Универсальные программаторы с LPT интерфейсом

Описание
Быстрый старт
Руководство пользователя

ChipProg+
ChipProg-2

ООО "Фирма Фитон"

Phyton

© 2009 Фитон, http://www.phyton.ru

Содержание

Часть I Введение 5

1 Обзор ... 5
2 Термины и определения ... 5
3 Требования к системе ... 7
4 Быстрый старт ... 7
Подключение программатора ... 7
Установка П.О. ChipProgLPT ... 7
5 Получение поддержки ... 9
Оперативные справки ... 9
Техническая поддержка ... 9
Решение проблем ... 10
Информация для контакта ... 10

Часть II Программаторы 11

1 Сравнительные характеристики программаторов ... 11
2 ChipProg+ ... 12
Основные характеристики ChipProg+ ... 13
Характеристики Аппаратуры ChipProg+ ... 13
Характеристики Программного Обеспечения ChipProg+ ... 13
3 ChipProg-2 ... 14
Основные характеристики ChipProg-2 ... 14
Характеристики Аппаратуры ChipProg-2 ... 14
Характеристики Программного Обеспечения ChipProg-2 ... 15
4 Работа с программатором ... 15
Установка микросхемы в колодку программатора ... 16
Программирование в плате пользователя ... 16
Микроконтроллеры PICmicro, особенности программирования ... 17
Микроконтроллеры AVR, особенности программирования ... 17
Микроконтроллеры Atmel 8051, особенности программирования ... 17
Программирование NAND Flash ... 17
Структура памяти NAND Flash ... 18
Плохие блоки (Bad blocks) ... 19
Маркирование плохих блоков ... 19
Обработка плохих блоков ... 19
Skip Bad Blocks ... 19
Reserved Block Area ... 20
Error Checking and Correction ... 20
Программирование NAND Flash на программаторах ChipProg (COPY) ... 20
Access Mode ... 21
Invalid Block Management ... 21
Spare Area Usage ... 21
Guard Solid Area ... 21
Tolerant Verify Feature ... 22
Invalid Block Indication Option ... 22
Access Mode Parameters ... 22
User Area ... 22
Solid Area ... 22
RBA Area ... 22
ECC Frame size ... 23
Acceptable number of errors ... 23
Карта плохих блоков ... 23
Функции процесса Программирования ... 23

© 2009 Фитон, http://www.phyton.ru

Проверить чистоту микросхемы ... 23
Стирание ... 23
Как Запрограммировать микросхему ... 23
Как Загрузить Файл, который должен быть записан в микросхему ... 24
Как Отредактировать Исходную Информацию ... 24
Как Сконфигурировать микросхему ... 24
Как записать информацию ... 24
Как Сравнить ... 24
Как Прочитать микросхему ... 25
Как Сохранить данные, прочитанные из микросхемы ... 25
Как Продублировать микросхему ... 25

Часть III Графический Интерфейс Пользователя 26

1 Меню ... 26

Обзор ... 26
Меню Файл ... 27
Конфигурационные файлы ... 28
Меню Просмотр ... 28
Меню Конфигурация ... 29
Диалог Выбор микросхемы ... 29
Диалог Буферы ... 29
Диалог Конфигурация буфера ... 30
Основной подслой ... 30
Дополнительный подслой ... 30
Диалог Опции ... 30
Диалог Опции Экрана ... 31
Шрифты Закладка ... 31
Цвета Закладка ... 32
Назначение клавиш экрана Закладка ... 32
Линейка управления Закладка ... 33
Прочие Закладка ... 33
Меню Команды ... 34
Калькулятор Диалог ... 34
Меню Сценарии ... 35
Диалог Файлы Сценария ... 36
Меню Окна ... 37
Меню Справка ... 38
О ChipProgLPT ... 38

2 Окна ... 38

Интерфейс Пользователя Обзор ... 38
Окно Программирование ... 39
Диалог Программирование ... 39
Автоматическое Программирование ... 40
Диалог Опции программирования ... 41
Чередование данных ... 41
Окно Редактор Параметров Микросхемы и Алгоритма Программирования ... 42
Команда Edit ... 43
Окно Дамп Буфера ... 44
Диалог Отобразить с адреса ... 45
Диалог Конфигурация буфера ... 45
Диалог Изменить Значение или Начальный Адрес ... 46
Операции с Блоками Памяти ... 47
Диалог Настройки окна Дамп ... 48
Диалог Загрузить Файл ... 49
Диалог Сохранить Файл из буфера ... 50
Форматы Файлов ... 50
Окно Информация о Микросхеме ... 51
Таблица соединений адаптера ... 51
Таблица подключения к адаптеру ... 52
Окно Консоль Сообщений ... 53
Окно Редактора ... 53

© 2009 Фитон, http://www.phyton.ru

Редактор Текста ... 54
Диалог Поиск текста ... 55
Диалог Поиск/Замена текста ... 56
Выражения поиска ... 57
Окно Переменные с автопросмотром ... 58
Настройки панели автопросмотра ... 58
Диалоги Установить закладку/Восстановить закладку ... 59
Диалог Отобразить с новой строки ... 59
Функция Быстрого просмотра ... 59
Действия с блоками ... 59
Окна, используемые в Файлах Сценария ... 60
Окно Переменные ... 60
Диалог Опции Отображения окна Переменные ... 61
Диалог Добавить Переменную в окно ... 62
Окно Информация о ChipProg ... 62

Часть IV Дополнительные Главы 63

1 Параметры Командной Строки ... 63
2 Окно Исходного Текста Файла Сценария ... 63
3 Окно Пользователя ... 64
4 Окно Потока Ввода / Вывода ... 64
5 Файлы Сценария ... 65
Простейший Пример Файла Сценария ... 65
Как Написать Файл Сценария ... 66
Как Запустить Файл Сценария ... 66
Как Отладить Файл Сценария ... 66
Описание языка Файлов Сценария ... 66
Встроенные Функции языка Файлов Сценария ... 67
Встроенные Переменные Файлов Сценария ... 68
Отличия языка Файлов Сценария от языка Си ... 69
Алфавитный Список встроенных Функций и Переменных в Файлах Сценария ... 71
6 Выражения ... 76
Операции с Выражениями ... 77
Числа ... 78
Символьные Имена ... 78
Примеры выражений ... 78

Алфавитный указатель 80

1 Введение

Семейство универсальных программаторов с LPT интерфейсом.

Все программаторы разработаны и производятся фирмой Фитон.

Серийный выпуск программаторов производится по ТУ 4034-001-17477019-05.
Сертификат соответствия системы сертификации ГОСТ Р Госстандарт России № РОСС RU. МЛ04.В01176
Программаторы соответствуют ГОСТ Р МЭК 60950-2002, ГОСТ Р 51318.22(24)-99, ГОСТ Р 51317.3.2(3)-99, предназначены для работы в нормальных климатических условиях.

Программаторы предназначены для программирования параллельных и последовательных микросхем EPROM, EEPROM, FLASH, NVRAM, микроконтроллеров с программируемой памятью данных и кодов, а также микросхем программируемой логики PAL, PLD, CPLD.

Все алгоритмы программирования реализованы в строгом соответствии со спецификациями, рекомендованными производителями микросхем.

ChipProg-+ ChipProg-2

ChipProg+ - универсальный программатор , предназначен для разработчиков. Обеспечивает программирование микроконтроллеров, FLASH, EPROM, EEPROM, NVRAM, PLD.

ChipProg-2 - универсальный программатор интерфейсом, предназначен для разработчиков. Обеспечивает программирование микроконтроллеров, FLASH, EPROM, EEPROM, NVRAM;

1.1 Обзор

ChipProgLPT - это интуитивная и лёгкая в использовании программа. Подробнее смотрите в Программаторы , Меню и Интерфейс пользователя.

Встроенный язык скриптов ChipProgLPT позволит вам автоматизировать много рутинных задач. Подробнее смотрите в Файлы скриптов .

ChipProgLPT работает на платформе IBM PC (см. Требования к системе).

1.2 Термины и определения

Идеология взаимодействия с микросхемами.

Программное обеспечение ChipProg в плане взаимодействия с микросхемами построено по классической схеме программаторов "файл" - "буфер" - "микросхема". В программной поддержке программатора реализованы буфера памяти. Буфер памяти является промежуточным звеном между файлом и микросхемой:

Файл <----> Буфер <----> Микросхема

Все операции с файлом (загрузка/сохранение файла) взаимодействуют только с буфером. Т.е. можно загрузить файл в буфер, а также сохранить содержимое буфера в файл. Аналогичным образом построено взаимодействие

© 2009 Фитон, http://www.phyton.ru

с микросхемой. Все манипуляции с микросхемой (как, например, чтение, запись, сравнение) используют только буфер. Таким образом, можно прочитать микросхему в буфер, записать содержимое буфера в микросхему, сравнить содержимое буфера и микросхемы и т.п. Взаимодействие файла и микросхемы напрямую не допускается.

Термины

ICP (in circuit programming) - программирование в плате пользователя.

ISP (in-system programming) - программирование в плате пользователя

ISP Mode - режим программирование в плате пользователя.

ISP BSL Mode - режим программирование в плате пользователя в режиме BSL.

ISP JTAG Mode - режим программирование в плате пользователя через JTAG порт.

ISP HV Mode - высоковольтный режим программирование в плате пользователя.

Проект - это некий контейнер, в котором сохраняется информация об установленном типе микросхемы, все настройки буферов программатора, настройки опций программирования, настройки по сериализации, контрольной сумме, ведению журнала. Дополнительно в проекте можно сохранить перечень загружаемых файлов и конфигурацию экрана программатора. Проекты различаются по именам. Проектами очень удобно пользоваться, если Вы работаете с несколькими микросхемами. Каждый проект настраивается под конкретную микросхему и в нем сохраняется все настройки программатора под конкретный тип микросхемы.

Адаптер - это специальный переходник под определенный тип корпуса микросхемы или разъем на плате пользователя.

Иногда на плате адаптера устанавливаются активные и пассивные компоненты.

Буфер памяти Буфер памяти - это объект, предназначенный для хранения данных. Буфер памяти программатора является промежуточным звеном между файлом и микросхемой.

Количество буферов Допускается создание бесконечного количества активных буферов.

Ограничением на количество может служить только отсутствие свободной памяти в системе.

Подслои буфера Каждый буфер в своем составе имеет определенное количество подслоев. Каждый подслой ассоциируется с конкретным адресным пространством установленной микросхемы. Пример использования подслоев смотрите ниже.

Пример использования подслоев буфера:

1. Микросхема Intel 87C51FA. Каждый буфер имеет в своем составе два подслоя: **подслой кодовой** памяти и **подслой таблицы кодирования** (encryption table).
2. Микросхемы Microchip PIC16F84. Каждый буфер имеет в своем составе три подслоя: **подслой кодовой** памяти, **подслой памяти данных** EEPROM, **подслой пользовательских идентификаторов**.

Такая гибкая реализация системы буферов позволяет пользователю очень легко манипулировать с несколькими разными массивами данных, поместив их в разные буфера.

© 2009 Фитон, http://www.phyton.ru

1.3 Требования к системе

Для работы программы ChipProgLPT необходимо использовать ПК, совместимый с IBM PC, в конфигурации не хуже, чем:

- Персональный компьютер, работающий с Windows 98/ME/2000/XP/Vista, для программатора ChipProg-G4 - Windows 2000/XP/Vista
- Pentium-III CPU или выше
- 256MB оперативной памяти RAM
- По крайней мере один порт LPT
- Жесткий диск со свободным пространством не менее 200MB

1.4 Быстрый старт

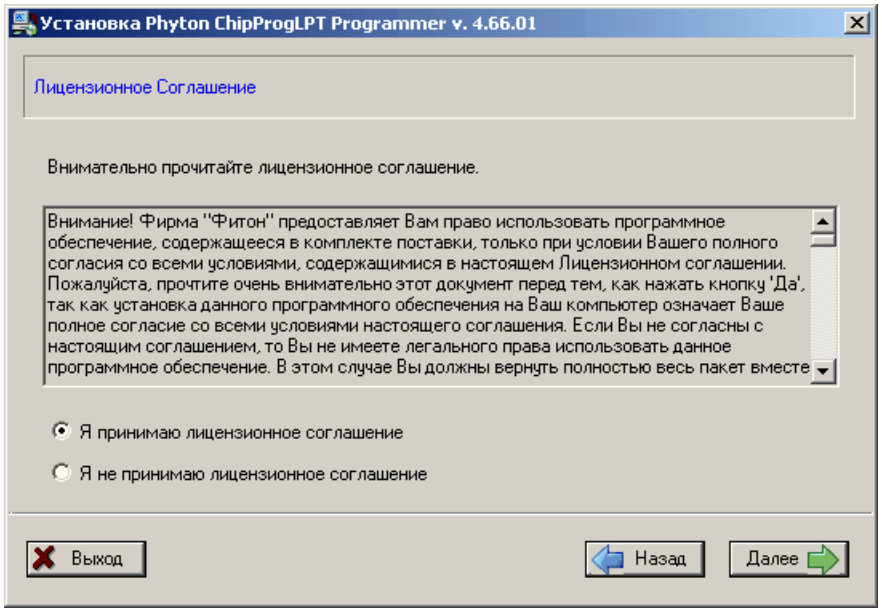
1.4.1 Подключение программатора

Установка программного обеспечения	Необходимо установить ПО с дистрибутивного диска
Подключить питание к программатору	Подключить блок питания в сетевую розетку. Разъем блока питания подключить в гнездо питания на задней панели программатора.
Подключение к компьютеру	С помощью LPT кабеля, входящего в состав комплекта поставки, необходимо соединить программатор со свободным принтерным портом компьютера. LPT разъем находится на задней панели программатора.
Запуск программы	Запустить программу **Phyton ChipProgLPT.**

Программатор готов к работе, можно начать программирование микросхем.

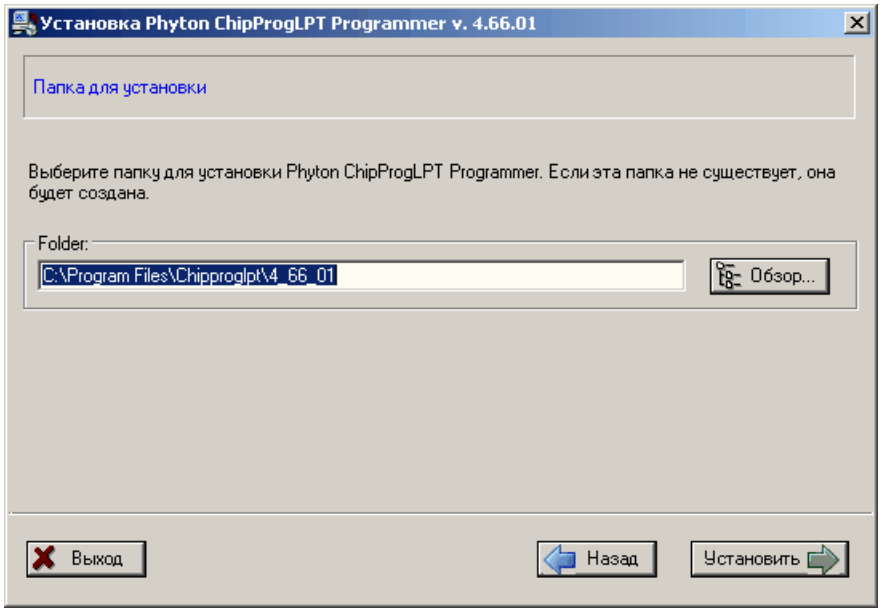
1.4.2 Установка П.О. ChipProgLPT

Вставьте диск ChipProgLPT в дисковод вашего компьютера. Далее следуйте по серии подсказок, которые проведут вас через процесс установки.

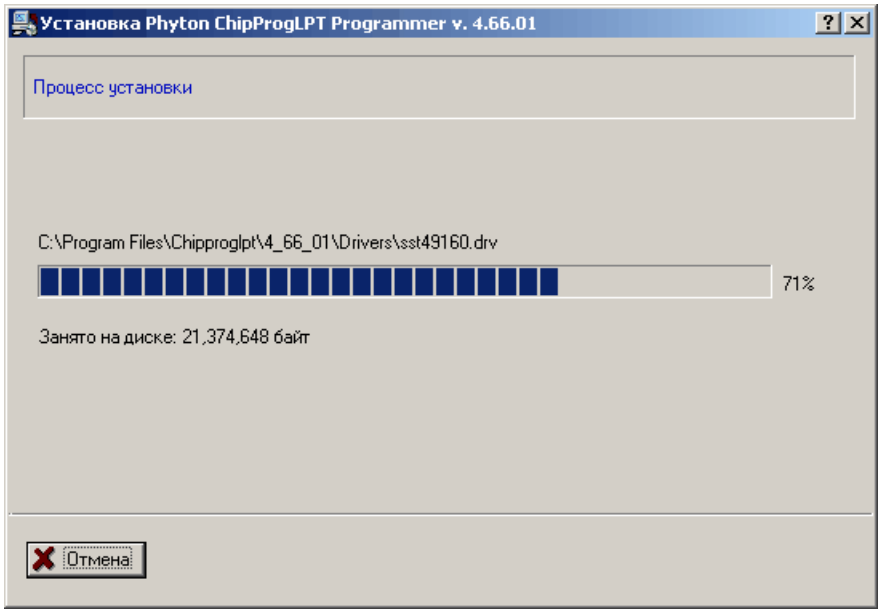


Принимать условия лицензионного соглашения

© 2009 Фитон, http://www.phyton.ru

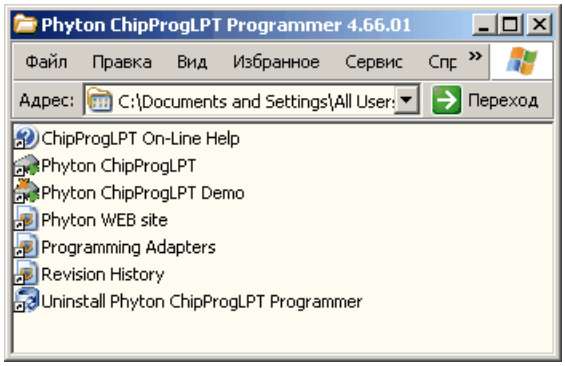


Выбрать папку для инсталляции



Процесс установки ...

В конце процесса инсталляции будет создана папка с программами и документами ChipProgLPT.



Папка Phyton ChipProgLPT

Перечень файлов поставки:

ChipProgLPT On-Line Help справочный файл программатора

© 2009 Фитон, http://www.phyton.ru

Phyton ChipProgLPT	исполняемый файл программатора
Phyton ChipProgLPT Demo	демонстрационная версия программатора
Phyton WEB site	ссылка на сайт фирмы Фитон
Programming Adapters	файл, в котором указаны таблицы соединений всех адаптеров, а также подключение к адаптерам для программирования микросхем в плате пользователя
Revision History	файл истории версий программатора
Phyton ChipProgLPT Programmer	программа деинсталляции программатора

1.5 Получение поддержки

1.5.1 Оперативные справки

Чтобы открыть окно оперативной справки, следует нажать клавишу **F1** клавиатуры или воспользоваться меню **Справка**. Мы постарались обеспечить окно справки для каждого диалога, окна сообщения или меню. Например, чтобы открыть окно справки для меню, откройте меню и нажмите **F1**.

Меню **Справка** содержит дополнительные команды для управления системой справок программы ChipProgLPT .

Также, используйте стандартную функцию поиска справочной системы. В большинстве случаев нужную тему можно найти по ключевому слову. Например, если в текстовом поле закладки **Поиск** (Find) напечатать «Программирование », то система покажет список тем, где говорится про программирование. Чтобы посмотреть конкретную тему, выделите её в списке и нажмите кнопку **Вывести** (Display).

Для последовательного изучения ChipProgLPT , можно читать файл справки, следуя оглавлению в закладке **Содержание** (Contents).

1.5.2 Техническая поддержка

Компания «Фитон» предоставляет бесплатную техническую поддержку. Наши специалисты с удовольствием помогут преодолеть трудности в работе с изделиями компании.

Наши программаторы являются мощными изделиями в своём классе и предоставляют Вам большие возможности. Пожалуйста, сообщайте нам обо всех замеченных ошибках, чтобы мы могли исправить их и предоставить Вам обновлённую версию изделия (бесплатно).

Если Вы только начинаете работать с ChipProgLPT , подробно ознакомьтесь с изделием. У программы ChipProgLPT вполне обычный и интуитивно понятный интерфейс пользователя. Тем не менее, для хорошего понимания функциональности изделия, пожалуйста, прочитайте этот файл справки.

Прежде, чем обращаться за помощью

Прежде, чем обращаться за помощью, пожалуйста:

- Проверьте, можно ли повторно воспроизвести обнаруженную ошибку, т.е., можно ли воспроизвести ситуацию, в которой происходит ошибка.
- Посмотрите справку по решению проблем, и если в ней уже затронута Ваша проблема, то воспользуйтесь предложенной рекомендацией.

При обращении за помощью

Пожалуйста, сообщите нам следующие сведения:

- Ваше имя, название компании, номер телефона (факса) и адрес электронной почты для контактов.
- Название изделия и его серийный номер.
- Дату покупки изделия.
- Номер версии и используемый объём памяти (по данным диалога **Информация**).
- Общие параметры используемого ПК и операционной системы.
- Список обнаруженных ошибок и их описание.

Направляйте запросы или сообщения по адресу: **info@phyton.ru** . Также, используйте другие способы коммуникации (смотрите адреса для контактов).

© 2009 Фитон, http://www.phyton.ru

1.5.3 Решение проблем

1.5.4 Информация для контакта

ООО «Фирма Фитон»

Тел/факс: (495) 730-7584

Email: **info**@phyton.ru, **sales**@phyton.ru, **phyton**@phyton.ru, **support**@phyton.ru

http://www.phyton.ru.

© 2009 Фитон, http://www.phyton.ru

2 Программаторы

Семейство программаторов ChipProgLPT фирмы Фитон:

ChipProg+ ;
ChipProg-2;

2.1 Сравнительные характеристики программаторов

Сравнительные характеристики программаторов выпускаемых ООО «Фирма Фитон»

	ChipProg-G4	ChipProg-48	ChipProg-40	ChipProg-ISP	ChipProg+	ChipProg-2
Тип программатора	Универсальный копировщик	Универсальный	Универсальный	Универсальный последовательный	Универсальный	Универсальный
Типы поддерживаемых микросхем	FLASH, EPROM, EEPROM, NVRAM, MCU, PLD	FLASH, EPROM, EEPROM, NVRAM, MCU, PLD	FLASH, EPROM, EEPROM, NVRAM, MCU	FLASH, EEPROM, MCU	FLASH, EPROM, EEPROM, NVRAM, MCU, PLD	FLASH, EPROM, EEPROM, NVRAM, MCU
Возможность мультипрограм-мирования	Да, 4 колодки на программаторе, К-во не ограничено.	Да, 1 колодка на программаторе, К-во не ограничено.	Да, 1 колодка на программаторе, К-во не ограничено.	Да, 1 ISP разъем, К-во не ограничено.	Нет	Нет
Тип сокетки	4 x 48 ZIF	48 ZIF	40 ZIF	IDC-14	40 ZIF	40 ZIF
Логические драйверы	Универсальные, 1.8V – 5.5V	Универсальные, 1.8V – 5.5V	Универсальные, 1.8V – 5.5V	Универсальные, 1.8V – 5.5V	Универсальные, 2.0V – 5.5V	Универсальные, 2.0V – 5.5V
Аналоговые драйверы	Универсальные	Универсальные	Специализирован-ные	Специализирован-ные	Универсальные	Специализирован-ные
Возможность поддержки новых микросхем	Любые	Любые	Ограничена аналоговыми драйверами	Любые	Любые	Ограничена аналоговыми драйверами
Программирование в схеме	Да	Да	Да	Да	Да	Да
Встроенный контроллер (частота)	32 битный (60 МГц)	32 битный (60 МГц)	32 битный (60 МГц)	32 битный (60 МГц)	8 битный (10 МГц)	8 битный (10 МГц)
Встроенная FPGA (частота)	Да (до 100 МГц)	Да (до 100 МГц)	Да (до 100 МГц)	Да (до 100 МГц)	Да (10 МГц)	Да (10 МГц)
Автоматическое распознавание присутствия микросхемы в колодке	Да	Да	Да	-	Нет	Нет
Тестирование правильности установки микросхемы	Да	Да	Да	-	Да	Да
Тестирование каждой ножки микросхемы на наличие контакта	Да	Да	Да	-	Нет	Нет
Режим сериализации, записи контрольной суммы и сигнатуры	Да	Да	Да	Да	Нет	Нет
Ведение журнала программирования	Да	Да	Да	Да	Нет	Нет
Поддержка проекта	Да	Да	Да	Да	Нет	Нет
Операционная система	Windows 2000/XP/2003/	Windows 98/ME/ 2000/XP/2003/	Windows 98/ME/ 2000/XP/2003/	Windows 98/ME/ 2000/XP/2003/	Windows 98/ME/ 2000/XP/2003/	Windows 98/ME/ 2000/XP/2003/

© 2009 Фитон, http://www.phyton.ru

	VISTA	VISTA	VISTA	VISTA	VISTA	VISTA
Интерфейс к компьютеру	USB 2.0	USB 2.0	USB 2.0	USB 2.0	LPT	LPT
Скорость программирования	Очень высокая	Очень высокая	Очень высокая	Очень высокая	Высокая	Высокая
Время программирование и сравнение (мин: сек)						
M25P20	00:05	00:05	00:05	00:05	00:07	00:07
SST39VF016Q	00:45	00:45	00:45	-	02:50	02:50
MX28F640C3BB	00:56	00:56	00:56	-	02:27	02:27
MX29LV017A	00:23	00:23	00:23	-	02:56	02:56
MX29LV160CT	00:16	00:16	00:16	-	01:17	01:17
SST49LF008A	00:19	00:19	00:19	-	01:43	01:43
PIC18LF8722	00:11	00:11	00:11	00:11	00:19	00:19
AT89S51	00:01	00:01	00:01	00:01	00:01	00:01

2.2 ChipProg+

Универсальный программатор ChipProg+ .

Программатор имеет небольшие габариты и предназначен для программирования последовательных и параллельных микросхемы памяти EPROM, EEPROM, FLASH, микроконтроллеров с программируемой памятью данных и кодов и микросхем программируемой логики PAL, PLD, CPLD. Программатор также поддерживает режим программирования микросхем в устройстве пользователя (In-System Programming).

ChipProg+

На программаторе установлена универсальная 40 выводная DIP сокетка с нулевым усилием (ZIF), позволяющая программировать все микросхемы в DIP корпусах без дополнительных адаптеров. Поддержка микросхем в корпусах, отличных от DIP (SOIC, SSOP, TSOP, PLCC, QFP, BGA и др.) осуществляется с помощью дополнительных адаптеров, производимых фирмой Фитон, а также сторонними производителями. С помощью специальных адаптеров программатор поддерживает программирование микросхем, установленных в устройстве пользователя.

Комплект поставки

Программатор	- 1
Блок питания	- 1
Кабель связи с PC	- 1
CD с программным обеспечением	- 1
Паспорт	- 1

Подробнее характеристики программатора ChipProg+ смотрите в разделе **Основные характеристики ChipProg+**.

© 2009 Фитон, http://www.phyton.ru

2.2.1 Основные характеристики ChipProg+

Основные характеристики программатора ChipProg+ можно представить следующими группами характеристик:

Характеристики Аппаратуры ChipProg+

Характеристики Программного Обеспечения ChipProg+

2.2.1.1 Характеристики Аппаратуры ChipProg+

1. Архитектура программатора построена на базе программируемой матрицы (FPGA). Расширение списка поддерживаемых микросхем производится путем простого обновления версии программного обеспечения.
2. Логические драйверы на базе FPGA, способные подвести логические сигналы низкого, высокого уровня, внешнего генератора, а также Pullup, Pulldown на любой вывод колодки.
3. 8-ми разрядные цифро-аналоговые преобразователи для программирования аналоговых источников напряжения.
4. Возможность программирования фронта подъема и спада аналогового напряжения.
5. Возможность подключения аналоговых напряжений питания и программирования на любой вывод микросхемы.
6. Возможность коммутации любого вывода микросхемы с «землей».
7. Наличие быстродействующих схем защиты от перегрузки по току, увеличивающих надежность программатора и не выводящих из строя неправильно подключенные микросхемы.
8. Самотестирование.

Поддерживаемые корпуса микросхем:

1. Поддержка всех микросхем в корпусах DIP в колодке программатора без дополнительных адаптеров.
2. Поддержка микросхем в корпусах до 40 выводов в универсальных адаптерах.
3. Поддержка микросхем в корпусах SDIP, PLCC, SOIC, SOP, PSOP, TSOP, TSOPII, TSSOP, QFP, TQFP, VQFP, QFN, SON, BGA, CSP с помощью дополнительных адаптеров.
4. Совместимость с адаптерами сторонних производителей.

2.2.1.2 Характеристики Программного Обеспечения ChipProg+

1. Работа под управлением Windows 95/98/ME/2000/XP/Vista.
2. Дружественный, интуитивно понятный, двуязычный интерфейс
3. Поддержка всех процедур работы с микросхемой: чтение, сравнение, контроль чистоты, запись и стирание, установка защиты, программирование конфигурационных битов, работа с памятью данных и т.п.
4. Многобуферный интерфейс с возможностью создания неограниченного числа буферов. Буфера разбиваются на подслои, имеющие структуру адресного пространства микросхем
5. Полноценный двоичный редактор с возможностью ручного редактирования данных, а также функции заполнения, сравнения, копирования, поиска и замены, инвертирования, вычисления контрольной суммы, логических операции OR, AND, XOR.
6. Загрузка и сохранение файлов в двоичном коде, Standard Extended Intel HEX, Motorla S-record, POF, JEDEC, PRG, Holtek OTP, ASCII HEC, ASCII OCTAL, Angstrem SAV форматах.
7. Встроенный язык сценариев, обеспечивающий доступ ко всем ресурсам программатора. Посредством применения этого языка можно значительно облегчить работу с программатором, автоматизируя рутинные операции.

© 2009 Фитон, http://www.phyton.ru

2.3 ChipProg-2

Универсальный программатор ChipProg-2.

Программатор имеет небольшие габариты и предназначен для программирования последовательных и параллельных микросхемы памяти EPROM, EEPROM, FLASH, микроконтроллеров с программируемой памятью данных и кодов и микросхем программируемой логики PAL, PLD, CPLD. Программатор также поддерживает режим программирования микросхем в устройстве пользователя (In-System Programming).

ChipProg-2

На программаторе установлена универсальная 40-ми выводная DIP сокетка с нулевым усилием (ZIF), позволяющая программировать все микросхемы в DIP корпусах без дополнительных адаптеров. Поддержка микросхем в корпусах, отличных от DIP (SOIC, SSOP, TSOP, PLCC, QFP, BGA и др.) осуществляется с помощью дополнительных адаптеров, производимых фирмой Фитон, а также сторонними производителями. С помощью специальных адаптеров программатор поддерживает программирование микросхем, установленных в устройстве пользователя.

Комплект поставки

Программатор	- 1
Блок питания	- 1
Кабель связи с PC	- 1
CD с программным обеспечением	- 1
Паспорт	- 1

Подробнее характеристики программатора ChipProg-2 смотрите в разделе **Основные характеристики ChipProg-2.**

2.3.1 Основные характеристики ChipProg-2

Основные характеристики программатора ChipProg-2 можно представить следующими группами характеристик:

Характеристики Аппаратуры ChipProg-2

Характеристики Программного Обеспечения ChipProg-2

2.3.1.1 Характеристики Аппаратуры ChipProg-2

1. Архитектура программатора построена на базе программируемой матрицы (FPGA). Расширение списка поддерживаемых микросхем производится путем простого обновления версии программного обеспечения.
2. Логические драйверы на базе FPGA, способные подвести логические сигналы низкого, высокого уровня, внешнего генератора, а также Pullup, Pulldown на любой вывод колодки.
3. 8-ми разрядные цифро-аналоговые преобразователи для программирования аналоговых источников напряжения.
4. Возможность программирования фронта подъема и спада аналогового напряжения.
5. Возможность подключения аналоговых напряжений питания и программирования на ограниченное количество

© 2009 Фитон, http://www.phyton.ru

выводов микросхемы.

6. Возможность коммутации ограниченного количества вывода микросхемы с «землей».
7. Наличие быстродействующих схем защиты от перегрузки по току, увеличивающих надежность программатора и не выводящих из строя неправильно подключенные микросхемы.
8. Самотестирование.

Поддерживаемые корпуса микросхем:

1. Поддержка всех микросхем в корпусах DIP в колодке программатора без дополнительных адаптеров.
2. Поддержка микросхем в корпусах до 40 выводов в универсальных адаптерах.
3. Поддержка микросхем в корпусах SDIP, PLCC, SOIC, SOP, PSOP, TSOP, TSOPII, TSSOP, QFP, TQFP, VQFP, QFN, SON, BGA, CSP с помощью дополнительных адаптеров.
4. Совместимость с адаптерами сторонних производителей.

2.3.1.2 Характеристики Программного Обеспечения ChipProg-2

1. Работа под управлением Windows 95/98/ME/2000/XP/Vista.
2. Дружественный, интуитивно понятный, двуязычный интерфейс
3. Поддержка всех процедур работы с микросхемой: чтение, сравнение, контроль чистоты, запись и стирание, установка защиты, программирование конфигурационных битов, работа с памятью данных и т.п.
4. Многобуферный интерфейс с возможностью создания неограниченного числа буферов. Буфера разбиваются на подслои, имеющие структуру адресного пространства микросхем
5. Полноценный двоичный редактор с возможностью ручного редактирования данных, а также функции заполнения, сравнения, копирования, поиска и замены, инвертирования, вычисления контрольной суммы, логических операции OR, AND, XOR.
6. Загрузка и сохранение файлов в двоичном коде, Standard Extended Intel HEX, Motorla S-record, POF, JEDEC, PRG, Holtek OTP, ASCII HEC, ASCII OCTAL, Angstrem SAV форматах.
7. Встроенный язык сценариев, обеспечивающий доступ ко всем ресурсам программатора. Посредством применения этого языка можно значительно облегчить работу с программатором, автоматизируя рутинные операции.

2.4 Работа с программатором

В этой главе описываются операции, которые могут быть осуществлены при помощи программатора.

© 2009 Фитон, http://www.phyton.ru

2.4.1 Установка микросхемы в колодку программатора

Микросхемы в DIP корпусах.

Практически все микросхемы в DIP корпусе не требуют дополнительного адаптера. Убедиться требуется ли адаптер или нет можно в окне Информация о микросхеме . Стандартная установка микросхемы в DIP корпусе в виде символического рисунка представлена как на верхней панели самого программатора, так и в окне Информация о микросхеме. Стандартная установка предполагает, что микросхемы с количеством ног меньшим количества выводов сокетки программатора смещаются в программаторе в сторону, противоположную от рычажка сокетки. В том случае, если микросхема устанавливается нестандартным образом, то при установке типа микросхемы выдается предупреждающее сообщение, а также в окне Информация о микросхеме показыаается как это нужно сделать.

Микросхемы в корпусах, отличных от DIP.

Установка микросхем в корпусах, отличных от DIP требует специального адаптера. Название этого адаптера, а также его установка в сокетку программатора указывается в окне Информация о микросхеме .

Установка микросхем в адаптер.

На адаптере всегда указан "ключ", символизирующий первый вывод адаптера. Микросхема **всегда** устанавливается в соответствии с этим ключом: первый вывод микросхемы должен соответствовать первому выводу адаптера.

2.4.2 Программирование в плате пользователя

Программатор ChipProg генерирует все необходимые сигналы, необходимые для программирования микросхем в плате пользователя.

Программирование микросхем в плате пользователя осуществляется с помощью специальных адаптеров. Название адаптера указано в окне Device Information. Схемы всех адаптеров, а также подключение к ним плат пользователя, представлены в файле adapters.chm поставки.

Подключение к устройству пользователя:

Основные требования

1. Подключение должно осуществляться в строгом соответствии со схемой соответствующего адаптера (см. файл adapters.chm),
2. Устройство пользователя должно быть разработано с таким учетом, чтобы не оказывать шунтирующего влияния на логические сигналы программатора.
3. Устройство пользователя должно предусматривать (если это необходимо) возможность подачи со стороны программатора напряжения программирования, превышающего напряжение питания.

Напряжение питания

Возможны два варианта:

1. Подача питания от программатора. В этом случае необходимо иметь в виду, что нагрузочная способность источника питания программатора ограничена током 80 мА. Источник питания программатора способен работать на емкостную нагрузку, которая не должна превышать 50 мкФ.
2. Подача питания на плату пользователя от внутреннего источника. В этом случае, питание от программатора на плату пользователя подаваться не должно. Необходимо иметь в виду, что логические сигналы со стороны программатора имеют высокий уровень, соответствующий его напряжению питания

ВНИМАНИЕ: категорически запрещается подавать питание на плату пользователя одновременно от программатора и от внутреннего источника.

Электрические характеристики сигналов программатора

Нагрузочную способность логических сигналов 5mA.

Нагрузочную способность напряжения питания - 80 мА.

Нагрузочную способность напряжения программирования - 80 мА.

© 2009 Фитон, http://www.phyton.ru

Таблица подключения к микросхеме.

Подключение устройства пользователя к программатору должно строго соответствовать таблице соединений для установленного типа микросхемы и адаптера. Чтобы посмотреть таблицу соединений, нужно активизировать иконку **Adapters Connection List** из папки, куда установлен программатор.

ВНИМАНИЕ: тщательно проверяйте правильность подключения программатора к плате пользователя. Неверное подключение может привести к выходу из строя аппаратуры программатора и платы пользователя.

Подробнее смотрите в разделах:

Особенности программирования микроконтроллеров PICmicro фирмы Microchip,
Особенности программирования микроконтроллеров AVR фирмы Atmel,
Особенности программирования микроконтроллеров MCS-51 фирмы Atmel.

2.4.2.1 Микроконтроллеры PICmicro, особенности программирования

Большинство микроконтроллеров семейства PICmicro фирмы Microchip программируется с использованием высоковольтного режима программирования HV ISP Mode (High-Voltage in-System Programming Mode). В этом режиме на вывод MCLR контроллера подается напряжение 13В. В устройстве пользователя должна быть предусмотрена возможность подачи этого напряжения. Например, в этом случае не допускается соединение вывода MCLR с питанием контроллера.

Еще одной особенностью программирования микроконтроллеров PICmicro является необходимость программирования микросхем при напряжении питания 5В и верификацией при минимально и максимально допустимых напряжениях питания микросхемы. В программаторе в поле 'Programming Options' диалога Program допускается корректировать напряжения верификации (Vccp min, Vccp max) после программирования. Напряжение питания при программировании нельзя изменять; оно равно 5В. Это обязательно нужно иметь в виду при проектировании схемы. Если устройство пользователя не допускает подачу напряжение питания равного 5В, то необходимо предусмотреть возможность подключения к этому напряжению только программируемого микроконтроллера.

2.4.2.2 Микроконтроллеры AVR, особенности программирования

Все микроконтроллеры семейства AVR фирмы Atmel программируется в режиме низковольтового программирования (т.е. при напряжении питания). Практически все микроконтроллеры в этом режиме требуют наличия тактового генератора. Если в устройстве пользователя такой генератор уже присутствует, то тактовый сигнал, поступающий с программатора, не должен быть соединен с платой пользователя. В противном случае, он должен быть подсоединен к соответствующему выводу микросхемы. В поле 'Programming Options' диалога Program необходимо указать значение тактовой частоты. Это значение используется в алгоритме программирования. Программатор генерирует тактовую последовательность частотой 3 МГц.

2.4.2.3 Микроконтроллеры Atmel 8051, особенности программирования

Все микроконтроллеры семейства 8051 фирмы Atmel программируется в режиме низковольтового программирования (т.е. при напряжении питания). Практически все микроконтроллеры в этом режиме требуют наличия тактового генератора. Если в устройстве пользователя такой генератор уже присутствует, то тактовый сигнал, поступающий с программатора, не должен быть соединен с платой пользователя. В противном случае, он должен быть подсоединен к соответствующему выводу микросхемы.

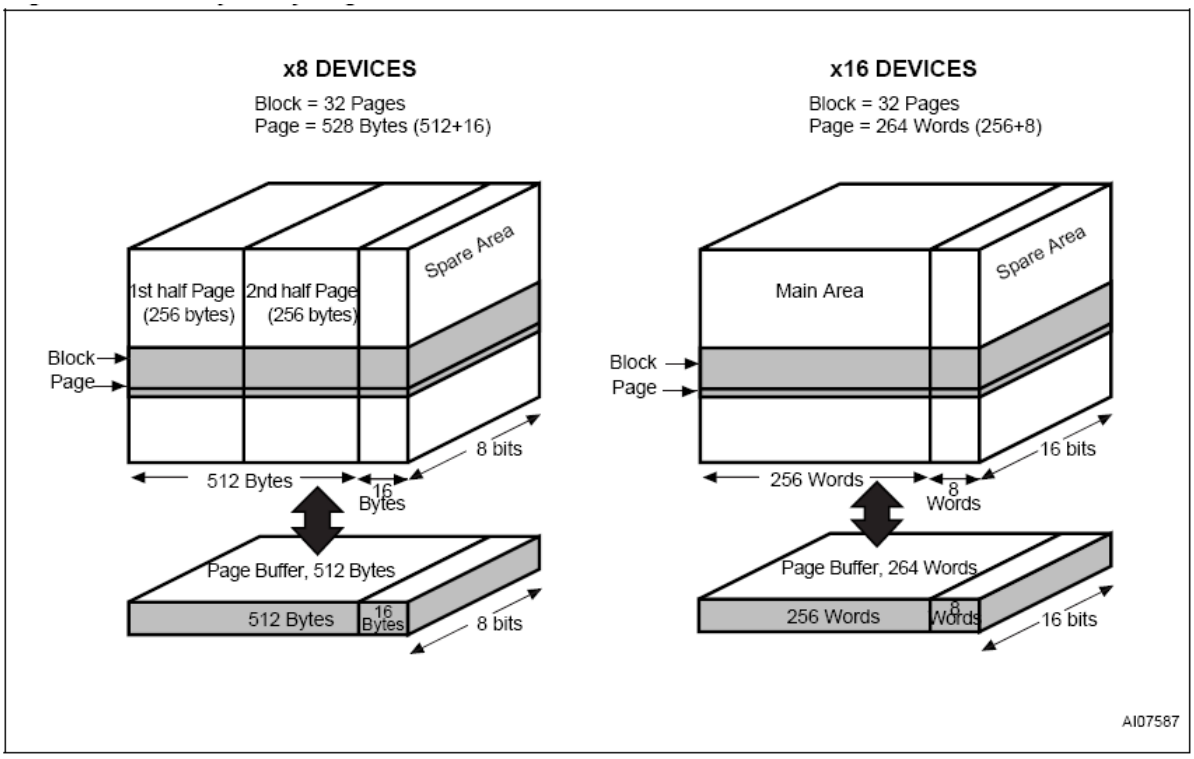
2.4.3 Программирование NAND Flash

В этом разделе описывается программирование микросхем NAND Flash. Очень важно правильно настроить все параметры программирования, в противном случае, вы можете запрограммировать микросхему неправильно.

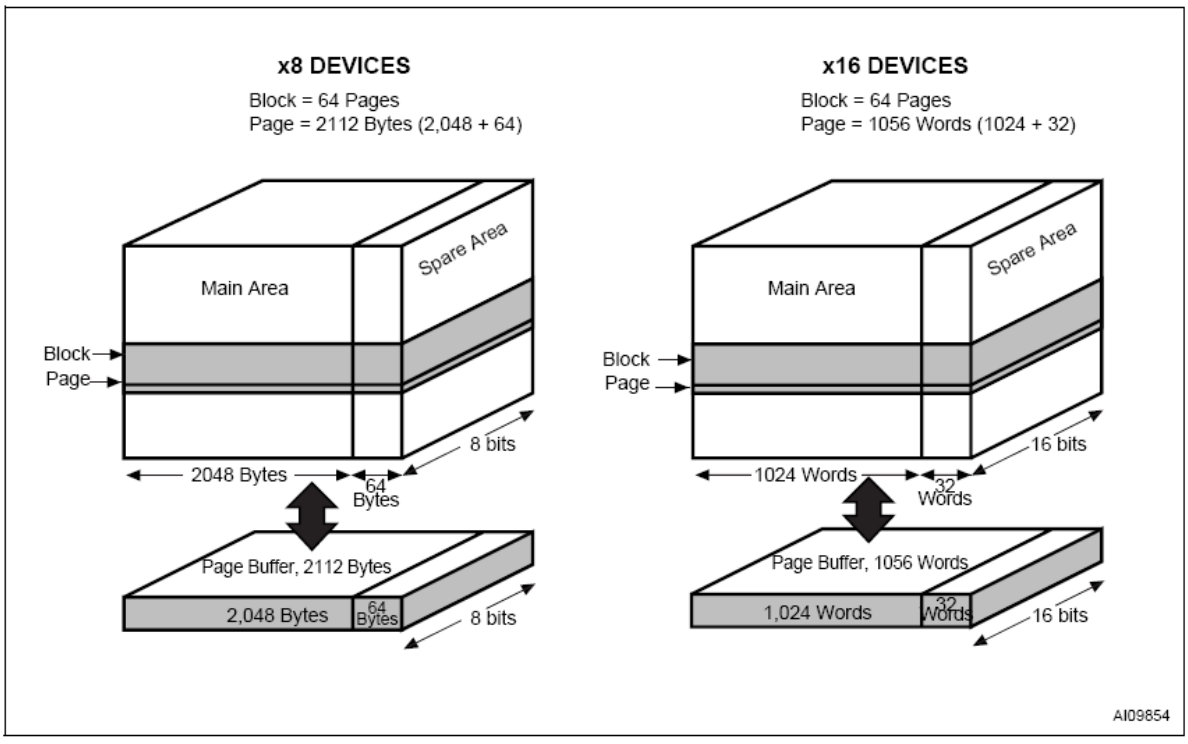
© 2009 Фитон, http://www.phyton.ru

2.4.3.1 Структура памяти NAND Flash

NAND Flash память подразделяется на блоки (Block) памяти, которые в свою очередь делятся на страницы (Page). Существуют микросхемы с маленьким размером страницы (small page) и с большим размером страницы (large page). Размер страницы зависит от общего размера микросхемы. Для маленькой страницы обычно характерны микросхемы объемом от 128Кбит до 512Кбит. Микросхемы с большим размером страницы имеют объем от 256Кбит до 32Гбит и выше. Маленький размер страницы равен 512 байтам для микросхем с байтной организацией и 256 словам для микросхем со словной организаций шины данных. Большая страница имеет размер 2048 байт для байтных микросхем и 1024 для словных. В последнее время появляются микросхемы с еще большим размером страницы. Она уже составляет 4096 байт для байтных микросхем.



Структура памяти микросхем с малым размером страницы фирмы STMicroelectronics.



Структура памяти микросхем с большим размером страницы фирмы STMicroelectronics.

Характерной особенностью микросхем NAND Flash является наличие плохих (дефектных) блоков (Bad blocks) как в новых микросхемах, так и появление таких блоков в процессе эксплуатации. Для маркирования плохих блоков, а также для сохранения дополнительной служебной информации или кодов коррекции, в архитектуре NAND Flash в дополнении к каждой странице памяти данных предусмотрена добавочная область Spare area. Для микросхем с малой страницей эта область имеет размер 16 байт / 8 слов. Для микросхем с большой страницей - 64 байта / 32 слова.

2.4.3.1.1 Плохие блоки (Bad blocks)

Характерной особенностью микросхем NAND Flash является наличие плохих блоков. Причем эти блоки могут появляться как в процессе эксплуатации, так и присутствовать в новой микросхеме. Обычно производитель микросхем гарантирует количество плохих блоков, не превышающее определенного размера.

Информация о плохих блоках поставляется производителем микросхем в определенном месте дополнительной области Spare Area.

2.4.3.1.1.1 Маркирование плохих блоков

Маркирование плохих блоков в микросхемах NAND Flash осуществляется записью обычно значения 0 по определенному адресу в области Spare Area нулевой страницы плохого блока.

Организация памяти	**Адрес в Spare Area**
Байтная организация, размер страницы - 512 байт.	5
Словная организация, размер страницы - 256 слов.	0
Байтная организация, размер страницы - 2048 байт.	0 или 5
Словная организация, размер страницы - 1024 слов.	0

Нужно иметь ввиду, что маркеры плохих блоков помещаются в обычные ячейки Flash памяти Spare Area, которые стираются при стирании всего блока памяти. Поэтому для сохранения информации о плохих блоках перед стиранием обязательно нужно сохранить эту информацию, а после стирания ее - восстановить. В программаторах ChipProg при установке опции InvalidBlockManagement в любое значение кроме Do Not Use сохранение и восстановление информации о плохих блоках происходит автоматически.

2.4.3.1.1.2 Обработка плохих блоков

Существует три наиболее распространенных способа обработки плохих блоков:

Пропуск плохих блоков

Область с резервными блоками

Контроль и коррекция ошибок

Алгоритм пропуска плохих блоков заключается в том, что при записи в микросхему анализируется в какой блок осуществляется запись. В случае наличия плохого блока, запись в этот блок не осуществляется, плохой блок пропускается, запись осуществляется в следующий после плохо блока блок.

© 2009 Фитон, http://www.phyton.ru

В этом методе память всей микрохемы делится на три области: User Block Area (UBA) - пользовательская область, Block Reservoir - резервная область, следуемая сразу за пользовательской областью, и таблицу соответствия плохих блоков хорошим (Reserved Block Area - RBA).

Алгоритм замены плохих блоков в этом методе таков: при выявлении плохого блока из области UBA блок переносится в область Block Reservoir, при этом в таблице RBA делается соответствующая запись замены блока.

Формат таблицы RBA:

2 байта	2 байта	2 байта	2 байта	....	2 байта	2 байта
Маркер	**Count Field**	**Invalid Block**	**Replaced Block**	....	**nvalid Block**	**Replaced Block**

Маркер имеет значение 0FDFEh.

Count Field начинается с 1 и значение этого поля увеличивается на 1 на каждой следующей странице данного блока.

Invalid Block - номер замененного плохого блока.

Replaced Block - номер блока, на который был заменен плохой блок.

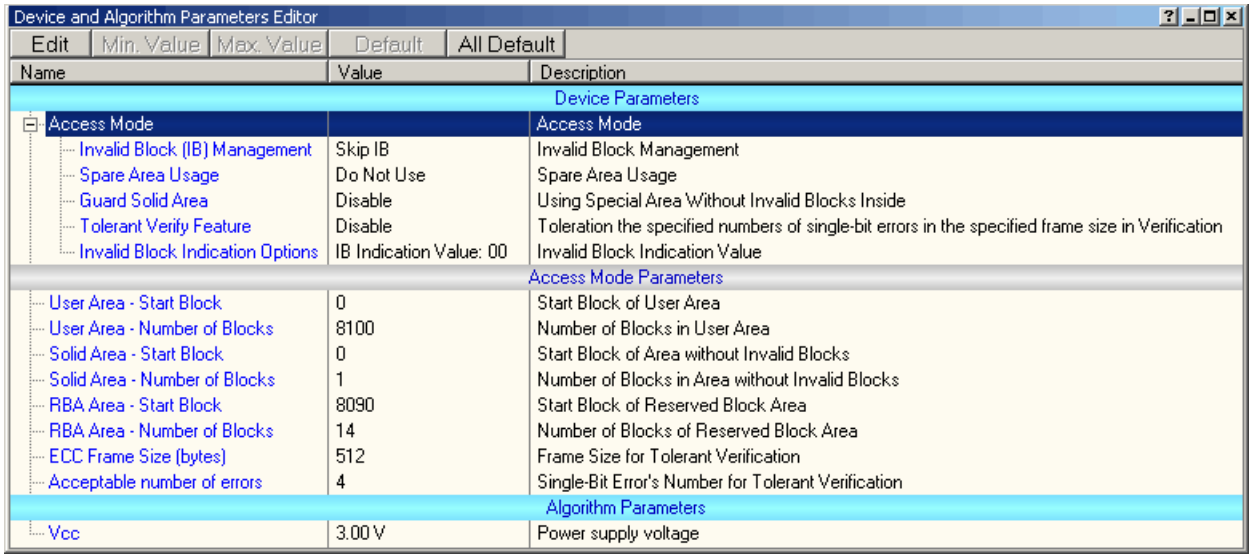
Пары Invalid Block и Replaced Block следуют одна за другой до конца страницы.

В области RBA находятся две таблицы в двух блоках. Таблица во втором блоке используется как резервная на случай, если информация в первой окажется недостоверной.

Для увеличения достоверности данных могут использоваться алгоритмы контроля и коррекции ошибок (Error Checking and Correction - ECC). Эта дополнительная информация может помещаться в свободное пространство Spare Area.

2.4.3.2 Программирование NAND Flash на программаторах ChipProg (COPY)

Перед работой с микросхемами NAND Flash в окне "Редактор параметров микросхемы и алгоритма программирования" обязательно нужно установить параметры алгоритма программирования программатора.



© 2009 Фитон, http://www.phyton.ru

2.4.3.2.1 Access Mode

Работа с плохими блоками

Использование области Spare Area

Использование специальной области без плохих блоков

Нечувствительность к ошибкам сравнения

Маркер плохих блоков

2.4.3.2.1.1 Invalid Block Management

В этом пункте выбирается алгоритм работы с плохими блоками.

Do Not Use	Не обрабатывать информацию о плохих блоках.
Skip IB	Пропускать плохие блоки.
Skip IB with Map in 0-th Block	Пропускать плохие блоки, в нулевой блок помещать карту плохих блоков.
RBA (Reserved Block Area)	Использовать алгоритм RBA.

2.4.3.2.1.2 Spare Area Usage

Do Not Use	Spare Area в микросхеме не используется. В микросхеме программируются страницы памяти без учета Spare Area.
User Data	Spare Area используется как пользовательская память. В этом случае при программировании микросхемы информация из буфера помещается сначала в основную страницу микросхемы, а затем в дополнительную область Spare Area. В этом случае буфер программатора выглядит как непрерывный поток основных страниц микросхемы и пристыкованных к ним областей Spare Area.
User Data with IB Info Forced	Spare Area интерпретируется аналогично предыдущему случаю за исключением того, что маркеры плохих блоков прописываются вместо информации пользователя.

2.4.3.2.1.3 Guard Solid Area

Это режим использования специальной области без плохих блоков. Обычно такие области используются в качестве загрузчиков микропроцессоров. В этой области недопустимо использование плохих блоков.

Эта опция используется совместно с параметрами:

Solid Area - Start Block - начальный блок области без плохих блоков.

Solid Area - Number of Blocks - количество блоков в этой области.

В случае, если внутри заданного диапозона **Solid Area** попадется плохой блок, программатор выдаст ошибку.

© 2009 Фитон, http://www.phyton.ru

2.4.3.2.1.4 Tolerant Verify Feature

Эта опция позволяет включить режим не чувствительности к ошибкам сравнения.

Обычно, эту опцию имеет смысл использовать, если в устройстве пользователя применяются алгоритмы контроля и коррекции ошибок (ECC). В этих случаях допускается наличия определенного количества ошибок на определенный размер массива данных.

Эти параметры и указываются в параметрах алгоритма программирования:

ECC Frame size (bytes) - размер массива данных.

Acceptable number of errors - допустимое количество однобитных ошибок.

2.4.3.2.1.5 Invalid Block Indication Option

В этой опции выбирается информация, которая используется в качестве маркера плохих блоков. Допускается выбрать либо значение 00h, либо 0F0h.

2.4.3.2.2 Access Mode Parameters

Пользовательская область

Параметры режима "Область без плохих блоков"

Область RBA

Размер фрэйма контроля и коррекции ошибок

Допустимое количество однобитных ошибок

2.4.3.2.2.1 User Area

Пользовательская область - это область микросхемы, с которой работают процедуры Программирования, Чтения и Сравнения. Процедуры Стирания и Контроля на чистоту работают со всем массивом микросхемы.

Пользователю необходимо установить параметры:

User Area - Start Block - начальный блок пользовательской области.

User Area - Number of Blocks - количество блоков в пользовательской области.

2.4.3.2.2.2 Solid Area

Это параметры режима Область без плохих блоков.

Solid Area - Start Block - начальный блок области без плохих блоков.

Solid Area - Number of Blocks - количество блоков в этой области.

2.4.3.2.2.3 RBA Area

Это параметры RBA таблицы алгоритма обработки ошибок Reserved Block Area.

RBA Area - Start Block - начальный блок таблицы RBA.

RBA Area - Number of Blocks - количество блоков в таблице RBA.

© 2009 Фитон, http://www.phyton.ru

2.4.3.2.2.4 ECC Frame size

Параметр режима Не чувствительность к ошибкам сравнения.

Этот параметр определяет размер массива данных, в котором допускаются однобитные ошибки.

2.4.3.2.2.5 Acceptable number of errors

Параметр режима Не чувствительность к ошибкам сравнения.

Этот параметр определяет количество однобитных ошибок, допустимых в массиве, размер, которого определяется параметром.

ECC Frame size.

2.4.3.2.3 Карта плохих блоков.

Карта плохих блоков создается в подслое Invalid Block Map. Карта блоков представляется как непрерывный массив бит. Хорошие блоки представляются значением 0, плохие блоки - 1.

Например, значение 02h по нулевому адресу говорит о том, что 0,2,3,4,5,6,7 блоки являются хорошими, 1-ый блок является плохим. Значение 01h по первому адресу говорит о том, что только 8-ой блок является плохим из группы блоков 8..15.

2.4.4 Функции процесса Программирования

В окне Программирование можно выполнить те команды для процесса программирования, которые вам необходимы.

2.4.4.1 Проверить чистоту микросхемы

1. Выберите тип микросхемы, с которым вы будете работать, нажав кнопку **Select Device** на главной инструментальной панели, или выберите команду **Меню > Конфигурация > Выбрать микросхему.**
2. Вставьте микросхему, выбранного типа, в разъем программатора
3. Нажмите кнопку **Check** на главной панели инструментов в окне Программирование, в поле **Информация об операциях** появится запись Контроль... Ok , если микросхема чистая. В противном случае в том же поле появится предупреждающая запись.

2.4.4.2 Стирание

1. Надо убедиться, что микросхему вообще можно стереть электрическим способом. Некоторые микросхемы можно запрограммировать только единожды, некоторые микросхемы можно стереть только при помощи ультрафиолета (UV), а некоторые микросхемы совсем не требуют стирания. В этих случаях кнопка **Erase** - заблокирована (окрашена серым цветом).
2. Если микросхему можно стереть электрическим способом, нажмите кнопку **Erase** на основной панели инструментов. Подождите до появления записи ***Erasing ... OK*** в поле **Информация об операциях** окна Программирование или предупреждающей записи, если микросхему не удалось очистить.

2.4.4.3 Как Запрограммировать микросхему

Для того чтобы запрограммировать микросхему надо последовательно выполнить несколько операций:

- загрузить файл, предназначенный для записи в микросхему;
- отредактировать файл (если необходимо);

© 2009 Фитон, http://www.phyton.ru

- сконфигурировать микросхему (если необходимо);
- записать подготовленную информацию в микросхему и сравнить записанную информацию с исходными данными.

2.4.4.3.1 Как Загрузить Файл, который должен быть записан в микросхему

1. В Главном Меню выберите команду **Файл >Загрузить.**
2. Запишите имя исходного файла, выберите формат файла, буфер загрузки, подслой и адреса.
3. Подождите до появления записи ***File loaded*** в поле **Информация об операциях** окна Программирование или предупреждающе й записи.

2.4.4.3.2 Как Отредактировать Исходную Информацию

1. Если вам необходимо изменить данные исходного файла, перед тем как записать информацию в микросхему, следует открыть окно **Дамп Буфера.** При этом нужно снять флаг **Только просмотр, редактирование запрещено** или отжать кнопку **View** на панели инструментов для того, чтобы разрешить редактирование.
2. Сделайте необходимые изменения в окне.

2.4.4.3.3 Как Сконфигурировать микросхему

1. Если вам необходимо изменить параметры, отображенные в окне Параметры Микросхемы и Алгоритма, подведите перемещаемую строку синего цвета к нужному имени.
2. Кликните по имени, которое вам надо изменить. Откроется окно, в котором можно выбрать нужный параметр. После того как параметр будет изменен, он поменяет цвет на красный.
3. Также следует поступать со всеми параметрами, которые надо изменить. Все установки будут исполняться при исполнении команд в окне Программирования.

2.4.4.3.4 Как записать информацию

1. Выберите закладку Опции, адреса, чередование в окне **Программирование.** В поле опции проверьте все установки. Лучше всего для надежного программирования устанавливать опции **Контроль чистоты перед программированием** и **Проверка после программирования**.
2. Выберите закладку Программирование. В поле **Операции** выберите функцию **Программирование** и дважды кликните по этой строке, чтобы начать процесс программирования. То же самое можно проделать, нажав кнопку **Program** на инструментальной линейке или выбрав в Меню команду **Меню> Команды> Запрограммировать микросхему.**
3. После успешного программирования в поле **Информация об операциях** появится запись **Программирование... Ok**. В случае ошибочного программирования том же появится информация об ошибке.
4. Далее идите по списку функций в поле **Операции**.
5. При выполнении последующих функций, надо дожидаться в поле **Информация об операциях** информации со словом Ok [xxxxx... Ok]
6. Продолжайте до полного успешного программироваия.

2.4.4.4 Как Сравнить

1. Обычно программатор сразу после программирования сравнивает содержимое микросхемы с содержимым исходного буфера.
2. Для дополнительного сравнения следует нажать кнопку **Verify** на основной панели инструментов**.**

© 2009 Фитон, http://www.phyton.ru

3. Подождите до появления записи ***Verifying ... OK*** в поле **Информация об операциях** окна Программирование или предупреждающей записи, если сравнение не прошло правильно..

2.4.4.5 Как Прочитать микросхему

1. Для того чтобы прочитать микросхему, следует нажать кнопку **Read** на основной панели инструментов**.**

2. Подождите до появления записи ***Reading... OK*** в поле **Информация об операциях** окна Программирование или предупреждающей записи.

2.4.4.6 Как Сохранить данные, прочитанные из микросхемы

1. При чтении данных из микросхемы, они записываются в буфер.Данные, которые читаются из микросхемы, помещаются в буфер. На панели инструментов можно нажать кнопку Save для сохранения данных из буфера.

2. В открывшемся диалоге можно определить имя и путь файла для записи, также можно выбрать формат файла, буфер и подслой, откуда сохранять файл. Здесь можно определить начальный и конечный адрес файла, который надо сохранить.

2.4.4.7 Как Продублировать микросхему

1. Установить исходную микросхему, которую требуется скопировать в разъем программатора.

2. Нажать кнопку **Read** на панели инструментов.

3. Подождите до появления записи ***Reading... OK*** в поле **Информация об операциях** окна Программирование . Убедитесь, что содержимое считанной микросхемы находится в текущем буфере.

4. Замените в разъеме считываемую микросхему на чистую, куда требуется записать информацию. Следует проверить чистоту микросхемы.

5. Вернитесь к разделу Как Редактировать Информацию для записи в микросхему далее действуйте по алгоритму записи в микросхему.

© 2009 Фитон, http://www.phyton.ru

3 Графический Интерфейс Пользователя

3.1 Меню

3.1.1 Обзор

Главное меню ChipProgLPT включает в себя следующие подменю:

- Меню **Файл**
- Меню **Просмотр**
- Меню **Конфигурация**
- Меню **Команды**
- Меню **Сценарии**
- Меню **Окна**
- Меню **Справка**

Чтобы открыть меню, используйте мышь или комбинацию клавиш **Alt+буква**, где «буква»—это подчёркнутая буква в названии пункта или команды меню.

© 2009 Фитон, http://www.phyton.ru

3.1.2 Меню Файл

Команды меню **Файл** выполняют действия с файлами разрабатываемого проекта. Если для некоторой команды есть кнопка на панели инструментов, её изображение показано в первом столбце. Если команду меню можно выполнить комбинацией клавиш, эта комбинация показана в меню справа от команды.

Кнопка	Команда	Описание
	Загрузить ...	Открывает диалог **Загрузить файл**.
	Перегрузить	Перезагружает последнюю из ранее загруженных программ.
	Записать...	Открывает диалог **Сохранить файл из буфера**.
	Конфигурационные файлы	Открывает суб–меню для действий с **файлами конфигурации**. .
	Выход	Закрывает сеанс работы ChipProgLPT . Также, можно использовать другие стандартные способы завершения работы приложения Windows (комбинации клавиш **Alt+F4** или **Alt+X**).

© 2009 Фитон, http://www.phyton.ru

3.1.2.1 Конфигурационные файлы

При завершении сеанса работы, ChipProgLPT автоматически сохраняет параметры своей конфигурации в нескольких файлах. В начале нового сеанса, он открывает эти ранее сохранённые файлы. Также, в любой момент времени любой из этих файлов можно сохранить или загрузить независимо друг от друга, через меню **Файл**, командой **Конфигурационные файлы**. Можно иметь несколько наборов файлов конфигурации, для разных задач отладки, и подгружать их «на ходу».

- Файл конфигурации экрана (**Desktop**) содержит значения параметров отображения на экране, расположение, размеры, цвета и шрифты всех специализированных окон отладчика. Расширение этого файла -(**.dsk).**
- Файл опций (**Option**) содержит данные о типе микроконтроллера, параметры отладки, точки останова, и т.д. Расширение этого файла -(**.opt).**
- Файл **Сессии** содержит данные о сеансе и указывает, какой файл конфигурации экрана и файл опций следует загрузить в начале следующего сеанса работы (если не используется файл проекта). Этот файл может быть загружен или сохранен при помощи команд **Загрузить сессию** и **Сохранить сессию** из субменю команды **Конфигурационные файлы.** Расширение этого файла -(**.ses).**
- Файл **История** содержит все значения, которые Вы вводили в текстовых полях всех диалогов программы. Этот файл невидимый для пользователя, но значения, которые ранее были сохранены, можно достать. Расширение этого файла -(**.hst).**

3.1.3 Меню Просмотр

Меню **Просмотр** обеспечивает доступ к окнам ChipProgLPT`a:

Кнопка	Команда	Описание
	Программирование	Открывается окно **Программирование**
	Параметры и алгоритмы микросхемы	Открывается окно **Параметры Микросхемы и Алгоритмы**
	Дамп буфера	Открывается окно **Дамп буфера**.
	Информация о микросхеме	Открывается окно **Информация о микросхеме**.
	Консоль сообщений	Открывается окно **Консоль сообщений**.

Локальные меню

В каждом окне есть собственное локальное (контекстное) меню. Чтобы открыть такое меню, надо нажать правую кнопку мыши в пределах окна или нажать комбинацию **Ctrl+Enter** или **Ctrl+F10**.

Для некоторых команд локального меню (но не для каждой), есть кнопка на панели инструментов этого окна.

© 2009 Фитон, http://www.phyton.ru

3.1.4 Меню Конфигурация

Это меню дает доступ ко всем диалогам конфигурации ChipProgLPT`a.

Кнопка	Команда	Описание
	Выбрать микросхему...	Открывает диалог **Выбор микросхемы**.
	История выбора микросхем	Список устройств предыдущего выбора
	Буфера	Открывает диалог **Буферы**
	Опции...	Открывает диалог **Опции**
	Опции экрана...	Открывает диалог **Опции Экрана**, в котором пять закладок: закладка **Шрифты**, закладка **Цвета**, закладка **Назначение клавиш**, закладка **Линейка управления** и закладка **Прочие**.

3.1.4.1 Диалог Выбор микросхемы

Элементы диалога	Описание
Показывать в списке типы:	В этом поле можно выбрать группы микросхем, которые будут показываться в списке поддерживаемых микросхем. К этим группам относятся EPROM, EEPROM, FLASH; PLD, PAL, EPLD; микроконтроллеры; ISP-программирование (программирование в устройстве пользователя).
Микросхемы	В этом поле показывается список микросхем в соответствии с маской поиска и опциями списка.
Маска поиска:	В этом поле указывается маска поиска микросхем. В этом поле можно использовать символ *, который символизирует любые символы в любом количестве. Например, маска 'PIC18*64' позволит найти микросхемы, начинающиеся с PIC18 и заканчивающиеся символами 64.
Корпуса/Адаптеры	В этом поле указываются поддерживаемые корпуса микросхемы и соответствующие им адаптеры.

3.1.4.2 Диалог Буферы

Элементы диалога	Описание
Список буферов:	
Добавить	Создать новый буфер памяти. Открывает диалог **Конфигурация буфера**
Удалить	Удалить буфер памяти
Редактировать	Редактировать опции буфера памяти. Открывает диалог **Конфигурация буфера**
Дамп	Перейти в окно дампа соответствующего буфера памяти
Выделение памяти	В этом поле указывается размер оперативной памяти компьютера, которой выделяется для использования буфера памяти.
Своп-файлы	Это временный файл для размещения на диске образа буфера памяти. В этом поле допускается выбрать размещение этих файлов автоматически или выбрать их размещение вручную.

© 2009 Фитон, http://www.phyton.ru

Использовать сетевые диски	Опция позволяет использовать сетевые диски компьютера для размещения своп-файлов.
Оставить на каждом диске не менее	Минимальное дисковое пространство, которое не занимается своп-файлами.

3.1.4.2.1 Диалог Конфигурация буфера

Диалог **Конфигурация буфера** позволяет сконфигурировать буферы и подслои в удобной для вас форме.

Диалог содержит содержит закладки. В первой закладке всегда присутствует основной подслой. Если в установленном типе микросхемы имеются программируемые дополнительные адресные пространства, реализованные в виде дополнительных подслоев буфера, то в диалоге конфигурации появляются дополнительные закладки для конфигурирования этих подслоев.

3.1.4.2.1.1 Основной подслой

В этой закладке можно сконфигурировать опции основного подслоя буфера.

Элементы диалога	**Описание**
Имя буфера	Поле, в которое надо ввести **Имя Буфера**.
Размер подслоя 'Code'	Размер подслоя можно выбрать из списка.
Заполнять подслой 'Code' значением:	Две опции, которые позволяют заполнять подслой значениями: перед загрузкой файла и/или после выбора микросхемы.
Значение заполнения:	Из двух кнопок надо выбрать одну либо заполнять предопределенным значением по умолчанию, либо написать то значение, которое вам необходимо.
Уменьшать размер буфера при выборе микросхемы	Эта опция позволяет уменьшать размер буфера до размера микросхемы.

3.1.4.2.1.2 Дополнительный подслой

В этой закладке можно сконфигурировать заполнение дополнительного подслоя буфера значениями.

Элементы диалога	**Описание**
Заполнять 'Область идентификатора' значением:	Две опции, которые определяют время загрузки **подслоя** значениями - это заполнять подслой перед загрузкой файла или заполнять подслой после выбора микросхемы.
Значения заполнения:	Из двух кнопок надо выбрать одну либо заполнять предопределенным значением по умолчанию, либо написать то значение, которое вам необходимо.

3.1.4.3 Диалог Опции

Элементы диалога	**Описание**
Опции	
Загружать последний файл при старте	Опция позволяет загружать последний файл при старте программатора.

© 2009 Фитон, http://www.phyton.ru

Выполнять самотестирование при старте	Опция позволяет выполнять самотестирование программатора при старте.
Звуки	
Звук при ошибке операции с микросхемой	Опция позволяет выбрать из списка звук, который будет происходить при ошибке операции с микросхемой.
Звук при завершении операции с микросхемой	Опция позволяет выбрать из списка звук, который будет происходить при завершении операции с микросхемой.

3.1.4.4 Диалог Опции Экрана

Диалог содержит закладки:

закладка **Шрифты**,

закладка **Цвета**,

закладка **Назначение клавиш**,

закладка **Линейка управления**,

закладка **Прочие**.

3.1.4.4.1 Шрифты Закладка

Закладка **Шрифты** диалога **Опции экрана** управляет шрифтами в окнах программы ChipProgLPT. Используются только шрифты фиксированной ширины (monospaced, non-proportional), по умолчанию-это Fixedsys. В качестве пользовательской настройки внешнего вида окна можно задать другой шрифт: либо единый шрифт для всех окон, либо индивидуальный шрифт для каждого окна.

В списке **Окна** перечислены все типы окон. Чтобы задать параметры для некоторого типа окон, выделите его в списке. Новые установленные параметры действуют для всех окон выделенного типа, включая уже открытые.

Элемент диалога	Описание
Окно имеет заголовок	Включает строку заголовка для окон данного типа. Когда флаг снят, окна получаются меньше размером за счёт отсутствующего заголовка. Также, смотрите примечания внизу.
Линейка управления окна	Управляет положением панели инструментов в окне данного типа.
Сетка	Включает отображение вертикальной и горизонтальной сетки в окне и разрешает изменение ширины столбцов (при включенной вертикальной сетке).
Интервал между строками	Задаёт межстрочное расстояние, которое будет добавлено к стандартному расстоянию между строками. Новое значение можно напечатать или выбрать из списка недавно использованных значений.
Выбрать шрифт	Открывает диалог **Шрифт**. Выбранный шрифт будет действовать для всех окон данного типа.
Этот шрифт для всех окон	Использует шрифт, установленный для окон данного типа, ко всем окнам в программе ChipProgLPT.

Примечания

1. Чтобы передвинуть окно, у которого нет строки заголовка, поместите курсор мыши на участок панели инструментов этого окна, где нет кнопок, и далее действуйте так, как будто панель инструментов является строкой заголовка. Также, можно обращаться к функциям управления окном через его системное меню, нажатием комбинации клавиш **Alt+<серый минус>**.

© 2009 Фитон, http://www.phyton.ru

2. В локальном меню каждого окна есть пункт **Свойства**. Пункты **Заголовок окна** и **Линейка управления** субменю **Свойства** переключают строку заголовка и панель инструментов для данного отдельного окна.

3.1.4.4.2 Цвета Закладка

Закладка **Цвета** диалога **Опции экрана** управляет цветом в специализированных окнах. По умолчанию, основные цвета заимствованы из Windows.

Элемент диалога	Описание
Схема цветов	Задаёт название цветовой схемы. Его можно напечатать или выбрать недавно использованное из списка с кнопкой. Кнопка **Сохранить** сохраняет используемую схему на диск. Кнопка **Удалить** удаляет её.
Цвета	Список названий групп цветов. Каждая группа состоит из нескольких цветов.
Стандартный цвет Windows	Когда флаг установлен, выделенный цвет позаимствован из Windows. Если в последующем Вы измените цвета, Windows через панель управления, этот цвет изменится соответственно. Этот флаг имеется только для цвета фона и текста.
Инвертированный цвет фона/текста	Когда флаг установлен, то инвертируются выделенные цвета окна (для текста и фона). Например, если в окне **Переменные** цвет фона белый и цвет текста чёрный, то для строки с выделенной переменной будет подсветка из черного фона и белого текста.
Выбрать	Открывает диалог **Цвет**, если флаги **Стандартный цвет Windows** и **Инвертированный цвет фона/текста** сняты для окон этого типа. Диалог **Color** также открывается, если дважды щелкнуть цвет в списке **Цвета**. .
Установить для всех	Задаёт использование данного цвета во всех окнах ChipProgLPT. Такая возможность удобна для цвета текста и фона. Например, если для окна **Редактора** выбрать голубой фон и жёлтый текст и потом нажать кнопку **Установить для всех**, то эти цвета будут заданы для фона и текста во всех окнах.
Шрифт	Для выделения синтаксиса в окне **Редактора** можно задать дополнительные атрибуты шрифта: «Жирный» и «Курсив». В некоторых случаях, Windows увеличивает размер символов при изображении жирного шрифта и шрифт становится неприменим, потому что жирный и обычный шрифт должны быть одного размера. В таких случаях атрибут «Жирный» игнорируется. Иногда, такой эффект случается со шрифтом Fixedsys. Если необходимо воспользоваться атрибутом «Жирный», выберите шрифт Courier New.

3.1.4.4.3 Назначение клавиш экрана Закладка

Закладка **Назначение клавиш** диалога **Опции экрана** позволяет присвоить комбинации клавиш вызова любой команды в ChipProgLPT. Столбец **Команды меню** отображает древообразную систему команд. Столбцы **Клав. 1** (**Клав. 2**) содержат соответствующие комбинации клавиш, назначенные командам. Все действия в этой закладке относятся к выделенной команде.

Элемент диалога	Описание

© 2009 Фитон, http://www.phyton.ru

Задать клав. 1 **Задать клав. 2**	Открывает диалог **Задать комбинацию клавиш**. В диалоге нажмите комбинацию клавиш, которую Вы собираетесь назначить данной команде, или нажмите **Отмена**. Также, этот диалог можно открыть двойным щелчком в «ячейке», где пересекается строка данной команды и столбец **Клав. 1** или **Клав. 2**.
Удалить клав. 1 **Удалить клав. 2**	Отменяет назначенную комбинацию клавиш для данной команды. Также, для отмены комбинации можно щелкнуть правой клавишей мыши в «ячейке», где пересекается строка данной команды и столбец **Клав. 1** или **Клав. 2**.

3.1.4.4.4 Линейка управления Закладка

Закладка **Линейка управления** диалога **Опции экрана** включает панели инструментов окна и их кнопки.

Элемент диалога	Описание
Группы	Содержит список всех панелей инструментов программы ChipProgLPT. Чтобы включить/выключить панель инструментов, установите её флажок в списке.
Кнопки / Команды	Список кнопок для панели инструментов, выделенной в списке **Группы.** Чтобы показать/убрать кнопку панели, установите её флажок в списке.
«Плоские» линейки управления окон	Переключает внешность кнопок между «плоской» и квази-3D для панели инструментов специализированных окон в программе ChipProgLPT. Кнопки панели инструментов всегда «плоские».
Настройки линейки управления одни и те же для всех проектов и файлов экрана	Распространяет значения параметров этой закладки на другие проекты или файлы, которые будут открыты потом.

3.1.4.4.5 Прочие Закладка

Закладка **Прочие** диалога **Опции экрана** управляет различными функциями окон и параметрами сообщений в ChipProgLPT.

Элемент диалога	Описание
Строка статуса главного окна	Управляет наличием и расположением статусной строки окна программы ChipProgLPT.
Включить функцию «быстрого просмотра»	Включает функцию быстрого просмотра.
Подсвечивать заголовок активной страницы в окнах	Включает подсветку текущей закладки (в стиле MS Windows) в окнах с закладками. Например, в окнах **Переменные**, **Дамп**, **Сообщения**, **Анализатор выполнения/доступа**, и других.
Двойной клик на флажках и переключателях в диалогах	Задает функцию двойного щелчка мышью, эквивалентную одиночному щелчку на соответствующем элементе диалога и нажатию кнопки **OK** данного диалога.
Показывать ‘горячие клавиши' во всплывающих описаниях	Включает/выключает отображение комбинаций клавиш быстрого доступа во всплывающей подсказке для кнопок панелей инструментов.
Не выдавать диалоги сообщений, если открыто окно консоли	Направляет все сообщения в окно **Консоль сообщений**, если оно открыто. Если закрыто, то сообщение будет послано в отдельном окошке.

© 2009 Фитон, http://www.phyton.ru

Всегда выдавать диалоги сообщений	Отображает все сообщения в отдельных окошках. Окно **Консоль сообщений** также отображает эти сообщения.
Курсор помещается на кнопку ОК	Если флаг установлен, то в каждом открытом окне сообщения курсор автоматически помещается на кнопку **ОК** этого окна. Данную функцию можно выключить, если Вы предпочитаете нажимать клавишу **Enter**, а не щёлкать **ОК** мышью.
Звуковое уведомление для сообщений об ошибках	Включает звук для сообщений об ошибках. Информационные сообщения всегда подаются без звука.
Записывать сообщения в файл журнала	Задаёт имя файла журнала. Все сообщения заносятся в этот файл. Способ записи выбирается переключателем, у которого есть следующие позиции:
Начинать файл журнала сначала при каждом старте	Указывает создавать новый файл для каждого сеанса и удалять прежний файл, если он существует.
Записывать сообщения в конец файла	Указывает добавлять сообщения в конец существующего файла. При этом размер файла будет неограниченно расти.

3.1.5 Меню Команды

Команда	Описание
Проверить стертость	Запускается процедура проверки микросхемы на стертость. Существует ряд микросхем, которые не требуют стирания для записи данных. В этих микросхемах эта процедура может отсутствовать.
Запрограммировать микросхему	Запускается процедура программирования микросхемы
Проверить правильность программирования	Запускается процедура сравнения данных в микросхеме и буфере программатора.
Прочитать микросхему в буфер	Запускается процедура чтения микросхемы в буфер программатора.
Стереть микросхему	Запускается процедура стирания микросхемы. Существует ряд микросхем, которые не имеют процедуры электрического стирания. В этих микросхемах процедура стирания будет отсутствовать.
Автоматическое программирование	Запускается процедура автоматического программирования .
Локальное меню	Вызов локального меню активного окна.
Калькулятор	Открывает диалог **Калькулятор**.

3.1.5.1 Калькулятор Диалог

Этот диалог служит для вычисления выражений и преобразования величин из одной системы счисления в другую. Результат действий можно скопировать в буфер обмена.

Элемент диалога	Описание

© 2009 Фитон, http://www.phyton.ru

Выражение	Поле для ввода выражения или числа.
Копировать в	Задаёт формат, в котором результат будет скопирован в буфер обмена.
Знаковые значения	Указывает, что результат надо интерпретировать и отображать, как величину со знаком (действует только для десятичных чисел).
Отображать незначащие нули	Включает отображение лидирующих нулей (в старших разрядах) в двоичных и шестнадцатеричных числах.
Копировать	Копирует результат вычисления в буфер обмена в формате, заданном переключателем **Копировать в**.
Clr	Очищает поле **Выражение**.
Bs	Удаляет один символ (цифру) слева от точки ввода (Backspace).
0x	Вставляет «0x».
>>	Выполняет сдвиг результата выражения вправо на указанное количество разрядов.
<<	Выполняет сдвиг результата влево на указанное количество разрядов.
Mod	Вычисляет остаток деления на заданное число.

В то время как Вы печатаете выражение в поле **Выражение**, ChipProgLPT старается вычислить результат и сразу же отображает его в различных форматах в панели **Результат**. Также, переключатель и два флага в этой панели управляют форматом результата.

Можно присваивать значения переменным в программе и регистрам специального назначения, напечатав выражение с оператором присвоения. Например, если напечатать SP = 66h, то число 66h будет записано в SP.

Примеры выражений:

```
0x1234
-126
main + 33h
(float)(*ptr + R0)
101100b & 0xF
```

3.1.6 Меню Сценарии

В этом меню собраны пользовательские команды, которые запускают файлы сценариев (ФС). Для определения файлов сценария используйте диалог **Файлы Сценария**.

Чтобы добавить новую команду в это меню, поместите её файл сценария в текущую папку или в папку, в которую установлена программа ChipProgLPT. Первая непустая строка файла сценария должна содержать три символа косой чёрточки и текст, который надо показывать в меню **Сценарии**:

```
/// Название команды меню
```

Когда программа ChipProgLPT строит меню **Сценарии**, она просматривает текущую папку и свою инсталляционную папку на предмет всех файлов *.CMD, содержащих «///» в первой строке (напоминаем, что «//» обозначает начало однострочного комментария), и добавляет текст, следующий за «///», в меню **Сценарии**.

Когда вы выбираете пункт меню **Сценарии**, запускается соответствующий файл сценария.

Также смотрите Простой пример файла Сценария.

© 2009 Фитон, http://www.phyton.ru

Кнопка	Команда	Описание
	Запуск...	Открывает диалог **Файлы Сценария**..
	Новый исходный текст файла сценария	Создать новый исходный текст Файла Сценария.
	Открыть окно переменных	Открывает окно **Переменные**
	Добавить переменную (watch)	Добавить переменную в окно Переменные
	Окно редактора	Открывает окно **Редактора**
	Редактор текста	

3.1.6.1 Диалог Файлы Сценария

Этот диалог заведует файлами сценариев (ФС). Подробнее об этих файлах—в главе Файлы сценариев .

В верхней части диалога находится список загруженных файлов сценариев с указанием текущего состояния каждого файла.

Состояние файла	Описание
Stopped	Выполнение файла временно приостановлено.
Running	Файл выполняется.
Waiting	Файл ожидает появления события. Данное состояние инициируется вызовом определённых функций ожидания (например, Wait) в тексте ФС.
Cancelled	Выполнение ФС прекращено, но файл ещё не выгружен из памяти.

Выделенный ФС обозначен цветом. Чтобы выделить файл сценария, щёлкните его кнопкой мыши. Четыре кнопки справа от списка управляют выделенным ФС:

Кнопка	Описание
Завершить	Выгружает выделенный ФС, когда его можно выгрузить. В противном случае, устанавливает флаг Unload Request для выделенного ФС (у таких файлов состояние **Cancelled**). Следует иметь в виду, что выгрузить файл сценария можно не всегда.
Завершить все	Выгружает все файлы сценариев.
Перезапустить	Повторно запускает выделенный файл сценария.
Отлаживать	Включает режим Debugger для выделенного файла сценария. Эта команда останавливает выполнение ФС (или, если ФС находится в состоянии ожидания, то выполнение остановится, когда ФС выполнит возврат из функции ожидания) и открывает окно **Текст сценария** для этого ФС.

Когда Вы используете несколько файлов сценариев одновременно и выгружаете или повторно запускаете

© 2009 Фитон, http://www.phyton.ru

некоторые из них, следует помнить, что ФС могут сообща пользоваться глобальными данными и функциями. Если один из ФС обращается к данным или функциям другого ФС, а этот другой уже выгружен, то интерпретатор подаст сообщение об ошибке и первый ФС тоже будет выгружен.

Кнопки и поля в нижней части диалога управляют запуском файлов сценариев:

Элемент диалога	**Описание**
Имя файла сценария	Указывает имя файла сценария.
Обзор	Открывает диалог **Load/Execute Script File**, чтобы найти на диске файл сценария, имя которого надо поместить в поле **Имя файла сценария**.
Defines	Определяет текстовые переменные препроцессора для компиляции. Подробнее об этом—внизу этой главы.
Директории для поиска #include-файлов	Задаёт папки для поиска файлов, указанных в директивах **#include <file_name>** в файле сценария. Можно указать несколько папок через точку с запятой. Также, будет выполнен поиск в текущей папке.
Отладка (открыть окно исходного текста)	Когда флаг установлен, после загрузки файла сценария он не будет выполняться. Вместо этого, ChipProgLPT включает режим Debugger для этого файла, т.е., открывает окно **Текст сценария** для этого ФС. Также, смотрите Отладка файлов сценариев.
Автосохранение исходных текстов сценариев	Указывает автоматически сохранять все файлы исходного текста сценариев, открытые в окнах **Текст сценария** на момент нажатия кнопки **Запуск**.
Запуск	Запускает выполнение файла сценария, указанного в поле **Имя файла сценария** .

Текстовые переменные препроцессора

Поле **Defines** эквивалентно директиве #define. Например, если в этом поле Вы напечатаете **DEBUG**, результат будет такой же, как от директивы **#define DEBUG** , помещённой в первую строку исходного текста ФС.

Можно задавать значения переменных. Например, **DEBUG=3** эквивалентно **#define DEBUG 3**.

Можно перечислить несколько переменных в строке, разделив их точкой с запятой. Например:

```
DEBUG;Passes=3;Abort=No
```

Также, смотрите Предопределенные символы при компиляции Файлов Сценариев .

3.1.7 Меню Окна

Команды этого меню управляют расположением окон в приложении. Также, в нижней части меню есть список открытых на данный момент окон—это стандартный способ для переключения между ними. При выборе название окна в этом списке, оно активизируется на экране компьютера. Этот способ удобен для перехода к окну, расположенному позади остальных.

Команда	**Описание**
Расположить черепицей	Изменяет размеры и располагает окна без наложения друг на друга. При этом размеры окон примерно одинаковы.
Расположить черепицей горизонтально	Располагает все окна горизонтально, без наложения друг на друга. При этом размеры окон примерно одинаковы.
Расположить каскадно	Располагает окна уступом

© 2009 Фитон, http://www.phyton.ru

Упорядочить иконки	Выстраивает иконки свёрнутых окон.
Закрыть все окна	Закрывает все окна

3.1.8 Меню Справка

Команды этого меню работают с системой оперативной справки. Также, смотрите главу **Оперативные справки** .

Команды	**Описание**
Оглавление	Открывает закладку **Содержание** файла справки
Поиск	Открывает закладку **Предметный указатель** файла справки.
Таблицы соединений адаптеров	Открывает Таблицу соединений Адаптеров
Проверка наличия новых версий	Открывает диалог **Проверка наличия новых версий**
Послать письмо в Фитон по e-mail...	Можно послать письмо в Фитон по электронной почте.
О ChipProg...	

3.1.8.1 О ChipProgLPT

Этот диалог показывает:

- номер версии ChipProgLPT;
- версия Windows-оболочки ChipProgLPT;
- тип микросхемы;
- производитель микросхемы.

Пожалуйста, держите под рукой эту информацию, когда обращаетесь за технической поддержкой. Вполне возможно, что ваши проблемы уже решены, и вам нужно только свериться с новой версией ChipProgLPT, которая находится на вебсайте Фитона.

3.2 Окна

3.2.1 Интерфейс Пользователя Обзор

В программе ChipProgLPT используется стандартный интерфейс MS Windows с некоторыми полезными дополнениями:

1. В каждом специализированном окне есть локальное контекстное меню. Чтобы открыть его, нажмите правую клавишу мыши в пределах окна, или нажмите комбинацию клавиш **Ctrl+Enter** или **Ctrl+F10**. Каждой команде такого меню присвоена комбинация «горячих» клавиш **Ctrl+<буква>**. По нажатию этих клавиш в активном окне исполняется соответствующая команда.
2. У каждого специализированного окна есть своя панель инструментов. Обычно, у каждой команды локального меню есть своя кнопка на этой панели. Действие кнопок панели инструментов

© 2009 Фитон, http://www.phyton.ru

специализированного окна распространяется только на это конкретное окно. У главного окна программы ChipProgLPT есть несколько панелей инструментов, которые можно включать/ выключать (в меню **Конфигурация** , в диалоге **Опции экрана**, закладке **Линейка управления**).

3. У каждой кнопки панели инструментов есть всплывающая подсказка: если навести курсор на кнопку и задержать на пару секунд, появляется маленькое жёлтое окошко с кратким описанием этой кнопки.
4. В целях экономии места на экране у каждого окна можно отключить строку заголовка. Для этого используйте команду **Свойства** локального меню.
   При отсутствии строки заголовка специализированные окна можно распознавать по их содержимому и положению на экране (а также, по цвету и шрифту, когда они заданы отдельно для каждого окна). Чтобы переместить окно, его можно тащить за панель инструментов: поместите курсор на участке панели, свободном от кнопок, нажмите левую клавишу мыши и перетащите окно на новое место.
5. Можно открыть любое количество окон одного типа. Например, несколько окон **Дамп** или несколько окон **Переменные** .
6. Во всех диалогах с полями для ввода текста есть списки предыдущих введённых значений. Эти списки сохраняются между сеансами работы.
7. Во всех диалогах с полями для ввода текста работает функция автоматическое завершение имени.
8. Во всех диалогах с флажками (check boxes) и переключателями (radio buttons) двойной щелчок на флажке или переключателе производит такое же действие, как одиночный щелчок на этом флажке (переключателе) и нажатие кнопки **OK**. Это удобно, когда сразу после установки флажка (переключателя) Вы собираетесь закрыть этот диалог.

3.2.2 Окно Программирование

Окно **Программирование** является по существу основным окном программатора. Именно из этого окна осуществляется взаимодействие с микросхемой - чтение, запись, сравнение и т.п.

Окно содержит закладки:

закладка Программирование ,
закладка Опции.

3.2.2.1 Диалог Программирование

Диалог служит для определения параметров и этапов программирования.

Здесь же отображается информация об операциях.

Элементы диалога	**Описание**
Используемый буфер:	В поле **Буфер** представлен список открытых буферов и выбранный буфер, с которым будет взаимодействовать выбранная функция программирования.
Операции	Поле операций содержит меню функций взаимодействия с аппаратурой. Именно в этом поле содержатся функции,соответствующие установленному типу микросхемы.
Контроль на чистоту	Контроль ячеек на чистоту (стертость)

© 2009 Фитон, http://www.phyton.ru

Программирование	Программирование ячеек
Чтение	Чтение ячеек в буфер
Сравнение	Сравнение содержимого ячеек и буфера
Автоматическое программирование	Автоматическое программирование - это наиболее употребимый набор операций над микросхемой
Адреса	В этом поле расположены адреса, с которыми работают функции программирования.
Начальный в микросхеме:	Адрес в микросхеме, с которого начинается программирование
Конечный в микросхеме:	Адрес в микросхеме, с которым заканчивается программирование
Начальный в буфере:	Начальный адрес в буфере, с которого выбираются данные
Выполнить	Для активизации функции в поле операций необходимо выбрать функцию из списка и запустить ее либо двойным нажатием левой кнопки мыши, либо нажатием на кнопку **Выполнить**, либо клавишей Enter.
Повторы:	В поле определяется количество повторов выполнения функции.
Редакт. Авто...	Редактирование этого набора операций можно осуществлять, нажав кнопку.
Информация об операциях	Вся информация о выполнении операций с микросхемами, смене типа микросхем, а также загрузке/выгрузке файлов в буфера памяти помещается в поле.

3.2.2.1.1 Автоматическое Программирование

Автоматическое программирование - это набор процедур из списка операций окна Программирования .

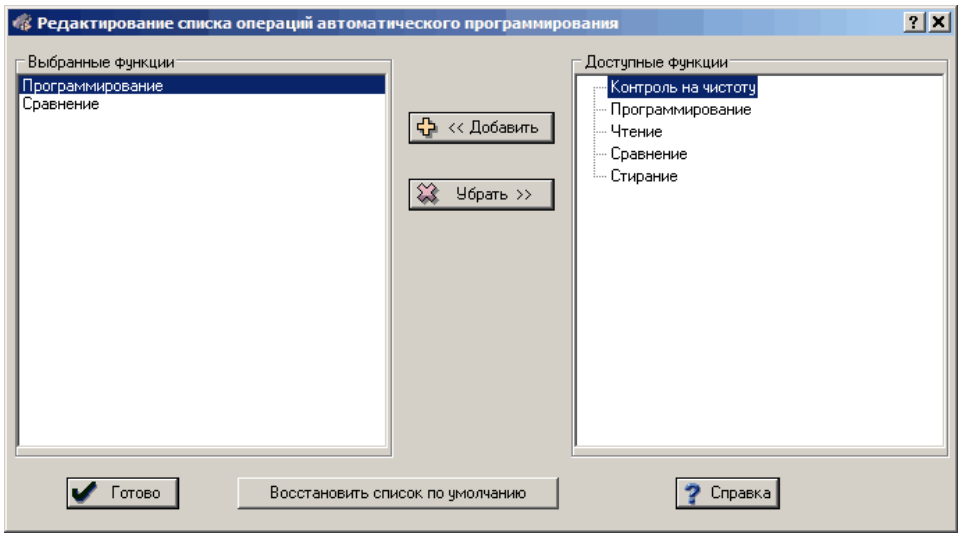
Каждая микросхема имеет список автоматического программирования по умолчанию. Обычно этот список содержит процедуры Стирания, Контролирование на стертость, Программирование, Сравнение, Установку защиты. Пользователь может редактировать список автоматического программирования по своему усмотрению.

В этом поле в древовидной форме представлен список операций с микросхемой. Последняя процедура в этом списке - это **Автоматическое программирование** . Эта процедура используется для вызова пакета функций, которые обычно выполняются с микросхемой. Каждая микросхема имеет список автоматического программирования по умолчанию. Обычно этот список содержит процедуры Стирания, Контролирования на стертость, Программирование, Сравнение, Установку защиты.

Если вы хотите изменить состав или порядок исполнения команд пакета используйте кнопку **Редакт.Авто...**, которая вызовет диалог **Редактирование списка операций автоматического программирования.**

Здесь можно отредактировать список команд при помощи кнопок **Добавить и Убрать.**

© 2009 Фитон, http://www.phyton.ru



Необходимые функции выбираются из поля **Доступные функции** и при помощи кнопки **Добавить** переносятся в поле **Выбранные функции**. Удаление происходит в обратном порядке: в поле **Выбранные функции** выбираются функции для удаления, затем кнопкой **Убрать** удаляются.

3.2.2.2 Диалог Опции программирования

В этой закладке помещены дополнительные опции программирования:

Элементы диалога	**Описание**
Чередование данных	Группа из семи радио кнопок в поле Чередования данных позволяет использовать восьмиразрядные микросхемы памяти в системах с 16- и 32-разрядной организацией памяти
Опции	
Тест присутствия микросхемы в колодке	При установленной опции, перед любой манипуляцией с микросхемой производится тестирование наличия микросхемы в колодке, при этом тестируется наличие контакта на каждом выводе микросхемы.
Проверять идентификатор микросхемы	При установленной опции, будет контролироваться внутренний идентификатор микросхемы.
Реверсировать порядок байт	При установленной опции при работе с микросхемой (программирование, чтение, сравнение) байты в шестнадцатибитном слове меняются местами. В дампе буфера данные местами не меняются и соответствуют загруженному файлу.).
Контроль чистоты перед программированием	При установленной опции каждое выполнение процедуры **Program** будет предварять процедура **Blank check**
Проверка после программирования	При установленной опции каждое выполнение процедуры **Program** будет заканчиваться процедурой **Verify**
Проверка после чтения	При установленной опции каждое выполнение процедуры **Read** будет заканчиваться процедурой **Verify**

3.2.2.2.1 Чередование данных

Группа из семи радио кнопок поля Чередования данных

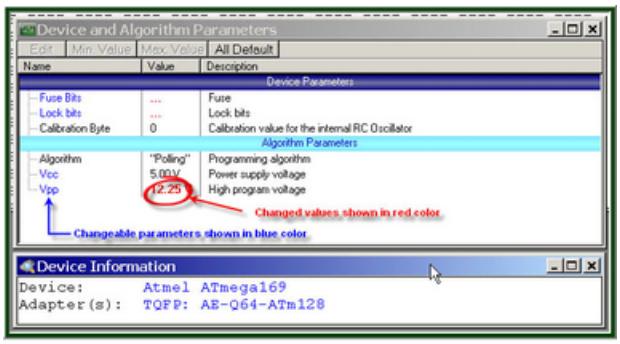
Элемент диалога	**Описание**

© 2009 Фитон, http://www.phyton.ru

Нет	Чередования данных нет. Буфер данных рассматривается как непрерывный массив данных.
Четный байт	Массив данных рассматривается как массив шестнадцатиразрядных слов, микросхема работает только с байтами, расположенными по **четным** адресам. Например, при чтении микросхемы в буфер нулевой байт микросхемы будет помещен в нулевой байт буфера, первый байт микросхемы - во второй байт буфера и т.д.
Нечетный байт	Массив данных рассматривается как массив шестнадцатиразрядных слов, микросхема работает только с байтами, расположенными по **нечетным** адресам. Например, при чтении микросхемы в буфер нулевой байт микросхемы будет помещен в первый байт буфера, первый байт микросхемы - в третий байт буфера и т.д.
Байт 0	Массив данных рассматривается как массив 32-разрядных слов, микросхема работает только с **нулевым байтом** 32-разрядного слова. Например, при чтении микросхемы в буфер нулевой байт микросхемы будет помещен в нулевой байт буфера, первый байт микросхемы - в четвертый байт буфера и т. д.
Байт 1	Массив данных рассматривается как массив 32-разрядных слов, микросхема работает только с **первым байтом** 32-разрядного слова. Например, при чтении микросхемы в буфер нулевой байт микросхемы будет помещен в первый байт буфера, первый байт микросхемы - в пятый байт буфера и т.д.
Байт 2	Массив данных рассматривается как массив 32-разрядных слов, микросхема работает только со **вторым байтом** 32-разрядного слова. Например, при чтении микросхемы в буфер нулевой байт микросхемы будет помещен во второй байт буфера, первый байт микросхемы - в шестой байт буфера и т.д.
Байт3	Массив данных рассматривается как массив 32-разрядных слов, микросхема работает только с **третьим байтом** 32-разрядного слова. Например, при чтении микросхемы в буфер нулевой байт микросхемы будет помещен в третий байт буфера, первый байт микросхемы - в седьмой байт буфера и т.д.

3.2.3 Окно Редактор Параметров Микросхемы и Алгоритма Программирования

В окне **Параметры Микросхемы и Алгоритма** представлена информация о параметрах микросхемы и параметрах алгоритма программирования.



Device and Algorithm parameters

Информация в этом окне разбита на две группы **Параметры Микросхемы** и **Параметры Алгоритма**. Эти группы разделены горизонтальными линиями голубого цвета.

Параметры Микросхемы	Это специфические для каждой микросхемы параметры. Обычно, эти параметры характерны для микроконтроллеров или микросхем FLASH памяти. Для микроконтроллеров в качестве этих параметров могут быть конфигурационные биты, значения калибрирующих констант, адреса старта микроконтроллеров и т.п. Для микросхем FLASH памяти в качестве этих параметров, обычно, выступают сектора микросхемы
Параметры Алгоритма	Это параметры алгоритма программирования выбранной микросхемы. В качестве параметров алгоритма могут выступать тип алгоритма и напряжения программирования.

ВАЖНО! Примечание: Любые изменения в этом окне **не вызывают** немедленные изменения в микросхеме. Установка параметров, сделанная в этом окне, только подготавливает конфигурацию программируемого устройства. Параметры микросхемы будут запрограммированы в микросхему только после активизации процедуры **Программирования** в списке **Параметры микросхемы** поля **Операции** окна Программирование .

Окно разделено на три вертикальных колонки:

1. **наименование параметра,**
2. **его значение,**
3. **краткое описание параметра.**

Наименование редактируемых параметров показано синим цветом, прочее - черным цветом; если пользователь изменил параметр, новое значение отображается красным цветом. Если значение велико для отображения, оно замещается тремя точками (‘…’). Если эти точки красные, значит, параметр был изменен.

Команды панели инструментов окна:

Кнопки панели	Описание
Edit	При помощи этой кнопки можно отредактировать параметр, который выбран активной линией. Чтобы отредактировать параметр, дважды щелкните левой кнопкой мыши по его имени, тогда появится Команда Edit. Для разных параметров вызываются разные Активные окна.
Min.Value	При помощи этой кнопки можно присвоить минимальное значение тому параметру, который выбран активной линией.
Max.Value	При помощи этой кнопки можно присвоить минимальное значение тому параметру, который выбран активной линией.
Default	При помощи этой кнопки можно присвоить значение по умолчанию тому параметру, который выбран активной линией.
All Default	При помощи этой кнопки можно присвоить значения по умолчанию всем параметрам.

3.2.3.1 Команда Edit

Редактирование параметров микросхемы и алгоритма программирования осуществляется в следующих форматах:

Параметры	Описание
Выпадающее меню	Редактирование в этом формате осуществляется в том случае, когда значение параметра выбирается из предопределенного списка.

© 2009 Фитон, http://www.phyton.ru

Диалог Check Box	Редактирование в этом формате осуществляется в том случае, когда при редактировании параметра предлагается включить/выключить опцию программирования.
Изменение значения параметра	Редактирование в этом формате осуществляется в том случае, когда при редактировании параметра программирования допускается изменить его значение. В заголовке диалога указывается допустимый диапазон изменения параметра. В этом формате, обычно, редактируются напряжения питания и программирования микросхем.

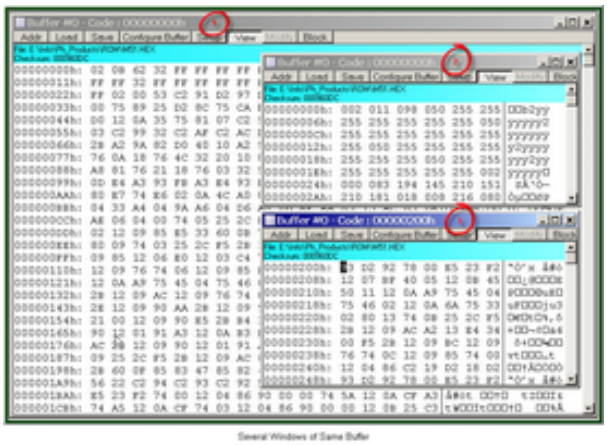
3.2.4 Окно Дамп Буфера

Окно **Дамп** отображает содержимое буфера памяти.

ChipProgLPT поддерживает гибкую структуру буфера:

- Вы можете создать бесконечное количество буферов. Количество буферов, которые вы можете открыть лимитировано только объемом RAM вашего компьютера.
- Каждый буфер имеет определенное количество подуровней, которое зависит от типа целевого устройства. Каждый подуровень связан с секцией адресного пространства целевого устройства. Например, для микроконтроллера Microchip PIC16F84 каждый буфер имеет три подуровня: 1) память кодов; 2) память данных EEPROM; 3) подуровень пользовательской идентификации.

Гибкая структура буфера позволяет простым образом манипулировать несколькими областями данных, которые располагаются в разных буферах. Чтобы открыть окно **Дамп Буфера** надо выбрать команду **Главное Меню > Просмотр > Дамп буфера.**



На рисунке изображены три открытых окна, которые представляют три части одного и того же буфера:

1) левое (самое большое окно) показывает содержимое буфера, начиная с адреса 0h;

2) второе окно показывает тоже содержание буфера с того же адреса, но данные представлены в десятичном формате;

3) третье окно показывает содержимое буфера, начиная с адреса 200h.

В каждом окне слева находится поле адреса. Значение адреса соответствует адресу первой ячейки строки. Адресация всегда увеличивается на один байт: 0, 1, 2…. Написание адреса заканчивается двоеточием (:). Когда вы изменяете размер окна, адреса автоматически изменяются в зависимости от количества байт данных в строке. Некоторые окна могут иметь справа поле, в котором данные представляются в ASCII формате. Сверху на панели инструментов перечислены команды, которыми можно пользоваться при работе с окном. Для этих же целей можно вызвать локальное меню, кликнув правой кнопкой мыши по окну.

Локальное меню

Локальное меню окна содержит следующие команды, для каждой команды имеется кнопка панели инструментов окна:

© 2009 Фитон, http://www.phyton.ru

Команда	Описание
Новый адрес...	Открывает диалог **Отобразить с адреса**. Можно ввести число
Загрузить файл в буфер...	Открывает диалог **Загрузить Файл**
Сохранить данные в файле...	Открывает диалог **Сохранить Файл**
Конфигурация буфера...	Открывает диалог **Конфигурация буфера**.
Опции отображения...	Открывает диалог **Настройки окна дампа**.
Только просмотр, редактирование запрещено	Если установить эту опцию, то редактирование в поле значений запрещается. Аналогично действует кнопка View, которая в этом случае будет утоплена. Если опция не установлена, в поле значений разрешено редактирование.
Изменить значение или начальный адрес	Открывает диалог **Отобразить с адреса** или Открывает диалог **Изменить значение**
Операции с блоком памяти	Открывает диалог **Операции с блоком памяти**, для действий с участками памяти.
Перемещение между полями	Позволяет переходить курсору из правого поля в левое и наоборот

3.2.4.1 Диалог Отобразить с адреса

Используйте этот диалог, чтобы задать новый адрес, начиная с которого следует отображать исходный текст в окне **Дамп**.

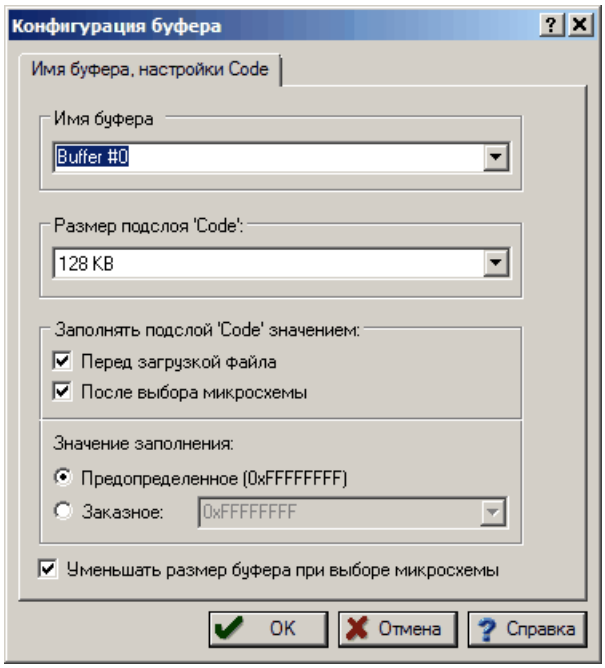
Элемент диалога	Описание
Введите новый адрес	Задаёт новый адрес.
История ввода	Список предыдущих наборов

3.2.4.2 Диалог Конфигурация буфера

Диалог **Конфигурация буфера** позволяет сконфигурировать буферы и подслои в удобной для вас форме.

По умолчанию в поле **Имя буфера** устанавливается ‘Buffer #0’. Следующий буфер будет называться ‘Buffer #1’ и так далее.

© 2009 Фитон, http://www.phyton.ru



Для того чтобы вызвать диалог **Конфигурация буфера**, нажмите кнопку Configure Buffer на панели инструментов окна **Дамп Буфера**. Для этих целей можно использовать локальное меню.

По умолчанию каждый буфер имеет минимальный размер 128K RAM и заполняется предопределенным значением (обычно 0FFh). Вы можете установить эти данные в соответствии со своими требованиями.

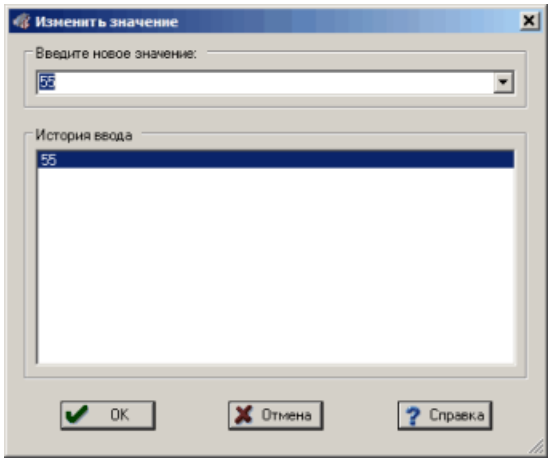
В этом диалоге вы можете ввести имя буфера по вашему усмотрению - для этого достаточно просто ввести имя в поле Имя буфера.

Определите значения заполнения буфера при инициализации: либо предопределенное, либо заказное.

3.2.4.3 Диалог Изменить Значение или Начальный Адрес

Изменить адрес

Для того чтобы изменить стартовый адрес, нажмите кнопку Addr на панели инструментов окна **Дамп Буфера.** Для этих целей можно использовать локальное меню. Затем надо ввести значение в поле **Введите новый адрес** или выбрать его из поля **Историю ввода.**



© 2009 Фитон, http://www.phyton.ru

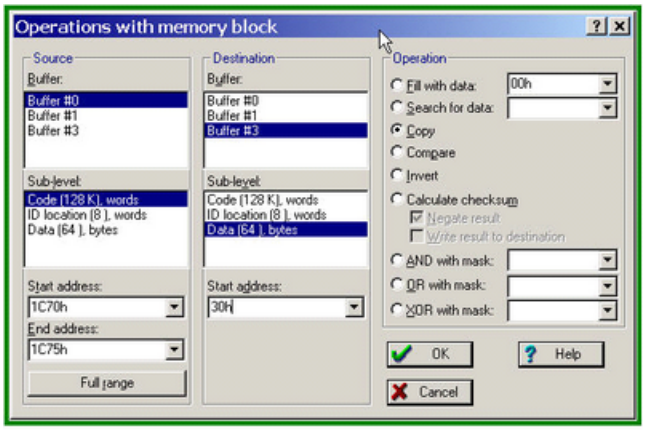
Изменить значение

Для того чтобы изменить значение ячейки памяти, нажмите кнопку Modify на панели инструментов окна **Дамп Буфера.** Для этих целей можно использовать локальное меню. Затем надо ввести значение в поле **Введите новое значение** или выбрать его из поля **Историю ввода.**

3.2.4.4 Операции с Блоками Памяти

Этот диалог выполняет действия с содержимым участков памяти. Программа ChipProgLPT обеспечивает выполнение сложных операций с блоками памяти.

Панель **Исходное адресное пространство** задаёт участок памяти с начальными данными для операции. Результат операции будет помещён в участок памяти, заданный в панели **Адресное пространство назначения** . По умолчанию, второй участок совпадает с первым участком. Если операция не использует участок памяти для результата, например, при Поиске и Замене, то параметры этой панели будут недоступны.



Operations with Memory Blocks

Элемент диалога	Описание
Начальный адрес	Начальный адрес участка памяти.
Конечный адрес	Конечный адрес участка памяти, включительно. Конечный адрес участка памяти определяется по размеру исходного участка.
Весь диапазон	Задаёт начальный и конечный адреса равными началу и концу всего адресного пространства памяти данного типа.

Диалог выполняет следующие операции. Каждая операция начинает выполняться, после того как вы нажмете кнопку **OK** .

Операция	Описание
Заполнить значением	Заполнить буфер значением или последовательностью значений, заданными в поле справа.
Поиск данных	Искать в участке с начальными данными значение или последовательность значений, заданную в поле справа.
Копировать	Копировать данные из участка памяти в другой участок. Можно скопировать блок в пределах памяти одного и того же типа, или в память другого типа.

© 2009 Фитон, http://www.phyton.ru

Сравнить	Сравнить содержимое двух участков памяти. Если есть различия, то появится окошко сообщения с адресами ячеек и значениями, которые не совпадают и вопросом о продолжении операции сравнения.
Инвертировать	Инвертировать содержимое участка памяти побитно.
Вычислить контрольную сумму	Вычислить контрольную сумму данных в исходном участке памяти. Смотрите примечание внизу.
Вычесть результат из нуля	Вычесть полученную контрольную сумму из нуля.
Записать результат в память	Записать 32-битный результат вычисления контрольной суммы по адресу, заданному в поле **Начальный адрес** для участка назначения. Если этот флаг снят, то ChipProgLPT только покажет контрольную сумму в виде сообщения.
AND со значением	Для заданного участка памяти выполнить побитно операцию И с операндом из текстового поля справа. Смотрите примечания внизу.
OR со значением	Для заданного участка памяти выполнить побитно операцию ИЛИ с операндом из текстового поля справа..
XOR со значением	Для заданного участка памяти выполнить побитно операцию ИСКЛЮЧАЮЩЕЕ ИЛИ с операндом из текстового поля справа.

Примечания

1. Исходный участок и участок назначения могут перекрываться: операции выполняются при помощи временного промежуточного буфера.
2. Команды **Копировать** и **Сравнить** используют участки, заданные в панелях **Исходное адресное пространство** и **Адресное пространство назначения** .
3. Контрольная сумма вычисляется как 32-битное число простым суммированием. Если данное адресное пространство организовано по байтам, то будут суммироваться 8-битные значения. Если оно организовано по словам, тогда суммироваться будут 16-битные значения.
4. Логические операции (И, ИЛИ, ИСКЛЮЧАЮЩЕЕ ИЛИ) выполняются над данными из участка, заданного в **Исходное адресное пространство** , а результат операции записывает в участок, заданный в **Адресное пространство назначения** . При этом размер операнда, указанного для операции, будет равным размеру единицы памяти, т.е., будет соответствовать организации памяти на устройстве пользователя.(16-бит для памяти **Prog, Data16, Reg** и **Stack**, 8-бит для памяти **Data8**).
5. Есть возможность выполнять более сложные действия с блоками памяти, с помощью файлов сценария.

3.2.4.5 Диалог Настройки окна Дамп

Диалог управляет параметрами отображения данных в окне **Дамп Буфера**.

Диалог **Настройки окна Дамп** можно вызвать из локального меню командой **Опции отображения** или кнопкой **Setup** на панели инструментов.

Элементы диалога	**Описание**
Буфер:	В поле **Буфер** представлен список открытых буферов и выбранный буфер, с которым будет взаимодействовать выбранная функция программирования.
Формат отображения	Группа из трех радио кнопок, которая позволяет выбрать формат отображения данных: двоичный, шестнадцатеричный или десятичный.
Отображать данные как:	Группа из четырех радио кнопок, которая позволяет выбрать размер отображения данных: байты, слова(16бит), двойные слова(32 бита), четверные слова(64 бита).

© 2009 Фитон, http://www.phyton.ru

Непечатаемые символы ASCII	
Заменять символы с кодами 0x00...0x20	Если выбрана эта опция, то символы из представленного ряда будут заменяться на '.' или пробел. Один из этих символов надо выбрать из двух радио кнопок **Заменять на:**
Заменять символы с кодами 0x80...0xFF	Если выбрана эта опция, то символы из представленного ряда будут заменяться на '.' или пробел. Один из этих символов надо выбрать из двух радио кнопок **Заменять на:**
Опции	Набор опций для удобства отображения данных.
Панель ASCII-кодов	Открывает или закрывает поле с правой стороны окна, в котором информация отображается в ASCII-кодах.
Отображать контрольную сумму	Если выбрана эта опция, то появляется информация о контрольной сумме в верхней части окна под панелью инструментов.
Ограничить размер дампа размером подслоя	Если выбрана эта опция, то будет отображаться кусок памяти размером в подслой.
Отображать десятичные и шестнадцатеричные данные как знаковые	Если выбрана эта опция, то данные будут отображаться как знаковые (старший бит=1 - отрицательное число; старший бит=0 - положительное число).
Всегда отображать '+' или '-'	Эта опция зависима от предыдущей опции. Если выбраны две эти опции, то будет отображаться и знак '+' и знак '-'.
Отображать лидирующие нули в десятичных числах	Если выбрана эта опция, то отображаются незначащие нули вначале десятичных чисел.
Реверсировать порядок байт в словах (сначала LSB)	Если выбрана эта опция, то порядок байт в словах будет реверсирован.
Реверсировать порядок слов в двойных словах	Если выбрана эта опция, то порядок слов в двойных словах будет реверсирован.
Реверсировать порядок двойных слов в четверных словах	Если выбрана эта опция, то порядок двойных слов в четверных словах будет реверсирован.

3.2.4.6 Диалог Загрузить Файл

Диалог определяет параметры файла для загрузки в Буфер.

Элементы диалога	**Описание**
Имя файла:	Поле, в котором надо написать имя файла с путем или найти нужный файл по кнопке **Обзор.**
Формат файла	Девять радио кнопок для выбора **Формата Файла**.
Буфер загрузки:	В поле **Буфер** представлен список открытых буферов и выбранный буфер, с которым будет взаимодействовать выбранная функция программирования.
Подслой загрузки:	Загрузить данные в подслой буфера
Адрес загрузки двоичного файла:	В этом поле определяют начальный адрес с которого будет записываться загружаемый файл

© 2009 Фитон, http://www.phyton.ru

Смещение адресов загрузки:	Файл загружается в адреса, указанные в файле плюс смещение, указанное в этом поле. Смещение может быть как положительным, так и отрицательным.

3.2.4.7 Диалог Сохранить Файл из буфера

Диалог определяет параметры файла для сохранения из Буфера.

Элементы диалога	Описание
Имя файла:	Поле, в котором надо написать имя файла с путем или найти нужный файл по кнопке **Обзор.**
Адреса сохраняемой области	**Начальный** и **Конечный** адреса определяют область, которую надо сохранить. Если надо сохранить все адреса используйте кнопку **Все**.
Формат файла	Девять радио кнопок для выбора **Формата Файла.**
Буфер, откуда сохранить файл	В поле **Буфер** представлен список открытых буферов и выбранный буфер, с которым будет взаимодействовать выбранная функция программирования.
Подслой, откуда сохранить файл	Сохранить данные из подслоя буфера

3.2.4.8 Форматы Файлов

Программатор поддерживает следующие форматы файлов:

Название	Описание
Standard/Extended Intel HEX (*.hex)	Стандартный формат фирмы Intel. HEX-файл представляет собой текстовый файл, содержащий загрузочные адреса данных, соответствующие данные, контрольные суммы и некоторую другую специальную информацию. Поддерживается как стандартный, так и расширенный формат файла.
Двоичный образ памяти (*.bin)	Двоичный образ памяти. Чтобы загрузить двоичный файл, нужно указать начальный адрес загрузки.
Motorola S-record (*.hex, *.s, *.mot)	HEX формат фирмы Motorola. HEX-файл представляет собой текстовый файл, содержащий загрузочные адреса данных, соответствующие данные, контрольные суммы и некоторую другую специальную информацию. Поддерживается весь спектр форматов Motorola S-record.
POF (*.pof)	Формат данных фирмы Altera. POF-файл представляет собой специальный файл, содержащий загрузочные адреса данных, соответствующие данные, контрольные суммы и некоторую другую специальную информацию.
JEDEC (*.jed)	Формат данных для программирования PLD (программируемых логических матриц). Этот формат является наиболее распространенным форматом для программирования PLD. JEDEC-файл представляет собой специальный файл, содержащий загрузочные адреса данных, соответствующие данные, информацию о тест-векторах и разнообразную специальную информацию.
PRG (*.prg)	Специальный формат фирмы Xilinx. Этот формат используется для программирования микросхем PLD (программируемых логических матриц) фирмы Xilinx. PRG-файл представляет собой специальный файл, содержащий загрузочные адреса данных, соответствующие данные, контрольные суммы и некоторую другую специальную информацию.

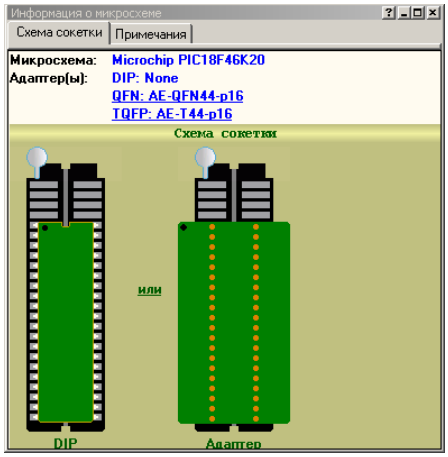
© 2009 Фитон, http://www.phyton.ru

Holtek OTR (*.otp)	Формат данных фирмы Holtek. Этот OTP-файл представляет собой специальный файл, содержащий загрузочные адреса данных, соответствующие данные, контрольные суммы и некоторую другую специальную информацию.
Angstrem SAV (*.sav)	Формат данных фирмы Angstrem. Этот SAV-файл представляет собой специальный файл, содержащий загрузочные адреса данных, соответствующие данные, контрольные суммы и некоторую другую специальную информацию.
ASCII Hex (*.txt)	Этот TXT-файл представляет собой текстовый файл, содержащий загрузочные адреса данных, соответствующие данные, контрольные суммы и некоторую другую специальную информацию.

3.2.5 Окно Информация о Микросхеме

В этом окне **Информация о Микросхеме** помещена информация о производителе и типе микросхемы. Если для работы с микросхемой требуется дополнительный адаптер, в этом окне появится сообщение, в котором будет указано название этого адаптера. Например, на рисунке ниже показаны два адаптера для выбранного микроконтроллера PIC18F46K20 - 'AE-QFN44-p16' (корпус QFN-44) и 'AE-T44-p16' (корпус TQFP-44). На картинке в нижней части окна схематично показано, как нужно устанавливать микросхему или адаптер в сокетку программатора.

Если кликнуть по адаптеру в этом окне, то открывается файл adapters.chm с таблицей соединений этого адаптера. При программировании в устройстве пользователя нужно посмотреть таблицу подключения к адаптеру.



3.2.5.1 Таблица соединений адаптера.

Таблица соединений всех адаптеров представлена в файле adapters.chm.

Формат таблиц:

Первая колонка	Разъем, устанавливаемый в программатор
Вторая колонка	Разъем (сокетка), в которую устанавливается микросхема
Третья и далее колонки	Дополнительные активные и пассивные элементы (если есть)
Строки таблицы	Соединения компонентов адаптера.

В качестве примера представим таблицу для адаптера AE-P20U.

DIP-20	PLCC-20
1	1

© 2009 Фитон, http://www.phyton.ru

2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	10
11	11
12	12
13	13
14	14
15	15
16	16
17	17
18	18
19	19
20	20

Первая колонка таблицы - это разъем, устанавливающийся в программатор, правая - разъем (сокетка), в которую устанавливается микросхема.

В нашем примере: со стороны программатора - разъем DIP-20, микросхема устанавливается в ZIF сокетку PLCC-20.

Строки таблицы символизируют соединения: первый вывод разъема DIP-20 соединен с первым выводом сокетки PLCC-20, второй - со вторым и т.д.

Бывают более сложные адаптеры, на которых установлены активные и пассивные элементы, но основной принцип описания адаптера остается прежним.

3.2.5.2 Таблица подключения к адаптеру.

Для адаптеров, используемых при программировании в устройстве пользователя, указаны таблицы подключения адаптера к

Формат таблицы:

Первая колонка Разъем, установленный на кабеле адаптера, который подключается к программируемой микросхеме

Вторая колонка Выводы микросхемы, которые должны быть соединены с кабелем адаптера.

Строки таблицы Соединения кабеля адаптера и программируемой микросхемы.

В качестве примера представим таблицу соединений микросхемы Zilog Z8Fxxx к адаптеру AE-ISP-U1.

BH10	Z8Fxxxx
1	Vcc
2	RESET
3	GND
4	DBG
5	GND
6	
7	
8	
9	
10	

© 2009 Фитон, http://www.phyton.ru

Как видно из таблицы подсоединяется всего 5 выводов: первый вывод разъема адаптера соединяется с выводом VCC микросхемы, второй - с выводом RESET и т.д.

3.2.6 Окно Консоль Сообщений

Окно Консоли отображает сообщения, которые ChipProgLPT выдает пользователю: сообщения об ошибках и информационные.

Если окно **Консоль сообщений** не открыто, то каждое сообщение будет отображаться в небольшом индивидуальном диалоговом окне. Если Вы откроете окно **Консоль сообщений**, то вместо упомянутых индивидуальных диалоговых окон все последующие сообщения будут попадать в это окно **Консоль сообщений**. Использовать окно **Консоль сообщений** удобнее, потому что:

- Можно просмотреть последние 256 сообщений и получить справку для каждого из них.
- Сообщения не требуют какого-либо Вашего отклика или реакции. Они собираются их в памяти PC, откуда могут быть извлечены в любой момент.

Окно должно быть достаточно большим, чтобы вместить несколько сообщений. Можно сохранить содержимое окна сообщений в отдельном файле. Для этого нужно поставить флаг и определить название файла в **Конфигурация>Опции Экрана>** закладка **Прочие**,

Окно **Консоль сообщений** сохраняет последние 256 сообщений даже тогда, когда оно закрыто. Его можно открыть в любой момент и посмотреть сообщения.

Выбранное сообщение выделено заданным цветом фона. Чтобы выбрать другое сообщение, щёлкните его кнопкой мыши или используйте клавиши управления курсором.

Локальное меню

Локальное меню окна содержит следующие команды, для каждой команды имеется кнопка панели инструментов окна:

Команда	Описание
Очистить окно	Удаляет из окна все сообщения.
Справка по сообщению	Открывает окно справки для выбранного сообщения.

3.2.7 Окно Редактора

Основное назначение окна **Редактора**—отображать и редактировать исходный текст программы.

Кнопка	Команда	Описание
	Новый файл	Открывает окно для записи нового файла.
	Открыть...	Открывает диалог **Открыть файл.**
	Сохранить	Сохраняет открытый файл.
	Сохранить как	Открывает окно **Сохранить файл как,** в котором можно определить имя и путь сохраняемого файла.
	Печать	Выводит файл на печать.
	Свойства	Открывает окно **Свойства:** . В окне имеется три закладки: закладка **Общие**, закладка **Безопасность** , закладка **Подробно**.

© 2009 Фитон, http://www.phyton.ru

3.2.7.1 Редактор Текста

Команды этого меню действуют в отношении активного в данный момент окна **Редактора**.

Кнопка	Команда	Описание
	Откат	Отменяет последнюю операцию редактирования текста в этом окне. Количество шагов, которое охватывает функция отмены действия (Undo).
	Сохранить файл на диске	Сохраняет текущий файл.
	Сохранить файл как...	Открывает диалог **Сохранить файл как...**
	Печать	Выводит файл на печать
	Скопировать	Копирует выделенный блок в буфер обмена. Формат текста в буфере стандартный, и этот блок доступен другим программам.
	Удалить	Удаляет выделенный блок и помещает его в буфер обмена.
	Вставить блок	Копирует блок из буфера обмена, начиная с курсора.
	Поиск текста	Открывает диалог **Поиск текста**.
	Повторить поиск	Повторяет поиск с теми же параметрами, которые были заданы для предыдущего поиска.
	Поиск/замена текста	Открывает диалог **Поиск/замена текста**.
	Показать результаты поиска по файлам	Повторно показывает результаты поиска в нескольких файлах в диалоге **Результаты поиска по файлам.**
	Перейти на строку с номером	Открывает диалог **Отобразить с новой строки**. Исходный текст будет показан, начиная с указанной строки.
	Поставить закладку	Открывает диалог **Установить закладку** для местных закладок
	Восстановить закладку	Открывает диалог **Восстановить закладку** для местных закладок.
	Режим сжатого текста	Переключает **Режим сжатого текста.** Режим сжатого текста предназначен для отображения только строк, удовлетворяющих некоторому заданному пользователем критерию.
	Параметры режима сжатого текста	Открывает диалог **Параметры режима сжатого текста.**
	Найти пару для скобки/ комментария	Находит парную скобку или метку комментария для скобки или метки в позиции курсора. Если пара найдена, то курсор переместится к ней. Скобки могут быть: круглыми **(** или **)**, квадратными **[** или **]** и фигурными **{** или **}**. Метки могут быть **/*** или ***/**.
	Восстановить контекст редактирования	Активизирует окно **Редактора** предыдущего редактирования и устанавливает курсор вместо последнего редактирования.

© 2009 Фитон, http://www.phyton.ru

Во время отладки обычно требуются частые переключения между несколькими файлами. Эта команда поможет Вам быстро перейти к месту последнего редактирования.

Скрипты пользователя

3.2.7.1.1 Диалог Поиск текста

Диалог задаёт параметры поиска текстовой фразы в файлах. Одноимённые параметры в этом диалоге и диалоге **Поиск/замена текста** эквивалентны. Можно указывать имена файлов с одним или несколькими символами маски. Вместе с именем файла можно указывать путь. Также, можно выполнять поиск сразу в нескольких файлах, используя параметры в панели **Поиск в файлах**.

Элемент диалога	Описание
Строка для поиска	Задаёт фрагмент текста для поиска (искомую строку).
Учитывать регистр	Включает проверку совпадения регистра букв. По умолчанию выключен.
Только целые слова	Искать только целые слова: строка будет считаться найденной, только если она находится между знаками препинания или разделительными знаками (пробелами, символами табуляции, запятыми, кавычками, и т.д.). По умолчанию выключен.
Использовать выражения	Указывает, что искомая строка является выражением поиска. По умолчанию выключен.
Вся область	Искать во всем файле. По умолчанию включен.
Только выделенный текст	Искать только в выделенном тексте.
С позиции курсора	Искать, начиная с текущей позиции курсора
Во всей области поиска	Искать от начала или конца файла (в зависимости от направления поиска). Включена по умолчанию.
Выполнить поиск в файлах	Включает поиск в нескольких файлах проекта (смотрите примечания внизу). Если флаг снят, то поиск будет только в активном окне **Редактора**.
Искать во всех исходных файлах проекта	Искать во всех исходных файлах, включённых в проект.
Искать в файлах зависимостей	Искать во всех исходных файлах, включённых в проект, и всех файлах, от которых исходные файлы зависят явно или неявно. Для языка Си - это заголовочные файлы (*.h).
Маски файлов	Содержит маску или маски имён файлов, в которых выполнять поиск. Если масок несколько, их надо разделить точками с запятой. Чтобы задать длинные имена, принятые в Windows, кавычек ставить не нужно. Пример: `*.txt;*.c;c:\prog\*.h`. Эта опция и **Искать во всех исходных файлах проекта** действуют независимо друг от друга: можно искать во всех файлах проекта. И в файлах, чьи имена удовлетворяют заданной маске (маскам).
Искать в поддиректориях	Искать в поддиректориях всех папок, заданных масками и опцией **Искать во**

© 2009 Фитон, http://www.phyton.ru

всех исходных файлах проекта.

Начальный путь — Начать поиск с заданной здесь папки. Эта папка служит общим путём и удобна, когда надо подставить несколько масок следующего вида:

`c:\prog\text\source\*.txt;c:\prog\text\source\*.doc`

В таком случае, используйте маски (`*.txt;*.doc`) и общий путь (`c:\prog\text\source`).

Примечания

1. Если задан поиск в файле, открытом в окне **Редактора**, то поиск будет выполнен в буфере окна, а не в файле на диске.
2. Если задан поиск в нескольких файлах, то поиск выполняется во всех исходных файлах проекта. По окончании поиска открывается диалог **Результаты поиска по файлам.**

3.2.7.1.2 Диалог Поиск/Замена текста

Диалог задаёт параметры поиска текстовой фразы в файлах. Одноимённые параметры в этом диалоге и диалоге **Поиск/замена текста** эквивалентны. Можно указывать имена файлов с одним или несколькими символами маски. Вместе с именем файла можно указывать путь. Также, можно выполнять поиск сразу в нескольких файлах, используя параметры в панели **Поиск в файлах**.

Элемент диалога	Описание
Строка для поиска	Задаёт фрагмент текста для поиска (искомую строку).
Заменить на	Задаёт фрагмент текста, который будет подставлен вместо найденного.
Учитывать регистр	Включает проверку совпадения регистра букв. По умолчанию выключен.
Только целые слова	Искать только целые слова: строка будет считаться найденной, только если она находится между знаками препинания или разделительными знаками (пробелами, символами табуляции, запятыми, кавычками, и т.д.). По умолчанию выключен.
Использовать выражения	Указывает, что искомая строка является выражением поиска. По умолчанию выключен.
Подтверждение замены	Указывает открывать диалог **Подтвердите замену** для подтверждения замены данного найденного фрагмента текста. Включён по умолчанию.
Вся область	Искать во всем файле. По умолчанию включен.
Только выделенный текст	Искать только в выделенном тексте.
С позиции курсора	Искать, начиная с текущей позиции курсора
Во всей области поиска	Искать от начала или конца файла (в зависимости от направления поиска). Включена по умолчанию.
Выполнить поиск в файлах	Включает поиск в нескольких файлах проекта (смотрите примечания внизу). Если флаг снят, то поиск будет только в активном окне **Редактора**.
Искать во всех исходных файлах проекта	Искать во всех исходных файлах, включённых в проект.

© 2009 Фитон, http://www.phyton.ru

Искать в файлах зависимостей	Искать во всех исходных файлах, включённых в проект, и всех файлах, от которых исходные файлы зависят явно или неявно. Для языка Си - это заголовочные файлы (*.h).
Маски файлов	Содержит маску или маски имён файлов, в которых выполнять поиск. Если масок несколько, их надо разделить точками с запятой. Чтобы задать длинные имена, принятые в Windows, кавычек ставить не нужно. Пример: `*.txt;*.c;c:\prog\*.h`. Эта опция и **Искать во всех исходных файлах проекта** действуют независимо друг от друга: можно искать во всех файлах проекта. И в файлах, чьи имена удовлетворяют заданной маске (маскам).
Искать в поддиректориях	Искать в поддиректориях всех папок, заданных масками и опцией **Искать во всех исходных файлах проекта**.
Начальный путь	Начать поиск с заданной здесь папки. Эта папка служит общим путём и удобна, когда надо подставить несколько масок следующего вида: `c:\prog\text\source\*.txt;c:\prog\text\source\*.doc` В таком случае, используйте маски (`*.txt;*.doc`) и общий путь (`c:\prog\text\source`).
Поиск	Заменить первый найденный экземпляр искомой строки.
Заменить все	Заменить все найденные экземпляры искомой строки.

Примечания

1. Если задан поиск в файле, открытом в окне **Редактора**, то поиск будет выполнен в буфере окна, а не в файле на диске.
2. Если задан поиск в нескольких файлах, то поиск выполняется во всех исходных файлах проекта. По окончании поиска открывается диалог **Результат поиска по файлам.**

3.2.7.1.3 Выражения поиска

Редактор текста поддерживает так называемые «выражения поиска», которые можно использовать для особых текстовых строк. Выражения поиска содержат управляющие символы в искомой текстовой строке:

? Означает один любой символ в этой позиции. Пример: если задать ?ell в качестве искомой строки, то будут найдены слова «bell», «tell», «cell», и т.д.

% Означает начало строки. Символы, следующие за «%» должны начинаться с позиции 1. Пример: %Counter —найти слово «Counter», которое начинается с первой позиции в строке.

**$** Конец строки. Символы, предшествующие «$», должны находиться в последних позициях в строке. Пример: Counter$ —найти слово «Counter» в конце строк.

@ Найти непосредственно символ; «@» позволяет задавать управляющие символы, как обычные буквы. Пример: @? –найти знак вопроса.

\xNN Шестнадцатеричное значение символа. Пример: \xA7 —найти символ, шестнадцатеричный код которого равен A7.

+ Неопределённое количество единиц предыдущего символа. Например, если указать 1T+2, то редактор найдёт строки, содержащие «1», следом за которой идёт «2», между которыми находится любое количество букв T.

[c1-c2] Равенство любому символу в интервале от c1 до c2. Пример: [A-Z] означает любую букву от A до Z

[~c1-c2] Равенство любому символу, не лежащему в интервале от c1 до c2, т.е., от 0 до c1–1 или от c2+1 до 255. Пример: [~A-Z] означает любой символ, кроме букв верхнего регистра.

text1|text2 Символ «|» обозначает логическое «ИЛИ» и редактор будет искать **text1** или **text2**. Пример: LPT|COM|CON —найти «LPT», или «COM», или «CON».

Не забудьте, что чтобы воспользоваться выражениями поиска, нужно в диалоге установить соответствующий

© 2009 Фитон, http://www.phyton.ru

флаг.

3.2.7.1.4 Окно Переменные с автопросмотром

Во время отладки программы, это окно автоматически отображает имена и значения переменных, которые в данный момент видны в окне **Редактора**.

Окно **Переменные с автопросмотром** поделено красной горизонтальной линией на две части, верхнюю и нижнюю. В каждый момент времени в окне **Редактора** есть строка, соответствующая текущему значению счётчика команд микроконтроллера. Эта строка выделена голубым цветом (по умолчанию). Имена переменных, находящихся в этой строке, и их текущие значения отображаются в верхней части окна **Переменные с автопросмотром**. Ниже красной линии, окно отображает списки: (a) параметры локальных переменных и параметры текущей функции; (b) прочие переменные, видимые в окне **Редактора**.

Содержимое окна **Переменные с автопросмотром** автоматически изменяется при перемещении в окне **Редактора**.

В отношении объекта, выделенного в окне, можно выполнить любую команду локального меню этого окна. Выделенный объект обозначен цветом фона (чёрным по умолчанию). Чтобы выделить объект, щёлкните его кнопкой мыши или используйте клавиши управления курсором.

Локальное меню

Локальное меню окна содержит следующие команды, у большинства команд есть кнопка на панели инструментов окна:

Команда	Описание
Изменить значение	Открывает диалог **Изменить значение** чтобы задать новое значение для выделенной переменной. Это же самое можно сделать, просто начав печатать новое значение на клавиатуре.
Скопировать в окно Переменных	Копирует выделенную переменную в окно **Переменные**.
Формат отображения	Переключает формат представления всех значений в окне между шестнадцатеричным, десятичным, двоичным и ASCII.
Не показывать функции и нетипизированные объекты	Установленный флаг выключает отображение имён и адресов функций и нетипизированных объектов. По умолчанию флаг установлен.

3.2.7.1.4.1 Настройки панели автопросмотра

Этот диалог управляет параметрами отображения выделенной переменной в окне **Переменные с автопросмотром** .

Элементы диалога	Описание
Формат отображения	Указывает формат представления для выделенного выражения: двоичный, шестнадцатеричный, десятичный или ASCII.
Всплывающее описание	Включает всплывающие описания для регистров специального назначения
Отображать битовую схему	Включает всплывающие описания битовой схемы регистров специального назначения, если они есть.
Отображать описания битов	Включает всплывающие описания для битов регистров специального назначения, если они есть.
Не отображать фунции, typedefs и нетипизированные объекты	Когда этот флаг установлен не отображаются фунции, typedefs и нетипизированные объекты.

© 2009 Фитон, http://www.phyton.ru

Авто-дописывание: добавить имена SFR и битов Когда этот флаг установлен, то автоматически дописываются имена SFR и имена битов.

3.2.7.1.5 Диалоги Установить закладку/Восстановить закладку

Закладки служат для возврата в последующем к помеченной позиции курсора в файле исходного текста. Бывают локальные и глобальные закладки. Локальные закладки действуют в пределах одного файла. Глобальные закладки хранят не только позицию курсора, но и имя файла.

С помощью этих диалогов можно задать и использовать до 10 локальных закладок. Каждой локальной закладке назначается индивидуальная кнопка с номером.

Чтобы открыть диалог **Установить закладку**, нажмите **Alt+[**. Чтобы открыть диалог **Восстановить закладку**, нажмите **Alt+]**. Чтобы установить закладку или перейти к ней, нажмите её кнопку с номером. Номер строки, в которой установлена закладка, позиция закладки в строке (в скобках) и текст этой строки отображены справа от кнопки.

Локальные закладки хранятся в файле конфигурации, к ним можно вернуться в следующем сеансе работы.

3.2.7.1.6 Диалог Отобразить с новой строки

Этот диалог выполняет переход в активном окне **Редактора** к отображению другой строки исходного текста. Следует указать номер строки или выбрать один из ранее использованных номеров в списке **История ввода**. Номер первой строки — 1.

3.2.7.1.7 Функция Быстрого просмотра

Функция быстрого просмотра работает следующим образом: если задержать курсор мыши над именем переменной в окне **Редактора**, или окне **Текст сценария**, то через полсекунды появится небольшое желтое окошко с текущим значением этой переменной. При дальнейшем движении курсором мыши это окошко исчезнет.

3.2.7.1.8 Действия с блоками

Операции с блоками предназначены для выполнения редактирования с более чем одним символом одновременно. Окно **Редактора** обеспечивает постоянные блоки (блоки с фиксацией выделения, persistent blocks) и выполняет полный набор действий с потоковыми (stream), вертикальными (column) и строчными блоками текста.

Непостоянные блоки Как только такой блок выделен, с ним надо сразу выполнять действие (удалять, копировать, и т.д.), потому что любое перемещение курсора снимает выделение блока. Если начать печатать текст в тот момент, когда выделен такой блок, то этот блок будет удалён и заменён напечатанным текстом.

Постоянные блоки Выделение блока сохраняется до тех пор, пока выделение не будет снято явно (клавишами **Shift+F3)** или блок не будет удалён (клавишами **Ctrl+X**). Для таких блоков операция вставки (paste) выполняется с особенностями. Для них есть две дополнительные операции: быстрое копирование и быстрое перемещение. Эти операции не используют буфер обмена и для выполнения им нужно меньше нажатий клавиш.

Стандартные блоки Потоковый блок представляет собой «поток текста», который начинается с начальной строки и позиции в этой строке и заканчивается на конечной строке и позиции в конечной строке.

Стандартные блоки включены по умолчанию (снят флаг **Вертикальные блоки** в диалоге **Опции редактора**).

Строчные блоки Такой блок состоит из строк текста целиком. Чтобы выделить строчный блок, поместите курсор в любую позицию в первой строке и нажмите **Alt+Z**, потом поместите курсор в любой позиции в последней строке блока и нажмите **Alt+Z** ещё раз (второе нажатие не нужно, если блок предстоит немедленно удалить или скопировать в буфер обмена).

Строчные блоки доступны всегда, независимо от состояния флага **Вертикальные блоки**.

Вертикальные блоки Вертикальные блоки содержат прямоугольный фрагмент текста. Если часть блока выходит за пределы строки, то соответствующие символы блока считаются пробелами. Вертикальные блоки удобны в случаях, как в следующем примере исходного текста:

```
char Timer0 far ;
char Timer1 far ;
```

© 2009 Фитон, http://www.phyton.ru

```
char Int0   far ;
char Int1   far ;
```

Предположим, что в каждой строке слово «far» надо перенести и поместить сразу после слова «char». Потоковые блоки здесь помогут мало. Но задача может быть легко выполнена одним вертикальным блоком. Выделите вертикальный постоянный блок, содержащий слово «far» в каждой строке, поместите курсор на первую букву слова «Timer0» и нажмите **Shift+F2** (быстрое перемещение блока):

Чтобы выделить блок, либо перемещайте мышь, нажав её левую кнопку, либо используйте клавиши со стрелками клавиатуры при нажатой клавише **Shift**. Чтобы снять выделение блока, нажмите **Shift+F3**.

Копирование / перемещение блоков

Выделенный блок можно скопировать или переместить в пределах этого же окна **Редактора** двумя способами: непосредственно (быстрое копирование, быстрое перемещение) и через буфер обмена (обычный способ Windows). Копирование и перемещение блоков между окнами **Редактора** или в другое приложение возможно только через буфер обмена.

Примечание. Результат копирования потокового или вертикального непостоянного блока зависит от режима вставки (INSERT). Если режим включён, то блок будет вставлен в текст в том месте, где находится точка ввода (мигающий курсор). В противном случае, копируемый блок будет записан поверх текста на участке такого же размера.

Быстрое копирование / перемещение

Быстрое копирование и перемещение блоков в пределах одного окна выполняется непосредственно, т.е., без использования буфера обмена, и поэтому удобно, поскольку для этого требуется только одно нажатие клавиш на одно действие. Выделите постоянный блок, потом поместите курсор в позицию, куда надо поместить блок, и нажмите **Shift+F1** для копирования, или **Shift+F2** для перемещения.

3.2.8 Окна, используемые в Файлах Сценария

При отладке Файлов Сценария можно использовать окно **Переменные** и окно **Переменные с автопросмотром.**

3.2.8.1 Окно Переменные

Во время отладки программы окно **Переменные** отображает текущие значения явно заданных переменных.

Каждый наблюдаемый объект занимает отдельную строку. В строке окно показывает имя, значение, тип и, если есть, адрес соответствующей переменной.

Только что открытое окно **Переменные** содержит только закладку **Главные**. Можно добавить дополнительные закладки, командой **Опции отображения** локального меню, а также изменить название любой из существующих страниц(tabs). Закладки работают независимо друг от друга, каждая закладка эквивалентна отдельному окну **Переменные**. Однако, если надо, можно открыть несколько окон **Переменные**.

На панели инструментов каждого из перечисленных выше окон есть кнопка **+Watch**, которая добавляет выбранный объект в окно **Переменные**.

Выбранный объект окно выделяет цветом. Чтобы выбрать другой объект, щёлкните его кнопкой мыши или используйте клавиши управления курсором.

Режимы отображения

В окне есть вертикальная и горизонтальная сетки, которые можно включить/выключить по отдельности.

Когда включена вертикальная сетка, данные в окне расположены столбцами и у каждого столбца есть заголовок в виде кнопки. Нажатие кнопок **Имя**, **Тип** и **Адрес** открывает диалог **Опции отображения** для выделенной в окне переменной. Нажатие кнопки **Значение** открывает диалог **Изменить значение** для выделенной переменной.

Когда вертикальная сетка выключена, двойной щелчок на строке с объектом открывает диалог **Изменить значение** для этой переменной.

Чтобы включить/выключить вертикальную сетку, используйте соответствующий флаг на закладке **Шрифты** (меню **Конфигурация**, команда **Опции экрана**).

Локальное меню

Локальное меню окна содержит следующие команды, для каждой команды имеется кнопка панели инструментов окна:

© 2009 Фитон, http://www.phyton.ru

Команда	Описание
Добавить переменную/ выражение	Добавляет один или несколько объектов в окно. Открывает диалог **Добавить переменную в окно** для выбора объекта по имени.
Удалить переменную из окна	Удаляет выбранный объект из окна.
Удалить все переменные из окна	Удаляет из окна все объекты.
Изменить значение	Открывает диалог **Изменить значение**, чтобы задать новое значение для выделенной переменной. Это же самое можно сделать, просто начав печатать новое значение на клавиатуре.
Переместить имя вверх	Перемещает выделенный в списке объект на одну строку вверх.
Переместить имя вниз	Перемещает выделенный в списке объект на одну строку вниз.
Опции отображения	Открывает диалог **Опции отображения**, чтобы изменить параметры отображения выделенного объекта, а также для добавления или удаления закладок в окне.
Добавить новую страницу	Открывает диалог **Добавить новую страницу в окно Переменных**

3.2.8.1.1 Диалог Опции Отображения окна Переменные

Этот диалог управляет параметрами отображения выделенной переменной в окне **Переменные**, а также добавляет или удаляет закладки в этом окне.

Элементы диалога	Описание
Имя переменной	Содержит имя выделенной переменной или выражение. Список накапливает ранее использованные выражения.
Формат отображения	Указывает формат представления для выделенного выражения: двоичный, шестнадцатеричный, десятичный или ASCII.
Всплывающее описание	Включает всплывающие описания для регистров специального назначения
Отображать битовую схему	Включает всплывающие описания битовой схемы регистров специального назначения, если они есть.
Отображать описания битов	Включает всплывающие описания для битов регистров специального назначения, если они есть.
Автоматический размер поля имени	Когда этот флаг установлен и включена вертикальная сетка (смотрите примечание внизу), окно автоматически подстраивает ширину столбца **Имя** под самую длинную запись в столбце.
Страницы	Список закладок в окне.
Добавить страницу	Открывает диалог **Добавить новую страницу в окно Переменные**, для ввода названия новой закладки. Окно создаст новую закладку по нажатию **OK**.
Удалить страницу	Удаляет закладку, выделенную в списке **Tabs**.
Редактировать страницу	Открывает диалог **Задать имя страницы**, для редактирования названия закладки.

© 2009 Фитон, http://www.phyton.ru

Глобальные опции отладки/ отображения	Открывает диалог **Загрузка кода**

3.2.8.1.2 Диалог Добавить Переменную в окно

Диалог предназначен для добавления символьных имён, например, имени переменной или выражения в окно **Переменные**. Окно содержит список символьных имён, определённых в программе или известных ей.

Элементы диалога	**Описание**
Введите имя переменной или выражения	Содержит имя выделенной переменной или выражение. Список накапливает ранее использованные выражения.
История ввода	Список предыдущих имен и выражений

3.2.9 Окно Информация о ChipProg

В этом окне выдается следующая информация:

- тип установленной микросхемы;
- производитель установленной микросхемы;
- версия драйвера программатора, поддерживающего данный тип;
- версия оболочки программатора;
- версия аппаратуры программатора;
- версия монитора драйверов программатора.

© 2009 Фитон, http://www.phyton.ru

4 Дополнительные Главы

4.1 Параметры Командной Строки

Перечисленные параметры командной строки (ключи) используются для запуска исполняемого файла ChipProgLPT по умолчанию. Написание параметров не зависит от регистра букв.

Ключ	Описание
/S<file>	Загрузить указанный файл сессии вместо файла по умолчанию. Файл сессии по умолчанию—это файл сессии, найденный в рабочей папке.
/D<file>	Загрузить указанный файл конфигурации экрана (desktop) вместо файла по умолчанию
/O<file>	Загрузить указанный файл опций вместо файла по умолчанию
/C"<название_микросхемы>"	Выбрать микросхему
/L<file>	Загрузить файл
/F<формат>	Указать формат загружаемого файла
/A	Автоматическое программирование
/I	Сделать окно программатора невидимым
/I1	Сделать окно программатора невидимым, сообщений об ошибках не выдавать, текст сообщения об ошибке скопировать в буфер обмена
/B	Не запрашивать номер порта и т.п.
/M	Демонстрационный режим .

Примечание. Имя файла <file> надо указывать сразу после ключа, без пробела.

Подробнее о файлах сессии, конфигурации экрана и опций—в главе Файлы конфигурации.

4.2 Окно Исходного Текста Файла Сценария

Окно этого типа предназначено для разработки исходного текста Файла Сценария (ФС) и отладки сценария. В действительности, это окно есть окно **Редактора** , и все обычные команды окна **Редактора исходного текста** работают здесь точно так же (например, **Поиск текста**, **Повторить поиск**, и так далее).

Кроме того, это окно располагает необходимыми функциями отладки. Имеется подсветка синтаксических конструкций и строк текста ФС, соответствующих текущему значению счётчика команд и установленным точкам останова.

Строки исходного текста, для которых компилятор сгенерировал исполняемый код, обозначены квадратиком в первой позиции.

Локальное меню

Локальное меню окна содержит следующие команды, для каждой команды имеется кнопка панели инструментов окна:

Команда	Описание
Step	Выполняет один оператор ФС.

© 2009 Фитон, http://www.phyton.ru

Run	Начинает непрерывное выполнение ФС.
Выполнить до Курсора	Выполнить ФС до адреса, соответствующего строке с курсором. По-другому можно дважды щелкнуть на нужной строке.
Origin	Показывает исходный текст со строки, чей адрес соответствует Программному Счетчику ФС. Эта команда не работает, когда для данных адресов не существует строк исходного текста.
Новый ПС	Устанавливает значение Программного Счетчика из ФС по адресу, соответствующему строке с курсором.
Переключить точку останова	Включает / Выключает Точку Останова по адресу, соответствующему строке с курсором.
Добавить в окно Переменные	Открывает окно **Переменные** (если оно не открыто) и помещает имя на позицию курсора.
Перезапуск	Перезапустить ФС.

Примечание. Для получения Справки по функциям или переменным подведите к ним курсор и щелкните. Более подробная информация в Как отладить Файл Сценария and Файл Сценария .

4.3 Окно Пользователя

Окно **Пользователя** используется вовремя работы с Файлами Сценариев (ФС). Оно позволяет:

- **создавать рисунки в окне при помощи функций графического вывода, доступных из ФС**
- **отображать текст в окне**
- **реагировать на события (смотрите WaitWindowEvent)**

Используя эти свойства, можно организовать работу окна в интерактивном режиме. За дополнительной информацией обращайтесь к разделу Файлы Сценариев .

Для того чтобы открыть окно ФС использует функцию **OpenUserWindow.**

Функции, работающие с окнами (включая окна **Пользователя**), получают идентификатор окна (handle) в качестве параметра. Поэтому одновременно можно держать несколько открытых окон этого типа.

На панели инструментов окна **Пользователя** имеется 16 кнопок (**0**…**F**). Нажатие на кнопку генерирует событие WE_TOOLBARBUTTON.

Окна этого типа не имеют локального меню.

4.4 Окно Потока Ввода / Вывода

Файлы скриптов используют окна такого типа для отображения потоков ввода/вывода в виде текста. Наиболее общие примеры потоков ввода/вывода - это вывод текста в окно и ввод символов с клавиатуры. Вы можете также переназначить вывод потоков в файлы и чтение данных из файлов. Для всех этих надобностей Файл Сценария (ФС) использует функции окна.

Чтобы открыть окно, ФС использует функцию **OpenStreamWindow** . За дополнительной информацией обращайтесь к разделу Файлы Сценариев .

Функции, управляющие окнами (включая окна **Потока Ввода/Вывода**) получают идентификатор окна (handle) в качестве параметра. Поэтому несколько окон одного типа могут быть открыты одновременно.

© 2009 Фитон, http://www.phyton.ru

Окно имеет два режима отображения: с автоматическим переносом строки (**Wrap**) и без него. В режиме автоматического перевода, каждая строка, не поместившаяся в окне, переносится на следующую позицию. В другом режиме конец не поместившейся строки находится за границей экрана и невидим. Кнопка **Wrap** на панели инструментов служит для переключения между этими режимами. Кнопка **Clear** очищает содержимое окна.

Окна этого типа не имеют локального меню.

4.5 Файлы Сценария

ChipProgLPT умеет выполнять так называемые Файлы Сценария (ФС), подобно тому, как DOS выполняет batch-файлы.

В основном, ФС используются для автоматизации работы пользователя с отладчиком. С помощью ФС можно в автоматическом (пакетном) режиме загружать программы, ставить точки останова, запускать программу на выполнение, манипулировать окнами и вообще выполнять почти все доступные пользователю действия. Можно также выдавать в окно **Консоль сообщения** и другие специальные окна, различные сообщения, строить пользовательские меню и т.п. Имеется возможность выводить любые графические данные в специальные окна.

Язык ФС - это Си-подобный язык: поддерживаются почти все Си конструкции, кроме структур и указателей. Здесь также много встроенных функций, таких как printf(), sin() и strcpy().

Файлы исходного текста ФС имеют расширение **.CMD**.

Простейший Пример Файла Сценария

Как написать Файл Сценария

Как запустить Файл Сценария

Как отладить Файл Сценария

Описание языка Файлов Сценария

Встроенные Функции языка Файлов Сценария

Встроенные Переменные Файлов Сценария

Отличия языка Файлов Сценария от языка Си

Алфавитный список Встроенных Функций и Переменных языка Файлов Сценария

4.5.1 Простейший Пример Файла Сценария

Этот пример просто загружает программу и запускает ее выполнение,предварительно сбросив процессор. Выполнение программы останавливается, когда программа достигает функции Init, и выводит на экран значение указателя стека SP:

```
#include <system.h>


{
 Reset();                                // сбросить процессор
 LoadProgram("TEST.SYM", LF_PICASM);     // загрузить программу
 RunTo(                                  // выполнить программу до
```

© 2009 Фитон, http://www.phyton.ru

```
       AddrExpr("Init")                  // адреса функции Init

      );

 printf("SP=%04X", $SP);                 // вывести на экран значение SP

}
```

4.5.2 Как Написать Файл Сценария

Файл сценария (ФС) очень похож на исходный текст программы на языке Си. Для создания и редактирования ФС выможете использовать встроенный редактор CodeMaster`а или любой другой текстовый редактор. Вы можете расположить ФС в своей рабочей директории или в директории, где инсталлирован CodeMaster.

Заметим, что не следует использовать в имени ФС символы, не допустимые для использования в именах переменных, т.е. знаки препинания, скобки и т.п.

4.5.3 Как Запустить Файл Сценария

Запустить файл сценария (ФС) можно несколькими способами:

1. При запуске CodeMaster автоматически выполняется файл **start.CMD,** подобно тому, как при старте DOS выполняется файл autoexec.bat. Если ФС **start.CMD** не найден в текущей директории, то запускается файл **start.CMD** из директории, где установлен CodeMaster.

2. Можно воспользоваться командой Главного меню **Сценарии** или нажать кнопку панели инструментов CodeMaster`а для того , чтобы запустить ФС. На экран выводится диалог **Файлы Сценария**, позволяющий манипулировать уже запущенными ФС и запускать новые.

3. Используйте функцию StartCommandFile() для запуска ФС из ФС.

4.5.4 Как Отладить Файл Сценария

Файлы сценария (ФС) можно отлаживать примерно так, как отлаживают программы с помощью окна **Редактора** и окна **Переменные**, т.е. выполнять ФС по шагам и "до курсора", ставить точки останова, просматривать значения переменных и т.п. Для отладки по исходному тексту предназначено окно **Исходного текста Файла Сценария.** Это окно открывается автоматически, если при запуске ФС через команду **Сценарии** Главного меню в диалоге **Файлы Сценария** установить опцию "Отладка".

При запуске одного ФС из другого функцией StartCommandFile() можно определить параметр, указывающий войти в режим отладки и открыть окно **Исходного Текста Файла Сценария.**

Можно также в любое место ФС вставить вызов функции Debug() . После вызова этой функции выполнение ФС будет приостановлено и включится режим отладки.

Чтобы посмотреть значение какой-либо переменной ФС в окне **Переменные**, можно воспользоваться командой **Добавить переменную в окно переменных** меню окна **Исходного Текста Файла Сценария** или иконкой на его линейке управления. Можно сделать это и вручную в окне **Переменные**. Например, если нужно посмотреть значение переменной addr, использующейся в ФС с именем TEST, нужно в окно **Переменные** поместить конструкцию #TEST#addr. Если addr об'явлена как public, т.е. вне функции, то нужно написать ##addr.

4.5.5 Описание языка Файлов Сценария

Язык ,на котором пишутся Файлы Сценария(ФС), очень похож на язык Си. Если Вы хорошо знаете Си, то эту главу можете не читать, а прочитать про отличия языка ФС от языка Си.

© 2009 Фитон, http://www.phyton.ru

Нижеследующий материал не содержитописания приемов программирования на языке ФС; для этого вы можете воспользоваться многочисленными книгами по языку Си.

Общий синтаксис Файлов Сценария

Основные Типы Данных

Порядок байт в Данных

Операции и выражения

Операторы

Функции

Описания

Директивы препроцессора языка Файлов Сценария

Предопределенные символы при компиляции Файлов Сценария

4.5.6 Встроенные Функции языка Файлов Сценария

Система Файлов Сценария (ФС) предоставляет пользователю большой набор встроенных функций для работы со строками и файлами, математических вычислений, доступа к ресурсам процессора и т.д. Описания встроенных функций помещены в файл **system.h,** который необходимо включить в исходный текст ФС директивой #include .

```
#include <system. h>
```

Встроенные функции используются так же, как и функции, определяемые пользователем.

Функции Ожидания События

Buffer access functions

Device programming control functions

Математические функции

Функции для работы со Строками

Функции для работы с Символами

Функции для работы с Файлами и Директориями

Функции для работы с Файлами-Потоками

Функции Форматированного Ввода-Вывода

Функции Загрузки/Выгрузки Файлов Сценария

Функции для работы с Текстом в окне Редактора

Функции управления Оболочкой Отладчика

[****]

Функции для работы с Windows и прочие Системные фунуции

Функции Графического Вывода

© 2009 Фитон, http://www.phyton.ru

Функции для работы с окнами Потоков Ввода-Вывода

Всякие другие Функции

Примечание: Чтобы получить справку о функции или переменной, во время редактирования исходного текста ФС с помощью встроенного редактора CodeMaster, поместите курсор на имя функции/переменной и нажмите **Alt+F1**.

4.5.7 Встроенные Переменные Файлов Сценария

Ко встроенным переменным Файлов Сценария(ФС можно обращаться так же, как к обычным глобальным переменным, однако некоторые встроенные переменные доступны только для чтения и при попытке записи в такую переменную PDS-52 выдаст сообщение об ошибке на этапе выполнения ФС.

Об'явления встроенных переменных содержатся в заголовочном файле **system.h.**

WorkFieldWidth

WorkFieldHeight

ApplName[]

DesktopName[]

SystemDir[]

errno

_fmode

MainWindowHandle

NumWindows

WindowHandles[]

SelectedString[]

[****]

LastMessageInt

LastMessageLong

Встроенные переменные текстового редактора:

InsertMode

CaseSensitive

WholeWords

RegularExpressions

BlockCol1

BlockCol2

BlockLine1

BlockLine2

BlockStatus

© 2009 Фитон, http://www.phyton.ru

CurLine

CurCol

LastFoundString

Programming variables:

InsertTest

ReverseBytesOrder

BlankCheck

VerifyAfterProgram

VerifyAfterRead

ChipStartAddr

ChipEndAddr

BufferStartAddr

LastErrorMessage

DialogOnError

4.5.8 Отличия языка Файлов Сценария от языка Си

Как уже говорилось, язык Файлов Сценария(ФС) - это Си-подобный язык, и не нужно требовать от него соответствия стандарту. Многие вещи не реализованы потому, что без них вполне можно обойтись, тогда как любое усложнение языка чревато ошибками в компиляторе (все же компилятор языка ФС - непростая штука).

- Непосредственно не поддерживаются указатели. Однако поддерживаются массивы, следовательно, "указатель" всегда можно сконструировать из массива и номера элемента. Заметьте, что, например, функции работы со строками типа strcpy получают в качестве параметров строку и номер байта (index), которые и формируют "указатель". В об'явлении функций index равен нулю по умолчанию.

- Не поддерживаются указатели на функции. При необходимости табличный вызов всегда можно заменить оператором **switch**.

- Не поддерживаются многомерные массивы. Если возникает острое желание, можно легко написать пару функций типа:

```
int GetElement(int array[], int index1, int index2);

void SetElement(int array[], int index1, int index2, int value);
```

- Не поддерживаются структуры (и союзы). По сути дела, без структур можно всегда обойтись. Единственно, структуры могут понадобиться при работе с API Windows и пользовательскими DLL, но, как правило, это делают только опытные программисты, которые знают, как добираться до членов структур (подсказка - есть функции типа memcpy, которые получают "указатель" на void).

- Не поддерживаются перечислимые типы (**enum**). Обойдитесь #define.

© 2009 Фитон, http://www.phyton.ru

- Не поддерживаются макросы препроцессора типа **#define half(x) (x / 2)**. Можно то же самое сделать с помощью функций.

- Не поддерживаются условный оператор типа **x = y == 2? 3 : 4;** а также оператор "запятая" вне объявлений переменных. Например,

int i = 0, j = 1; поддерживается, но

for (i = 0, j = 1; ...) не поддерживается.

- Не поддерживаются **пользовательские** функции с переменным числом параметров. Имеется, однако, большое количество системных функций типа printf с переменным числом параметров.

- Не поддерживаются объявления **пользовательских** параметров функции типа **void array[].** Системные функции типа memcpy имеют такие параметры.

- Логические выражения всегда вычисляются полностью. Это очень важно помнить, т.к. ситуация типа

```
char array[10];

if (i < 10 && array[i] != 0)

  array[i] = 1;
```

вызовет ошибку на этапе выполнения, если i будет больше 9, т.к. выражение array[i] будет вычислено. В обычном компиляторе оно вычислено не будет, т.к. условие **i < 10** отменит дальнейшую обработку выражения.

- Константные выражения всегда вычисляются во время выполнения, т.е. **int i = 10 * 22** будет вычислено не во время компиляции, а при выполнении.

- Отсутствует ключевое слово **const** .

- Класс размещения static не поддерживается.

Однако

- Допускается объявление переменных в любом месте, а не только до первого выполняемого оператора. Например:

```
void main()

{

GlobalVar = 0;

int i = 1;      // будет как в C++

}
```

- Допускаются вложенные комментарии.

© 2009 Фитон, http://www.phyton.ru

- Допускаются выражения типа **array = "1234"** .

- Допускаются значения параметров по умолчанию в об'явлениях функций, как в C++. Например, **void func (char array[], int index = 0);.** В качестве значения по умолчанию может быть и выражение, например, **void func(char array[], int index = func1() + 1);.**

- Допускаются выражения в инициализаторах глобальных переменных. Например:

```
float table[] = { sin(0), sin(0.1) };

void main()

{

  ...

}
```

4.5.9 Алфавитный Список встроенных Функций и Переменных в Файлах Сценария

API
ActivateWindow
AddButton
AddSymbol
AddWatch
[****]
ApplName[]
BackSpace
BlockBegin
BlockCol1
BlockCol2
BlockCopy
BlockDelete
BlockEnd
BlockFastCopy
BlockLine1
BlockLine2
BlockMove
BlockOff
BlockPaste
BlockStatus
CaseSensitive
[****]
ClearWindow
CloseProject
CloseWindow
Cr
CurChar
CurCol
CurLine
Curcuit
Debug
DelChar
DelLine
[****]

© 2009 Фитон, http://www.phyton.ru

DesktopName[]
DisplayText
DisplayTextF
Down
Ellipse
Eof
Eol
ExecMenu
ExecScript
ExitProgram
[****]
FileChanged
FillRect
[****]
FindWindow
FirstWord
[****]
ForwardTill
ForwardTillNot
FrameRect
FreeLibrary
[****]
GetFileName
GetLine
GetMark
[****]
GetScriptFileName
GetWindowHeight
GetWindowWidth
[****]
GotoXY
[****]
InitCOM
InsertMode
[****]
InvertRect
LastChar
LastEvent
LastEventInt{1...4}
LastFoundString
[****]
LastMessageInt
LastMessageLong
LastString
Left
LineTo
LoadDesktop
LoadLibrary
LoadOptions
LoadProgram
LoadProject
MainWindowHandle
[****]
MessageBox
MessageBoxEx
[****]
MoveTo
MoveWindow
NumWindows
OpenEditorWindow
OpenStreamWindow
OpenUserWindow
OpenWindow

© 2009 Фитон, http://www.phyton.ru

Polyline
ReceiveCOM
Rectangle
RedrawScreen
RegularExpressions
ReloadProgram
RemoveButtons
[****]
ResetSymbolTable
[****]
Right
[****]
SaveData
SaveDesktop
SaveFile
SaveOptions
Search
SearchReplace
SelectBrush
[****]
SelectFont
SelectPen
SelectedString[]
SendCOM
SetBkColor
SetBkMode
[****]
SetCaption
[****]
SetFileName
SetMark
[****]
SetPixel
SetTextColor
SetToolbar
SetUpdateMode
SetWindowFont
SetWindowSize
SetWindowSizeT
[****]
StatusCOM
[****]
SystemDir[]
TerminateAllScripts
TerminateScript
Text
[****]
Tof
Up
UpdateWindow
Wait
WaitExprChange
WaitExprTrue
WaitGetMessage
WaitMemoryAccess
WaitSendMessage
WaitStop
WaitWindowEvent
WholeWords
WindowHandles[]
WindowHotkey
WordLeft
WordRight

© 2009 Фитон, http://www.phyton.ru

WorkFieldHeight
WorkFieldWidth
_GetWord
_ff_attrib
_ff_date
_ff_name
_ff_size
_ff_time
_fmode
_fullpath
_printf
abs
acos
asin
atan
atof
atoi
ceil
chdir
chsize
clearerr
close
cos
creat
creatnew
creattemp
delay
difftime
dup
dup2
eof
errno
exec
exit
exp
fabs
fclose
fdopen
feof
ferror
fflush
fgetc
fgets
filelength
fileno
findfirst
findnext
floor
fmod
fnmerge
fnsplit
fopen
fprintf
fputc
fputs
fread
freopen
frexp
fscanf
fseek
ftell
fwrite
getc

© 2009 Фитон, http://www.phyton.ru

getcurdir
getcwd
getdate
getdfree
getdisk()
getenv
getftime
gettime
getw
inport
inportb
isalnum
isalpha
isascii
isatty
iscntrl
isdigit
isgraph
islower
isprint
ispunct
isspace
isupper
isxdigit
itoa
lock
locking
log
log10
lseek
ltoa
memccpy
memchr
memcmp
memcpy
memicmp
memmove
memset
mkdir
movmem
open
outport
outportb
peek
peekb
poke
pokeb
pow
pow10
printf
pscanf
putc
putenv
putw
rand
random
randomize
read
rename
rewind
rmdir
scanf
searchpath

© 2009 Фитон, http://www.phyton.ru

setdisk
setftime
setmem
setmode
sin
sprintf
sqrt
srand
sscanf
stpcpy
strcat
strchr
strcmp
strcmpi
strcpy
strcspn
stricmp
strlen
strlwr
strncat
strncmp
strncmpi
strncpy
strnicmp
strnset
strpbrk
strrchr
strrev
strset
strspn
strstr
strtol
strtoul
strupr
tan
tanh
tell
toascii
tolower
toupper
ultoa
unlink
unlock
wgetchar
wgethex
wgetstring
wprintf
write

4.6 Выражения

Выражения в ChipProgLPT —это математические конструкции для вычисления результата, содержащие один или более операндов. ChipProgLPT поддерживает различные операции и выражения. Допускаются следующие операнды:

- числа
- символьные имена
- пример выражения

Когда требуется число, вы можете ввести выражение. ChipProgLPT возьмет значение этого выражения. Например, когда вы используете команду **Modify** в окне **Переменные,** вы можете ввести новое значение в виде числа или арифметического выражения.

© 2009 Фитон, http://www.phyton.ru

Интерпретация результата выражения

Результат вычисления выражения интерпретируется в зависимости от контекста, в котором он используется.

В диалоге, где требуется указать адрес (например, адрес точки останова по коду), ChipProgLPT старается интерпретировать результат выражения как адрес. Если указать имя переменной, результатом такого выражения будет адрес переменной, а не её значение.

Если в диалоге требуется указать именно число (например, в диалоге **Изменить значение** окна **Дамп**), то результат выражения будет интерпретирован как число. То есть, если здесь указать имя переменной, то результатом будет значение этой переменной, а не её адрес.

Тем не менее, есть обходные пути:

Если предполагается адрес переменной, а Вам необходимо использовать её **значение**, тогда можно написать простое выражение типа «var + 0». В данном случае, в выражение попадёт значение переменной.

Если необходимо использовать **адрес** переменной, можно воспользоваться операцией взятия адреса (**&**). Например, «&var».

4.6.1 Операции с Выражениями

ChipProgLPT поддерживает все арифметические и логические операции, имеющиеся в языке Си, а также операции с указателями и операции взятия адреса:

Обозначение	Описание
`( )`	Скобки (более высокий приоритет)
`[ ]`	Выделение элемента массива
`.`	Выделение элемента структуры или союза
`->`	Выделение элемента структуры или союза, адресуемого указателем
`!`	Одноместная операция логического отрицания
`~`	Одноместная операция побитового инвертирования
`-`	Одноместная операция изменения знака
`&`	Определение адреса
`*`	Обращение по адресу
`(тип)`	Преобразование типа в явной форме
`(sizeof)`	Определение размера в байтах
`*`	Умножение
`/`	Деление
`%`	Деление по модулю
`+`	Сложение
`-`	Вычитание
`<<`	Сдвиг влево
`>>`	Сдвиг вправо
`<`	Меньше, чем
`<=`	Меньше или равно
`>`	Больше, чем
`>=`	Больше или равно
`==`	Равно
`!=`	Не равно
`&`	Побитовая операция И
`^`	Побитовая операция ИСКЛЮЧАЮЩЕЕ ИЛИ
`|`	Побитовая операция ИЛИ
`&&`	Логическая операция И
`||`	Логическая операция ИЛИ

© 2009 Фитон, http://www.phyton.ru

=	Присваивание

Тип операнда преобразуется в соответствии со стандартом ANSI.

Результатом логической операции является 0 (ЛОЖЬ) или 1 (ИСТИНА).

Разрешенные типы преобразования:

- Операнды могут быть преобразованы к простым типам (char, int, ... float).
- Указатели могут быть преобразованы к простым типам (char *, int *, ... float *) и к указателям на структуры и союзы.
- Слово "struct"не обязательно (MyStruct *).

4.6.2 Числа

По умолчанию, все числа считаются десятичными. Значения целых чисел должны умещаться в 32 бита, значения чисел с плавающей точкой—в формат одинарной точности (32 бита).

Поддерживаются следующие форматы чисел:

1) Целый десятичный.

Пример: `126889`

2) Десятичный с плавающей запятой.

Примеры: `365.678`; `2.12e-9`

3) Шестнадцатеричный, в формате Си или в формате ассемблера.

Примеры: `0xF6D7`; `0F6D7H`; `0xFFFF1111`

4) Двоичный. Числа, вводимые в двоичном формате, должны оканчиваться буквой «B».

Примеры: `011101B`; `11111111111111000011B`

5) Символьный (ASCII).

Примеры: `'a'`; `'ab'`; `'$B%8'`.

4.6.3 Символьные Имена

Если не загружена программа в виде HEX или двоичного файла, либо программа не имеет отладочной информации ; то имена, определенные в загруженной программе отсутствуют.

4.6.4 Примеры выражений

Примеры выражений

```
#test#i + #test#j << 2

(unsigned char)#test#i + 2

sizeof(##array) > 200


main

i + j << 2 / :CW0x1200

(unsigned char)i + 2

sizeof(array) > 200

(a == b && a <= 4) || a > '3'

sptr -> Member1 -> a[i]

*p
```

© 2009 Фитон, http://www.phyton.ru

```
*((char *)ptr)
```

© 2009 Фитон, http://www.phyton.ru

Алфавитный указатель

- A -

Alphabetical List of Script Language Built-in Functions and Variables 71
Angstrem SAV 50
ANSI 77
ASCII Hex 50

- B -

Binary image 50

- C -

ChipProg
  Главное Меню 26
ChipProg+ 12
  Комплект поставки 12
  Основные характеристики 13
  Характеристики Аппаратуры 13
  Характеристики Программного Обеспечения 13
ChipProg-2 14
  Комплект поставки 14
  Основные характеристики 14
ChipProg-40
  Характеристики Аппаратуры 14
  Характеристики Программного Обеспечения 15

- D -

Description of Script Language 66

- H -

Holtek OTR 50
Hot Keys 32
How to debug a script file 66
How to start a script file 66
How to write a script file 66

- I -

ICP 5
ISP
  ISP HV Mode 5
  ISP Mode 5

- J -

JEDEC 50

- M -

MODULE1 78
Motorola S-record 50
MS Windows 37

- P -

POF 50
PRG 50

- S -

Script 36
Script Language Built-in Functions 67
Script Language Built-in Variables 68
Standard/Extended Intel HEX 50

- Z -

Автомаизация 65
Автоматическое Программирование 40
Адресное пространство 47
Адресное пространство исходное 47
Адресное пространство назначения 47
Байт 0 41
Буферы
  Диалог 29
Быстрый просмотр 59
Быстрый Старт 7
Ввод/Вывод 64
Вертикальные блоки 59
Выбор микросхемы 29
  Диалог 29
Выбрать цвет 32
Выбрать шрифт 31
Выделение памяти 29
Выражения 76
Выражения поиска 57
Дамп Буфера
  локальное меню 44
Двоичный 78
Десятичный 78
Диалог Поиск текста 55
Диалог Поиск/Замена текста 56
Диалоги Установить закладку/Восстановить закладку 59
Добавить Переменную в окно

© 2009 Rumon, http://www.phyton.ru

Добавить Переменную в окно
диалог 62
Дополнительный подслой 30
Заголовок активной страницы
подсвечивать 33
Загрузить программу 27
Загрузить сессию 28
Загрузить Файл 23
диалог 49
Загрузить Файл для записи 24
Задать комбинацию клавиш
диалог 32
Записать информацию 24
Запрограммировать Устройство 23
Изменить Значение
диалог 46
Имя буфера, Настройки 'Code' 30
Интерфейс Пользователя Обзор 38
Информация для контакта 10
Информация о ChipProg
окно 62
Информация о Микросхеме
окно 51
Информация об операциях 39
История 28
Как получить оперативную справку 9
Калькулятор
диалог 34
Клавиша
задать 32
удалить 32
Командная строка Ключи 63
Командная строка Параметры 63
Команды
меню 34
Команды меню 26
Команды панели инструментов
Кнопка Edit 43
Консоль Сообщений
окно 53
Контакты с Фитоном 9
Контрольная сумма
Отображать 48
Конфигурационные Файлы 28
Конфигурация
меню 29
Опции редактора 29
Опции экрана 29
Конфигурация буфера
диалог 30, 45
имя буфера 30
настройки Code 30
область идентификатора 30
Копирование блоков 59
Корпуса/Адаптеры 29
Линейка управления
закладка 33
Маска поиска 29
Меню Команды 34
Меню Окна 37
Меню Просмотр 28
Меню Справка 38
Меню Файл 27
Микроконтроллеры AVR Особенности
Программирования 17
Микроконтроллеры MCS-51 Особенности
Программирования 17
Микроконтроллеры PICmicro Особенности
Программирования 17
Назначение клавиш экрана
закладка 32
Настройки 'Code' 30
Настройки окна Дамп 48
Непечатаемые симвоы ASCII 48
Непостоянные блоки 59
Нечетный байт 41
Новый Программный Счетчик 63
О ChipProg
информация 38
Обзор 5
Интерфейс Пользователя 38
Область идентификатора 30
Окна 37
Окно
Информация о Микросхеме 51
Окно Дамп Буфера 44
Окно Информация о ChipProg 62
Окно Информация о Микросхеме 51
Окно исходного текста Файла Сценария 63
Окно Консоль Сообщений 53
Окно Параметры Микросхемы и Алгоритма 42
Окно Переменные 60
Опции отображения 61
Окно Переменные с автопросмотром 58
Окно Пользователя 64
Окно потока Ввода/Вывода 64
Окно Программирование 39
Окно Редактора 53
Окно Справка 9
Окно Файла Сценария 63
Оперативная справка 9
Оперативные справки 38
Операции 77
Операции с блоками 59
Операции с Блоками Памяти 47
Операции с выражениями 77
Определения
адаптер 5
буфер 5
буфер памяти 5
подслой 5

© 2009 Фитон, http://www.phyton.ru

Опции
диалог 30
Опции отображения окна Переменных 61
Опции Экрана
диалог 31
Опции, Адреса, Чередование
Диалог 41
Основной подслой 30
Особенности Программирования
микроконтроллеры AVR 17
микроконтроллеры MCS-51 17
микроконтроллеры PICmicro 17
Отображать контрольную сумму 48
Отобразить с адреса
диалог 45
Отобразить с новой строки
диалог 59
Отредактировать Информацию для записи 24
Параметры Алгоритма 42
Параметры Микросхемы 42
Параметры Микросхемы и Алгоритма
Команды панели инструментов 42
Переключить 63
Переменную в окно
добавить 62
Переменные
добавить/удалить 60
окно 60
Переменные с автопросмотром
диалог 58
Перемещение блоков 59
Поиск текста
диалог 55
Поиск/Замена текста
диалог 56
Постоянные блоки 59
Примеры выражений 78
Проверить Стертость 23
Програматоры
Сравнительные характеристики 11
Программатор ChipProg+ 12
Программатор ChipProg-2 14
Программаторы 11
Программирование
Адреса 39
диалог 39
Загрузить Файл для записи 24
Записать информацию 24
Запрограммировать Устройство 23
Информация об операциях 39
Окно 39
операции 39
Отредактировать Информацию для записи 24
Проверить Стертость 23
Продублировать Устройство 25
Прочитать Устройство 25
Сконфигурировать Устройство 24
Сохранить Данные, прочитанные из Устройства 25
Сравнить 24
Стирание 23
функции процесса 23
Программирование в плате пользователя 16
Программная установка ChipProg 7
Программный Счетчик 63
Программный Счетчик ФС 63
Продублировать Устройство 25
Просмотр 28
Простой пример Файла Сценария 35
Прочие установки
закладка 33
Прочитать Устройство 25
ПС 63
Работа с программатором 15
Редактор Текста 54
Редактора
окно 53
Своп-файлы 29
Семейство программаторов 11
Символьные имена 78
Символьный 78
Сконфигурировать Устройство 24
Сообщения
выдавать диалоги 33
записывать в конец файла 33
записывать в файл журнала 33
звуковое уведомление 33
Сохранить Данные, прочитанные из Устройства 25
Сохранить сессию 28
Сохранить Файл 27
Сохранить Файл из буфера
диалог 50
Справка
меню 38
Сравнить 24
Стандартные блоки 59
Стирание 23
Строка Главного меню 26
Строка статуса главного окна 33
Строчные блоки 59
Сценарии
меню 35
Сценарий 36
Таблица подключения к адаптеру.
Подключение к адаптеру. 52
Таблица соединений адаптера.
Соединения адаптера. 51
Текст 63

© 2009 Фирма «Тайтон» http://www.chyton.ru

Текст сценария
окно 36
Терминология 5
Термины и определения 5
Техническа Поддержка 9
Требования к Системе 7
Установить закладку/Восстановить закладку
диалоги 59
Установка ChipProg 7
Установка микросхемы 16
Файл меню 27
Файл Сценария 64
Файлы поставки 7
Файлы Сценариев 65
Файлы Сценария
диалог 36
имя 36
Фитон связаться 9
Формат отображения 48
Форматы Файлов 50
ФС 63, 64
Функцию «быстрого просмотра»
включить 33
Функция Быстрого просмотра 59
Цвет 32
Цвета
закладка 32
Чередование данных 41
Четный байт 41
Числа 78
Шестнадцатеричный 78
Шрифт 31
Шрифты
закладка 31

© 2009 Фитон, http://www.phyton.ru

Фирма "Фитон”

тел./факс: (495) 730-75-84 (многоканальный)

http://www.phyton.ru
email: phyton@phyton.ru

Адрес: Россия, 127015, Москва, ул. Новодмитровская,
д. 5а, оф. 1103