразовывает ASCII строку **S** в 16-разрядное целое число (-32768 до 32767). Входная строка должна быть в базе 10 (десятичной системе счисления) и может начинаться с символов с указанием знака («+» или «-»). Переполнение Результаты не определены. Эта функция является расширением MPLAB C18 для ANSI стандартные библиотеки.

Возвращаемые значения: 16-разрядное целое число для всех строк в диапазоне (-32768 до 32767).

Имя файла: atoi.asm

//**//

atol

Функция: Преобразование строки в длинное целое представление.

Включает в себя: stdlib.h

Прототип:

long **atol**(const char * s);

Аргументы: s

Указатель на ASCII строки должны быть преобразованы.

Примечания: Эта функция преобразует ASCII строке **S** с в течение длительного значения. входной Строка должна быть в базе 10 (десятичной системе счисления) и может начинаться с тличающаяся

тер с указанием знака («+» или «-»). Переполнение результат не определен. Это Функция является продолжением MPLAB C18 для стандартных библиотек ANSI.

Возвращаемые значения: Функция возвращает преобразованное значение.

Имя файла: atol.asm

//***//

btoa

Функция: Преобразование 8-разрядное байт в строку.

Включает в себя: stdlib.h

Прототип:

```
char * btoa( signed char value,
        char * string );
```

Аргументы:

перемення

8-разрядное байт.

строка

Указатель на ASCII строкой, которая будет содержать результат. Строка должна быть длинной достаточно, чтобы провести представление ASCII, включая знаковый характер для отрицательных значений и задней нулевого символа.

Примечания: Эта функция преобразует 8-разрядное байт в значение аргумента в ASCII строка представление.

Эта функция является расширением MPLAB C18 из ANSI требуется библиотеки.

Возвращаемое значение: указатель на результирующую строку.

Имя файла: btoa.asm

//***//

itoa

Функция: Преобразование 16-разрядное целое число в строку.

Включает в себя: stdlib.h

Прототип:

http://www.microchip.com/

char * **itoa**(int value, char * string);

Аргументы:

значение

8-разрядное байт.

строка

Указатель на ASCII строкой, которая будет содержать результат. Строка должна быть длинной достаточно, чтобы провести представление ASCII, включая знаковый характер для отрицательных значений и задней нулевого символа.

Примечания: Эта функция преобразует 16-разрядное целое число в значение аргумента до представления ASCII строки.

Эта функция является расширением MPLAB C18 из ANSI требуется библиотеки.

Возвращаемое значение: указатель на результирующую строку.

Имя файла: itoa.asm

//***//

ltoa

Функция: Преобразование длинное целое в строку.

Включает в себя: stdlib.h

Прототип:

char * **ltoa**(long value, char * string);

Аргументы:

значение

Длинное целое, чтобы быть преобразованы.

строка

Указатель на ASCII строкой, которая будет содержать результат.

Примечания: Эта функция преобразует длинное целое в значение аргумента до представления ASCII строки. Строка должна быть достаточно длинной, чтобы держать представление ASCII, включая знаковый характер для отрицательной ценности и конечные нулевой символ. Эта функция является MPLAB C18 расширение до требуемых ANSI библиотек.

Возвращаемое значение: указатель на результирующую строку.

Имя файла: ltoa.asm

http://www.microchip.com/

//***//

rand

Функция: Генерация псевдослучайное целое.

Включает в себя: stdlib.h

Прототип:

int **rand**(void);

Примечания: Звонки на эта функция возвращает псевдослучайных целых значений в диапазоне [0,32767]. Для эффективного использования этой функции необходимо отобрать случайным генератор чисел с помощью функции **srand** (). Эта функция будет всегда возвращают ту же последовательность чисел, когда идентичны seed значения используются.

Возвращаемые значения: целое значение псевдо-случайного.

Имя файла: rand.asm

//***//

srand

Функция: Установка отправной семя для последовательности псевдослучайных чисел.

Включает в себя: stdlib.h

Прототип:

void **rand**(unsigned int seed);

Аргументы: **seed**

Начальное значение для последовательности псевдослучайных чисел.

Примечания: Эта функция устанавливает начальную **seed** для псевдослучайных чисел последовательность генерируется с помощью функции **rand**().**rand**() функция всегда возвращают ту же последовательность чисел, когда идентичны **seed** значения используются. Если **rand**() вызывается без **srand** (), имеющих первую был называется последовательность чисел, сгенерированных будет таким же, как если бы **srand** () был вызван с начальным значением 1.

Имя файла: rand.asm

//***//

http://www.microchip.com/

ToLower

Функция: Преобразование символ в нижний регистр алфавитном ASCII символа.

Включает в себя: ctype.h

Прототип:

char **tolower**(char ch);

Аргументы: **ch**

Персонаж для преобразования.

Примечания: Эта функция преобразует **ch** к нижним регистром алфавитном ASCII символа при условии, что аргумент является действительным в верхнем регистре алфавита.

Вернуться Значение: Эта функция возвращает строчную характер, если аргумент был верхний случай для начала; в противном случае исходный символ возвращается.

Имя файла: tolower.c

//***//

ToUpper

Функция: Преобразование символ в верхний регистр алфавитном ASCII символа.

Включает в себя: ctype.h

Прототип:

char **toupper**(char ch);

Аргументы: **ch**

Персонаж для преобразования.

Примечания: Эта функция преобразует **ch** к верхний регистр алфавитном ASCII символа при условии, что аргумент является действительным строчная буквы в алфавитном порядке.

Вернуться Значение: Эта функция возвращает символ верхнего регистра, если аргумент был нижний регистр, чтобы начать с; в противном случае исходный символ возвращается.

Имя файла: toupper.c

//***//

4.4 Память и Функции обработки строк

http://www.microchip.com/

За исключением указанных в описании функций, эти функции в соответствии с ANSI (1989) стандартной библиотеки языка Си функции с тем же именем. Следующие функции при условии:

//**//

ultoa

Функция: Преобразование длинное целое без знака в строку.

Включает в себя: stdlib.h

Прототип:

char * **ultoa**(unsigned long value,

char * string);

Аргументы:

значение

Неподписанных долгое целое число для преобразования.

строка

Указатель на ASCII строкой, которая будет содержать результат.

Примечания: Эта функция преобразует длинное целое без знака в аргументе Значение в представлении ASCII строки. Строка должна быть достаточно длинной, провести представление ASCII, в том числе задней нулевого символа. Это Функция является продолжением MPLAB C18 до требуемых ANSI библиотек.

Возвращаемое значение: указатель на результирующую строку.

Имя файла: ultoa.asm

//**//

ТАБЛИЦА 4-3: ПАМЯТЬ И Функции обработки строк

Функция Описание

memchr

memchrpgm Поиск значения в заданной области памяти.

memcmp

http://www.microchip.com/

memcmppgm

memcmppgm2ram

memcmpram2pgm Сравните содержимое двух массивов.

memcpy

memcpypgm

memcpypgm2ram

memcpyram2pgm Скопируйте буфер.

memmove

memmovepgm

memmovepgm2ram

memmoveram2pgm Скопируйте буфер, где источник и пункт назначения могут перекрываться.

MemSet

memsetpgm Инициализировать массив с одной повторной стоимости.

Strcat

strcatpgm

strcatpgm2ram

strcatram2pgm Добавить копию исходной строки в конце назначения строка.

strchr

strchrpgm Найдите первое вхождение значения в строку.

strcmp

strcmppgm

strcmppgm2ram

strcmpram2pgm Сравнение двух строк.

http://www.microchip.com/

strcpy

strcpypgm

strcpypgm2ram

strcpyram2pgm Скопируйте строку из данных или памяти программ в память данных.

strcspn

strcspnpgm

strcspnpgmram

strcspnrampgm Рассчитать количество последовательных символов в начале Строка, не содержатся в наборе символов.

StrLen

strlenpgm Определить длину строки.

strlwr

strlwrpgm Конвертировать все символы верхнего регистра в строке в нижний регистр.

strncat

strncatpgm

strncatpgm2ram

strncatram2pgm Добавляет указанное количество символов из исходной строки в концом строки назначения.

strncmp

strncmppgm

strncmppgm2ram

strncmpram2pgm Сравнение двух строк, до определенного числа символов.

http://www.microchip.com/

strncpy

strncpypgm

strncpypgm2ram

strncpyram2pgm Скопируйте символы исходной строки в строку назначения, до в указанное число символов.

strpbrk

strpbrkpgm

strpbrkpgmram

strpbrkrampgm Поиск строку для первого вхождения символа из набора символов.

strrchr

strrchrpgm Найдите последнюю вхождение указанного символа в строке.

strspn

strspnpgm

strspnpgmram

strspnrampgm Рассчитать количество последовательных символов в начале строка, которая содержится в наборе символов.

strstr

strstrpgm

strstrpgmram

strstrrampgm Найдите первое вхождение строки внутри другой строки.

Strtok

strtokpgm

strtokpgmram

strtokrampgm Перерыв строку на подстроки или жетонов, вставив нулевые символы вместо указанных разделителей.

http://www.microchip.com/

strupr

struprpgm Конвертировать все строчные символы в строке в верхний регистр.

//***//

4.4.1 Описания функций

memchr

memchrpgm

Функция: Найдите первое вхождение значения байта в указанной памяти область.

Включает в себя: string.h

Прототип:

void * **memchr**(const void *mem,

unsigned char c,

size_t n);

rom char * **memchrpgm**(const rom char *mem,

const unsigned char c,

sizerom_t n);

Аргументы: mem

Указатель на область памяти.

c

Байт значение найти.

n

Максимальное число байтов для поиска.

Примечания: Эта функция поиски до **n** байт региона **mem** найти первый Возникновение **c**.

Эта функция отличается от указанной функции ANSI в том, что **c** определяется как беззнаковое параметра char, а не параметром десятичного.

http://www.microchip.com/

Вернуться Значение: Если **c** появляется в первых **n** байт мем, эта функция возвращает указатель на характер в **mem**. В противном случае, она возвращает пустой указатель.

Имена файлов:

memchr.asm

mchrpgm.asm

//***//

memcmp

memcmppgm

memcmppgm2ram

memcmpram2pgm

Функция: Сравнение содержимого двух массивов байтов.

Включает в себя: string.h

Прототип:

signed char **memcmp**(

    const void * buf1,

    const void * buf2,

    size_t memsize);

signed char **memcmppgm**(

    const rom void * buf1,

    const rom void * buf2,

    sizerom_t memsize);

signed char **memcmppgm2ram**(

    const void * buf1,

    const rom void * buf2,

    sizeram_t memsize);

signed char **memcmpram2pgm**(

    const rom void * buf1,

    const void * buf2,

http://www.microchip.com/

sizeram_t memsize);

Аргументы:

buf1 Указатель на первом массиве.

buf2 Указатель на второй массив.

Memsize Количество элементов в сравнении в массивах.

Примечания: Эта функция сравнивает первый **Memsize** количество байтов в **buf1** к первый Memsize количество байтов в **buf2** и возвращает значение указывающий буферы меньше, равно или больше, чем друг к другу.

Возвращаемое значение: значение, которое:

<0, если **buf1** меньше **buf2**

== 0, если **buf1** такой же, как **buf2**

> 0, если **buf1** больше **buf2**

Имена файлов:

memcmp.asm

memcmpp2p.asm

memcmpp2r.asm

memcmpr2p.asm

//**********//

memcpy

memcpypgm

memcpypgm2ram

memcpyram2pgm

Функция: Скопируйте содержимое исходного буфера в буфер назначения.

Включает в себя: string.h

Прототип:

void * **memcpy**(

void * dest,

http://www.microchip.com/

const void * src,

size_t memsize);

rom void * **memcpypgm**(

rom void * dest,

const rom void * src,

sizerom_t memsize);

void * **memcpypgm2ram**(

void * dest,

const rom void * src,

sizeram_t memsize);

rom void * **memcpyram2pgm**(

rom void * dest,

const void * src,

sizeram_t memsize);

Аргументы:

dest Указатель на массив назначения.

SRC Указатель на исходном массиве.

Memsize Количество байтов **SRC** массива скопировать в **dest** .

Примечания: Эта функция копирует первую **Memsize** количество байтов в **SRC** в Массив **dest** . Если **SRC** и **Dest** перекрытия, поведение не определено.

Вернуться Значение: Эта функция возвращает значение **dest**.

Имена файлов:

memcpy.asm

memcpyp2p.asm

memcpyp2r.asm

memcpyr2p.asm

//***//

memmove

http://www.microchip.com/

memmovepgm

memmovepgm2ram

memmoveram2pgm

Функция: Скопируйте содержимое исходного буфера в буфер назначения, даже если области перекрываются.

Включает в себя: string.h

Прототип:

```
void * memmove( void * dest,
        const void * src,
        size_t memsize );
rom void * memmovepgm(
        rom void * dest,
        const rom void * src,
        sizerom_t memsize );
void * memmovepgm2ram(
        void * dest,
        const rom void * src,
        sizeram_t memsize );
rom void * memmoveram2pgm(
        rom void * dest,
        const void * src,
        sizeram_t memsize );
```

Аргументы:

dest

Указатель на массив назначения.

SRC

Указатель на исходном массиве.

Memsize

Количество байтов **SRC** массива скопировать в **dest**.

http://www.microchip.com/

Примечания: Эта функция копирует первую **Memsize** количество байтов в ***SRC*** в Массив приемник **dest**. Эта функция выполняет правильно, даже если **SRC** и приемник перекрытие.

Вернуться Значение: Эта функция возвращает значение **dest**.

Имена файлов: memmove.asm

memmovp2p.asm

memmovp2r.asm

memmovr2p.asm

//**//

MemSet

memsetpgm

Функция: Скопируйте указанный символ в массив назначения.

Включает в себя: string.h

Прототип:

```
void * memset( void * dest,
        unsigned char value,
        size_t memsize );
rom void * memsetpgm(
        rom void * dest,
        unsigned char value,
        sizerom_t memsize );
```

Аргументы:

приемник **dest**

Указатель на массив назначения.

значение **value**

Значение символов для копирования.

Memsize

Количество байтов **Dest**, в которую копируется значение.

http://www.microchip.com/

Примечания: Эта функция копирует значения символа в первый байт **Memsize** из массива **dest**. Эта функция отличается от указанного ANSI

Функция в этом значение определяется как символ без знака, а не в качестве Int параметр.

Вернуться Значение: Эта функция возвращает значение **dest**.

Имя файла: memset.asm

memsetpgm.asm

//**//

Strcat

strcatpgm

strcatpgm2ram

strcatram2pgm

Функция: Добавить копию исходной строки в конце строки назначения.

Включает в себя: string.h

Прототип:

char * **strcat**(char * dest,

const char * src);

rom char * **strcatpgm**(

rom char * dest,

const rom char * src);

char * **strcatpgm2ram**(

char * dest,

const rom char * src);

rom char * **strcatram2pgm**(

rom char * dest,

const char * src);

Аргументы:

приемник **dest**

Указатель на массив назначения.

http://www.microchip.com/

SRC

Указатель на исходном массиве.

Примечания: Эта функция копирует строку в **SRC** в конце строки в **dest**. **SRC** строка начинается на нулевой в **dest**. Пустой символ добавляется в конец полученной строки в **Dest**. Если **SRC** и приемник перекрытия, поведение не определено.

Вернуться Значение: Эта функция возвращает значение **dest**.

Имена файлов:

strcat.asm

scatp2p.asm

scatp2r.asm

scatp2p.asm

//**//

strchr

strchrpgm

Функция: Найдите первое вхождение указанного символа в строке.

Включает в себя: string.h

Прототип:

char * **strchr**(const char * str,

unsigned char c);

rom char * **strchrpgm**(

const rom char * str,

unsigned char c);

Аргументы:

str

Указатель на строку для поиска.

c

Характер найти.

http://www.microchip.com/

Примечания: Эта функция ищет строку строка, чтобы найти первое вхождение характер с. Эта функция отличается от указанной функции ANSI в том, что с определяется как беззнаковое параметра char, а не параметром десятичного.

Вернуться Значение: Если **с** появляется в **str** эта функция возвращает указатель на этот символ в **str**. В противном случае, она возвращает пустой указатель.

Имена файлов:

strchr.asm

schrpgm.asm

//***//

strcmp

strcmppgm

strcmppgm2ram

strcmpram2pgm

Функция: Сравнение двух строк.

Включает в себя: string.h

Прототип:

signed char **strcmp**(

    const char * str1,

    const char * str2);

signed char **strcmppgm**(

    const rom char * str1,

    const rom char * str2);

signed char **strcmppgm2ram**(

    const char * str1,

    const rom char * str2);

signed char **strcmpram2pgm**(

    const rom char * str1,

    const char * str2);

Аргументы:

http://www.microchip.com/

str1

Указатель на первую строку.

str2

Указатель на второй строке.

Примечания: Эта функция сравнивает строку в **str1** в строку в **str2** и возвращает значение, указывающее, если **str1** меньше, равно или больше, чем **str2**.

Возвращаемое значение: значение, которое:

<0, если **str1** меньше, чем **str2**

== 0, если **str1** же, как и **str2**

> 0, если **str1** больше, чем **str2**

Имя файла:

strcmp.asm

scmpp2p.asm

scmpp2r.asm

scmpr2p.asm

//**//

strcpy

strcpypgm

strcpypgm2ram

strcpyram2pgm

Функция: Копирует исходную строку в строку назначения.

Включает в себя: string.h

Прототип:

```
char * strcpy( char * dest,
        const char * src );
rom char * strcpypgm(
        rom char * dest,
        const rom char * src );char *
```

http://www.microchip.com/

char * **strcpypgm2ram**(

char * dest,

const rom char *src);

rom char * **strcpyram2pgm**(

rom char * dest,

const char * src);

Аргументы:

dest

Указатель на строку назначения.

SRC

Указатель на исходной строки.

Примечания: Эта функция копирует строку в **SRC** в **dest**. Символы в **SRC** являются скопированы до, и в том числе, пустой символ завершения в **SRC**. Если **SRC** и **dest** перекрываются, поведение не определено.

Вернуться Значение: Эта функция возвращает значение **dest**.

Имя файла:

strcpy.asm

scpyp2p.asm

scpyp2r.asm

scpyr2p.asm

//**//

strcspn

strcspnpgm

strcspnpgmram

strcspnrampgm

Функция: Подсчитать количество последовательных символов в начале Строки, не содержатся в наборе символов.

Включает в себя: string.h

Прототип:

http://www.microchip.com/

size_t **strcspn**(const char * str1,

const char * str2);

sizerom_t **strcspnpgm**(

const rom char * str1,

const rom char * str2);

sizerom_t **strcspnpgmram**(

const rom char * str1,

const char * str2);

sizeram_t **strcspnrampgm**(

const char * str1,

const rom char * str2);

Аргументы:

str1

Указатель на строку для поиска.

str2

Указатель на строку, что рассматривается как набор символов.

Примечания: Эта функция будет определять количество последовательных символов из начало **str1**, которые не содержатся в **str2**. Например:

str1 результат **str2**

"Hello" "AEIOU" 1

"Антилопы" "AEIOU" 0

"Антилопы" "XYZ" 8

Вернуться Значение: Эта функция возвращает количество последовательных символов из начало **str1**, которые не содержатся в **str2**, как показано на Приведенные выше примеры.

Имена файлов:

strcspn.asm

scspnpp.asm

scspnpr.asm

scspnrp.asm

http://www.microchip.com/

//***//

StrLen

strlenpgm

Функция: Вернуть длину строки.

Включает в себя: string.h

Прототип:

size_t **strlen**(const char * str);

sizerom_t **strlenpgm**(const rom char * str);

Аргументы:

str

Указатель на строку.

Примечания: Эта функция определяет длину строки, не включая Пустой символ завершения.

Возвращаемые значения: Эта функция возвращает длину строки.

Имя файла:

strlen.asm

slenpgm.asm

//***//

strlwr

strlwrpgm

Функция: Преобразование все символы верхнего регистра в строке в нижний регистр.

Включает в себя: string.h

Прототип:

char * **strlwr**(char * str);

rom char * **strlwrpgm**(rom char * str);

Аргументы:

str

Указатель на строку.

http://www.microchip.com/

Примечания: Эта функция преобразует все символы верхнего регистра в **str** к нижнему регистру символов. Все персонажи, которые не в верхнем регистре (А до Я) не пострадавших.

Вернуться Значение: Эта функция возвращает значение **str**.

Имя файла:

strlwr.asm

slwrpgm.asm

//**//

strncat

strncatpgm

strncatpgm2ram

strncatram2pgm

Функция: Добавляет указанное количество символов из исходной строки в назначения строка.

Включает в себя: string.h

Прототип:

```
 char * strncat( char * dest,
                 const char * src,
                 size_t n );
rom char * strncatpgm(
                 rom char * dest,
                 const rom char * src,
                 sizerom_t n );
char * strncatpgm2ram(
                 char * dest,
                 const rom char * src,
                 sizeram_t n );
rom char * strncatram2pgm(
                 rom char * dest,
                 const char * src,
```

http://www.microchip.com/

sizeram_t n);

Аргументы:

dest

Указатель на массив назначения.

SRC

Указатель на исходном массиве.

n

Количество символов для добавления.

Примечания: Эта функция добавляет ровно **n** символов из строки в **SRC**, чтобы конец строки в **dest**. Если нулевой символ копируется до **n** персонажи были скопированы, нулевые символы будут добавлены в **dest** пока точно N символов не были приложены. Если SRC и **Dest** перекрытия, поведение не определено. Если нулевой символ не встречается, то нулевой символ не добавляется.

Вернуться Значение: Эта функция возвращает значение **dest**.

Имена файлов:

strncat.asm

sncatp2p.asm

sncatp2r.asm

sncatr2p.asm

//**********//

strncmp

strncmppgm

strncmppgm2ram

strncmpram2pgm

Функция: Сравнение двух строк, до определенного числа символов.

Включает в себя: string.h

Прототип:

signed char **strncmp**(const char * str1,

const char * str2,

size_t n);

signed char **strncmppgm**(

const rom char * str1,

const rom char * str2,

sizerom_t n);

signed char **strncmppgm2ram**(

const char * str1,

const rom char * str2,

sizeram_t n);

signed char **strncmpram2pgm**(

const rom char * str1,

const char * str2,

sizeram_t n);

Аргументы:

str1

Указатель на первую строку.

str2

Указатель на второй строке.

n

Максимальное количество символов для сравнения.

Примечания: Эта функция сравнивает строку в **str1** в строку в **str2** и возвращает значение, указывающее, если **str1** меньше, равно или больше, чем **str2**. Если **n** символов сравниваются и никаких различий не обнаружено, это функция возвращает значение, указывающее, что строки эквивалентны.

Возвращаемое значение: значение, основанное на первом символе, который отличается от **str1** и **str2**. Она возвращает:

<0, если **str1** меньше, чем **str2**

== 0, если **str1** же, как и **str2**

> 0, если **str1** больше, чем **str2**

Имя файла:

http://www.microchip.com/

strncmp.asm

sncmpp2p.asm

sncmpp2r.asm

sncmpr2p.asm

//***//

strncpy

strncpypgm

strncpypgm2ram

strncpyram2pgm

Функция: Копирование персонажей из исходной строки в строку назначения, до заданное число символов.

Включает в себя: string.h

Прототип:

char * **strncpy**(char * dest,

const char * src,

size_t n);

rom char * **strncpypgm**(

rom char * dest,

const rom char * src,

sizerom_t n);

char ***strncpypgm2ram**(

char * dest,

const rom char * src,

sizeram_t n);

rom char * **strncpyram2pgm**(

rom char * dest,

const char * src,

sizeram_t n);

http://www.microchip.com/

Аргументы:

dest

Указатель на строку назначения.

SRC

Указатель на исходной строки.

n

Максимальное количество символов для копирования.

Примечания: Эта функция копирует строку в **SRC** в адр. Символы в **SRC** являются не копируется в **Dest** до пустого символа завершения или н символов были скопированы. Если **n** символы не были скопированы и нулевой символ был найден, то **Dest** не будет нулем.

Если копирование происходит между объектами, которые перекрываются, поведение не определено.

Вернуться Значение: Эта функция возвращает значение **Dest**.

Имя файла: strncpy.asm

sncpyp2p.asm

sncpyp2r.asm

sncpyr2p.asm

//**//

strpbrk

strpbrkpgm

strpbrkpgmram

strpbrkrampgm

Функция: Ищет в строке первого вхождения персонаже указано набор символов.

Включает в себя: string.h

Прототип:

```
char * strpbrk( const char * str1,
        const char * str2 );
rom char * strpbrkpgm(
        const rom char * str1,
```

http://www.microchip.com/

const rom char * str2);

rom char * **strpbrkpgmram**(

const rom char * str1,

const char * str2);

char * **strpbrkrampgm**(

const char * str1,

const rom char * str2);

Аргументы:

str1

Указатель на строку для поиска.

str2

Указатель на строку, что рассматривается как набор символов.

Примечания: Эта функция будет искать **str1** для первого вхождения символа содержится в **str2**. Возвращаемые значения: Если персонаж в **str2** найден, указатель на этот символ в **str1** является вернулся. Если ни один символ из **str2** не найден в **str1**, пустой указатель вернулся.

Имена файлов:

strpbrk.asm

spbrkpp.asm

spbrkpr.asm

spbrkrp.asm

//***//

strrchr

Функция: местонахождение последнего вхождение указанного символа в строке.

Включает в себя: string.h

Прототип:

char * **strrchr**(const char * str,

const char c);

Аргументы:

http://www.microchip.com/

str

Указатель на строку для поиска.

c

Характер найти.

Примечания: Эта функция поиск строки **str**, включая завершающий нуль характер, чтобы найти последнее вхождение символа **c**. Эта функция отличается от указанной функции ANSI в том, что с определяется как беззнаковое параметра char, а не параметром десятичного.

Вернуться Значение: Если с появляется в **str** эта функция возвращает указатель на этот символ в **str**. В противном случае, она возвращает пустой указатель.

Имена файлов: strrchr.asm

//**//

strspn

strspnpgm

strspnpgmram

strspnrampgm

Функция: Подсчитать количество последовательных символов в начале строки, которая содержится в наборе символов.

Включает в себя: string.h

Прототип:

```
size_t strspn( const char * str1,
        const char * str2 );
sizerom_t strspnpgm(
        const rom char * str1,
        const rom char * str2 );
sizerom_t strspnpgmram(
        const rom char * str1,
        const char * str2 );
sizeram_t strspnrampgm(
        const char * str1,
```

http://www.microchip.com/

const rom char * str2);

Аргументы:

str1

Указатель на строку для поиска.

str2

Указатель на строку, что рассматривается как набор символов.

Примечания: Эта функция будет определять количество последовательных символов из начало **str1**, которые содержатся в **str2**. Например:

str1 результат **str2**

"Банан" "б" 2

"Банан" "АБН" 6

"Банан" "" 0

Вернуться Значение: Эта функция возвращает количество последовательных символов из начало **str1**, которые содержатся в **str2**, как показано на Приведенные выше примеры.

Имена файлов:

strspn.asm

sspnpp.asm

sspnpr.asm

sspnrp.asm

//***//

strstr

strstrpgm

strstrpgmram

strstrrampgm

Функция: Найдите первое вхождение строки внутри другой строки.

Включает в себя: string.h

Прототип:

char * **strstr**(const char * str,

http://www.microchip.com/

const char * substr);

rom char * **strstrpgm**(

const rom char * str,

const rom char * substr);

rom char * **strstrpgmram**(

const rom char * str,

const char * substr);

char * **strstrrampgm**(

const char * str,

const rom char * substr);

Аргументы:

str

Указатель на строку для поиска.

substr

Указатель на строку шаблона для которого нужно искать.

Примечания: Эта функция будет найти первое вхождение строки подстрока (за исключением нулевой символ) в строку строка.

Возвращаемые значения: Если строка находится, указатель на эту строку в **str** будут возвращены.

В противном случае возвращается пустой указатель.

Имена файлов:

strstr.asm

sstrpp.asm

sstrpr.asm

sstrrp.asm

//**//

Strtok

strtokpgm

http://www.microchip.com/

strtokpgmram

strtokrampgm

Функция: Перерыв строку на подстроки или жетонов, вставив нулевые символы в место указанных разделителей.

Включает в себя: string.h

Прототип:

```
char * strtok( char * str,
        const char * delim );
rom char * strtokpgm(
        rom char * str,
        const rom char * delim );
char * strtokpgmram(
        char * str,
        const rom char * delim );
rom char * strtokrampgm(
        rom char * str,
        const char * delim );
```

Аргументы:

str

Указатель на строку для поиска.

delim

Указатель на набор символов, указывающих на конец знак.

Примечания: Эта функция может использоваться, чтобы разделить строку на подстроки по замена указанные символы с нулевыми символами. В первый раз это функция вызывается на конкретной строки, что строка должна быть передана в **str**. После первого раза, эта функция может продолжить разбор строку от последнего разделителя запустив его с нулевым значением, переданным в **str**. Когда **Strtok** вызывается с ненулевым параметром для **str**, он начинает поиск **str** с самого начала. Это пропускает все ведущие символы, появляются в строке **delim**, затем пропускает все символы не входящие в разделитель, затем устанавливает следующий символ NULL.

Когда **Strtok** вызывается с нулевым параметром для **str**, он ищет Строка, которая совсем недавно исследовали, начиная с характером после того, который был установлен в нуль во время

предыдущего вызова. Это пропускает все персонажи не появляются в **delim**, затем устанавливает следующий символ до нуля.

Если Strtok находит конец строки, прежде чем он находит разделитель, он делает не изменять строку.

Набор символов, который передается в **delim** не обязательно должно быть то же самое для каждый вызов **Strtok**.

Возвращаемые значения: Если в качестве разделителя был найден, то эта функция возвращает указатель на ул чтобы Первый символ, что был проведен обыск, что, казалось, не в наборе символы **delim**. Этот символ представляет первый символ маркер, который был создан в результате вызова.

Если нет разделитель не было найдено до пустой символ завершения, нулевой указатель возвращается из функции.

Имена файлов:

strtok.asm

stokpgm.asm

stokpr.asm

stokrp.asm

//***//

strupr

struprpgm

Функция: Преобразование всех строчных букв в строке в верхний регистр.

Включает в себя: string.h

Прототип:

char * **strupr**(char * str);

rom char * **struprpgm**(rom char * str);

Аргументы:

str

Указатель на строку.

Примечания: Эта функция преобразует все строчные символы в **str** в верхний регистр символов. Все персонажи, которые не строчными (a to z) не пострадавших.

Вернуться Значение: Эта функция возвращает значение **str**.

http://www.microchip.com/

Имя файла:

strupr.asm

suprpgm.asm

//***//

4,5 Функция задержки

Функции задержки выполнения кода для определенного числа циклов команд процессора.

Для задержек на основе временных, рабочая частота процессора должны быть приняты во внимание.

Следующие функции предоставляются:

Таблица 4-4: Функция задержки

Функция Описание

Delay1TCY Задержка на один цикл команды.

Delay10TCYx задержки в упаковке 10 командных циклов.

Delay100TCYx задержки в упаковке 100 циклов команд.

Delay1KTCYx задержки в упаковке 1000 циклов команд.

Delay10KTCYx задержки в упаковке 10000 циклов команд.

4.5.1 Описания функций

//***//

Delay1TCY

Функция: Задержка 1 цикл команды (TCY).

Включает в себя: delays.h

Прототип: void **Delay1TCY**(void);

http://www.microchip.com/

Примечания: Эта функция на самом деле является #define для обучения NOP. Когда встречаются в исходном коде, компилятор просто вставляет NOP.

Имя файла: #define в delays.h

//**//

Delay10TCYx

Функция: Задержка в упаковке 10 командных циклов (TCY).

Включает в себя: delays.h

Прототип: void **Delay10TCYx**(unsigned char unit);

Аргументы: блок

Значение блока может быть любой 8-битное значение.Значение в диапазоне [1255] задержит (единица * 10) циклов. Значение 0 вызывает задержку 2560 циклов.

Примечания: Эта функция создает задержку в упаковке 10 командных циклов.

Имя файла:

d10tcyx.asm

//**//

Delay100TCYx

Функция: Задержка в упаковке 100 циклов команд (TCY).

Включает в себя: delays.h

Прототип: void **Delay100TCYx**(unsigned char unit);

Аргументы:

блок

Значение блока может быть любой 8-битное значение.Значение в диапазоне [1255] задержит (единица * 100) циклов. Значение 0 вызывает задержку 25600 циклов.

Примечания: Эта функция создает задержку в упаковке 100 циклов команд. Это Функция использует глобально размещенную переменную, DelayCounter1. Если это функция используется в обоих прервать и магистральных код, переменную DelayCounter1 должен быть сохранен и восстановлен в прерывания обработчик. Обратитесь к экономии = п от #pragma прерывания или директивы #pragma interruptlow получить дополнительные сведения. Обратите внимание, что другие функции задержки также использовать глобально выделенный DelayCounter1 переменная.

Имя файла:

http://www.microchip.com/

d100tcyx.asm

//***//

Delay1KTCYx

Функция: Задержка в упаковке 1000 циклов команд (TCY).

Включает в себя: delays.h

Прототип: void **Delay1KTCYx**(unsigned char unit);

Аргументы:

блок

Значение блока может быть любой 8-битное значение.Значение в диапазоне [1255] задержит (единица * 1000) циклов. Значение 0 вызывает задержку 256000 циклов.

Примечания: Эта функция создает задержку несколько 1000 циклов команд. Эта функция использует глобально распределенные переменные, DelayCounter1 и DelayCounter2. Если эта функция используется в обоих прерывания и магистральных код, эти переменные, DelayCounter1 и DelayCounter2, должны быть сохранены и восстановлены в прерывания обработчик. Обратитесь к экономии = п от #pragma прерывания и директивы #pragma interruptlow получить дополнительные сведения. Обратите внимание, что другие функции задержки также использовать глобально выделенный DelayCounter1 переменная.

Имя файла:

d1ktcyx.asm

//***//

Delay10KTCYx

Функция: Задержка в упаковке 10000 циклов команд (TCY).

Включает в себя: delays.h

Прототип: void **Delay10KTCYx**(unsigned char unit);

Аргументы:

блок

Значение блока может быть любой 8-битное значение.Значение в диапазоне [1255] задержит (единица * 10000) циклов. Значение 0 вызывает задержку 2560000 циклов.

Примечания: Эта функция создает задержку несколько 10000 циклов команд.

http://www.microchip.com/

Эта функция использует глобально размещенную переменную, DelayCounter1. Если эта функция используется в обоих прервать и магистральных код, переменная DelayCounter1 должен быть сохранен и восстановлен в прерывания обработчик. Обратитесь к экономии = п от #pragma прерывания или директивы #pragma interruptlow получить дополнительные сведения. Обратите внимание, что другие функции задержки также использовать глобально выделенный DelayCounter1 переменная.

Имя файла:

d10ktcyx.asm

//**//

4.6 Функции сброса

Функции сброса могут быть использованы, чтобы помочь определить источник сброса или пробуждения событий и для перенастройки состояние процессора после сброса. В следующем

Процедуры предоставляются:

Таблица 4-5: функции сброса

Функция Описание

isBOR Определите, если причиной сброс Сброс цепи Браун отъезда.

isLVD Определите, если причиной сброса был низкого напряжения определить состояние.

isMCLR Определите, если причиной сброс контактный MCLR.

ISPOR Обнаружить сброса по включению питания состояние.

isWDTTO Определите, если причиной сброс сторожевого таймера тайм-аут.

isWDTWU Определите, если причиной пробуждения был таймер Watchdog.

isWU Обнаруживает если микроконтроллер просто просыпаются от сна от MCLR контактный или прерывания.

StatusReset Установите ПОР и БОР биты.

Примечание: Если вы используете Brown-из Reset (BOR) или сторожевой таймер (WDT), вам должны определить включить макросы (#define BOR_ENABLED и #define WDT_ENABLED, соответственно) в файле заголовка reset.h и пересобрать исходный код.

4.6.1 Описания функций

http://www.microchip.com/

//***//

isBOR

Функция: Определите, если причиной сброс схема сброса Браун отъезда.

Включает в себя: reset.h

Прототип:

char **isBOR**(void);

Примечания: Эта функция определяет, микроконтроллер был сброшен в связи с Браун-из Сброс цепи. Это состояние обозначается следующим Биты состояния:

POR = 1

БОР = 0

Возвращаемые значения:

1, если сброс в связи с Reset цепи Brown-из

0 в противном случае

Имя файла:

isbor.c

//***//

isLVD

Функция: Определите, если причиной сброса был низкого напряжения определить состояние.

Включает в себя: reset.h

Прототип:

char **isLVD**(void);

Примечания: Эта функция определяет, если напряжение устройства стала ниже, чем Значение, указанное в LVDCON регистре (LVDL3:. LVDL0 бит)

Возвращаемые значения:

1, если Сброс было связано с LVD при нормальной работе

0 в противном случае

Имя файла:

http://www.microchip.com/

islvd.c

//***//

isMCLR

Функция: Определите, если причиной сброс контактный MCLR.

Включает в себя: reset.h

Прототип:

char **isMCLR**(void);

Примечания: Эта функция определяет, микроконтроллер был сброшен с помощью пальца MCLR в то время как в нормальном режиме. Эта ситуация указаны следующими Биты состояния:

POR = 1

Если Браун отъезда включена, БОР = 1

Если WDT включен, К = 1

PD = 1

Возвращаемые значения:

1, если Сброс было связано с MCLR при нормальной работе

0 в противном случае

Имя файла:

ismclr.c

//***//

ISPOR

Функция: Обнаружить сброса по включению питания состояние.

Включает в себя: reset.h

Прототип: char **isPOR**(void);

Примечания: Эта функция определяет, микроконтроллер просто оставил сброса по включению питания.

Это состояние обозначается следующими биты состояния:

POR = 0

http://www.microchip.com/

БОР = 0

К = 1

PD = 1

Это условие также может произойти по MCLR при нормальной эксплуатации и когда инструкция CLRWDT выполняется.

После ISPOR называется, StatusReset следует называть установить ПОР и БОР бит.

Возвращаемые значения:

1, если устройство просто оставил сброса по включению питания

0 в противном случае

Имя файла:

ispor.c

//***//

isWDTTO

Функция: Определите, если причиной сброс сторожевого таймера (WDT) время вне.

Включает в себя: reset.h

Прототип: char **isWDTTO**(void);

Примечания: Эта функция определяет, микроконтроллер был сброшен в связи с WDT во время нормальной работы. Это состояние обозначается следующим Биты состояния:

POR = 1

БОР = 1

К = 0

PD = 1

Возвращаемые значения:

1, если сброс в связи с WDT при нормальной работе

0 в противном случае

Имя файла:

iswdtto.c

//***//

http://www.microchip.com/

isWDTWU

Функция: Определите, если причиной пробуждения был сторожевой таймер (WDT).

Включает в себя: reset.h

Прототип: char **isWDTWU**(void);

Примечания: Эта функция определяет, микроконтроллер был выведен из сна на WDT. Это состояние обозначается следующими биты состояния:

POR = 1

БОР = 1

К = 0

PD = 0

Возвращаемые значения:

1, если устройство было принесено из сна на WDT

0 в противном случае

Имя файла:

iswdtwu.c

//***//

isWU

Функция: Определяет если микроконтроллер просто просыпаются от сна через MCLR контактный или прерывания.

Включает в себя: reset.h

Прототип:

char **isWU**(void);

Примечания: Эта функция определяет, микроконтроллер был выведен из сна на контактный MCLR или прерывания. Это состояние обозначается следующие биты состояния:

POR = 1

БОР = 1

К = 1

PD = 0

Возвращаемые значения:

http://www.microchip.com/

1, если устройство было принесено из сна штифтом MCLR или прервать

0 в противном случае

Имя файла:

iswu.c

//***//

StatusReset

Функция: Установка биты POR и BOR в реестре CPUSTA.

Включает в себя: reset.h

Прототип: void **StatusReset**(void);

Примечания: Эта функция устанавливает биты POR и BOR в реестре CPUSTA. Эти бит должен быть установлен в программном обеспечении после сброса по включению питания произошло.

Имя файла:

statrst.c

//***

4.7 ХАРАКТЕР выходных функций

Выходные характер функции обеспечивают центральное семейство функций для обработки Выход на периферийные устройства, буферов памяти и других потребителей символьных данных.

При обработке вызова fprintf, Printf, Sprintf, vfprintf, vprintf или vsprintf, MPLAB C18 всегда будет обрабатывать переменную часть длины аргумента Список с целыми акциях разрешены (см раздел "Целое Акции" из MPLAB® C18 C Compiler Руководство пользователя для получения дополнительной информации). Это позволяет стандарт Библиотека для взаимодействия с компилятором чисто и с последовательным поведением для форматирование выходе как следовало бы ожидать от этих функций.

4.7.1 Выходные Потоки

Выход основан на использовании потока назначения. Поток может быть периферической, буфер памяти, или любой другой потребитель данных и обозначается указатель на объект типа файла. MPLAB C18 определяет два потока из стандартной библиотеки:

http://www.microchip.com/

_H_USER Выход через пользовательской функции вывода _user_putc.

_H_USART Выход через функции _usart_putc выходного библиотека.

Текущая версия библиотеки поддерживает только эти два выходных потоков. Оба потока всегда считается открытым и не требует использования таких функций, как FOPEN, fclose и т.д.

Глобальные переменные стандартный вывод и STDERR определяются библиотеки и имеет значение по умолчанию Значение _H_USART. Чтобы изменить место быть _H_USER, назначить это значение переменная. Например, чтобы изменить стандартный вывод на использование определенного пользователем выход функция:

стандартный вывод = _H_USER;

Таблица 4-6: ХАРАКТЕР выходных функций

Функция Описание

fprintf отформатированную строку вывода в поток.

fputs Строка вывода в поток.

printf отформатированную строку на стандартный вывод.

putc Характер выход в поток

puts ставит выходной строки на стандартный вывод.

Sprintf отформатированную строку вывода в буфер памяти данных.

vfprintf отформатированную строку вывода в поток с аргументами для обработки Строка формата подается через stdarg объекта.

vprintf Форматированный вывод строки на стандартный вывод с аргументами для обработки Строка формата подается через stdarg объекта.

vsprintf отформатированную строку вывода в буфер памяти данных с аргументами для обработки строки формата поставляемый через stdarg объекта.

_usart_putc Одновыходовой характер в USART (USART1 для устройств, которые иметь более одного USART).

_user_putc Одновыходовой характер в прикладной определяется способом.

//***

4.7.2 Описания функций

http://www.microchip.com/

//**//

fprintf

Функция: отформатированную строку вывода в поток.

Включает в себя: stdio.h

Прототип:

int **fprintf** (FILE *f, const rom char *fmt, ...);

Примечания: Выход форматы функциональные fprintf, попутный символы для указанный поток через putc функции. Строка формата обрабатывается один символ за один раз и персонажи выводятся, как они появляются в строка формата, для спецификаторов формата исключением.Спецификатор формата является, ука лаборантом в строке формата с помощью знака процента,%; следующее, что, хорошо сформирован спецификатор формата имеет следующий components.1 за исключением операция преобразования, все спецификаторы формата являются обязательными:

1. символов Flag (порядок не имеет значения), где флаг символ является одним из #, -, +, 0 или пространство.

2 ширина поля, который является десятичное целое значение константы Звездочка, *.

3 точность поле, которое представляет собой период (.), Необязательно с последующим десятичное целое число или звездочкой, *.

4 спецификация размер, который является одним из спецификаторов ч, H, HH, J, Z, Z, T, T или л.

5 операция преобразования, которая является одним из C, B, B, D, I, N, O, p, P, S, S, U, X, X или%.

примечание: Не все компоненты действительны для всех конверсионных операций. Подробности

при условии, в описании операторов преобразования.

Flag Персонажи

Альтернативная форма результата будет представлена. Для о конверсии мер, альтернативная форма является как если точность были увеличены такие что первая цифра результата вынужден быть нулевой. Для x конверсия, ненулевой результат будет иметь префикс 0x добавили к нему. Для X Преобразование, ненулевой результат будет иметь префикс 0X добавили к нему. Для Преобразование б, в результате ненулевое будет иметь 0b префикс добавляется к нему.Для преобразования B, ненулевой результат будет иметь префикс 0B добавил к нему. Для других преобразований, флаг игнорируется.

- Результат будет левому краю. Если этот флаг не указан, то результат будет правильно оправдано.

http://www.microchip.com/

+ Для знаковых преобразований, результат всегда будет начинаться с + или - подписать. По умолчанию, символ знака добавляется только к результату, если результат отрицательный. Для других преобразований, флаг игнорируется. пространство для знаковых преобразований, если результат не является отрицательным или имеет себе персонажи, пространство будет иметь префикс к результату. Если пространство и + флаги как указано, пространство флаг будут игнорироваться. Для друга Преобразования, флаг игнорируется.

0 Для целочисленных преобразований (д, I, O, U, B, B, X, X), ведущими нулями начинаются в результате (после любых знаковых и / или базовых показателей) таким образом, что результат заполняет ширину поля. Нет места обивка не Per- формируется. Если - флаг также указано, то флаг 0 игнорируется. Если точность определена, то флаг 0 игнорируется. Для друга Преобразования, флаг игнорируется.

поле Ширина

Ширина поля определяет минимальное количество символов для контуров Значение преобразуется. Если преобразованное значение короче, чем ширина поля, а затем значение дополняется, чтобы иметь количество символов равным ширина поля. По умолчанию, ведущие пробелы используются для заполнения; флаг символы используются для изменения площадку характер и оправданность значение.

Если ширина поля символ звездочки, *, Int аргумент читать указать ширину поля. Если значение отрицательное, это как если бы - флаг были указано, с последующим положительным ширины поля.

поле Precision

Точность поле определяет минимальное количество цифр, которые будут присутствует в переоборудованном значение для объявления, я, о, у, б, B, X или преобразования X, или максимальное количество символов в преобразованного значения для х преобразования.

Если ширина поля символ звездочки, *, Int аргумент читать указать ширину поля. Если значение отрицательное, это как если бы в точности были неопределенные.

Для D, I, O, U, B, B, X или преобразования X операторы, высокоточными по умолчанию Сион 1. Для всех других операторов преобразования поведение, когда точности не определен как описано ниже.

Размер Технические характеристики

Спецификация размер символа относится к целочисленного преобразования спектрометре ifiers, д, I, O, U, B, B, X или X, и преобразования указатель спецификаторы, р и Р. Если присутствует любой другой оператор преобразования, он игнорируется.

hh Для целых спецификаторов преобразования, аргумент, который будет преобразован является подписали символ или неподписанные символ argument.**2** Для н конверсии Сион спецификатор, спецификатор обозначает указатель на подписанного гольца Аргумент.

h Для целых спецификаторов преобразования, аргумент, который будет преобразован является Короткая Int или беззнаковое короткое внутр. Для преобразования спектров н ifier, спецификатор обозначает указатель на короткий аргумент десятичного. Как равнина Int имеет такой же размер,

как короткий междунар для MPLAB C18, этого Опция не имеет реальный эффект и присутствует для совместимости только. Для указателей спецификаторов преобразования, аргумент, чтобы быть переделанный является 16-битный указатель.

H Для целых спецификаторов преобразования, аргумент, который будет преобразован является короткий длинный Int или без знака короткий длинный внутр. Для н CON- Версия спецификатор, спецификатор обозначает указатель на короткое длительного Int Аргумент. Для указателей спецификаторов преобразования, аргумент, чтобы быть переделанный является 24-битный pointer.**3** Например, при выводе далеко ром символ *, спецификатор размера H следует использовать (% HS).

J Для целых спецификаторов преобразования, аргумент, который будет преобразован в intmax_t или uintmax_t аргумент. Для преобразования спектров н ifier, спецификатор обозначает указатель на intmax_t аргумента. Для MPLAB C18, это равносильно тому, размера л спецификатора.

l Для целых спецификаторов преобразования, аргумент, который будет преобразован является долго Int или неподписанных долгое Int. Для преобразования н спецификатора, Спецификатор обозначает указатель на долгое десятичного аргумента. Для указатель спецификаторы преобразования, спецификатор размера игнорируется.

t Для целых спецификаторов преобразования, аргумент, который будет преобразован в ptrdiff_t аргумент. Для спецификатором преобразования н, образце литель обозначает указатель на целое число со знаком тип, соответствующий ptrdiff_t аргумент. Для MPLAB C18, это равносильно тому, ч спецификатор размера.

T Для целых спецификаторов преобразования, аргумент, который будет преобразован в ptrdiffrom_t аргумент. Для спецификатором преобразования н, Спецификатор обозначает указатель на целое число со знаком типа, соответствующий чтобы ptrdiffrom_t аргумент. Для MPLAB C18, это равносильно тому, Размер H specifier.**4**

r Для целых спецификаторов преобразования, аргумент, который будет преобразован в size_t аргумент. Для спецификатором преобразования н, спецификатор обозначает указатель на целое число со знаком типа, соответствующий size_t Аргумент. Для MPLAB C18, это равносильно тому, ч спецификатор размера.

Z Для целых спецификаторов преобразования, аргумент, который будет преобразован в sizerom_t аргумент. Для спецификатором преобразования н, образце литель обозначает указатель на целое число со знаком тип, соответствующий sizerom_t аргумент. Для MPLAB C18, это равносильно тому, H размер specifier.**5**

примечание 2 что целые акции будут по-прежнему применяться, когда аргумент прошло. Этот спецификатор вызывает аргумент должны быть поданы обратно в 8 бит в размере предварительного к стоимости используется.

примечание 3 The размер H спецификатор определенный добавочный MPLAB C18 для ANSI C.

примечание 4 The размер T спецификатор определенный добавочный MPLAB C18 для ANSI C.

примечание 5 The размер Z спецификатор определенный добавочный MPLAB C18 для ANSI C.

http://www.microchip.com/

Операторы преобразования

c Int аргумент преобразуется в символьное значение без знака и символ, представленный на эту величину написано.

d, Ithe Int аргумент имеет формат подписанного десятичной с высокоточными мер с указанием минимального количества цифр, которые будут написаны. Если преобразованное значение имеет меньше цифр, то добавляется нулями. Если преобразованное значение равно нулю, а точность равна нулю, нет символов будет записать.

o неподписанных Int аргумент преобразуется в беззнаковое восьмеричное с точность, указывающее минимальное количество цифр для записи.

Если преобразованное значение имеет меньше цифр, то добавляется с ведущими нули. Если преобразованное значение равно нулю, а точность равна нулю, нет символы будут записаны.

u неподписанных Int аргумент имеет формат без знака десятичной с точностью, указывающий минимальное количество цифр, чтобы быть написано. Если преобразованное значение имеет меньше цифр, то предваряться нули. Если преобразованное значение равно нулю, а точность равна нулю, нет символы будут записаны.

b неподписанных Int аргумент имеет формат без знака двоичного с точность, указывающее минимальное количество цифр для записи. Если преобразованное значение имеет меньше цифр, то добавляется нулями. Если преобразованное значение равно нулю, а точность не равна нулю, персонажи будет written.**6**

В неподписанный междунар аргумент имеет формат без знака двоичного с точность, указывающее минимальное количество цифр для записи.

Если преобразованное значение имеет меньше цифр, то добавляется нулями. Если преобразованное значение равно нулю, а точность не равна нулю, персонажи будет written.**7**

x неподписанных Int аргумент имеет формат без знака шестнадцатеричное мал с точностью указывает минимальное количество цифр, чтобы быть написано. Символы ABCDEF которые используются для представления, если десятичные числа 10 через 15 Если преобразованное значение имеет меньше цифр, оно добавляется нулями. Если полученное значение нулю, а точность равна нулю, никакие символы не будут записаны.

X неподписанных Int аргумент имеет формат без знака шестнадцатеричное мал с точностью указывает минимальное количество цифр, чтобы быть написано. Символы ABCDEF используются для представления десятичные числа 10 через 15 Если преобразованное значение имеет меньше цифр, оно добавляется нулями. Если полученное значение нулю, а точность равна нулю, никакие символы не будут записаны.

сек Символы из массива памяти данных Чар аргумента Обозначая десять, пока либо конечном '\ 0' не видно ('\ 0' характер не написано) или число записанных символов равна с заданной точностью. Если точность определена, чтобы быть больше чем размер массива или не задан, массив должен содержать прекращения '\ 0'.

S Символы из массива памяти программ Чар аргумента не написано, пока не встретится завершающий '\ 0' рассматривается ('\ 0' характер не написано) или число записанных символов

равна с заданной точностью. Если точность определена, чтобы быть больше чем размер массива или не задан, массив должен содержать прекращения '\ 0' character.**8** При выводе дальний ром символ *, убедитесь, что использовать H размер спецификатор (т.е.% УГ).

примечание 6 The оператор преобразования б это определенный добавочный MPLAB C18 для ANSI C.

примечание 7 The оператор преобразования B является определенный добавочный MPLAB C18 для ANSI C.

примечание 8 The оператор S преобразования конкретных расширение MPLAB C18 для ANSI C.

р Указатель аннулировать (данные или программную память) аргумент свя- преобразовывались к эквивалентного размера целого числа без знака типа, и это значение является обработаны, как если бы было указано оператор х конверсии. Если Размер H спецификатор присутствует, указатель находится 24-разрядный указатель, иначе это 16-битный указатель.

Р указатель на аннулированию (данные или программную память) аргумент свя- преобразовывались к эквивалентного размера целого числа без знака типа, и это значение является обработаны, как если бы было указано оператор преобразования х. Если Размер H спецификатор присутствует, указатель находится 24-разрядный указатель, иначе это 16-битный pointer**.9**

н Количество символов до сих пор хранятся на положении ние, на который ссылается аргумент, который является указателем на целое введите в память данных.Размер целого типа пределяется спецификатор размера присутствует для преобразования, или простой 16-разрядного целого числа Если размер спецификатор не присутствует.

% Написано буквальное символ%.Спецификация преобразования должны быть %% Только, никакие флаги или другие спецификаторы не могут присутствовать.

Если спецификатор преобразования является недействительным (например, флаг символ присутствует на %% спецификатор преобразования), поведение не определено.

Возвращаемые значения: fprintf возвращает EOF, если происходит ошибка, в противном случае возвращает количество выводимых символов.

Имя файла: fprintf.c

Пример кода:

```
#include <stdio.h>

void main (void)

{

 far rom char * S = ”Hello, World!”;
```

http://www.microchip.com/

```
 int n = 0x1234;

 fprintf (_H_USART, "test output to USART\n");

 fprintf (_H_USER, "test output to application”

       “defined function\n" );

 fprintf (stdout, "hex output: %#x", n);

 fprintf (stderr, “%HS\n”, S);

}
```

примечание 9. О оператор преобразования P является определенный добавочный MPLAB C18 для ANSI C.

//**//

fputs

Функция: String выход в поток.

Включает в себя: stdio.h

Прототип:

int **fputs** (const rom char *s, FILE *f);

Примечания: ЕриЕз выводит оканчивающихся нулем строку в указанный выходной поток, один символ за один раз через putc. Символ новой строки добавляется в выход. Нуль-терминатор не выводится.

Возвращаемые значения fputs возвращает EOF, если происходит ошибка, в противном случае возвращает неотрицательное Значение.

Имя файла: fputs.c

//**//

printf

Функция: отформатированную строку на стандартный вывод.

Включает в себя: stdio.h

Прототип:

int **printf** (const rom char *fmt, ...);

Примечания: Выход форматы функциональные E, попутный символы для стандартный вывод через putc функции. Строка формата обрабатывается как в описании функции fprintf.

http://www.microchip.com/

Возвращаемые значения: E возвращается EOF, если происходит ошибка, в противном случае возвращает количество Выход символов.

Имя файла: printf.c

Пример кода:

```
#include <stdio.h>

void main (void)

{

  / * Выведет через стандартный вывод (_H_USART по умолчанию) * /

 printf ("Hello, World!\n");

}
```

//**//

putc

Функция: выход Персонаж в поток.

Включает в себя: stdio.h

Прототип: int **putc** (char c, FILE *f);

Примечания: putc выводит один символ на указанный выходной поток.

Вернуться Значение: putc возвращает EOF, если происходит ошибка, в противном случае возвращает символ который был выход.

Имя файла: putc.c

//**//

puts

Функция: String выход на стандартный вывод.

Включает в себя: stdio.h

Прототип: int **puts** (const rom char *s);

Примечания: ставит выводит оканчивающихся нулем строку на стандартный вывод один символ Время через putc. Символ новой строки добавляется к выходу. Нуль-терминатор не выход.

http://www.microchip.com/

Возвращаемые значения: ставит возвращается EOF, если происходит ошибка, в противном случае возвращает неотрицательное Значение.

Имя файла: puts.c

Пример кода:

```
#include <stdio.h>

void main (void)

{

  puts ("test message");

}
```

//**//

Sprintf

Функция: отформатированную строку вывода в буфер памяти данных.

Включает в себя: stdio.h

Прототип: int sprintf (char *buf, const rom char *fmt, ...);

Примечания: Функция Sprintf форматирует вывод, хранения символов в назначения буфер памяти данных, buf. Строка формата, FMT, является обрабатывают, как описано для функции fprintf.

Возвращаемые значения: Sprintf возвращает EOF, если происходит ошибка, в противном случае количество Выход символов возвращается.

Имя файла: sprintf.c

Пример кода:

```
#include <stdio.h>

void main (void)

{

  int i = 0xA12;

  char buf[20];

  sprintf (buf, "%#010x", i);

   / * buf будет содержать строку "0x00000a12"

}
```

http://www.microchip.com/

//**//

vfprintf

Функция: отформатированную строку вывода в поток с аргументами для обработки Строка формата подается через stdarg объекта.

Включает в себя: stdio.h

Прототип:

int **vfprintf** (FILE *f, const rom char *fmt,

va_list ap);

Примечания: Выход форматы функциональные vfprintf, попутный символы для указано выходной поток, F, через putc функции. Строка формата, FMT, обрабатывается, как описано для функции fprintf исключением того, что аргументы, потребляемые при обработке строки формата получить через stdarg переменной объекта длина аргумента.

Вернуться Значение:

vfprintf возвращает EOF, если происходит ошибка, в противном случае число Выход символов возвращается.

Имя файла: vfprintf.c

//**//

vprintf

Функция: после форматирования выходной строки на стандартный вывод с аргументами для обработки Строка формата подается через stdarg объекта.

Включает в себя: stdio.h

Прототип:

int **vprintf** (const rom char *fmt, va_list ap);

Примечания: Выход форматы функциональные vprintf, попутный символы для стандартный вывод через putc функции. Строка формата, FMT, обрабатывается как в описании функции fprintf исключением того, что логичные рассуждения разряжался при обработке строки формата извлекаются через stdarg Переменная объект длина аргумент.

Вернуться Значение:

vprintf возвращает EOF, если происходит ошибка, в противном случае число Выход символов возвращается.

Имя файла: vprintf.c

http://www.microchip.com/

//**//

vsprintf

Функция: отформатированную строку вывода в буфер памяти данных с аргументами для обработки строки формата поставляемый через stdarg объекта.

Включает в себя: stdio.h

Прототип:

int **vsprintf** (char *buf, const rom char *fmt,

va_list ap);

Примечания: Выход форматы функциональные vsprintf, хранящие символы в назначения буфер памяти данных, buf. Строка формата, FMT, является обрабатывают, как описано для функции fprintf исключением того, что Аргументы, потребляемые при обработке строки формата извлекаются через stdarg объекта переменной длины-аргумента.

Вернуться Значение:

vsprintf возвращает EOF, если происходит ошибка, в противном случае число Выход символов возвращается.

Имя файла: vsprintf.c

//**//

_usart_putc

Функция: Единственный выход характер в USART (USART1 для устройств, которые имеют более чем одной УСАПП).

Включает в себя: stdio.h

Прототип:

int _usart_putc (char c);

Примечания: _usart_putc является выходная функция библиотеки вызывается putc когда это _H_USART Является поток назначения. Характер быть выход назначен на передающий регистр (TXREG), когда УСАПП готов к Выход (TRMT установлен).

Если USART не включен, когда _usart_putc называется (TXSTA немного TXEN ясно), USART будет включен (TXEN и SPEN будет установлен) и установить на максимальную мощность скорости передачи (SPBRG будет присвоено значение нуля). Такая конфигурация позволяет библиотечные функции вывода символов , которые будут использоваться при поддержке MPLAB IDE для USART отладочных сообщений без явного периферической конфигурации.

http://www.microchip.com/

Вернуться Значение:

_usart_putc возвращает значение символа, который был выход.

Имя файла:

_usart_putc.c

//***//

_user_putc

Функция: Единственный выход характер в прикладной определяется способом.

Включает в себя: stdio.h

Прототип: int _user_putc (char c);

Примечания: _user_putc является применение функции определены. Она будет называться по Выходные характер функции для каждого символа, чтобы быть выход, когда Поток назначения _H_USER.

Вернуться Значение:

_user_putc возвращает значение символа, который был выход.

//***//

Глава 5. математические библиотеки

5.1 ВВЕДЕНИЕ

В этой главе документы математические функции библиотеки. Она включает в себя две секции:

• 32-бит с плавающей точкой Math Library

• Функции C Стандартная библиотека Math

5.2 32-бит с плавающей точкой математической библиотеки

Основным пунктом операций с плавающей запятой-сложение, вычитание, умножение, деление и преобразование между целыми и-совместимы со стандартом IEEE 754 для одинарной точности плавает с двумя исключениями. Исключения будут обсуждаться в рамках Subnormals (Раздел 5.2.1.2 "Subnormals") и Округление (раздел 5.2.2 "Округление"). расширенный режим и традиционный режим использовать те же представления с плавающей точкой и Результаты операций с плавающей точкой и то же. Стандарт IEEE для двоичной арифметики с плавающей точкой опубликован в 1985 году стал как известно, официально ANSI / IEEE Std 754-1985 [IEEE85]. Стандарт имеет три важные требования:

http://www.microchip.com/

• согласуется представление чисел с плавающей точкой на всех машинах, принимающих стандарт;

• правильно округлые операций с плавающей запятой, используя различные режимов округления;

• согласуется лечение исключительных ситуациях, таких как деление на ноль.

5.2.1 плавающей запятой Представление

C18 представление числа с плавающей запятой следует одинарной точности IEEE 754 стандарт.Число с плавающей точкой состоит из четырех частей:

1 знак

2 мантисса

3 базы

4 показатель

Эти компоненты имеют вид

x = ± d0.d1.d2.d3 • • • d23 Ч 2E

где ± это знак, d0.d1.d2.d3 • • • d23 является мантисса, и E показатель, которому

Основание 2 поднимается. Каждый ди это цифра (0 или 1). Показатель E представляет собой целое число в диапазоне Emin к Emax, где Emin = -126 и Emax = 127.

Номера Одноместный формата использовать 32-битное слово, организованная в 1-битной знак, 8-битный предвзятое показатель e = E + 127, и 23-разрядную фракция, которая является дробная часть мантисса.

Наиболее значимым битом мантиссы (D0) не сохраняется. Это возможно потому, что его значение может быть выведено из значения показателя: если оказывают влияние значение показателя равно 0, то d0 = 0, в противном случае d0 = 1 Используя эту конвенцию позволяет 24 бита точности будут храниться в 23 физических бит.

В реализации C18, с D0 = 0 числа не используются (см раздел 5.2.1.2 "Subnormals").

5.2.1.1 нормалей

Все линии в таблице 5.2, за исключением первого и последнего см нормированных чисел. показатель немного строка e7e6e5. . . e0 использует предвзятое представление;битовая строка сохраняется как двоичного представления E + 127, где E беспристрастным показатель. число 127, которая добавляется к экспоненты E, называется смещения показатель. Например, Число 1 = (1,000... 0) 2 20 хранится в виде Здесь показатель битовая строка является двоичным представлением для 0 + 127 и доля немного строка бинарное представление для 0 (дробная часть 1,0).

http://www.microchip.com/

Диапазон битовых строк показатель поля для нормированных чисел 00000001 в 11111110 (десятичные цифры от 1 до 254), что составляет фактические показатели с Эмин = -126 чтобы Emax = 127.

Наименьшее положительное, ненулевой нормализованное число, которое может храниться представлена и это обозначается

Nmin = (1,000 ... 0) 2 x 2-126 = 2-126 ~ 1,2 x 1038

Постоянная Nmin доступна С программистов, использующих манифеста константу FLT_MIN определены в <float.h>.

Наибольшее нормализованное число (то же самое, самое большое конечное число) представлен и это обозначается Nmax = (1,111 1...) 2 x 2127 = (2 - 2-23) x 2127 ~ 2128 ~ 3,4 x 1038

Постоянная Nmax доступна С программистов, использующих манифеста константу FLT_MAX определены в <float.h>.

Войдите 8-битный предвзятым показатель E 23-разрядное беззнаковое фракции F ± 7e6e5ee4e3e2e1e0 d0d1d2d3 • • • d23

0 01111111 00000000000000000000000

Таблица 5-1: IEEE-754 едином формате

Предвзятость Показатель числа, представленного

(00000000) 2 = (00) 16 = (0) 10 ± (0.d1d2d3... D23) 2 X 2-126

(00000001) 2 = (01) 16 = (1) 10 ± (1.d1d2d3... D23) 2 X 2-126

(00000010) 2 = (02) 16 = (2) 10 ± (1.d1d2d3... D23) 2 X 2-125

(00000011) 2 = (03) 16 = (3) 10 ± (1.d1d2d3... D23) 2 X 2-124

↓↓

(01111110) 2 = (7E) 16 = (126) 10 ± (1.d1d2d3.. .d23) 2 X 2-1

(01111111) 2 = (7F) 16 = (127) 10 ± (1.d1d2d3.. .d23) 2 X 20

(10000000) 2 = (80) 16 = (128) 10 ± (1.d1d2d3.. .d23) 2 X 21

↓↓

(11111100) 2 = (FC) 16 = (252) 10 ± (1.d1d2d3.. .d23) 2 X 2125

(11111101) 2 = (FD) 16 = (253) 10 ± (1.d1d2d3.. .d23) 2 X 2126

(11111110) 2 = (FE) 16 = (254) 10 ± (1.d1d2d3.. .d23) 2 X 2127

http://www.microchip.com/

(11111111) 2 = (УС) 16 = (255) 10 ± ∞, если d1. . . д 23 = 0

NaN, если d1. . .d23 ≠ 0

0 00000001 00000000000000000000000

0 11111110 11111111111111111111111

5.2.1.2 SUBNORMALS

Наименьшее нормированное число, которое может быть представлено в 2-126.IEEE 754 Стандарт использует сочетание нулевой предвзятым экспоненты е и ненулевой фракцию F Чтобы представляют меньшее число называемые субнормальные числа. Структура субнормальная Номера показано в строке 1 таблицы 5.2. В реализации флоат C18, субнормальная номера всегда преобразуются в подписанном нуля. IEEE 754 использует два различных нулевые представления: + 0 и -0.+0 Представлена все нулевые биты.-0 Представлена все нули для знакового бита кроме.

Если результат операции флоат меньше наименьшего нормированного числа, результат установлен в ноль со знаком, прежде чем он будет возвращен. Так, в реализации C18, не поплавка операция может создать субнормальная, субнормальная появится только тогда, когда она построена явно, как буквальный, или генерируется в некотором роде, кроме стандартной поплавковой опеции. Если субнормальная значение используется в операции флоат, он автоматически преобразуется в подписан нулю, прежде чем он используется в работе.

5.2.1.3 Значения NaN

В дополнение к поддержке подписанных бесконечности, подписали нули и подписан ненулевую конечную цифры, формат IEEE с плавающей точкой определяет кодировку для паттернов ошибок. Эти шаблоны не являются числами, но записи о том, что недопустимая операция была пытался. Любая такая картина является показателем ошибки, не число с плавающей точкой и так называют не число, или NaN. Недопустимые операции определяются IEEE Стандарт включает:

• Величина вычитание бесконечностей, таких как (+ ∞) + (-∞)

• Умножение нуля на бесконечности, например, (+ ∞) x (+ ∞)

• Деление нуля или бесконечности на ноль или бесконечность, соответственно, такие как (+ ∞) / (-∞) или (+ ∞) / (+ ∞)

Значения NaN имеют предвзятое экспоненту 255, который также показатель используется для кодирования бесстей. Интерпретация когда показатель оказывают влияние 255: если доля равна нулю, Кодирование представляет собой бесконечность; Если доля не равен нулю, то кодирование представляет NaN (не число). Не обращая внимания на знаковый бит, который в стандарте не интерпретировать для NaNs, есть поэтому 223 - 1 возможных Значения NaN. Реализация C18 возвращает NaN шаблон 7FFF FFFF16 в ответ на неправильной операции. То есть, бит знака равен 0, экспоненты 255, и фракцию, биты все 1с.

http://www.microchip.com/

5.2.2 Округление

Стандарт IEEE-754 требует, чтобы операции правильно округлены. стандарт определяет правильно округленное значение x, которая обозначается круглого (x), следующим образом: если x является число с плавающей точкой, то круглые (x) = x. В противном случае, правильно-округлые значение зависит от того, какой из четырех режимов округления, по сути. Реализация поплавок C18 использует округляется до ближайшего режиме с небольшой модификацией к стандарту IEEE 754. Порог округления составляет около 0,502 вместо точно 0,5. Это дает небольшое уклон в сторону округления в сторону нуля. Это изменение приводит к существенной экономии пространство кода и время выполнения практически без последствий для реального мира расчеты.

5.3 C STANDARD БИБЛИОТЕКА математических функций

Все математические функции из библиотеки стандартного C вернется NaN, если один или несколько из его аргументы:

• является NaN.

• находится вне диапазона значений, для которых функция имеет определенную реальную ценность, для Пример квадратный корень из отрицательного числа.

Таблица 5-2 перечислены математические функции.

5-2: Math Library ФУНКЦИИ

Функция Описание

acos Вычислить обратное косинус (арккосинус).

ASIN Вычислить обратное синус (Арксинус).

atan Вычислить арктангенс (арктангенс).

atan2 Вычислить арктангенс (арктангенс) из соотношении.

Ceil Вычислить потолок (наименьшее целое).

cos потому Вычислить косинус.

cosh Вычислить гиперболический косинус.

exp вычисляется экспоненциальная напр.

fabs Вычислить абсолютное значение.

floor Вычислить этаж (наибольшее целое).

fmod Вычислить остаток.

frexp Сплит в фракции и экспоненты.

http://www.microchip.com/

ieeetomchp Преобразование значение с плавающей запятой 32-битный формат IEEE-754 в Microchip 32-битный формат с плавающей точкой.

ldexp нагрузки показатель - вычислить x * 2n.

log вычислить натуральный логарифм.

log10 Вычислить общее (основание 10) логарифм.

mchptoieee Преобразование значения формата Microchip 32-бит с плавающей точкой в IEEE-754 32-битный формат с плавающей точкой.

modf Вычислить модуль.

Pow вычисляется экспоненциальная xy.

sin Вычислить синус.

sinh Вычислить гиперболический синус.

SQRT вычисления квадратного корня.

tan Вычислить касательной.

tanh Вычислить гиперболический тангенс.

//**********

5.3.1 Описания функций

acos

Функция: Вычислить арккосинуса (арккосинуса)

Включает в себя: math.h

Прототип: float **acos**(float x);

Примечания: Эта функция вычисляет арккосинус (арккосинуса) аргумента x, которые должны быть между -1 и +1. Аргументы за пределами допустимого Диапазон возникать ошибки доменных и результат NaN.

Возвращаемые значения: возвращаемым значением является Арккосинус в радианах, и находится между 0 и π.

Имя файла: acos.c

//**********

ASIN

Функция: Вычислить арксинус (Арксинус).

http://www.microchip.com/

Включает в себя: math.h

Прототип:

float **asin**(float x);

Примечания: Эта функция вычисляет арксинус (Арксинус) аргумента x, которое должно быть между -1 и +1. Аргументы за пределами допустимого Диапазон возникать ошибки доменных и результат NaN.

Вернуться Значение: возвращаемым значением является Арксинус в радианах, и между -π/2 и π/2.

Имя файла: asin.c

//***//

atan

Функция: Вычислить арктангенс (арктангенс).

Включает в себя: math.h

Прототип:

float **atan**(float x);

Примечания: Эта функция вычисляет арктангенс (арктангенс) из Аргумент x. Если x является NaN, ошибка домена происходит и значение Возвращается NaN.

Вернуться Значение: возвращается значение в радианах, и между -π / 2 и π / 2.

Имя файла: atan.c

//***//

atan2

Функция: Вычислить арктангенс (арктангенс) из соотношении.

Включает в себя: math.h

Прототип:

float **atan2**(float x, float y);

Примечания: Эта функция вычисляет арктангенс (арктангенс) из г / x. Если x или y является NaN, домен происходит и возвращенное значение NaN. Если x является NaN, или, если x = y = 0, или, если x = y = ∞, ошибка возникает домена и значение Возвращается NaN.

Возвращаемые значения: возвращается значение в радианах, и между - π и π.

http://www.microchip.com/

Имя файла: atan2.c

//***//

Ceil

Функция: Вычислить потолок (хотя бы целое).

Включает в себя: math.h

Прототип:

float **ceil** (float x);

Замечания: Нет.

Возвращаемое значение: наименьшее целое число, большее или равное x.

Имя файла: ceil.c

//***//

cos

Функция: Вычислить косинус.

Включает в себя: math.h

Прототип:

float **cos** (float x);

Примечания: Вычислить косинус x (в радианах). Происходит ошибка домена от Аргумент, что бесконечно или NaN. Оба случая вернуться NaN.

Вернуться Значение: Косинус аргумента x.

Имя файла: cos.c

//***//

cosh

Функция: Вычислить гиперболический косинус.

Включает в себя: math.h

Прототип:

float **cosh** (float x);

http://www.microchip.com/

Замечания: Нет.

Вернуться Значение: гиперболический косинус аргумента x.

Имя файла: cosh.c

//**//

exp

Функция: вычисляется экспоненциальная напр.

Включает в себя: math.h

Прототип:

float **exp** (float x);

Примечания: Возникает ошибка диапазон, если величина x является слишком большим. Диапазон этого Функция ограничена значениями для показателей между примерно -103.2789 И 88,722283. Минимальное значение результата есть 2-149 и максимум 2127.

Вернуться Значение: Значение экспоненциальной exp.

Имя файла: exp.c

//**//

fabs

Функция: Вычислить абсолютное значение.

Включает в себя: math.h

Прототип:

float **fabs**(float x);

Примечания: Для аргументов с плавающей запятой, которые нули и бесконечности, возвращение значение аргумент со знаком бит очищается.

Вернуться Значение: Абсолютное значение x.

Имя файла: fabs.c

//**//

floor

Функция: Вычислить floor (наибольшее целое).

http://www.microchip.com/

Включает в себя: math.h

Прототип:

float **floor**(float x);

Замечания: Нет.

Возвращаемое значение:

наибольшее целое, меньшее или равное x.

Имя файла: floor.c

//**//

fmod

Функция: Вычислить остаток.

Включает в себя: math.h

Прототип:

float **fmod**(float x, float y);

Замечания: Нет.

Вернуться Значение: Остальная для x по модулю y.

Имя файла: fmod.c

//**//

frexp

Функция: Разделить на фракции и экспоненты.

Включает в себя: math.h

Прототип:

float **frexp**(float x, int *pexp);

Примечания: Разделяет аргумент x на две части, которые соответствуют этой формуле:

x = frexp (x, * Pexp) x 2 * Pexp

Целочисленное значение, которое хранится в местоположении Pexp, выбирается так, чтобы дробная часть результата находится между S и 1.

Возвращаемые значения:

http://www.microchip.com/

Дробное результат, который удовлетворяет перечисленным выше условиям.

Имя файла: frexp.c

//**//

ieeetomchp

Функция: Преобразование IEEE-754 формата 32-битное значение с плавающей запятой в Microchip 32-битный формат с плавающей точкой.

Включает в себя: math.h

Прототип:

unsigned long **ieeetomchp**(float v);

Примечания: Эта функция регулирует знаковый бит представлении с плавающей точкой в быть расположены в соответствии с требованиями формата Microchip:

EB f0 f1 f2

IEEE-754 32-разрядная Seee ээээ Exxx xxxx xxxx xxxx xxxx xxxx

Microchip 32-бит ээээ ээээ Sxxx xxxx xxxx xxxx xxxx xxxx

с = бит знака е = показатель х = мантисса

Вернуться Значение: преобразуется 32-битное значение.

Имя файла: ieeetomchp.c

//**//

ldexp

Функция: Нагрузка показатель - вычислить х * 2n.

Включает в себя: math.h

Прототип:

float **ldexp**(float x, int n);

Замечания: Нет.

Возвращаемое значение: значение х * 2n.

Имя файла: ldexp.c

http://www.microchip.com/

//**//

log

Функция: Вычислить натуральный логарифм.

Включает в себя: math.h

Прототип:

float **log**(float x);

Примечания: Возникает ошибка домена, если аргумент не в интервале [0, + ∞].

Вернуться Значение: Натуральный логарифм от x.

Имя файла: log.c

//**//

log10

Функция: Вычислить общую (база 10) логарифм.

Включает в себя: math.h

Прототип:

float **log10**(float x);

Примечания: Возникает ошибка домена, если аргумент не в интервале [0, + ∞].

Возвращаемые значения: log10x.

Имя файла: log10.c

//**//

mchptoieee

Функция: Преобразование 32-битное значение формата Microchip с плавающей запятой в IEEE-754 32-битный формат с плавающей точкой.

Включает в себя: math.h

Прототип:

float **ieeetomchp**(unsigned long v);

Примечания: Эта функция регулирует знаковый бит представлении с плавающей точкой в быть расположены в соответствии с требованиями стандарта IEEE формате:

http://www.microchip.com/

EB f0 f1 f2

IEEE-754 32-разрядная Seee ээээ Exxx xxxx xxxx xxxx xxxx xxxx

Microchip 32-бит ээээ ээээ Sxxx xxxx xxxx xxxx xxxx xxxx

c = бит знака e = показатель x = мантисса

Вернуться Значение:

преобразуется значение с плавающей точкой.

Имя файла: mchptoieee.c

//**//

modf

Функция: Вычислить модуль.

Включает в себя: math.h

Прототип: float **modf**(float x, float *ipart);

Примечания: Эта функция отделяет аргумент x в целую и дробную части. Дробная часть возвращается, и целая часть сохраняется в местоположении параметрического ряда. Если аргумент является NaN, результаты для обоих дробная и целая часть будет NaN также.

Вернуться Значение:

дробная часть x.

Имя файла: modf.c

//**//

Pow

Функция: вычисляется экспоненциальная xy.

Включает в себя: math.h

Прототип:

float **pow**(float x, float y);

Примечания: Ошибки домена произойти, если x конечна и отрицательный, и y конечна, а не целое; Кроме того, если x равен нулю, а Y меньше или равна нулю.диапазон происходит ошибка, если xy слишком большой или слишком маленький, чтобы быть представлены. В таком случай, правильно подписан бесконечность или возвращается ноль и ошибка диапазон сигнал.

Возвращаемые значения: xy.

http://www.microchip.com/

Имя файла: pow.c

//**//

sin

Функция: Вычислить синус.

Включает в себя: math.h

Прототип:

float **sin**(float x);

Примечания: Вычислить синус x (в радианах). Происходит ошибка домена от Аргумент, что бесконечно или NaN. Оба случая вернуться NaN.

Вернуться Значение: Синус x.

Имя файла: sin.c

//**//

sinh

Функция: Вычислить гиперболический синус.

Включает в себя: math.h

Прототип:

float **sinh**(float x);

Замечания: Нет.

Вернуться Значение: гиперболический синус аргумента x.

Имя файла: sinh.c

//**//

SQRT

Функция: вычисления квадратного корня.

Включает в себя: math.h

Прототип:

http://www.microchip.com/

float **sqrt**(float x);

Примечания: Возникает ошибка домена, если аргумент x является строго отрицательным.прин-приятель квадратный корень существует и вычислим для каждого неотрицательного плавание номер точки x.

Вернуться Значение: Квадратный корень из x.

Имя файла: sqrt.c

//***//

tan

Функция: Вычислить касательной.

Включает в себя: math.h

Прототип:

float **tan**(float x);

Примечания: Вычислить тангенс x (в радианах).Ошибка возникает, если доменом Аргумент бесконечен или NaN. Оба случая вернуться NaN.

Вернуться Значение: Тангенс x.

Имя файла: tan.c

//***//

tanh

Функция: Вычислить гиперболический тангенс.

Включает в себя: math.h

Прототип:

float **tanh**(float x);

Примечания:

Если аргумент является NaN, возвращается значение NaN.

Вернуться Значение:

гиперболический тангенс x.

Имя файла: tanh.c

http://www.microchip.com/

//**********//

Список

Символы

http://www.microchip.com/