Микроконтроллеры MCS-196

Общая характеристика

***

Руководство по работе в среде проектирования

ProjectBuilder

Новосибирск
1995

Оглавление

Предисловие ... 3
Обзор микроконтроллеров серии MCS-196 ... 6
1. Семейство HSIO ... 6
1.1. 8XC196KB ... 7
1.2. 8XC196KC ... 7
1.3. 8XC196KD ... 7
1.4. Ключевые особенности микроконтроллеров семейства HSIO ... 8
2. Семейство EPA ... 9
2.1. 8XC196KR ... 9
2.2. 8XC196KT ... 10
2.3. 8XC196NT ... 10
2.4. 8XC196NP ... 10
2.5. 8XC196NU ... 11
2.6. Ключевые особенности микроконтроллеров семейства EPA ... 11
3. Семейство микроконтроллеров для управления движением ... 12
3.1. 8XC196MC ... 12
3.2. 8XC196MD ... 13
3.3. Ключевые особенности микроконтроллеров семейства Motion Control ... 14
4. Документация ... 16
5. Средства поддержки ... 16
ApBuilder. Введение ... 19
1. Как запустить программу "*ApBuilder*"? ... 19
2. Команды меню *ApBuilder'a* ... 20
3. Функциональная клавиатура *ApBuilder'a* ... 22
4. Что дальше? ... 27
ModelBuilder. Введение ... 28
1. Использование шаблонов производительности при моделировании приложений ... 28
1.1. Модель прикладной системы ... 29
1.2. Шаблоны для оценки производительности ... 29
1.3. Загрузка процессора ... 29
1.4. Как это делается ... 29
2. Моделирование ... 30
3.Хронометрирование программ пользователя ... 31
3.1. Требования к хронометрируемым программам ... 32
3.2. Запуск хронометрирования модели ... 33
4. Пример моделирования приложения ... 34
4.1. Описание моделируемого приложения ... 34
4.2. Моделирование приложения ... 35
5. Работа с *ModelBUILDER*'ом ... 36
5.1. Настройка системы ... 37
5.2. Установка параметров модели ... 37

5.3. Альтернатива аппаратным средствам моделирования 38
5.4. Команды меню *ModelBUILDER*'a 39
5.5. Функциональная клавиатура *ModelBUILDER*'a 40
5.6. Некоторые дополнительные соображения 41
Debug Monitor. Введение 42
Обозначения 42
1. Особенности Монитора mon96.exe 43
1.1. Ограничения: 43
2. Описание экрана пользователя 43
3. Работа с *Монитором* 44
3.1. Настройка системы 44
3.2. Загрузка программ пользователя 45
3.3. Дизассемблирование загруженного кода 45
3.4. Просмотр содержимого памяти 47
3.5. Исполнение программ 48
3.6. Пошаговое исполнение программ 49
3.7. Особенности реализации пошагового режима работы 50
3.8. Использование точек останова 52
3.9. Изменение содержимого ячеек памяти и специальных регистров микроконтроллера 53
3.10. Встроенный ассемблер 54
3.11.Сохранение содержимого памяти оценочного модуля на диске 55
3.12. Другие возможности *Монитора* 55
Приложение 56
1. Демонстрационный модуль (Target Board) 56
2. Оценочный модуль (Evaluation Board) 59
Литература: 60

Предисловие

Первые микроконтроллеры серии ***MCS-96*** выпущены на рынок корпорацией ***Intel*** в конце 70-х годов прошлого века и стали, по существу, промышленным стандартом в классе 16-разрядных микроконтроллеров, предназначенных для встраиваемых приложений. Эта серия непрерывно развивается и в настоящее время состоит из нескольких семейств, объединяющих более двух десятков моделей, реализованных с учетом требований конкретных областей применения.

К первому семейству серии MCS-96 относятся микроконтроллеры ***8X9X***, выполненные по HMOS технологии. Впоследствии они были вытеснены более совершенными CHMOS изделиями ***8XC196***, о которых и пойдет речь далее.

Архитектура микроконтроллеров MCS-96 и MCS-196, называемая архитектурой типа **"регистр-регистр"**, принципиально отличается от архитектуры 8-разрядных микроконтроллеров серий ***MCS-48*** и ***MCS-51***, а также многих микропроцессоров фирмы Intel. Как известно, такая архитектура обеспечивает достижение более высокой производительности и упрощает работу с периферией. Этим, в первую очередь, объясняются возможности эффективного решения на базе микроконтроллеров MCS-196 достаточно сложных задач управления в реальном времени такими объектами, как, например, приводы жестких дисков и принтеров, модемы, двигатели, системы сбора и обработки данных, системы распознавания образов и т.п.

Главным отличительным признаком архитектуры микроконтроллеров серии MCS-196 является ядро, базирующееся на регистровом файле. Большое число универсальных легко доступных регистров исключает "узкие места", свойственные архитектуре, использующей специальные регистры-аккумуляторы, и обеспечивает быстрое переключение контекста. Все устройства MCS-196 реализуют операции с байтами, словами, несколько операций с 32-битовыми операндами а также оепрации перехода по битам.

Кроме того, микроконтроллеры содержат разнообразные встроенные периферийные блоки - многоканальные аналого-цифровые преобразователи, широтно-импульсные модуляторы, таймеры, последовательные порты и т.п., которые могут функционировать автономно, практически не загружая центральный процессор. Для иллюстрации некоторых возможностей микроконтроллера реагировать на события, происходящие во внешней среде, ниже приведена таблица, в которой приведены характеристики разрешающей способности при восприятии и формировании сигналов периферийными блоками микроконтроллеров - устройствами высокоскоростного ввода-вывода типа ***HSIO (High Speed Input/Output)*** и типа ***EPA (Event Processer Array*** - матрица процессоров событий).

Разрешающая способность во времени	16MHz	20MHz	24MHz
HSI (высокоскоростной ввод)	1,125µs	900ns	-
HSO (высокоскоростной вывод)	1µs	800ns	-
EPA (матрица процессоров событий)	250ns	200ns	167ns

Характерные особенности контроллеров шины устройств серии MCS-196 - программное задание числа тактов ожидания, разрядности шины данных (8-ми или 16-разрядной), а также протокол HOLD/HLDA для многопроцессорных систем. Некоторые из последних моделей микроконтроллеров серии MCS-196, 8XC196NP и 8XC196NU, позволяют динамически переключать режимы работы шин (мультиплексировать или демультиплексировать адреса и данные), кроме того, они содержат блок выборки внешних устройств ***(Chip Select Unit),*** что упрощает построение схем сопряжения с внешней периферией***.***

Эти и многие другие замечательные свойства микроконтроллеров серии MCS-196 привлекают внимание специалистов ведущих фирм мира, занимающихся проектированием встраиваемых систем для различных отраслей техники. К сожалению, подавляющее большинство российских инженеров и научно-технических сотрудников до недавних пор были лишены возможности не только применять эти микроконтроллеры, но даже познакомиться с их особенностями - наша электронная промышленность оказалась не в состоянии или не сочла необходимым их воспроизвести (как обычно делалось, если находился достаточно крупный заказчик). Кроме того, насколько нам известно, на русском языке не издавалась техническая литература, в которой рассматривалась бы архитектура микроконтроллеров серии MCS-196. Сейчас стали доступны практически любые современные компоненты, но их распространение, а, следовательно, и развитие отечественной системотехники сдерживаются, в основном, из-за отсутствия соответствующих информационных материалов на русском языке. Предлагаемое Вашему вниманию пособие призвано облегчить "русскоязычным" специалистам освоение столь мощной и достаточно современной элементной базы, каковой являются микроконтроллеры серии MCS-196.

При отборе материалов мы решили в первую очередь представить информацию *не о структуре и характерных особенностях конкретной модели микроконтроллера*, а об *универсальных инструментальных средствах, предназначенных для изучения архитектуры и проектирования систем* на базе любых микроконтроллеров этой серии (и, кстати, многих других изделий фирмы Intel).

Объясняется наше решение следующим. Во-первых, объем информации, касающейся микроконтроллеров рассматриваемого типа, очень велик - по каждой из разновидностей микроконтроллеров фирмой выпущены весьма подробные руководства по применению, много обзорной и справочной литературы; перевод даже минимального комплекта документации и ее последующее изучение требует значительных затрат времени. В то же время, при недостатке информации даже предварительный выбор одного из многочисленных представителей серии MCS-196 является непростой задачей. Во вторых, корпорация Intel создала уникальные базы данных, экспертные и обучающие системы, системы моделирования и отладки, с помощью которых решение упомянутой задачи и собственно задач проектирования существенно упрощается. Ускорить и облегчить освоение этих систем и является основной целью пособия. Научившись работать с предлагаемым Intel инструментарием, пользователь сам сможет выбрать нужный ему материал и изучать с той степенью подробности и с той скоростью, которые покажутся ему оптимальными.

В пособие включены краткий **обзор** семейств микроконтроллеров серии MCS-196, **описание и краткие руководства по работе в системе проектирования *ProjectBuilder***, в том числе **- руководства по работе в экспертной системе *ApBuilder***, **системе моделирования *ModelBuilder***, **отладчике *Debug Monitor***, приведены также минимальные **сведения о простейших аппаратных средствах - оценочных (демострационных) модулях *Evualution Board*** и ***Demo Board*** (называемом также ***Target Board*** или ***Development Board***), необходимых при изучении архитектуры микроконтроллеров и проектировании простейших систем на их основе.

При составлении пособия использована техническая документация и все упомянутые выше программные средства Intel, любезно предоставленные ее официальным дистрибьютером - ***АО "Новые технологии",*** сотрудникам которого мы выражаем искреннюю признательность. Отбор материалов и форма изложения продиктованы опытом, полученном при освоении изделий MCS-196, преподавании соответствующей дисциплины ***на кафедре систем сбора и обработки данных Новосибирского государственного технического университета***, а также в процессе создания реальных технических систем. Следует отметить, что в оригинале (на английском языке) предлагаемые Вам руководства существуют только в "электронном" виде, однако мы считаем, что на первых этапах работы весьма полезно иметь их машинописную версию. Впоследствии возможно создание и "электронных" учебников в гипертекстовом формате.

Желаем успеха!

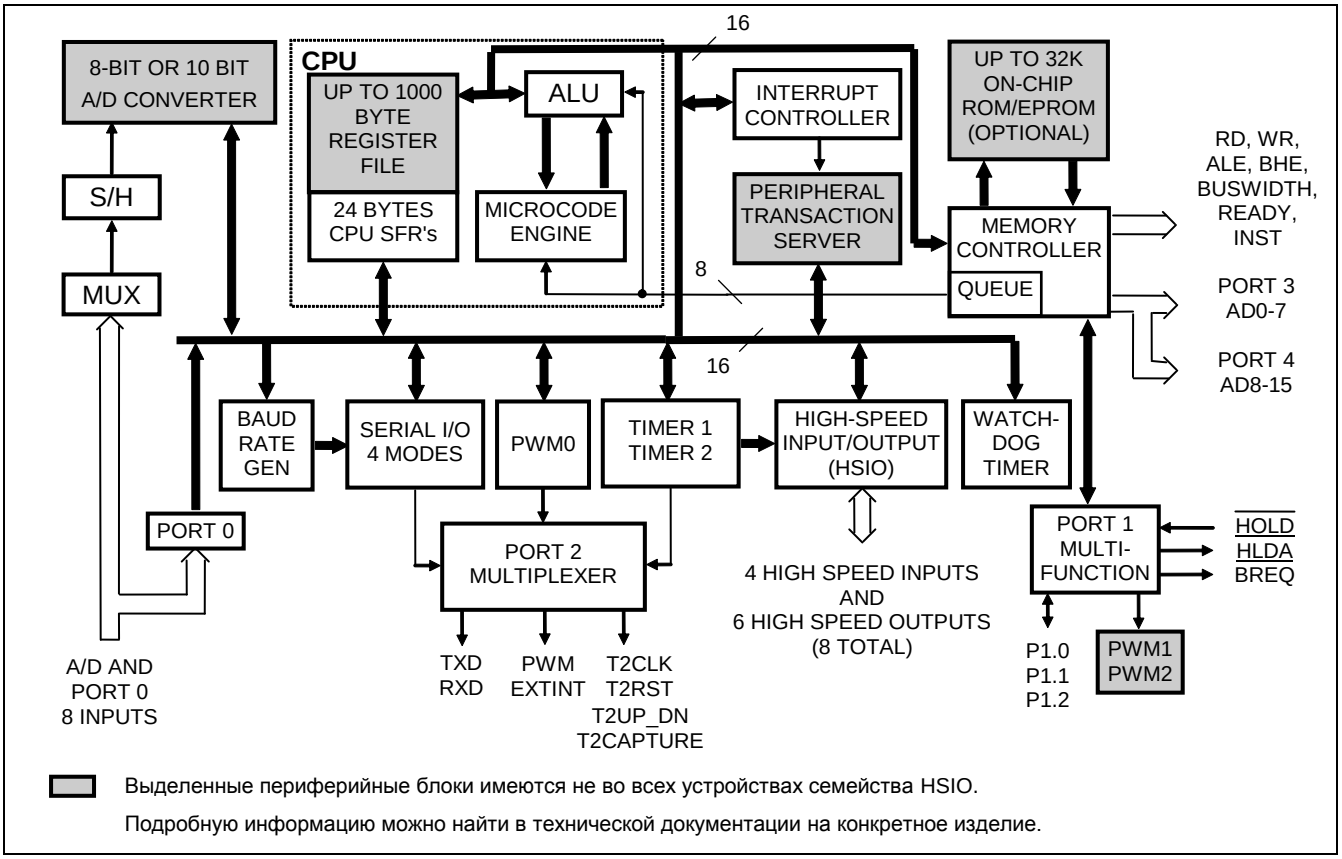
Обзор микроконтроллеров серии MCS-196

Все устройства серии MCS-196 группируются в три семейства. Семейство ***HSIO*** состоит из устройств, отличительным признаком которых является субсистема высокоскоростного ввода/вывода (HSIO). Большинство же современных моделей составляют семейство ***EPA***-микроконтроллеров. Это семейство имеет развитую периферию, отличающуюся более гибкой и быстродействующей системой ввода/вывода, выполненной на основе матрицы процессоров событий (EPA). Третье семейство - ***Motion Control*** - образуют устройства, оптимизированные для решения задач, связанных с управлением двигателями. Микроконтроллеры этого семейства также использует систему ввода/вывода типа EPA.

Рассмотрим особенности каждого семейства подробнее.

1. Семейство HSIO

Семейство HSIO состоит из микроконтроллеров типа 8XC196KB, 8XC196KC и 8XC196KD. Обобщенная структурная схема устройств этого семейства приведена ниже:



Блок-схема устройств семейства HSIO

1.1. 8XC196KB

Микроконтроллер типа 8XC196KB является первым представителем серии MCS-196, выполненным на основе CHMOS технологии. Эта модель доступна в вариантах, содержащих 8KB ПЗУ (ROM) или 8KB однократно программируемого ПЗУ (OTPROM), а также в варианте без ПЗУ. В любом исполнении микроконтроллер содержит регистровое ОЗУ емкостью 232 байта.

Как уже отмечалось, для работы с событиями устройство 8XC196KB использует блок высокоскоростного ввода/вывода типа HSIO, в котором можно запрограммировать до 4-х входных и до 6-ти выходных линий; временн***у***ю базу для них образует один из двух таймеров/счетчиков. К дополнительным особенностям этой модели относят аппаратно реализованный широтно-импульсный модулятор ***(Pulse Width Modulator - PWM),*** полный дуплексный последовательный порт ввода/вывода ***(Serial I/O Port - SIO),*** сторожевой таймер ***(Watchdog Timer)*** и 8-канальный аналого-цифровой преобразователь ***(A/D)*** с 10-разрядным разрешением.

Общее количество линий ввода/вывода ***(I/O)*** 8XC196KB - 48, причем многие из них - многофункциональные и могут использоваться либо в качестве каналов простого ввода/вывода либо как выводы периферийных устройств.

1.2. 8XC196KC

8XC196KC - следующий шаг в CHMOS-технологии серии MCS-196. Это изделие выпускается в модификациях с ПЗУ или однократно программируемым ПЗУ емкостью 16KB, а также без ПЗУ. Во всех модификациях регистровое ОЗУ имеет емкость 488 байт.

Модель 8XC196KC, работающая на частоте в 20MHz, позволяет увеличить производительность по сравнению с 16MHz устройствами на 25%.

Микроконтроллер 8XC196KC отличается от 8XC196KB усовершенствованным набором периферийных устройств: количество аппаратно реализованных широтно-импульсных модуляторов увеличено до трех, аналого-цифровой преобразователь может работать в режимах с программируемым временем выборки и преобразования при 8-ми или 10-разрядном разрешении. Кроме того, в состав микроконтроллера включен сервер периферийных транзакций ***(Peripheral Transaction Server - PTS).*** Этот блок обеспечивает значительное уменьшение загрузки центрального процессора при обслуживании прерываний - обработка запросов на прерывание осуществляется в блоке PTS на микропрограммном уровне независимо от процессора.

1.3. 8XC196KD

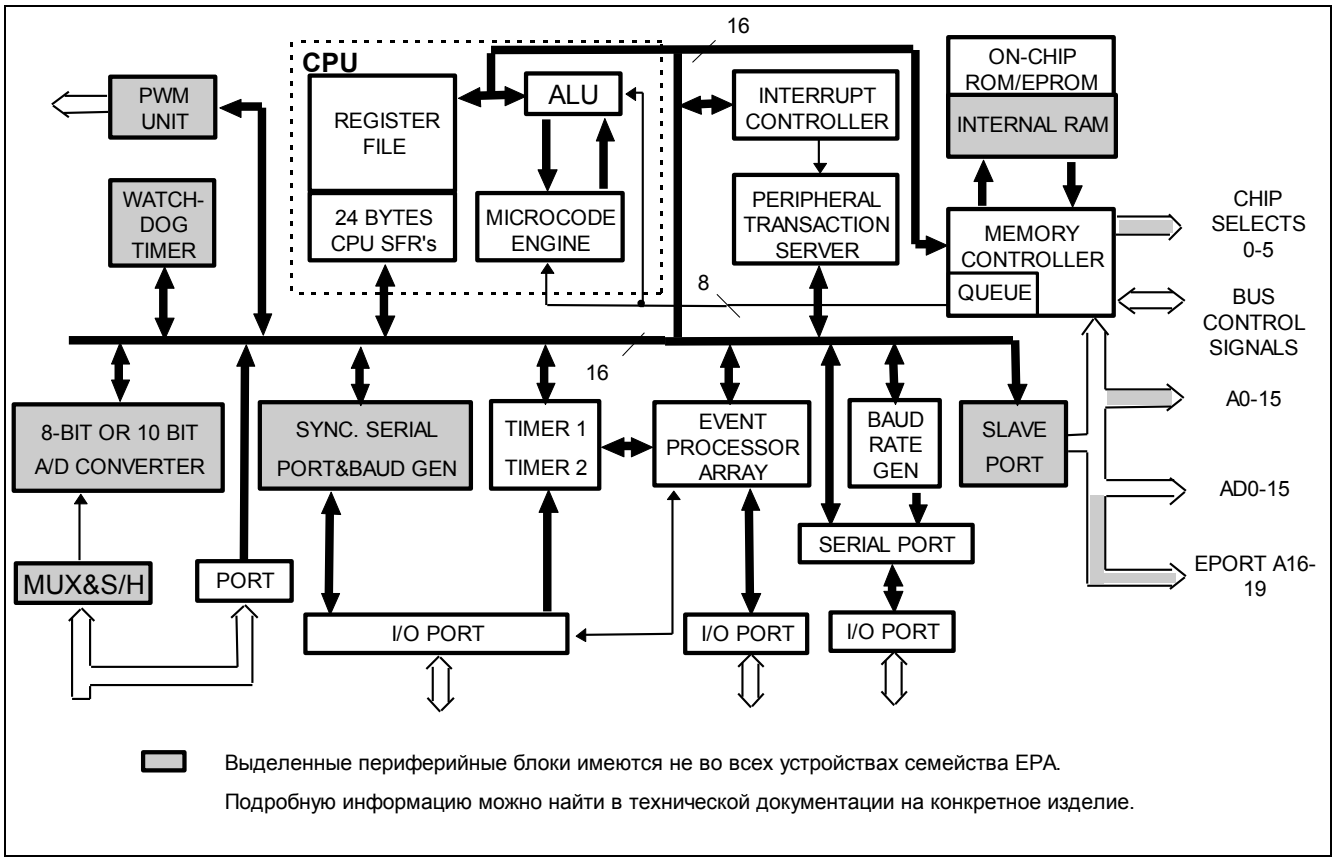
8XC196KD обладает всеми характерными признаками микроконтроллера 8XC196KC при существенно расширенной внутренней памяти. Модель 8XC196KD доступна в версиях с ПЗУ или однократно программируеммым ПЗУ емкостью 32KB при емкости регистрового ОЗУ 1000 байт. Увеличенный объем памяти делает более практичными системы проектирования программного обеспечения на языках высокого уровня. 8XC196KD предлагается с тактовой частотой 20MHz.

1.4. Ключевые особенности микроконтроллеров семейства HSIO

- Функционирование на частотах до 20MHz
- Быстродействующая архитектура типа "регистр-регистр"
- Регистровое ОЗУ емкостью до 1000 байт
- Внутреннее однократно программируемое ПЗУ емкостью до 32KB
- Динамически изменяемая разрядность шины данных - 8 или 16 бит
- Протокол захвата шины HOLD/HLDA
- 8-канальная субсистема высокоскоростного ввода/вывода (HSIO)
- 16-разрядный таймер
- 16-разрядный счетчик
- До трех широтно-импульсных модуляторов
- Полный дуплексный последовательный порт
- 16-разрядный сторожевой таймер
- 8-канальный аналого-цифровой преобразователь с разрешением 8 или 10 бит
- Пять 8-разрядных портов ввода/вывода
- Режимы холостого хода (IDLE) и пониженного энергопотребления (POWERDOWN)
- Сервер периферийных транзакций (PTS) в моделях KC и KD

2. Семейство EPA

Представителями семейства EPA являются микроконтроллеры типа 8XC196KR, 8XC196KT, 8XC196NT, 8XC196NP и 8XC196NU.



Блок-схема устройств семейства EPA

2.1. 8XC196KR

Микроконтроллер 8XC196KR содержит ПЗУ или однократно программируемое ПЗУ емкостью 16KB, 488 байт регистрового ОЗУ, а также внутреннее ОЗУ с произвольной выборкой ***(RAM)*** емкостью 256 байт. Это ОЗУ может быть использовано для хранения данных или исполняемых программ.

Для контроля событий и управления ими в состав 8XC196KR включена матрица процессоров событий (EPA). При тактовой частоте 16MHz блок EPA имеет разрешение 250ns по каждому из 10 универсальных модулей фиксации/сравнения, а также по двум дополнительным специализированным модулям сравнения. Предусмотрена возможность формирования каналами EPA сигналов широтно-импульсной модуляции (PWM).

Сервер периферийных транзакций (PTS) микроконтроллера 8XC196KR обеспечивает поддержку генерации сигналов PWM.

Новый блок в рассматриваемой модели - ведомый порт ***(Slave Port)*** - используется для связи с другими шинами системы. Этот порт упрощает разработку на базе 8XC196KR разнообразных периферийных устройств, подключаемых к шинам персональных компьютеров.

В составе 8XC196KR имеются также два последовательных порта, первый - стандартный последовательный модуль ввода/вывода ***(SIO),*** такой же, как и в микроконтроллере 8XC196KB, а второй - синхронный последовательный порт ввода/вывода ***(SSIO),*** обеспечивающий возможность полной дуплексной связи. Оба порта снабжены собственными программируемыми генераторами скорости обмена информацией.

Замечательной особенностью аналого-цифрового преобразователя 8XC196KR является наличие дополнительных по сравнению с моделью 8XC196KC режимов, с помощью которых можно программно корректировать погрешности масштаба и смещения нуля характеристики преобразования.

2.2. 8XC196KT

8XC196KT - это улучшенная версия 8XC196KR с ПЗУ или однократно программируемым ПЗУ (по выбору) емкостью 32KB, содержащая 1000-байтное регистровое ОЗУ и внутреннее ОЗУ с произвольной выборкой емкостью 512 байт. **Контроллер шины 8XC196KT реализует новые режимы, которые позволяют работать с медленной внешней памятью без тактов ожидания.**

2.3. 8XC196NT

Модель 8XC196NT отличается от 8XC196KT возможностью **адресации внешней памяти емкостью до 1MB.** Количество входов аналого-цифрового преобразователя уменьшено до четырех. За счет этого в состав микроконтроллера введен порт расширения адреса ***(Extended Adress Port - EPORT).*** Четыре вывода этого порта могут быть использованы в качестве дополнительных линий адреса (A16 - A19) или как линии стандартного порта ввода/вывода, а также как комбинация тех и других. Рабочая частота 8XC196NT - 20MHz.

2.4. 8XC196NP

8XC196NP - первый представитель серии микроконтроллеров MCS-196 **с динамическим выбором режима работы шины - с мультиплексированием или демультиплексированием адресов/данных и низким напряжением питания в рабочем режиме (3,3V при 16MHz).** Важными отличительными признаками 8XC196NP являются **блок выбора внешних устройств *(Chip Selects Unit),*** адресное пространство до 1MB, 3 выхода широтно-импульсной модуляции, функционирование на частоте 25MHz при напряжении питания 5V.

2.5. 8XC196NU

Эта модель, первые образцы которой появились на рынке в марте 1995г., отличается от изделий 8XC196NP **вдвое увеличенной тактовой частотой - 40÷50MHz** и, соответственно, более высокой производительностью. Одновременно **улучшилось разрешение каналов EPA - до 80ns и быстродействие блока PWM - до 97.6 KHz. Ускорение обработки сигналов обеспечивается благодаря специальному 32-разрядному умножителю-аккумулятору**.

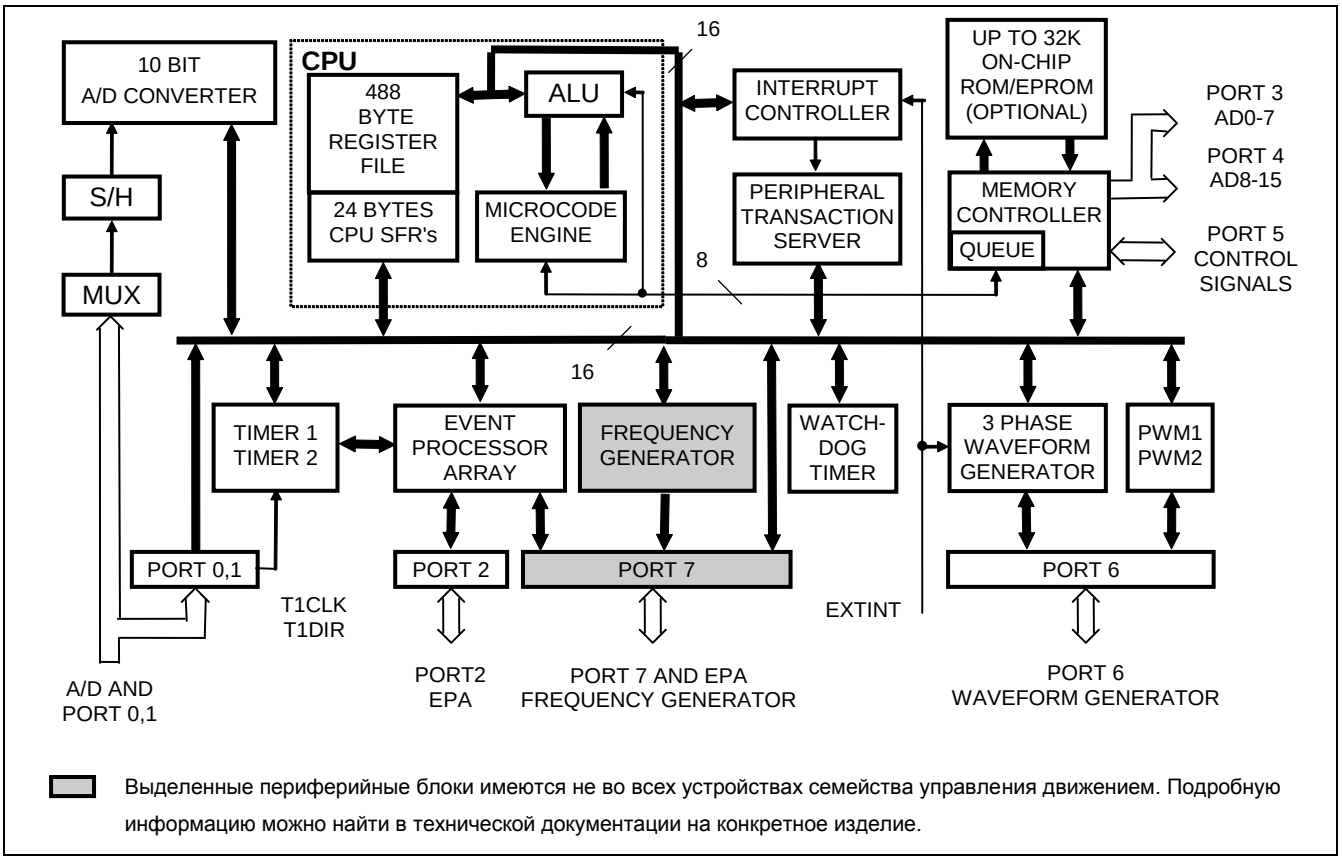
Исключение из состава микроконтроллера ПЗУ позволило резко снизить стоимость, сохранив при этом совместимость с предыдущими моделями.

2.6. Ключевые особенности микроконтроллеров семейства EPA

- Функционирование на частотах до 25MHz (до 50 MHz - только NU)
- Работа при напряжении питания 3,3V (только NP)
- Быстродействующая архитектура типа "регистр-регистр"
- Регистровое ОЗУ емкостью до 1000 байт
- Внутреннее ОЗУ с произвольной выборкой емкостью до 512 байт
- Внутреннее однократно программируемое ПЗУ емкостью до 32KB (кроме NU)
- Адресация внешней памяти до 1MB
- Динамически конфигурируемая разрядность шины данных - 8 или 16 бит
- Улучшенный контроллер шины
- Динамическое мультиплексирование/демультиплексирование шины (только NP и NU)
- Блок выборки внешних устройств (дешифратор адресов)
- Протокол захваты шины HOLD/HLDA
- 10-канальная матрица процессоров событий
- Два 16-разрядных таймера/счетчика с предделителями и режимами квадратурного счета
- Полный дуплексный последовательный порт с независимым генератором скорости обмена
- Полный дуплексный синхронный последовательный порт
- Ведомый порт для прямого межпроцессорного обмена
- 16-разрядный сторожевой таймер
- До 8-каналов аналого-цифрового преобразователя с разрешением в 8 или 10 бит
- Режимы холостого хода (IDLE) и пониженного энергопотребления (POWERDOWN)
- Сервер периферийных транзакций (PTS)
- 3 выхода PWM (широтно-импульсной модуляции) (только NP и NU)

3. Семейство микроконтроллеров для управления движением

В это семейство входят микроконтроллеры 8XC196MC и 8XC196MD.



Блок-схема устройств семейства для управления движением

3.1. 8XC196MC

8XC196MC открывает семейство микроконтроллеров серии MCS-196, предназначенных для управления движением. В этот микроконтроллер встроены периферийные устройства, оптимизированные для управления трехфазными индукционными двигателями переменного тока и инверторами напряжения. 8XC196MC доступен в модификациях с внутренними ПЗУ или однократно программируемым ПЗУ емкостью 16KB, а также без внутреннего ПЗУ. Все модификации содержат регистровое ОЗУ емкостью 488 байт.

В микроконтроллере 8XC196MC имеется уникальное периферийное устройство, называемое **генератором сигналов *(Waveform Generator - WFG),*** которое используется для формирования трехфазных широтно-модулированных сигналов.

Блок WFG генерирует три неперекрывающиеся комплементарные последовательности импульсов с управляемой длительностью (PWM сигналы) с разрешением в 125ns (при запуске по фронту) или 250ns (для центрированной оценки). Особенностями WFG-генератора являются программируемые частота, время рабочего цикла и время блокировки. Для каждой фазы блок WFG имеет по два выхода с повышенной нагрузочной способностью, при этом полярность выходных сигналов может

программироваться. Схема защиты позволяет отключать все выходы генератора WFG одновременно в ответ на внешние события.

Кроме того, 8XC196MC содержит два аппаратно реализованных широтно-импульсных модулятора, для которых используется общий программируемый источник опорной частоты, но длительность рабочего цикла для каждого выхода программируется независимо.

Для работы с событиями в 8XC196MC используется матрица процессоров событий (EPA), состоящая из 6 модулей фиксации/сравнения и 6 модулей сравнения. Блок EPA имеет разрешение в 125ns.

Аналого-цифровой преобразователь 8XC196MC представляет собой 13-канальный вариант АЦП микроконтроллера 8XC196KR. Он также работает в режимах с 8-ми или 10-разрядным разрешением, с программируемыми длительностями выборки и преобразования, предусмотрены возможности коррекциии погрешностей масштаба и смещения нуля.

Сервер периферийных транзакций (PTS) поддерживает обработку запросов на прерывание на микропрограммном уровне, не требуя вмешательства процессора. Специальный режим работы PTS обеспечивает поддержку функций последовательного ввода/вывода (SIO).

8XC196MC имеет 53 линии ввода/вывода, которые могут быть использованы встроенными периферийными устройствами. В корпусе с уменьшенным количеством выводов микроконтроллер предлагается для приложений, не требующих всех стандартных функций.

3.2. 8XC196MD

8XC196MD - новая модель в семействе микроконтроллеров MCS-196, предназначенных для управления движением. Этот микроконтроллер отличается от устройства 8XC196MC следующими усовершенствованиями:

Новый блок - генератор частот - позволяет формировать квадратурные сигналы программируемой частоты, необходимые при дистанционном управлении с использованием инфракрасных каналов связи.

В блок EPA дополнительно введены два модуля фиксации/сравнения и два модуля сравнения, что расширяет возможности восприятия и генерации событий.

Добавлены также 8 дополнительных линий ввода/вывода, 2 линии ввода и 1 вход аналого-цифрового преобразователя.

Микроконтроллер 8XC196MD совместим по выводам с устройством 8XC196MC, позволяя легко усовершенствовать существующие разработки.

3.3. Ключевые особенности микроконтроллеров семейства Motion Control

- 3-фазный генератор сигналов широтно-импульсной модуляции
- Генератор частот (только для MD)
- Функционирование на частотах до 16MHz
- Быстродействующая архитектура типа "регистр-регистр"
- Регистровое ОЗУ емкостью до 488 байт
- Внутреннее ПЗУ или однократно программируемое ПЗУ емкостью 16KB
- Динамически конфигурируемая разрядность шины данных - 8 или 16 бит
- Протокол захваты шины HOLD/HLDA
- 12-канальная матрица процессоров событий
- Два 16-разрядных таймера/счетчика с предделителями и режимами квадратурного счета
- 16-разрядный сторожевой таймер
- До 14 каналов аналого-цифрового преобразователя с разрешением в 8 или 10 бит
- До 8-ми 8-разрядных портов ввода/вывода
- Режимы холостого хода (IDLE) и пониженного энергопотребления (POWERDOWN)
- Сервер периферийных транзакций (PTS)

Знакомство с микроконтроллерами серии MCS-196 завершим сводной таблицей, в которой представлены основные характеристики наиболее распространенных моделей микроконтроллеров.

Устройство	ROM/ OTPROM (KB)	Регистр. ОЗУ (байт)	ОЗУ программ (байт)	Таймер/ счетчик	Каналы аналог. ввода	Линии ввода/ вывода	Процессор событий	Послед. порт	Рабочая частота (MHz)	Тип корпуса	Режим внутри-схемной эмуляции	Адресное прост-ранство	Ключевые особенности
8XC198	8K	232	НЕТ	2	0, 4	48	HSIO	1	16	N-52, S-80	ЕСТЬ	64K	8-битная шина; версия KB с 0 или 4 каналами АЦП
8XC196KB	8K	232	НЕТ	2	8	48	HSIO	1	12, 16	N-52, S-80	ЕСТЬ	64K	малое потребление, CMOS-технология
8XC196KC	16K	488	НЕТ	2	8	48	HSIO	1	16, 20	N-52, S-80, SB-80	ЕСТЬ	64K	PTS, PWM, 16K OTPROM, 488 байт RAM
8XC196KD	32K	1000	НЕТ	2	8	48	HSIO	1	16, 20	N-52, S-80, SB-80	ЕСТЬ	64K	версия KC с 32K OTPROM и 1000 байт RAM
8XC196MC	16K	488	НЕТ	2	13	53	8EPA	1	16	N-52, S-80, U-64	ЕСТЬ	64K	PTS , PWM , 3-фазный генератор сигналов
8XC196MD	16K	488	НЕТ	2	14	64	12EPA	1	16	N-52, S-80, U-64	ЕСТЬ	64K	развитие MC с генератором частоты
8XC196KR	16K	488	256	2	8	56	10EPA	2	16	N-68	ЕСТЬ	64K	гибкий ввод/ вывод с использованием матрицы процессоров событий (EPA)
8XC196KT	32K	1000	512	2	8	56	10EPA	2	16	N-68	ЕСТЬ	64K	версия KR с развитым контроллером шины
8XC196NT	32K	1000	512	2	4	56	10EPA	2	20	N-68	ЕСТЬ	1M	1Мбайт линейного адресного пространства
8XC196NP	4K	1000	НЕТ	2	0	33	4EPA	1	25	S-100, SB-100	ЕСТЬ	1M	питание 3.3V, 6 сигналов Chip Select, Mux/Demux динамическая шина, 3 PWM, PTS
8XC196NU	НЕТ	1000	НЕТ	2	0	32	4EPA	1	40,50	S-100, SB-100	ЕСТЬ	1M	то же, что NP + 32-разр. MAC, до 50 MHz

Приведенной информации, как правило, достаточно для принятия решения о целесообразности применения микроконтроллеров серии MCS-196 в планируемых разработках, а может быть, и для выбора определенного семейства (HSIO, EPA или MC) или даже конкретной модели микроконтроллера. Однако, очевидно, что до того, как приступить к проектированию аппаратных и программных средств новой системы, необходимо более основательно изучить особенности архитектуры микроконтроллера, выполнить пробные задания, на качественном уровне оценить эффективность предполагаемых результатов, а лучше - получить хотя бы ориентировочные количественные оценки характеристик реализуемого проекта. Для этого существует несколько способов:

- традиционный - по литературным источникам изучить архитектуру и на основании имеющихся опыта и интуиции приступить к проектированию, надеясь "угадать" желаемые результаты. По-видимому, такой подход чреват б***о***льшим количеством ошибок и меньшей степенью оптимальности решений при значительных затратах времени.

- изучать микроконтроллеры с помощью гипертекстовых элекронных справочников и руководств, а требуемые проверки выполнять с использованием программных систем моделирования. Очевидно, это позволит существенно быстрее и полнее ознакомиться с особенностями архитектуры микроконтроллеров, отмоделировать некоторые программные модули, уменьшив вероятность ошибок.

- и освоение элементной базы, и начальные этапы проектирования выполнять с помощью интегрированных пакетов, содержащих, кроме упомянутых выше средств, демонстрационные и обучающие "инструменты", аппаратно-программные моделирующие и отладочные системы и т.п.

Рассмотрим вкратце средства, доступные для реализации каждого из подходов к проектированию.

4. Документация

Корпорация Intel выпускает исчерпывающую техническую документацию на все изделия серии MCS-196: справочные данные ***(Data Sheets)***, краткие справочные руководства ***(Quick Reference Fact Sheets)***, различные материалы по архитектуре (путеводители, обзоры, справочники по системе команд и т.п.), описания инструментальных средств проектирования ***(Tools)***, пособия и руководства по применению ***(User's Guides and Manuals)***.

5. Средства поддержки

Для микроконтроллеров серии MCS-196 возможна полная эмуляция создаваемых приложений. Необходимые для этого средства выпускаются как корпорацией Intel, так и многими фирмами, специализирующимися в области создания инструментальных средств. Достаточно полную систему проектирования приложений на базе микроконтроллеров MCS-196 предлагает фирма ***BSO Tasking***:

-
- Макроассемблер ***ASM96***
- Компилятор ***C96***
- Компоновщик объектных модулей и релокатор ***RL96***
- Преобразователь объектного кода в шестнадцатиричный ***OH96***
- Библиотека подпрограмм обработки 32-разрядных операндов с плавающей точкой

Ассемблеры и С-компиляторы предлагают также фирмы ***Avocet Systems***, ***Archimedes Software*** и др. Для многих приложений можно воспользоваться готовыми операционными системами реального времени ***(U.S.Software, BSO Tasking*** и др.), многозадачными операционными системами ***(CMX Company, Forth***). Фирмой ***U.S.Software*** создана библиотека подпрограмм ***GoFast 80196*** для обработки операндов в формате с плавающей точкой, а фирмой ***Tektronix*** - дизассемблеры. Для отладки программ несколько фирм предлагают симуляторы, эмуляторы EPROM, отладчики-анализаторы, а для комплексной отладки приложений - разнообразные внутрисхемные эмуляторы и логические анализаторы с набором адаптеров, клипс и т.п. Окончательная подготовка изделий к применению осуществляется с помощью программаторов фирм ***Data I/O, Logical Devices, SMS*** и многих других.

Даже беглое (и неполное!) перечисление средств, предлагаемых для изучения, освоения и применения микроконтроллеров серии MCS-196, убедительно демонстрирует, что решение прикладных задач с использованием этих изделий корпорации Intel обеспечивается мощной и разносторонней поддержкой на всех стадиях проектирования, и это, безусловно, существенно облегчает и ускоряет создание новых эффективных систем. С другой стороны, столь широкий спектр инструментальных средств, ориентированных именно на серию MCS-196 и созданных многими солидными фирмами, свидетельствует об огромной популярности микроконтроллеров этой серии и их высокой эффективности в самых разных областях применений.

Как выбрать подходящие средства проектирования? Задача, на первый взгляд, очень не простая при таком разнообразии возможностей, решается достаточно просто. Корпорация ***Intel*** для начинающих предлагает недорогой комплект разработчика ***ProjectBuilder***, который является средой, содержащей и программные, и аппаратные инструментальные средства, эффективные на первых этапах освоения микроконтроллеров серии MCS-196 и полезные в дальнейшей профессиональной деятельности.

В комплект ***ProjectBuilder*** входят:

- экспертная система ***ApBUILDER***;
- система моделирования приложений ***ModelBUILDER***;
- гипертекстовое руководство по микроконтроллерам ***196KC/KD Fact Sheet***;
- технические характеристики микроконтроллеров ***Data Sheet*** (электронные справочники***)***;
- макроассемблер ***DEMOASM96***;
- гипертекстовое руководство программиста по языку ассемблера ***ASM96 Manual***;
- символьный монитор-отладчик ***Debug Monitor,*** совместимый с разными типами оценочных модулей (***Evualation Board*** или ***Target Board***);
- гипертекстовое руководство пользователя и система помощи для систем ***ApBUILDER***, ***ModelBUILDER*** и ***Debug Monitor***;
- ***исходные тексты 9-ти программ-шаблонов*** для анализа производительности;
- ***модуль проектирования 196KD20 Target Board;***
- ***руководство по аппаратной части модуля проектирования*** с принципиальными схемами в формате ***OrCAD;***

- принципиальная схема в формате ***OrCAD*** для 16-разрядной платы расширения векторной памяти;

- ***библиотека OrCAD*** для 114-ти встраиваемых процессоров корпорации Intel;

- ***исходный текст программы-монитора*** с сокращенной системой команд ***iRISM***.

Кроме того, могут поставляться демонстрационная версия компилятора ***C96*** с примерами программ, утилиты компоновщика (релокатора), построителя и т.п. Комплект ProjectBuilder регулярно расширяется и пополняется информационными материалами и руководствами на другие изделия серии MCS-196, созданы оценочные модули на базе микроконтроллеров семейств EPA и MC, усовершенствуется и программное обеспечение. Некоторые утилиты комплекта, например, ApBUILDER, поддерживают и другие компоненты фирмы Intel, в частности устройства серий 51,251,186 и 386EX.

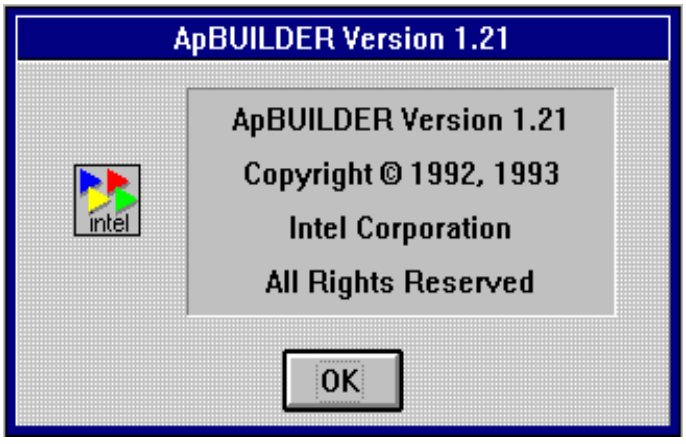
Таким образом, ProjectBuilder представляет собой достаточно мощную интегрированную универсальную среду для работы с микроконтроллерами серии MCS-196. Следует также отметить чрезвычайно удобный и дружественный интерфейс пользователя, обеспечивающий возможность самостоятельного изучения компонентов серии MCS-196 даже начинающим разработчикам. Поэтому мы рекомендуем приступать к работе с микроконтроллерами этой серии, используя в качестве основного инструмента комплект ProjectBuilder. Чтобы помочь русскоязычному пользователю быстрее освоиться с англоязычным пакетом программ, ниже приводится авторизованный перевод материалов, касающихся особенностей работы в среде ProjectBuilder, оформленный как краткое руководство по применению основных утилит комплекта, а именно, ApBuilder, ModelBuilder и Debug Monitor.

ApBuilder. Введение

ApBUILDER - это мощная экспертная система для встраиваемых микроконтроллеров фирмы ***Intel***, в которой содержится вся информация, необходимая не только при изучении микроконтроллеров, но и при проектировании приложений на их основе. С помощью "мыши" ***ApBuilder*** позволяет быстро и легко получить информацию о назначении любого блока, регистра, команды, бита, о формате и времени выполнения команд и многих других свойствах микроконтроллера и его составных частей. Каждый экран ***ApBuilder***'а чувствителен к контексту и связан с гипертекстовым руководством и техническими характеристиками процессора, к которым обеспечивается мгновенный доступ. Более того, ***ApBuilder*** с помощью одной только "мыши" позволяет **генерировать текст программы (!)** как на языке ***C96***, так и на языке ассемблера ***ASM96***, настраивая окружение и отображая все необходимое в соответствующих окнах дисплея. Так называемые маски программирования предотвращают ошибки и позволяют разрабатывать приложения непосредственно в процессе изучения архитектуры ***MCS-196***.

1. Как запустить программу "*ApBuilder"?*

ApBuilder работает в операционной среде *Windows*, поэтому, если Вы находитесь в среде ***DOS***, наберите на клавиатуре *"win"* и нажмите клавишу "Enter". После того, как загрузилась среда "Windows", найдите в окне ***"Диспетчер Программ"*** группу программ с именем "***Intel Microcontrollers***". Активизируйте эту группу двукратным нажатием левой клавиши "мыши", а затем - в развернутом окне группы щелкните два раза по "иконке" с именем ***"ApBuilder XX".*** Через некоторое время в центре экрана появится заставка



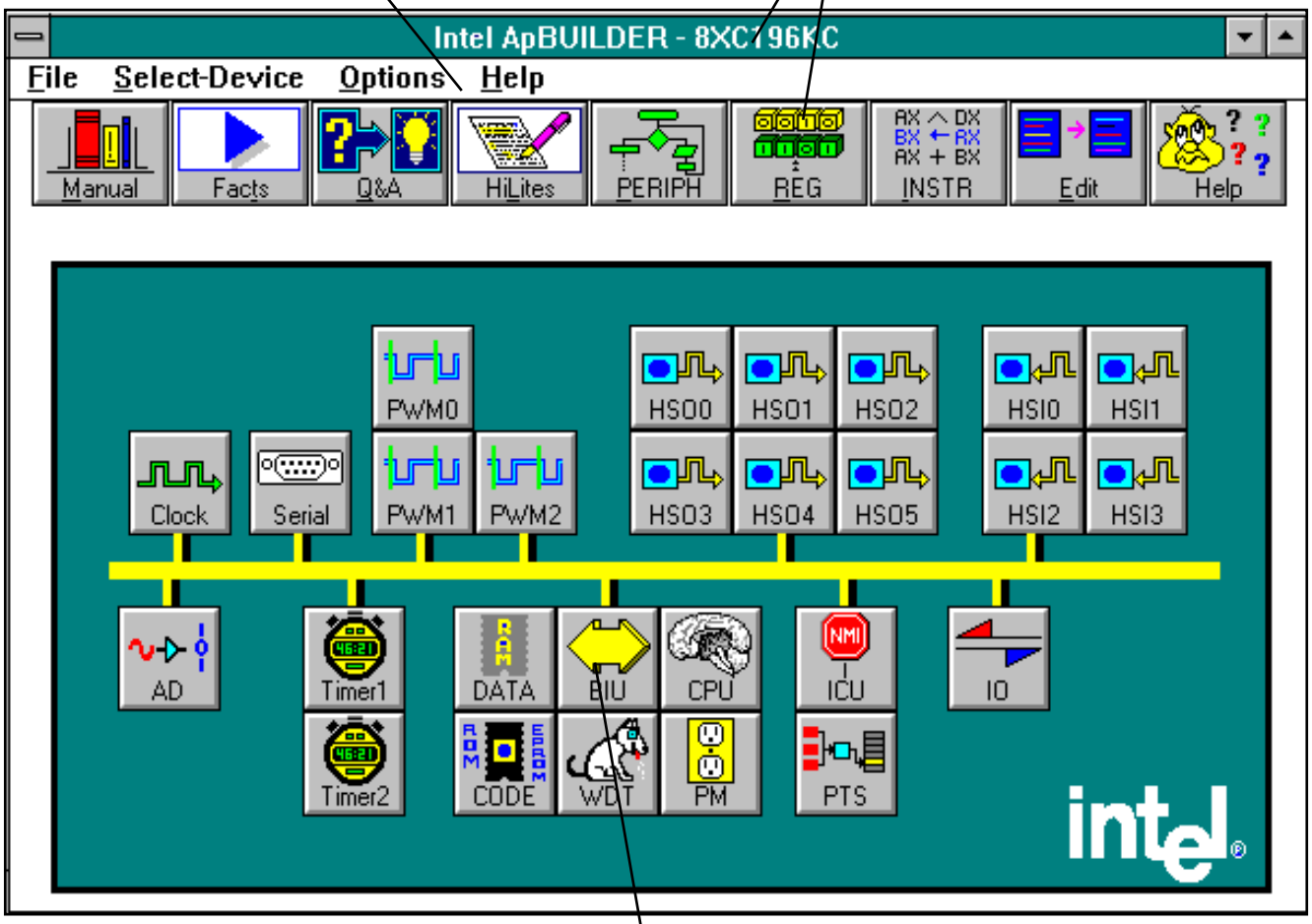
и Вы можете приступать к работе, "нажав" на экране кнопку ***"OK".***

После того, как Вы выполнили описанные выше операции, на экране развертывается рабочее поле:

Строка главного меню управления ***ApBUILDER'ом***

Строка заголовка с информацией о типе изучаемого микроконтроллера

Строка "горячих клавиш" управления ***ApBUILDER'ом***



а это один из функциональных блоков выбранного нами микроконтроллера 8XC196KC

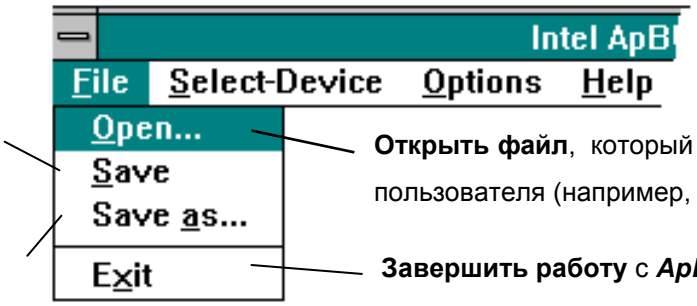
2. Команды меню *ApBuilder'a*

В главном меню щелчком кнопки "мыши" Вы можете выбрать:

File	**- Работа с файлами**
Select-Device	**- Выбор устройства**
Options	**- Опции (Настройка)**
Help	**- Справочники (Помощь)**

Выбор любого из пунктов главного меню, в свою очередь, открывает соответствующее подменю:

File - Работа с файлами:



Сохранить настройки пользователя в файле

Сохранить настройки пользователя **в новом файле**

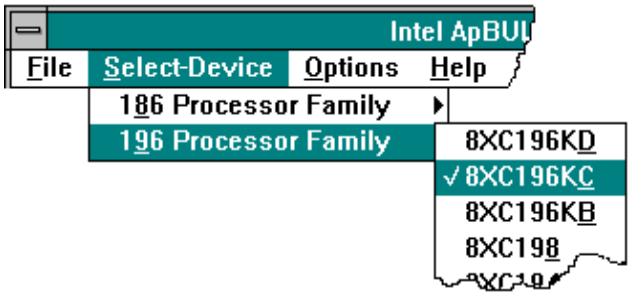
Открыть файл, который хранит настройки пользователя (например, my196kc.sav)

Завершить работу с ***ApBuilder'ом***

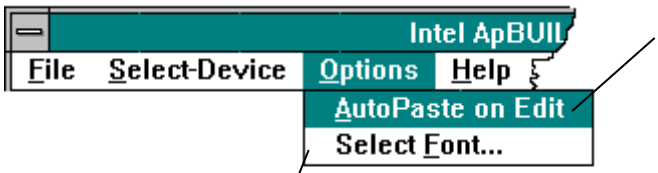
Select-Device - Выбор микроконтроллера:

Из предложенных серий (186, 196, 51 и т.п.) выберите интересующую Вас, после чего отметьте нужный тип микроконтроллера (например, 80C196KC).

На экране это выглядит примерно так:



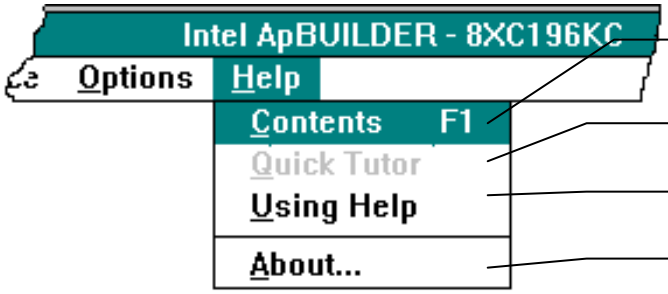
Options - Опции:



Автоматическая Вставка содержимого буфера обмена в текст программы (что находится в буфере, будет ясно позже)

Выбор шрифта, его размера и цвета (только для символов на структурной схеме)

Help - Помощь:



Содержание Справки

Краткий учебник по ***ApBuilder'y***

Использование справочника

Информация о программе (наименование, разработчик, версия, дата выпуска)

3. Функциональная клавиатура *ApBuilder'a*.

Функциональная клавиатура состоит из 9 "иконок":

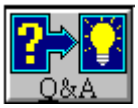
Manual	**- Руководство по применению**
Facts	**- Технические характеристики**
Q & A	**- Вопросы и Ответы**
HiLites	**- Краткая информация по узлам микроконтроллера**
PERIPH	**- Редактор периферии**
REG	**- Редактор регистров**
INSTR	**- Редактор команд**
Edit	**- Редактор текста (Блокнот)**
Help	**- Справочник**

Manual - Руководство по применению

"Иконка" ***"Manual"*** позволяет получить доступ к информации из "Руководства по применению". Первый кадр ***Manual*** активизируется при помощи "иконки" ***"Manual"***; последующие кадры открываются кнопкой ***"Manual"***. Если Вы предварительно выбрали какой-либо блок микроконтроллера (щелчком кнопки "мыши"), а затем щелкнули по "иконке" ***"Manual"***, то ***ApBuilder*** отобразит информацию о выбранном Вами блоке. Аналогично можно вызвать необходимый раздел из "Руководства по применению", отметив соответствующее поле в каком-либо редакторе ***ApBuilder'a*** и нажав **кнопку"*"Manual"*** в рабочем поле этого редактора. В процессе работы с Руководством доступны стандартные функции гипертекстовых справочников (Содержание, Поиск, Закладка, Аннотация и др.).

Facts - Технические характеристики

"Иконкой" ***"Facts"*** открывается меню обзорной информации по всем изделиям серии. Здесь Вы можете также узнать об инструментальных средствах, предлагаемых фирмой Intel для разработки программного обеспечения, внутрисхемных эмуляторах ***(ICE)*** и др., о ценах на эти изделия.

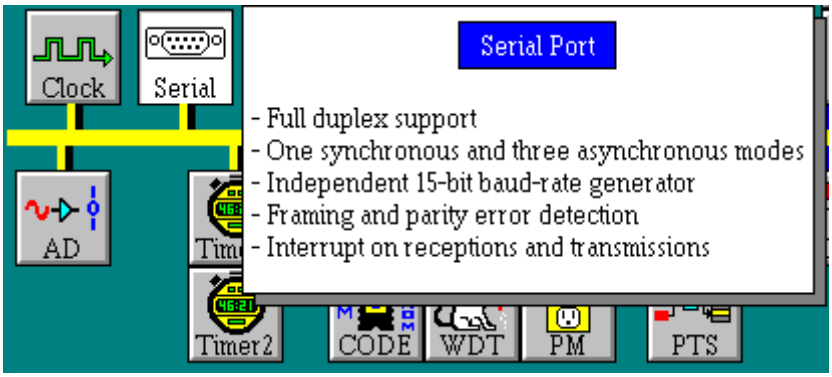


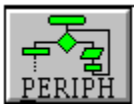
Question's & Answer's - Вопросы и Ответы

"Иконка" ***"Q & A"*** отсылает Вас к разделу ***ApBuilder'***а, который содержит ответы на вопросы, возникавшие при разработке прикладных систем у Ваших предшественников. При щелчке по "иконке" ***"Q & A" ApBuilder*** приводит список разделов для каждого функционального блока микроконтроллера. Если же Вы выбрали блок до того, как щелкнули по иконке "Q & A", то ***ApBuilder*** сразу перейдет к разделу, относящемуся к активизированному блоку.

HiLites - Краткая информация о блоках микроконтроллера

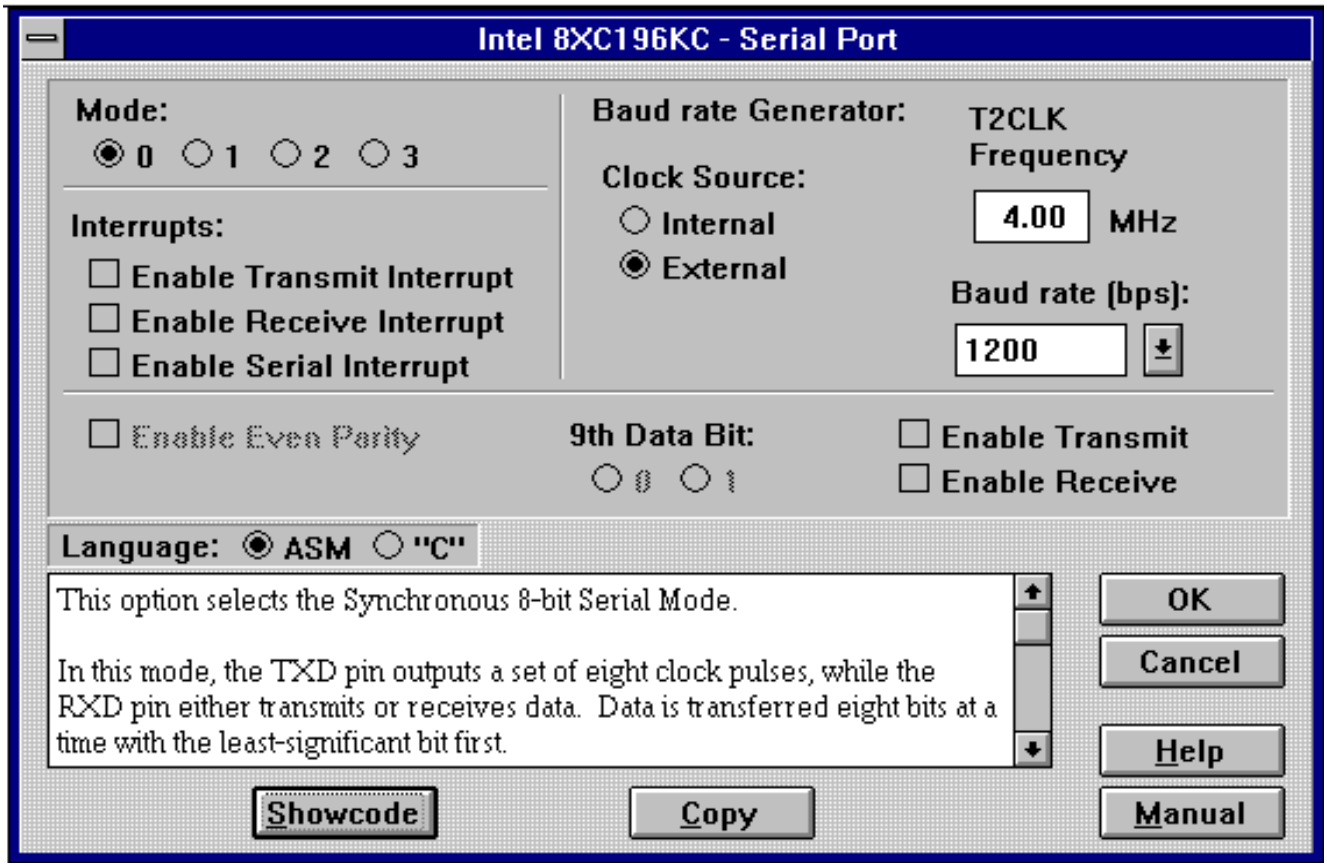
"Иконка" ***"HiLites"*** выдает краткую справку о выбранном блоке. ***ApBuilder*** не отреагирует на щелчок по "иконке" ***"HiLites"***, пока Вы не отметите какой-либо блок микроконтроллера - в приведенном ниже примере - последовательный порт.



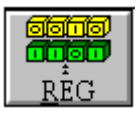


PERIPH - Редактор периферии

"Иконка" ***"PERIPH"*** вызывает редактор периферии для работы с выбранным блоком. Если Вы предполагаете не только посмотреть, как управлять теми или иными функциональными узлами микроконтроллера, но и создать программу для последующей компиляции и отладки в среде ***"Debug Monitor***", необходимо в главном меню ***ApBuilder***'а в пункте ***"Options"*** отметить режим ***"Autopaste on Edit".***

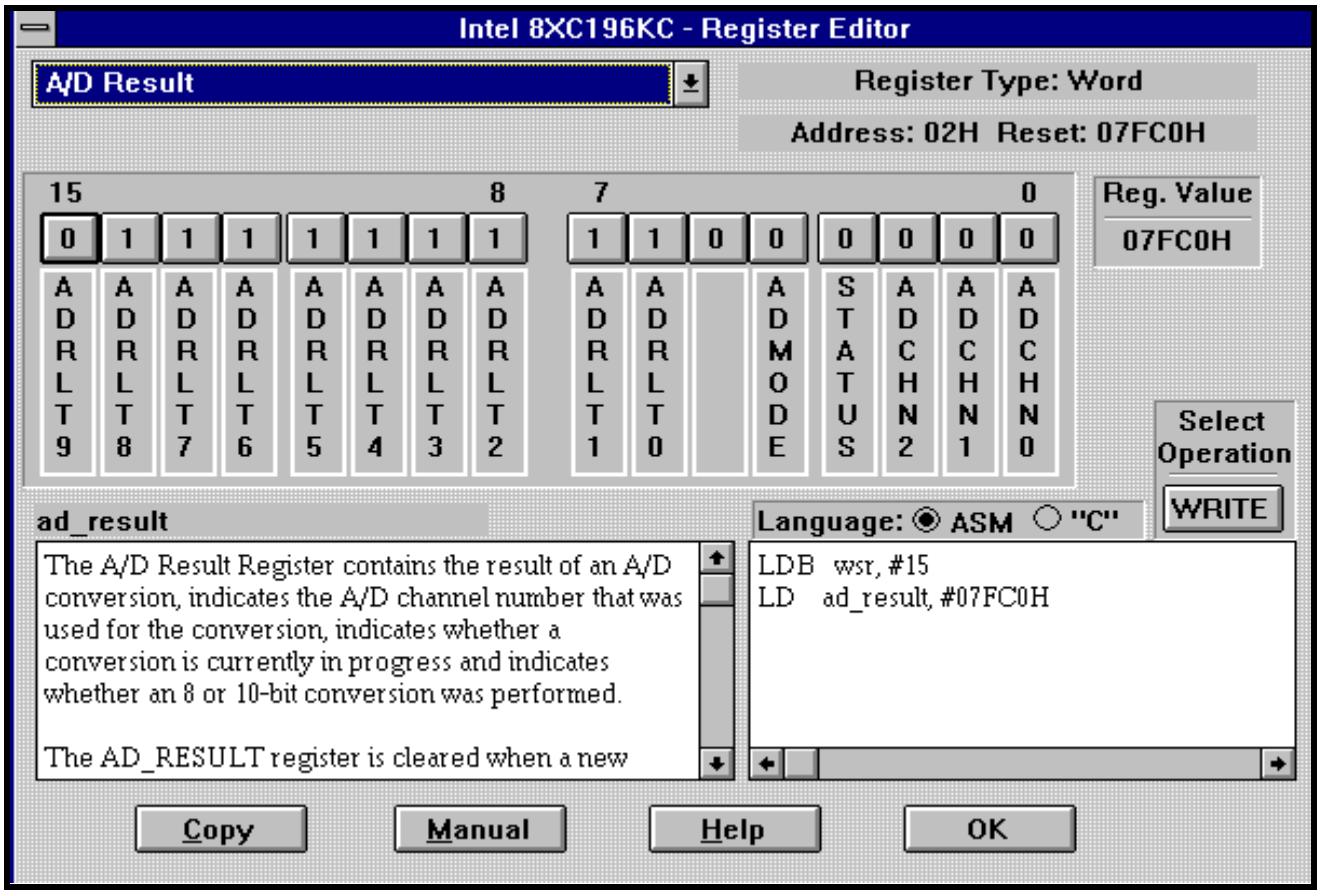


Имеется возможность изменения параметров режима работы выбранного блока - в нашем примере - установить необходимую тактовую частоту процессора, режим обмена данными и их формат, скорость передачи информации для порта, разрешить или запретить прерывания и т. д. Кроме того, редактор позволяет сгенерировать практически законченную программу для выполнения выбранной Вами функции блока. Предварительно необходимо указать, на каком языке Вы хотите реализовать эту функцию - на языке ***Ассемблера*** или на ***C***. Выбор языка осуществляется в окне ***"C" - "ASM".*** Для просмотра сгенерированного кода достаточно нажать на кнопку ***"Showcode",*** а для переноса его в буфер обмена, в котором можно скомпоновать несколько программных модулей, реализующих различные функции соответствующих блоков - кнопку ***"Copy".*** Сформированную программу можно сохранить в файле с предлагаемым системой расширением ***"*.c"*** или ***"*.a96".*** В любой момент времени Вам доступны соответствующие справочные данные по работе с редактором (***Help***) и руководство по применению (***Manual***).



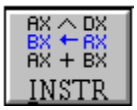
REG - Редактор регистров

Редактор регистров отличается от редактора периферии тем, что позволяет работать не с блоками микроконтроллера, а непосредственно с их регистрами.



После выбора конкретного регистра из предлагаемого системой набора (левое верхнее окно редактора) Вы можете получить исчерпывающую информацию о назначении регистра и его отдельных разрядов, адресе регистра и его содержимом после сброса, изменить состояние любых доступных разрядов и т.п.

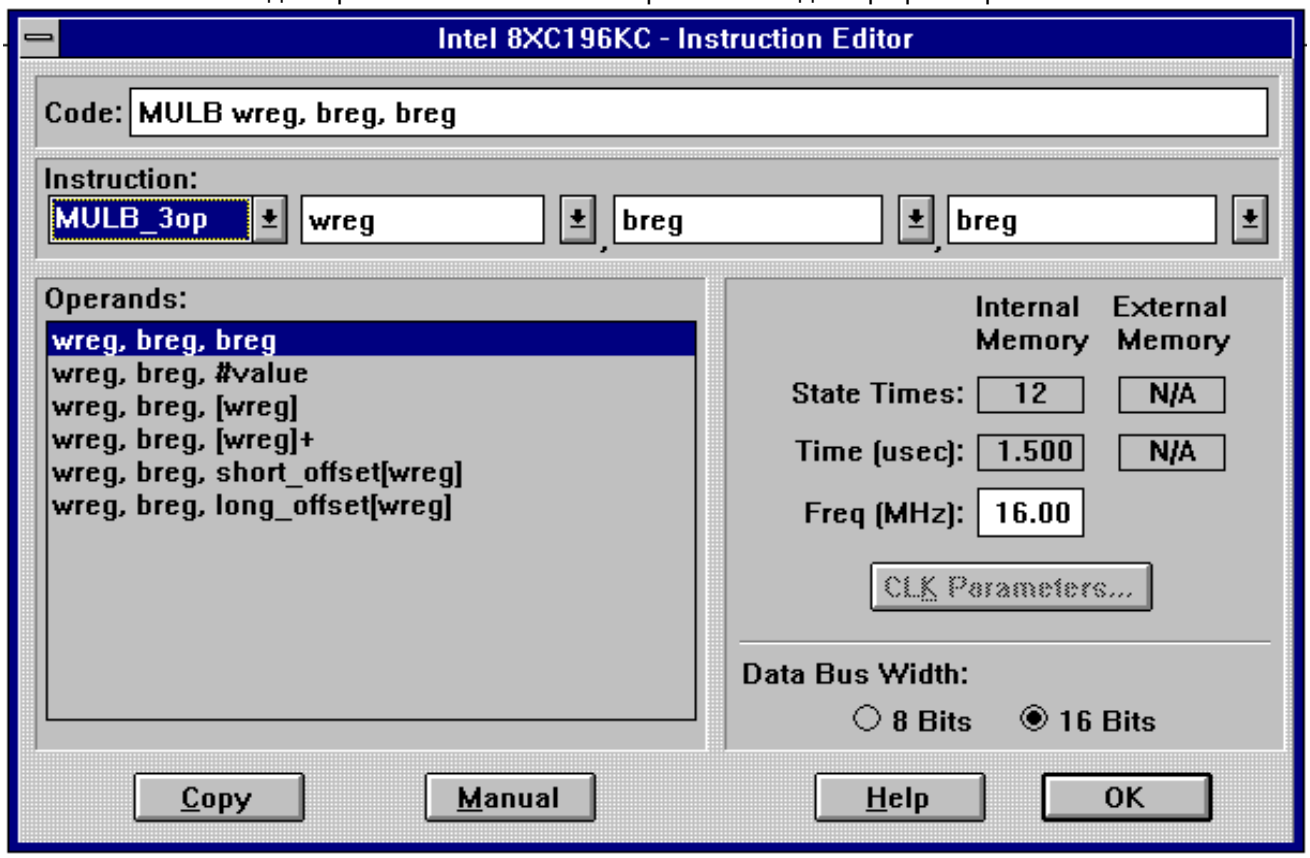
Кроме того, с помощью кнопки в окне "***Select Operation***" можно посмотреть, в каком окне регистрового файла доступен регистр по записи или чтению, какими инструкциями можно реализовать те или иные логические операции (***AND, XOR***, ***TestZ...)*** и т.д. Редактор регистров не имеет клавиши "***Showcode***", так как генерируемый код отображается здесь же в соответствующем окне экрана. Для того чтобы скопировать сгенерированный системой код из этого окна в буфер обмена для последующего использования в блокнотном или ином редакторе, Вам необходимо нажать кнопку ***"Copy***".



INSTR - Редактор команд

"Иконка" ***"INSTR"*** вызывает Редактор команд, в котором можно посмотреть мнемонику инструкции, исследовать особенности выполнения команд при разных методах адресации, оценить количество машинных циклов и время исполнения команды при разных тактовой частоте и разрядности шины данных и т.д.

Работа с этим Редактором мало отличается от работы в Редакторе регистров.



Edit - Редактор текста (Блокнот)

"Иконка" ***"Edit"*** обеспечивает доступ к текстовому редактору. Здесь Вам предоставляется возможность просмотреть и отредактировать скомпонованный из других редакторов ***ApBuilder***'а текст программы.

Существуют два метода использования блокнота для пересылки команд, сгенерированных ***ApBuilder***'ом в Редакторах Команд, Регистров и Периферии в создаваемый Вами файл. Сначала Вы должны скопировать код из упомянутых редакторов в буфер нажатием кнопки ***"COPY".*** Далее искомый результат достигается следующим образом:

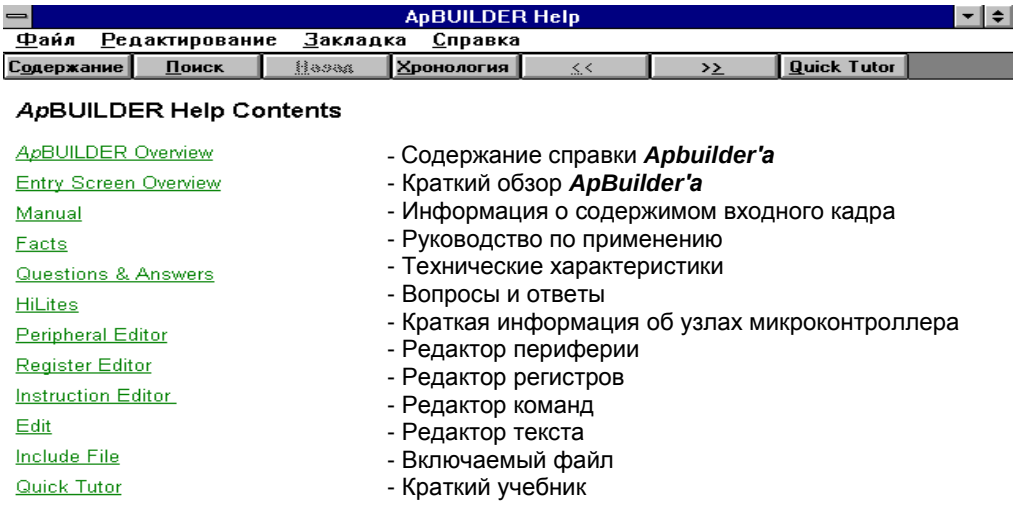
1. Если Вы **установили опцию *"Autopaste on Edit"*** и щелкнули по иконке ***"Edit",*** чтобы обратиться к Вашему файлу программы, то ***ApBuilde*r автоматически вставит** из буфера **содержимое выделенного текста в блокнот**, начиная с текущей позиции курсора.

2. Если Вы **не установили опцию *"Autopaste on Edit"*** и щелкнули по иконке ***"Edit",*** чтобы обратиться к создаваемому файлу программы, то ***содержимое буфера Вам* придется вставлять самостоятельно**, используя средства блокнотного или другого редактора.

Используйте в меню *"Options"* режим *"Autopaste on Edit"* для указания метода, который Вы предпочитаете.

Help - Справочник (Помощь)

Активизация "иконки" ***"Help"*** представляет Вашему вниманию небольшой справочник,



ознакомившись с которым Вы быстрее постигнете все возможности программы ***ApBuilder***.

Используя ***Help***, Вы получите исчерпывающую информацию по всем возможностям экспертной системы "***ApBuilder***", о многих из которых мы даже не упомянули в этом кратком ПУТЕВОДИТЕЛЕ, рассчитывая, что для начала Вам будет достаточно приведенных выше сведений, а с остальным - справитесь самостоятельно.

4. Что дальше?

*Итак, в среде **ApBuilder** Вы познакомились с архитектурой микроконтроллера, сделали предварительный выбор семейства, а возможно, и типа микроконтроллера, поняли, как реализовать требуемые в Вашем приложении функции и, наверное, попробовали даже скомпоновать из предлагаемых **ApBuilder**'ом масок фрагменты программ для Вашего проекта.*

*Что же дальше? Можно посмотреть, как выполняется спроектированная программа - для этого надо научиться работать с отладчиком **Debug Monitor**. А можно попытаться оценить эффективность выбранного Вами микроконтроллера в создаваемой системе с помощью программы **ModelBUILDER**. Более того, если Вы знаете основные функции создаваемого приложения, **ModelBUILDER** позволит уточнить требуемую тактовую частоту микроконтроллера, разрядность шины, быстродействие памяти и некоторые другие характеристики разрабатываемой системы. Это, как правило, уменьшает количество итераций при проектировании изделия, если Вы стремитесь его оптимизировать.*

*Поэтому мы предлагаем вначале познакомиться с **ModelBUILDER**'ом, а уж затем перейти к освоению средств отладки - **Debug Monitor**'а.*

ModelBuilder. Введение

Система моделирования ***ModelBUILDER*** предназначена для оценки производительности систем, разрабатываемых на базе микроконтроллеров ***MCS-196.*** В состав системы входит программа ***mbdl96kd.exe*** и аппаратные средства - оценочные (демонстрационные) модули, с помощью которых еще до того, как создаваемый проект будет реализован "в железе" (и даже до разработки его принципиальной схемы!) Вы можете проанализировать, справится ли микроконтроллер при заданных тактовой частоте, разрядности внешней шины и числе тактов ожидания с поставленными перед ним задачами. Более того, программы-шаблоны, поставляемые в составе ***ModelBUILDER'***а, позволяют выполнить этот анализ до того, как будет написана программа, которую должен выполнять микроконтроллер! Для этого пакет ***ProjectBUILDER*** комплектуется девятью **программами-шаблонами с аннотированным исходным текстом,** а также демонстрационной версией Макроассемблера **ASM96** фирмы **Intel.**

Как уже отмечалось, в систему ***ProjectBUILDER*** входит **монитор-отладчик *Debug Monitor***, функционирующий по запросам низкого уровня вместе с оценочным модулем **(Target Board, Demo Board** или **Evualation Board**). Ядро монитора, зашитое в ПЗУ микроконтроллера, позволяет опрашивать его регистры и ячейки памяти и модифицировать их во время работы CPU. Отлаживаемые программы загружаются из персонального компьютера через последовательный порт со скоростью до 57,6 Кбод. По мере развития Вашего приложения Вы можете заменить статическое ОЗУ в оценочном модуле на запрограммированное ПЗУ (EPROM или FLASH) и эксплуатировать модуль как автономное устройство. Через внешние разъемы модулей доступны интерфейс высокоскоростного ввода/вывода, аналоговые входы, цифровые входы/выходы и внешняя память. Перечисленные функции отладчика обеспечивают возможность достаточно полной отладки алгоритма и основных программных модулей приложения, так что к моменту изготовления макета Вашего устройства в него можно будет загрузить отлаженную программу. Ядро монитора-отладчика настраивается на микроконтроллеры типов 196KB/KC/KD с любой тактовой частотой.

1. Использование шаблонов производительности при моделировании приложений

Набор подпрограмм, поставляемых с ***ModelBUILDER***'ом, разработан так, чтобы обеспечить разработчика разнообразными базовыми примерами программ, которые могут использоваться для построения моделей различных приложений. Для того, чтобы модели Вашей системы имели смысл, Вы должны хорошо понимать, **что** делает каждая из поставляемых подпрограмм и **как** ***ModelBUILDER*** использует их на системном уровне для получения информации о производительности Вашей прикладной системы.

1.1. Модель прикладной системы

Модель приложения - это программный аналог проектируемой целевой системы. Построение модели сводится к компановке по определенным правилам некоторых подпрограмм-шаблонов, которые выполняют функции, **подобные тем**, которые будут нужны Вашему приложению, и установке параметров модели (частоты синхронизации, разрядности шины данных, периодов выполнения программы и т.д.). Это позволяет оценить ожидаемую загрузку процессора и тем самым определить возможность реализации Вашего приложения, а кроме того - облегчить поиск компромиссных решений при проектировании аппаратных средств.

1.2. Шаблоны для оценки производительности

Программы-шаблоны для анализа производительности оптимизированы с целью наиболее эффективного использования CPU. Детальное обсуждение принципов построения шаблонов проводится в разделе **Набор Программ-Шаблонов Производительности** (см. также описание подпрограмм).

1.3. Загрузка процессора

Производительность П оценивается степенью загрузки процессора, которая вычисляется как относительная величина от возможной загрузки:

$$П = \frac{\text{реальное время, требуемое для решения задачи}}{\text{период, с которым должно появляться решение задачи}} * 100\%$$

Хронометрирование производится для каждой из подпрограмм приложения, а суммарный процент загрузки процессора отображается в виде диаграммы.

Загрузка оптимальна, если процессор выполняет код в течение 100% требуемого времени. Очевидно, что приложение будет функционировать правильно, пока загрузка меньше 100% (лучше с некоторым запасом).

1.4. Как это делается

Аппаратные средства проектируемой Вами системы могут не соответствовать возможностям оценочного модуля, на котором производится моделирование. ***ModelBUILDER*** это учитывает при вычислении времени выполнения программ и загрузки процессора. Когда ***ModelBUILDER*** производит хронометраж, он выполняет программу в "пошаговом" режиме на моделирующем устройстве и суммирует времена выполнения каждой команды, выраженные в машинных циклах. Затем используется обширная база данных, которая позволяет учесть зависимость времени выполнения каждой команды от разрядности шины данных, модели памяти и количестве тактов ожидания для реальных массивов данных, получаемых от модуля, на котором осуществляется моделирование.

Результаты вычислений сохраняются для последующих расчетов с помощью планировщика задач.

При известном числе тактов, требуемых для выполнения кода, легко может быть вычислено время выполнения программы для любой частоты синхронизации. Аналогично учитывается влияние положения кода или переменной в памяти на время обращения к ним. Таким образом, действительное значение времени, требуемого для выполнения той или иной программы, может быть измерено практически для любой системы, независимо от аппаратных средств, на которых определяется время исполнения того или иного кода.

2. Моделирование

или "Как мне приспособить ваши шаблоны для анализа моего приложения?"

ModelBUILDER поддерживается следующими программами-шаблонами:

1. Шаблоны для оценки производительности CPU:

- ***Cube*** - вращение координат вершин куба
- ***IIR*** - фильтр с бесконечной импульсной характеристикой
- ***MAC*** - умножение-накопление
- ***Matrix*** - перемножение матриц
- ***PID*** - пропорциональное интегрально-дифференциальное преобразование

2. Шаблоны для анализа производительности периферии

- ***Analog*** - аналого-цифровое преобразование - аналоговый ввод
- ***Keyboard*** - подпрограмма сканирования клавиатуры
- ***PWM*** - широтно-импульсная модуляция - аналоговый вывод
- ***Tachometer*** - тахометр

Шаблоны для анализа производительности процессора выполняют функции математической обработки и цифровой фильтрации. Шаблоны анализа производительности периферии выполняют функции цифрового и аналогового ввода/вывода информации. После изучения поставляемых подпрограмм Вы сможете моделировать большинство необходимых Вам прикладных программ.

Ниже рассматриваются примеры использования шаблонов.

ПРИМЕР 1: Ваша задача - производить измерения по 4-м каналам АЦП с периодом выборки 200μs на канал. Вы могли бы загрузить подпрограмму ANALOG 4 раза и для каждой установить период в 200μs, но тогда Вы не сможете загружать другие подпрограммы. Эквивалентный результат можно получить загрузив подпрограмму ANALOG один раз и установив период выборки 50μs (200μs/4).

ПРИМЕР 2: Необходимо сканировать матрицу клавиатуры 3x3 каждые 10ms. Имеется подпрограмма-шаблон для матрицы 4x4. Загрузка процессора этой задачей так низка, что различие в числе строк и столбцов не имеет значения, поэтому можно использовать поставляемую программу, не обращая внимания на некоторое несовпадение исходных данных. Рассчитанная загрузка будет немного выше, чем загрузка процессора Вашим реальным приложением, так что это просто добавит некоторый запас к расчетной загрузке Вашей модели.

ПРИМЕР 3: Приложение должно формировать один синусоидальный сигнал частотой 60Hz. Имеющийся в ***ProjectBUILDER***'е PWM-шаблон вырабатывает 6 таких сигналов. Исследование PWM подпрограмм показывает, что при некотором их объеме загрузка, связанная с генерацией сигнала через систему высокоскоростного вывода (HSO), не зависят от числа каналов, но загрузка на обслуживание прерываний прямо связана с числом каналов. Требуемый период для сигнала частотой 60Hz - 1280μs. Нам необходимы 1/6 от доступных в шаблоне выходов и загрузка будет в 6 раз меньше, но, будучи консерваторами, примем, что один канал PWM загружает CPU до 1/4 загрузки, определенной для шаблона. Тогда значение периода 1280x4=5120μs. Используем значение 5000μs. Обратите внимание, что форма генерируемого сигнала здесь не влияет на точность определения результирующей загрузки CPU.

ПРИМЕР 4: Необходимо сформировать два синусоидальных сигнала частотой 60Hz с 6 отсчетами на цикл, но соответствующая подпрограмма-шаблон формирует один сигнал по 13 отсчетам. Для аппроксимации числа каналов можно, как и в примере 3, воспользоваться масштабированием. Примем, что 2 канала будут требовать 0.4 от времени выполнения исходной подпрограммы, тогда правильное значение загрузки получается для периода 1280μs/0.4=3200μs. Количество выборок на период - 6 для нашего приложения против 13 для подпрограммы-шаблона, или 6/13 от шаблона. Итак, для эквивалентной загрузки необходимо взять время, определенное с помощью масштабирующего коэффициента для числа каналов (3200μs), разделить его на 6/13, расчетное значение периода - 6933μs. Используем ближайшее меньшее значение периода из возможных - 5000μs.

3.Хронометрирование программ пользователя

Могут быть случаи, когда невозможно согласовать все Ваши требования с имеющимися шаблонами. Тогда можно создать Вашу собственную подпрограмму и загружать ее как подпрограмму пользователя ***(User Routine***). ***ModelBUILDER*** поддерживает подпрограммы пользователя, написанные на ассемблере (ASM-96) или на C (C-96), так что Вы можете даже исследовать влияние используемого языка программирования на производительность.

Демонстрационная версия Макроассемблера ASM-96, поставляемая в составе ***ProjectBUILDER***, совместима с полной версией снизу вверх, но имеет ограниченные возможности - объектный код генерируется в абсолютном формате и не может превышать 8 Кбайт.

Тем не менее уникальные возможности ***ModelBUILDER'a***, позволяющие объединять стандартные шаблоны производительности с кодом пользователя, делают эти ограничения несущественными для достаточно точной оценки производительности весьма сложных приложений.

3.1. Требования к хронометрируемым программам

Для анализа времени выполнения программ используется оценочный модуль с программой-монитором. Обязательное условие для проведения анализа - хронометрируемый участок программы пользователя должен быть выделен метками метками ***T_BEGIN*** и ***T_END.*** В процессе прогона программы время выполнения кодов между этими метками, выраженное в машинных циклах, фиксируется и передается в ***ModelBUILDER***.

Для решения задачи хронометрирования требуется:

- подготовить текст Вашей программы таким образом, чтобы **метки *T_BEGIN* и *T_END* были достижимы во время выполнения программы**
- оттранслировать или скомпилировать код пользователя **с использованием опции *DEBUG***, чтобы символьная информация оказалась включенной в объектный модуль;
- инсталлировать монитор-отладчик ***mon96.exe***;
- подключить оценочный модуль (**Target Board**) или (**Evaluation Board**), совместимый с проектируемой системой, и включить питание;
- средствами ***ModelBUILDER***'а запустить хронометрируемую задачу на выполнение.

Пример корректный программы:

для Target Board	*для Evaluation Board*	
$DEBUG	$DEBUG	
CSEG AT 0C000H	CSEG AT 2080H	
ljmp START	ljmp START	
START:	START:	
nop	nop	
nop	nop	
T_BEGIN:	T_BEGIN:	; **код программы или метка *T_BEGIN*** ; **после метки *START* - это правильно**

Пример некорректной программы:

для Target Board	*для Evaluation Board*	
$DEBUG	$DEBUG	
CSEG AT 0C000H	CSEG AT 2080H	
ljmp START	ljmp START	
T_BEGIN:	T_BEGIN:	;**эти строки не могут быть выполнены**
nop	nop	;**этот код недостижим**
START:	START:	
...	...	

Если нужно хронометрировать подпрограмму обработки прерывания, прерывания должны быть запрещены, а исследуемая подпрограмма должна вызываться директивой *lcall*.

Пример правильного хронометрирования прерывания:

```
T_BEGIN:                    ; метка начала хронометрирования
        clrc
        nop
        lcall   DO_SLOPE    ; вызов подпрограммы прерывания
T_END:
```

Таким образом может быть получена достаточно точная верхняя оценка времени обслуживания прерывания.

3.2. Запуск хронометрирования модели

Если Вы загрузили планировщик задач шаблонами производительности, запустить хронометраж модели на оценочном модуле можно щелчком по кнопке ***"Stopwatch"*** в нижнем правом углу окна планировщика задач. Это также может быть выполнено из меню ***"Options"*** в пункте ***"Time Selected Tasks"*** с помощью "мыши" или клавиатуры (комбинация **ALT-O-T**).

Внимание: не забудьте подсоединить оценочный модуль к компьютеру, включить питание модуля; правильно установить скорость пересылки данных между модулем и компьютером (из Debug Monitor'a). Учтите также, что хронометрирование времени исполнения кодов требует значительной загрузки линии связи между целевой платой и главной ЭВМ, поэтому хронометраж сложных подпрограмм может потребовать больших временных затрат. Для ускорения процесса моделирования используйте как можно более высокую скорость обмена.

4. Пример моделирования приложения

Напомним, что одна из основных задач, для которых разрабатывался ***ModelBUILDER*** - моделирование приложений. В процессе моделирования с помощью стандартных подпрограмм-шаблонов можно оценить производительность вновь создаваемого приложения, не имея ни макета аппаратных средств, ни прикладных программ. Однако от Вас требуется четкое понимание требований к Вашей системе и знание особенностей доступных шаблонов производительности, из которых строится модель. Анализ загрузки процессора выполняется ***ModelBUILDER***'ом для выбранных пользователем подпрограмм при различных значениях периода их повторения, частоты синхронизации микроконтроллера, разрядности шины данных, скорости передачи данных, и т.д. Одновременно в ***ModelBUILDER*** могут быть загружены до пяти подпрограмм со своими периодами повторения.

4.1. Описание моделируемого приложения

Замечание: рассматривается не реальная прикладная программа, а абстрактный пример моделирования системы для измерения давления крови.

В этой системе используется один преобразователь давления как для измерения давления, так и для обнаружения пульса. Нагнетание воздуха в манжету производится с помощью трехфазного микродвигателя, управление микродвигателем и выпускным клапаном манжеты осуществляется микроконтроллером. Сигнал преобразователя давления усиливается и подается на один из каналов аналого-цифрового преобразователя. Индикация осуществляется на жидкокристалическом табло, а для управления системой используется клавиатура.

Система работает следующим образом:

Пациенту накладывается манжета и нажимается клавиша "Старт". Выпускной клапан манжеты закрывается и в нее нагнетается воздух, при этом контролируются давление и пульс. При пропадании пульсового сигнала выпускной клапан открывается, и фиксируется давление, при котором восстанавливается пульсовой сигнал. Таким образом определяются частота пульса и давление крови, после чего воздух выпускается, а результаты измерений выводятся на индикатор.

4.2. Моделирование приложения

Для того, чтобы приступить к моделированию, необходимо знать не только основные функции приложения, но и полосу частот входного сигнала. Знание функций требуется для выбора соответствующих моделирующих подпрограмм-шаблонов, а характеристики входного сигнала позволяют определить параметры, которые должны быть воспроизведены этими шаблонами.

Обычно управление микродвигателем осуществляется с использованием широтно-импульсной модуляции. Поэтому выберем в качестве первой составляющей модели приложения стандартную программу широтно-импульсного модулятора - ***PWM-***шаблон. Номинальная скорость прерывания для этого шаблона, соответствующая периоду в 1280μs, приемлема для наших условий. Т.к. нам не требуется высокой частоты выходного сигнала управления микродвигателем, то достаточно 13-ти выборок на период синусоиды. (Подпрограмма генерирует синусоиду частотой 60Hz).

Для моделирования ввода с клавиатуры воспользуемся шаблоном ***KEYBD*** и, поскольку 20-ти опросов в секунду достаточно для сканирования клавиатуры, выберем период функционирования для этого шаблона, равный 50000μs.

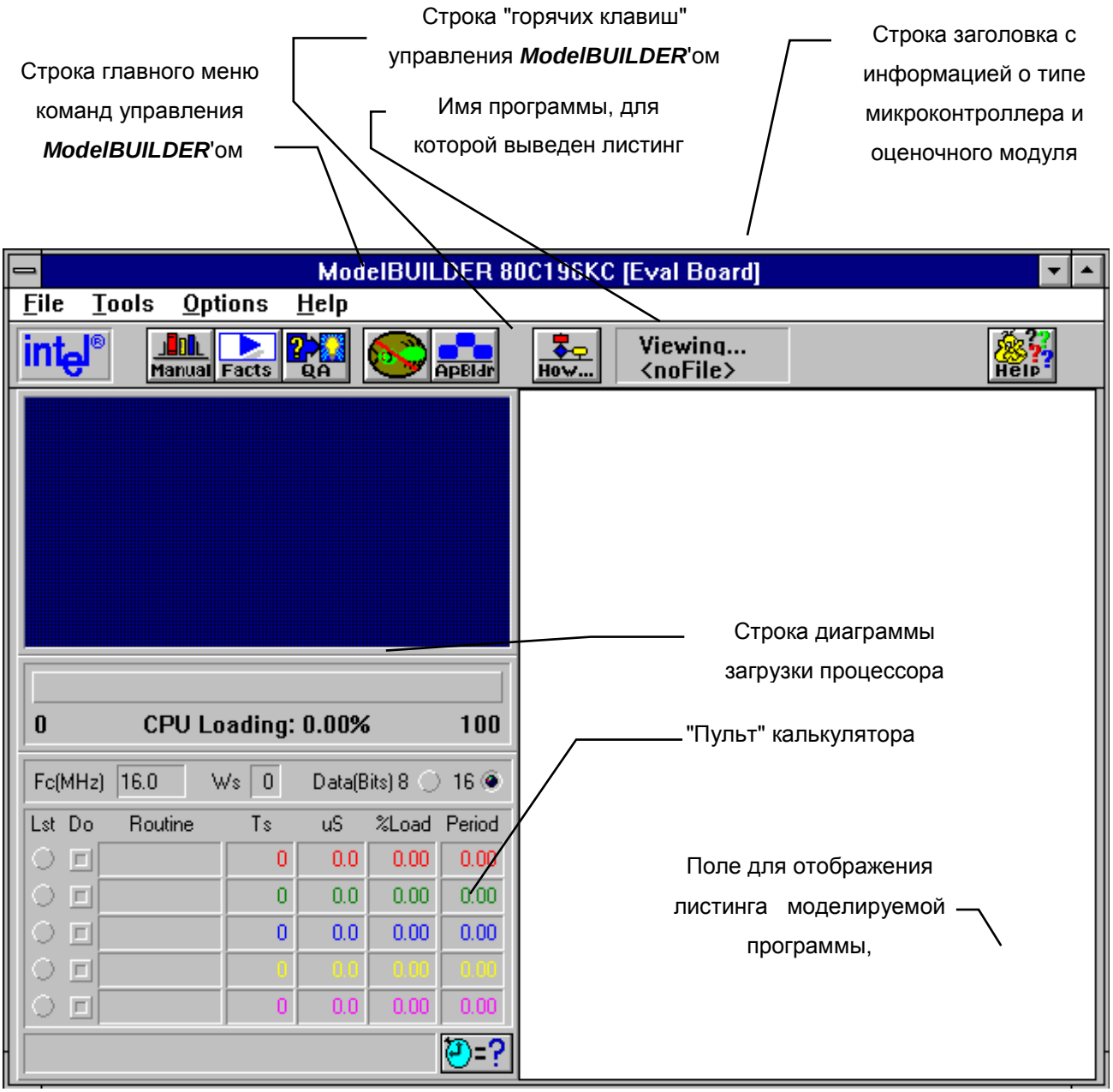
Пульсовой сигнал может составлять до 300 ударов в минуту, причем известно, что сигнал существенно искажен помехами. Поэтому для выделения полезного сигнала потребуется применение цифровой фильтрации. Выборки сигнала должны производиться с частотой, по крайней мере в 10 раз большей частоты пульса. Тремстам ударам в минуту соответствует частота 5Hz, следовательно частота дискретизации для такого сигнала с учетом соотношения Найквиста и приведенных выше соображений может быть принята равной 10x2x5Hz=100Hz, а период выборок - 10000μs. Моделирование процесса дискретизации будем проводить с помощью подпрограммы-шаблона ***ANALOG*** с периодом повторения 10000μs. Обращаем внимание, что более высокая частота выборки позволила бы повысить разрешающую способность, но это, очевидно, избыточно и мы будем использовать частоту 100Hz.

Как уже отмечалось, выделение полезного сигнала требует применения низкочастотной фильтрации входного сигнала. Возможное решение этой задачи обеспечивается фильтром с конечной импульсной характеристикой (IIR), для моделирования этой функции воспользуемся ***IIR***-шаблоном. Фильтрация производится путем цифровой обработки каждой выборки, поэтому период для этого шаблона должен быть равен периоду работы шаблона ***ANALOG***, т.е. 10000μs.

И, наконец, мы должны измерить пульс. Достаточно близкая задача решается подпрограммой-шаблоном ***TACH***, предназначенной для измерения скорости вращения некоторого вала. В нашем случае можно использовать упрощенную версию шаблона ***TACH***. Заданный по умолчанию период в 20000μs обеспечивает достижение более высокой разрешающей способности, чем требуется, но мы используем предлагаемое значение периода, поскольку увеличение разрешающей способности практически не влияет на результат расчета загрузки CPU (измерение производится блоком высокоскоростного ввода/вывода автономно, без участия процессора).

5. Работа с *ModelBUILDER*'ом

Продемонстрируем последовательность действий при моделировании рассмотренного выше примера. Все подготовительные операции - подключение оценочного модуля (Target Board или Evaulation Board) и запуск ***ModelBUILDER***'а уже выполнены? Тогда найдите в окне ***Intel Microcontrollers*** "иконку" ***ModelBUILDER*** и щелкните по ней. На экране развертывается рабочее поле:



В главном меню щелчком кнопки "мыши" Вы можете выбрать:

File **- Работа с файлами**

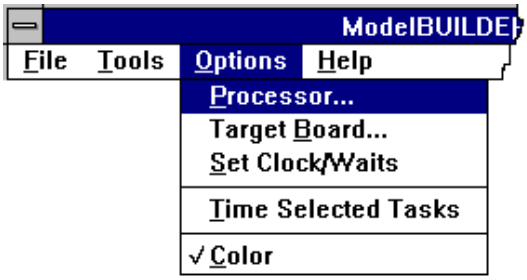
Tools **- Выбор инструментария**

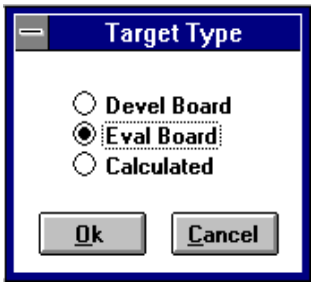
Options **- Выбор оценочного модуля и режимов работы**

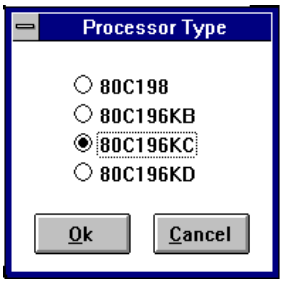
Help - Справочник (Помощь)

5.1. Настройка системы

Выберите в основном меню позицию ***Option***. С помощью команды ***Target Board***... выберите тип оценочного модуля, подключенного к Вашему компьютеру а затем с помощью команды ***Processor...*** - тип процессора в оценочном модуле.





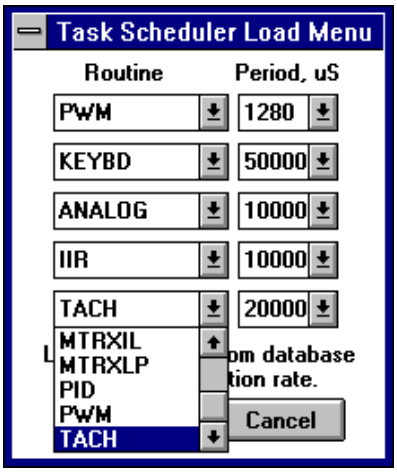


5.2. Установка параметров модели

Установить временные параметры для моделируемой программы - тактовую частоту процессора и количество состояний ожидания Вы можете в окне ***Task Timing Setup*** после выполнения команды ***Set Clock/Waits*** из меню ***Options***: щелкните по кнопке  рядом с окном для соответствующего параметра, и отметьте требуемое значение в развернувшемся меню допустимых значений, после этого не забудьте "щелкнуть" по кнопке ***"Ok".***

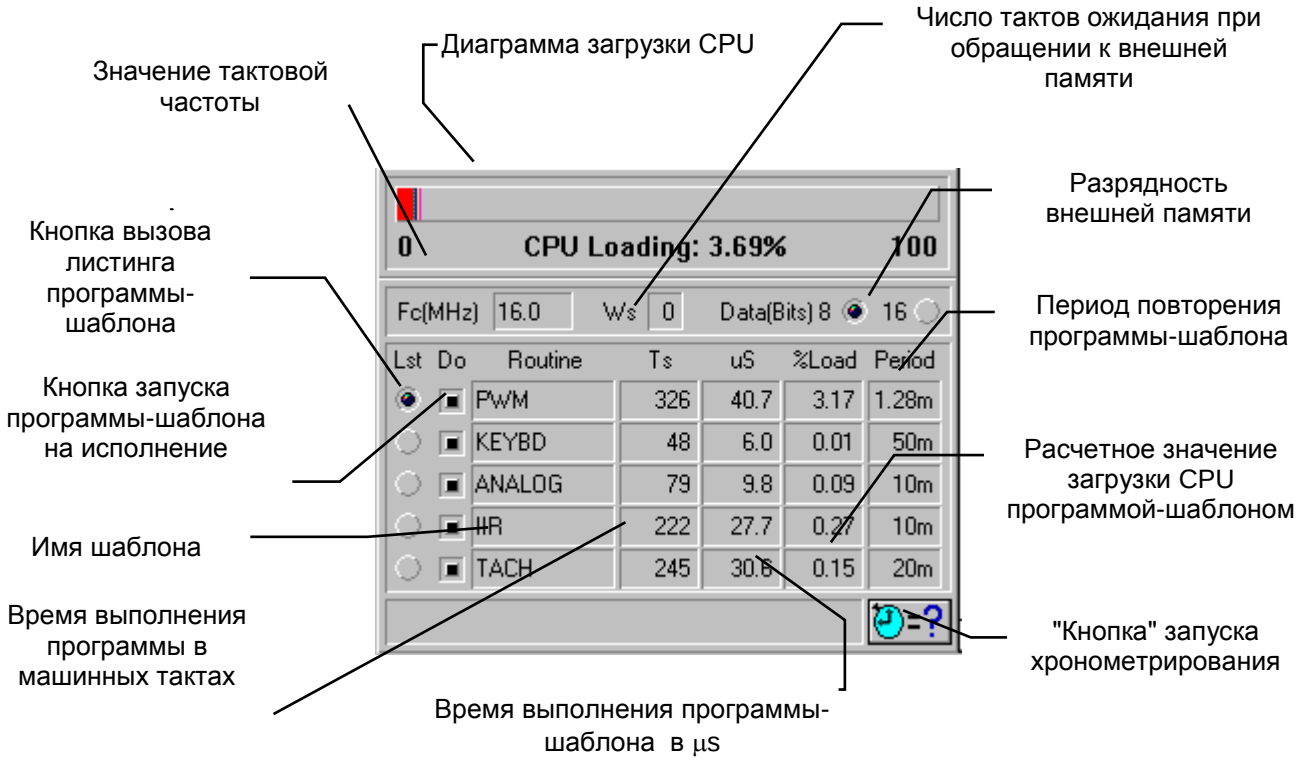
На этом заканчивается подготовка аппаратной среды для моделирования и можно приступать к загрузке программ шаблонов производительности. Для этого выберите в главном меню пункт ***File***, а в нем - команду ***Load Templates.*** При этом "всплывает" окно ***Task Scheduler Load Menu*** - загрузки задач-шаблонов постановщика. В этом окне Вы можете выбрать до пяти различных программ, в том числе - разработанную Вами.

Щелкните в разделе ***Routine*** по одной из кнопок  , после чего откроется каталог файлов, в котором Вы должны выбрать файл с требуемым шаблоном. Вид окна загрузки постановщика задач перед выбором последнего из пяти возможных шаблонов показан справа. Одновременно с включением имени файла в перечень загружаемых в разделе ***Period*** в соответствующем окне появляется заданный по умолчанию период выполнения программы-шаблона. При необходимости Вы можете изменить этот период, нажав кнопку  в разделе ***Period*** и отметив требуемое значение периода в предложенном наборе значений.



Закончив подготовку шаблонов, щелкните по кнопке "***Ok"*** в окне загрузки постановщика задач. Имена выбранных Вами программ и параметры их исполнения перенесутся в соответствующие позиции окна-калькулятора.

Ниже показано окно - "пульт" калькулятора. Рассмотрим подробнее основные компоненты этого окна.



Результаты моделирования показывают, что в нашем примере CPU микроконтроллера 80C196KC с 8-разрядной шиной данных, тактовой частотой 16MHz при работе с памятью, не требующей состояний ожидания, загружен вышеупомянутыми подпрограммами (с указанными в окне калькулятора периодами повторения) только на 3.69%!

5.3. Альтернатива аппаратным средствам моделирования

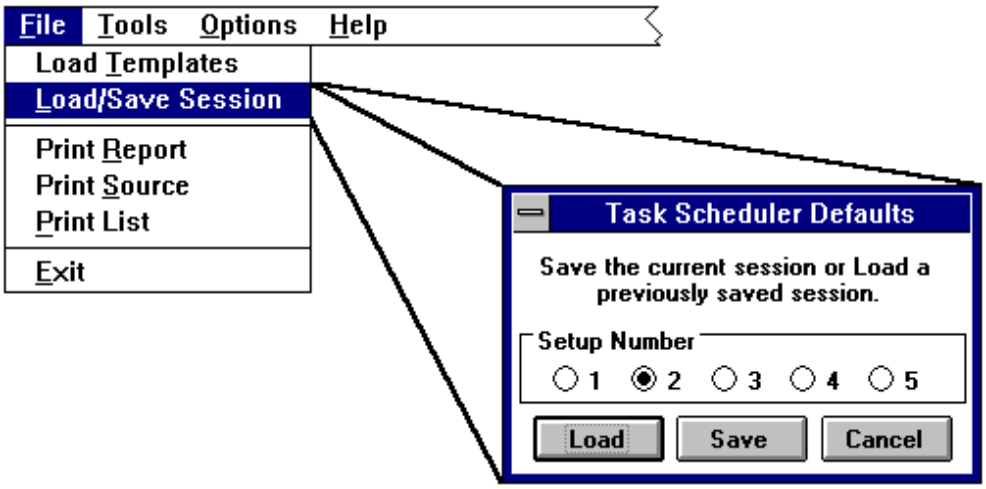
Поскольку от устройства требуются малая потребляемая мощность и минимальная цена, можно исследовать вопрос снижения тактовой частоты, а для уменьшения числа компонентов системы - возможность применения 8-разрядной внешней памяти. Кроме того, может оказаться целесообразным применение более простых и дешевых микроконтроллеров типа 80C196KB или 80C198. Анализ ответов на вопросы **"А что если ?.."** может производиться с использованием всех доступных параметров: пункт ***Clock/Waits*** позволяет изменять тактовую частоту CPU и число тактов ожидания при обращении к внешней памяти, в окне ***Load Templates*** можно изменять периоды повторения программ-шаблонов для упрощения проекта или улучшения производительности.

Заметим, что даже если необходимо иметь максимально возможные скорости выборки, мы можем это реализовать при более низких частотах синхронизации процессора без потери общей производительности нашей системы. 16-разрядная архитектура процессора поддерживает эффективные вычислительные операции, периферийные устройства отвечают требованиям быстрого ввода-вывода,

а запас производительности процессора таков, что программирование на языке высокого уровня, например, ***C***, не будет приводить к существенному снижению эффективности решения поставленной задачи при существенном уменьшении времени проектирования программного обеспечения.

5.4. Команды меню *ModelBUILDER*'а

Что делать, если Вы не успели за один прием проанализировать возможные варианты реализации Вашего приложения? В любое время можно повторить моделирование с теми же или измененными параметрами без существенных дополнительных затрат времени, сохранив текущий сеанс под определенным номером. Для этого в подменю ***File*** главного меню выберите пункт ***Load/Save Session*** - на экране "всплывет" окно постановщика задач (по умолчанию).



Отметьте с помощью "мыши" номер, под которым Вы хотите сохранить текущий сеанс и нажмите кнопку ***"Save"***.

В последующем достаточно вызвать это окно, отметить требуемый номер сеанса и нажать кнопку ***"Load"*** - ***ModelBUILDER*** загрузит параметры модели, используемые в ранее сохраненном сеансе.

Существует более быстрый способ вызова окна постановщика задач - установите курсор в любое свободное от кнопок место на "пульте" калькулятора и дважды щелкните по левой клавише "мыши".

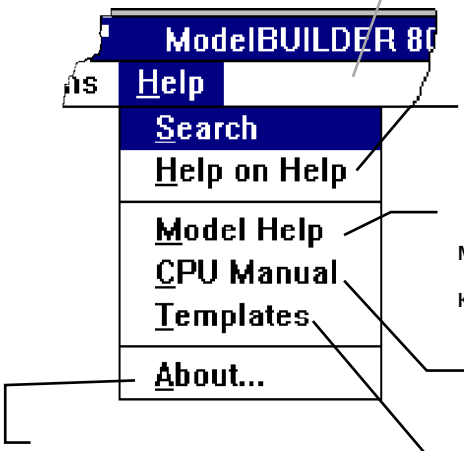
В этом же разделе меню Вам предоставляется возможность задокументировать результаты работы:

Print Report	- **Печать отчета** по результатам моделирования
Print Source	- **Печать исходного текста** моделируемой программы
Print List	- **Печать листинга** трансляции исходного текста программы

Печатаются только те файлы текущего сеанса моделирования, которые отмеченные кнопкой *LST* в пульте калькулятора. Файлы исходного текста (расширение *.a96* или *.c96*) и листинга трансляции программы (*.lst*) должны находиться в том же каталоге, что и соответствующий обьектный файл (*.obj*).

При входе в подменю ***Tools*** Вы можете обратиться за "советом" к ***ApBUILDER'***у или перейти в ***Monitor*** для исследования особенностей исполнения моделируемой программы.

Если Вам в процессе работы что-либо непонятно или Вы испытываете затруднения при выполнении некоторых действий - ***ModelBUILDER*** поможет - отметьте в главном меню пункт ***Help*** - и перед Вами откроется обширная справочная система, в которой имеется несколько разделов:



Поиск - здесь можно выбрать тему, по которой необходима справка

Справка о справке - информация о том, как пользоваться стандартной системой ***Help*** в ***Windows***

Информация о программе *ModelBUILDER* - как осуществлется моделирование, что представляют собой программы- шаблоны, как выполняются те или иные функции модели, глоссарий и т.п.

Руководство по применению микроконтроллера - обеспеспечивает мгновенный доступ к гипертекстовому руководству

Шаблоны производительности - более быстрый доступ к информации об этих программах, чем из раздела ***Model Help***

Информация о программе *ModelBUILDER* - разработчик, версия программы, дата и пр.

5.5. Функциональная клавиатура *ModelBUILDER*'а

Кроме рассмотренных функций главного меню, Вы можете использовать функции, реализуемые с помощью клавишной строки ***ModelBUILDER'a***, которая расположена под строкой главного меню. Это в ряде случаев позволит упростить и ускорить работу в ***ModelBUILDER***'а. Клавиатура содержит следующие кнопки:

Manual - Руководство по применению выбранного предварительно микроконтроллера.

Facts - Технические характеристики микроконтроллера

Question's & Answer's - Вопросы и Ответы

Раздел, вызываемый этой кнопкой, содержит ответы на некоторые общие вопросы, возникавшие у разработчиков.

Monitor - Передача управления программе *Monitor Debug*

ApBldr - Передача управления программе *ApBUILDER*

How - "Как..." Эта кнопка позволяет узнать, как реализуется та или иная функция моделируемой программы. Чтобы получить соответствующую справку, необходимо вызвать листинг исходного тектста программы (кнопка ***"LST"*** калькулятора), "нажать" кнопку ***How,*** установить курсор на интересующий Вас фрагмент текста и щелкнуть один раз "мышью". Собственно, тот же результат может быть достигнут и без кнопки ***How,*** если дважды щелкнуть мышью по фрагменту программы (в любом месте фрагмента!).

Help - Справочник (Помощь)

В ряду клавиатуры есть небольшое окно, в котором индицируется наименование той из моделируемых программ, листинг которой "распечатан" в соответствующем окне ***ModelBUILDER***'а.

5.6. Некоторые дополнительные соображения

Погрешность времени исполнения программы, определенного с помощью ***ModelBUILDER***'а, находится в пределах ± 5% - это достаточно точный результат с некоторым запасом в большую сторону по сравнению с реально достижимым временем исполнения на Ваших аппаратных средствах. Это обеспечивается благодаря нескольким предположениям, сделанным при хронометраже:

- выбранное число тактов ожидания применяется ко всей внешней памяти - и кода, и данных
- выбранный режим работы с 8-разрядной памятью также применяется ко всей внешней карте памяти.
- для получения наиболее точных результатов Вы должны выбрать разрядность шины такую же, какую имеет память программ.

ПРИМЕР: Если Ваше устройство выполняет команды из внутреннего ПЗУ (EPROM), но использует внешнюю память данных, выберите 16-разрядную шину и нулевое число тактов ожидания. Микроконтроллер выполняет коды из внутренней памяти с нулевым числом тактов ожидания, используя внутреннюю 16-разрядную шину данных, поэтому выбираемые разрядность внешней шины и число тактов ожидания будут влиять только на чтение или запись в память данных. Влияние разрядности шины и числа тактов ожидания будет мало, если только Ваше приложение не очень часто обращается к внешней памяти. В этом случае при моделировании Вы можете использовать масштабирование частоты синхронизации (снижая ее), чтобы отразить некоторую потерю производительности.

Поработав в среде ***ModelBUILDER*** и разобравшись с программами-шаблонами, Вы теперь, несомненно, гораздо яснее представляете возможности микроконтроллеров ***MCS-196*** и можете приступить к проектированию и отладке собственных программ. С особенностями трансляторов и других средств разработки программ целесообразно познакомиться после того, как Вы освоите следующий инструмент - отладчик ***Debug Monitor.***

Debug Monitor. Введение

Важнейшей частью системы проектирования ***ProjectBUILDER*** является среда отладки программ ***Debug Monitor***, называемая далее ***Монитором***. Программы, написанные для устройств ***8XC196***, из этой среды могут быть загружены в оценочные модули ***(Development Board*** или ***Evaluation Board)*** и запускаться в них на исполнение в различных режимах, обеспечивающих возможность наблюдать за функционированием всех компонентов микроконтроллера, а также изменять их состояние.

Основные отладочные функции *Монитора* поддерживаются двумя независимыми программами, одна из которых - *Intel's Reduced Instruction Set Monitor* (***iRISM)*** - монитор с сокращенной системой команд, выполняется оценочным модулем, а другая - рассматриваемый здесь *Монитор* ***mon96.exe*** - управляет процессом отладки из персонального компьютера.

Связь между управляющим компьютером и оценочным модулем производится через асинхронный последовательный порт с использованием протокола, специально разработанного для этих целей. Разделение отладочной системы на две подсистемы имеет ряд достоинств:

- Система легко адаптируется для новых устройств MCS-196, поскольку программа, которая выполняется на оценочном модуле (***iRISM-96***), очень проста и занимает менее 300h байт;
- Ресурсы оценочного модуля не накладывают ограничений на удобство и гибкость пользовательского интерфейса, т.к. он реализуется в управляющем персональном компьютере;
- Обе программы работают параллельно, что позволяет получать информацию о модуле и управлять состоянием микроконтроллера в процессе отладки.

Обозначения

Во избежание путаницы, названия ***Команд***, ***Меню***, ***Клавиш***, ***Окон*** и других элементов *Монитора* выделены полужирным курсивом и **только первые буквы** этих слов - **прописные**; названия ***КОМАНД***, ***ТИПОВ ДАННЫХ***, ***РЕЖИМОВ РАБОТЫ*** и других характеристик архитектуры микроконтроллеров MCS-196 выделены полужирным курсивом и пишутся **прописными буквами**.

1. Особенности Монитора mon96.exe

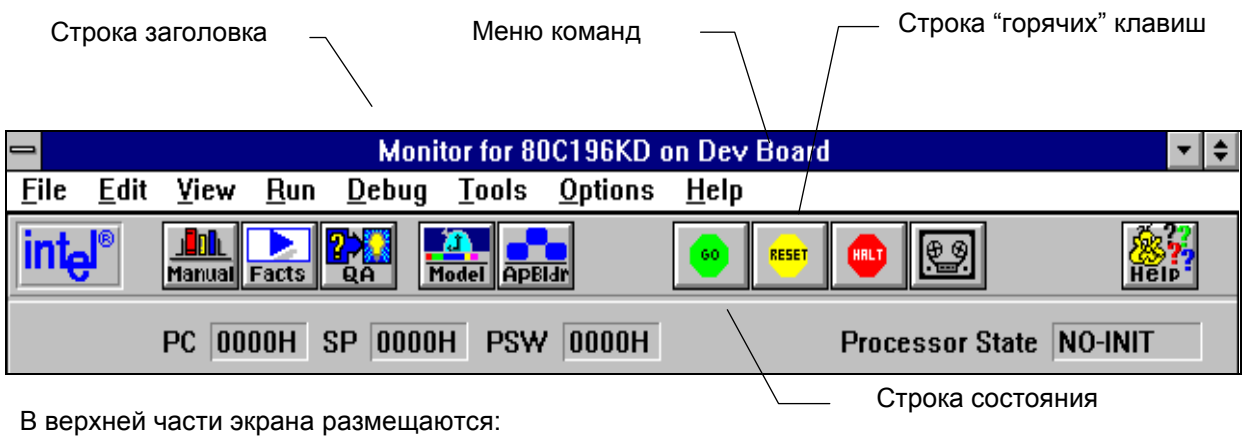
В качестве управляющего компьютера используется ***IBM-совместимый персональный компьютер***, способный работать с версией ***Windows 3.0*** и выше. Оценочный модуль подключается к компьютеру через порт COM1 или COM2 и обменивается с ним информацией со скоростью до 57 Kb/s. В процессе отладки *Монитор* обеспечивает:

- Задание до 16-ти точек останова
- Одновременное отслеживание состояния памяти и регистров микроконтроллера
- Отображение данных в форматах ***BYTE, WORD, LONG и REAL***
- Использование встроенного *Ассемблера/Дизассемблера*
- Фиксацию процесса отладки на диске в формате ***OMF***

1.1. Ограничения:

- Шесть байт стека микроконтроллера зарезервированы для использования ***iRISM***-программой. Некоторые другие ячейки памяти и регистры также используются этой программой. Все сведения об этих и других ограничениях, связанных с конкретным типом оценочного модуля приведены в Приложении.
- Команда ***TRAP*** зарезервирована для выполнения основных функций Монитора.
- Нельзя использовать точки останова и пошаговый режим работы, если пользовательская программа размещается в EPROM или другой неизменяемой памяти.

2. Описание экрана пользователя



В верхней части экрана размещаются:

- Строка заголовка, содержащая информацию о типе используемого отладочного модуля, типе процессора, установленного в модуле, и объектном файле пользователя, который загружен в память модуля.

- Строка меню. С помощью команд этого меню производится управление работой *Монитора*.

- Клавишная строка. Кнопки из этой строки обеспечивают быстрый вызов некоторых команд из строки меню.

- Строка состояния предназначена для отображения информации о текущем состоянии оценочного модуля и пользовательской программы.

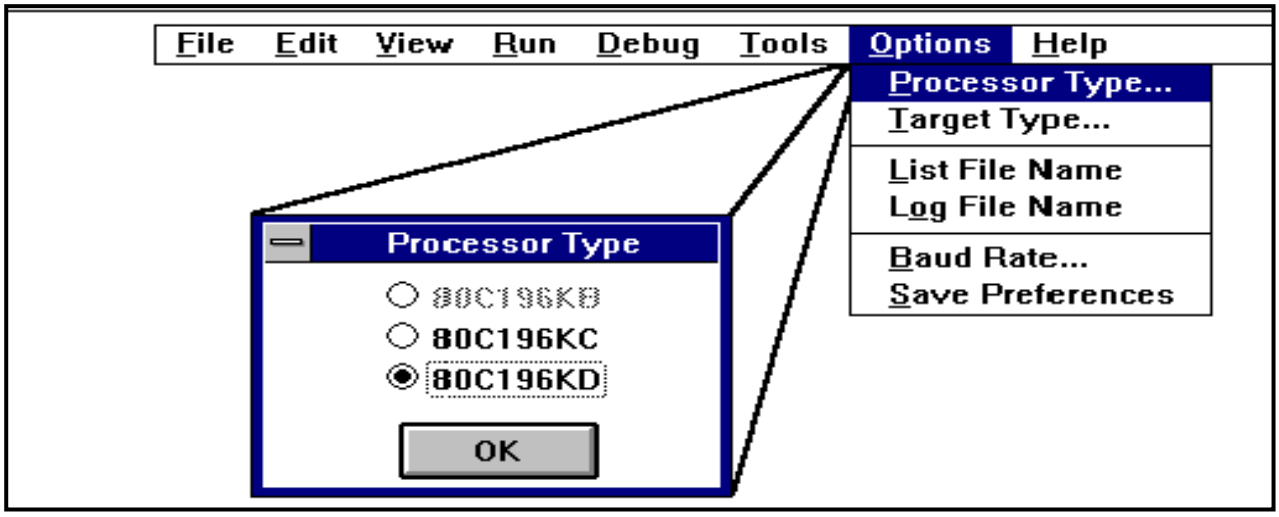
3. Работа с *Монитором*

3.1. Настройка системы

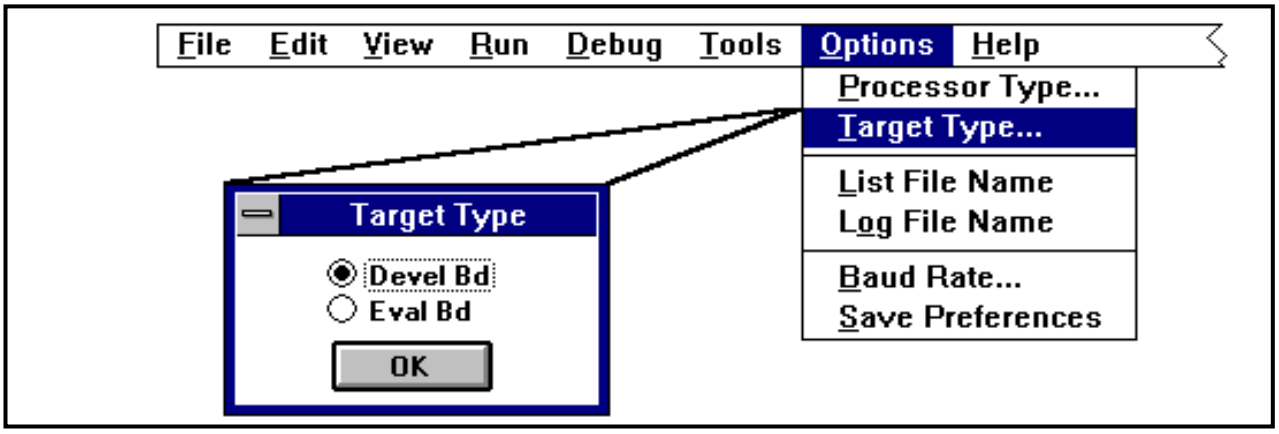
Перед началом работы с *Монитором* подключите к компьютеру оценочный модуль, включите компьютер и блок питания модуля. Затем необходимо настроить *Монитор* на работу с конкретным типом оценочного модуля. Для этого:

1. Выберите в строке меню пункт ***Options***, а в нем пункт ***Processor Type***.

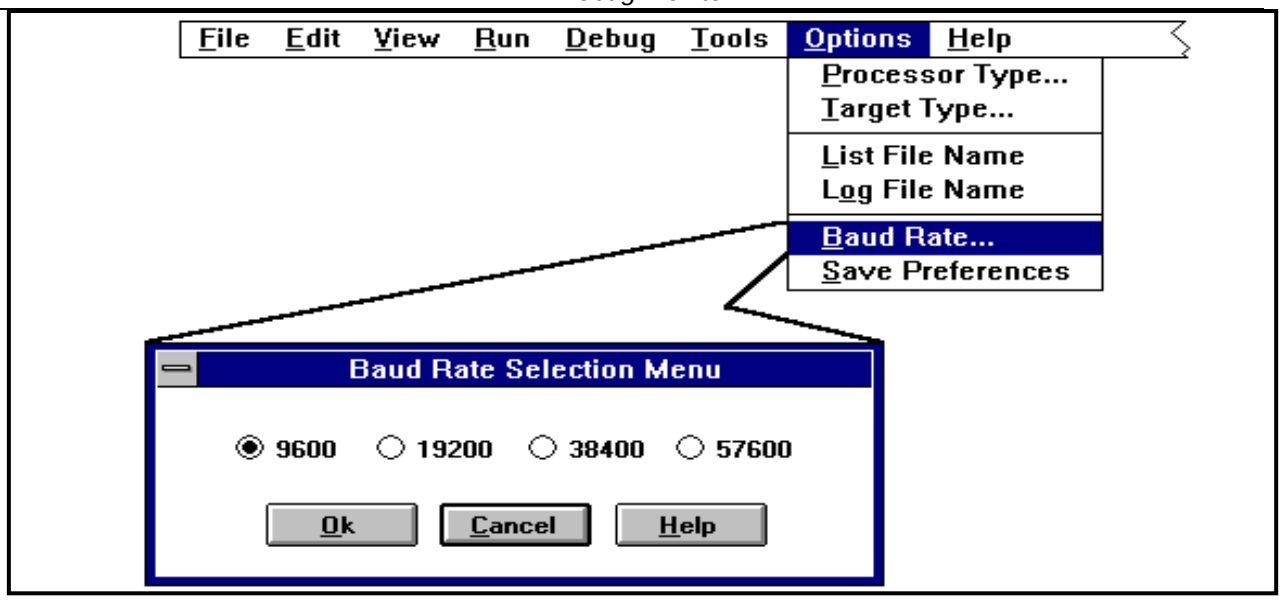
2. В открывшемся окне установите тип процессора оценочного модуля, с которым будет работать *Монитор:*



3. В этом же меню выберите пункт ***Target Type*** и установите тип используемого модуля (***Evaluation Board*** или ***Development Board***):



4. Аналогично п.2, вызвав окно ***Baud Rate,*** установите скорость обмена между управляющим компьютером и оценочным модулем. Информация о том, какую скорость обмена использует ***iRISM***-программа, "зашитая" в ПЗУ модуля конкретного типа, приводится в руководстве пользователя, поставляемом вместе с модулем:



5. **Сохраните выполненные установки**, используя команду ***Save Preferences*** в том же подменю.

Внимание: при изменении типа процессора и типа модуля, система сбрасывается в исходное состояние, закрывая все открытые окна и выполнив аппаратный сброс модуля.

Типы выбранного модуля и процессора отображаются в строке заголовка.

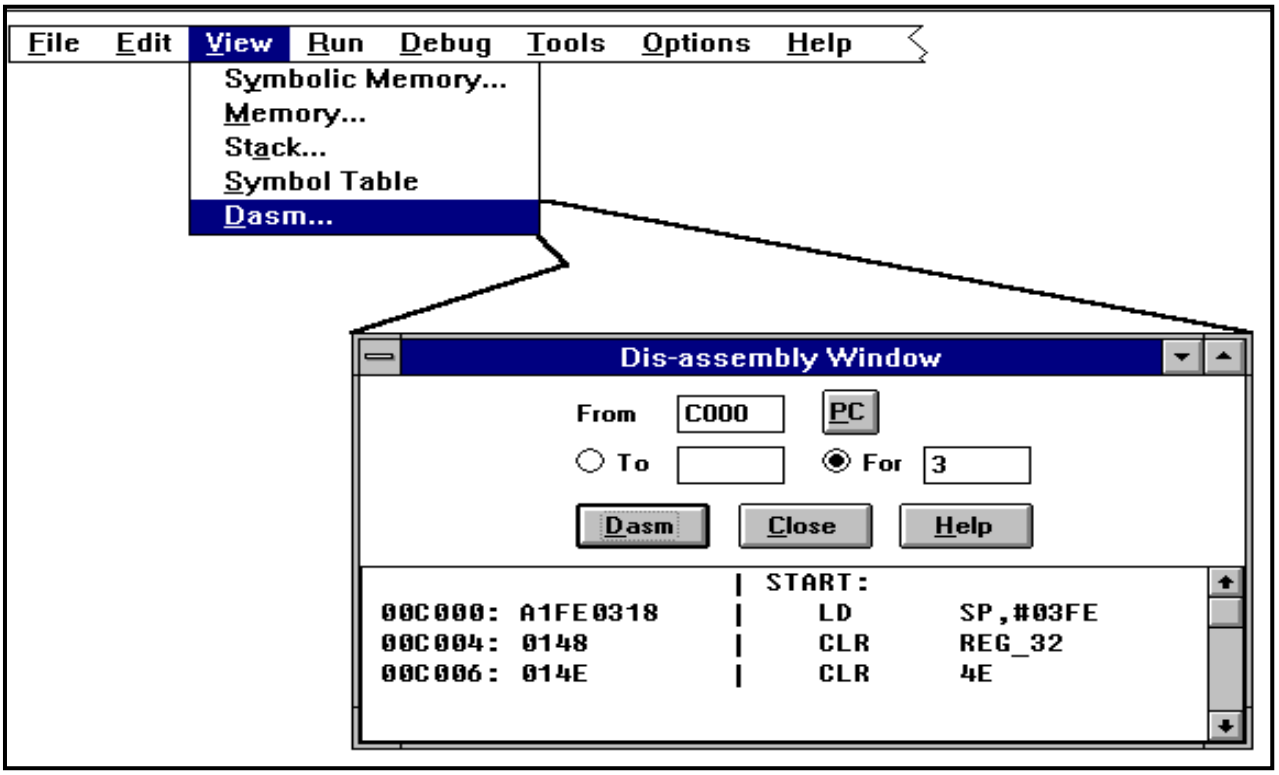
3.2. Загрузка программ пользователя

Чтобы загрузить в память модуля объектный код созданной Вами программы, выберите в меню ***File*** команду ***Load***. В открывшемся окне выберите устройство, на котором находится Ваши каталог и файл программы. После того, как Вы выберете требуемый файл и дважды щелкнете кнопкой "мыши" или нажмете клавишу "***Enter***", он загрузится в память модуля. При этом в строке состояния в окне счетчика команд ***PC*** отобразится начальный адрес программы, а в окне состояния процессора ***Processor State*** - сообщение ***"Stopped"***. В строке заголовка отображается имя загруженного файла.

Внимание: загружаемая программа должна учитывать ограничения, накладываемые оценочным модулем. Несоблюдение этих ограничений может привести к "зависанию" системы.

3.3. Дизассемблирование загруженного кода

При отладке программы полезно иметь перед глазами текст программы в формате ассемблерных команд. Однако, поскольку некоторые инструкции Ассемблера могут транслироваться в несколько различных эквивалентных машинных команд (например, инструкция ***BR***), а также поскольку программы для микроконтроллеров могут проектироваться на языках высокого уровня ***(C, PL/M),*** целесообразно работать не только с исходным текстом программы, но и с текстом, полученным дизассемблированием машинных команд, загруженных в память оценочного модуля. Для дизассемблирования программы, помещенной в память модуля, или какой-то части этой программы, нужно выбрать команду ***Dasm*** из меню ***View***. При этом на экране появится окно *Дизассемблера*:



В этом окне необходимо указать границы участка памяти, в пределах которого должно производиться дизассемблирование. Адрес нижней границы задается в окне с надписью ***From***. В том случае, если Вы хотите дизассемблировать машинные коды от текущего положения счетчика команд, Вы можете загрузить окно ***From*** значением счетчика команд, нажав кнопку ***PC*** справа от окна ***From***.

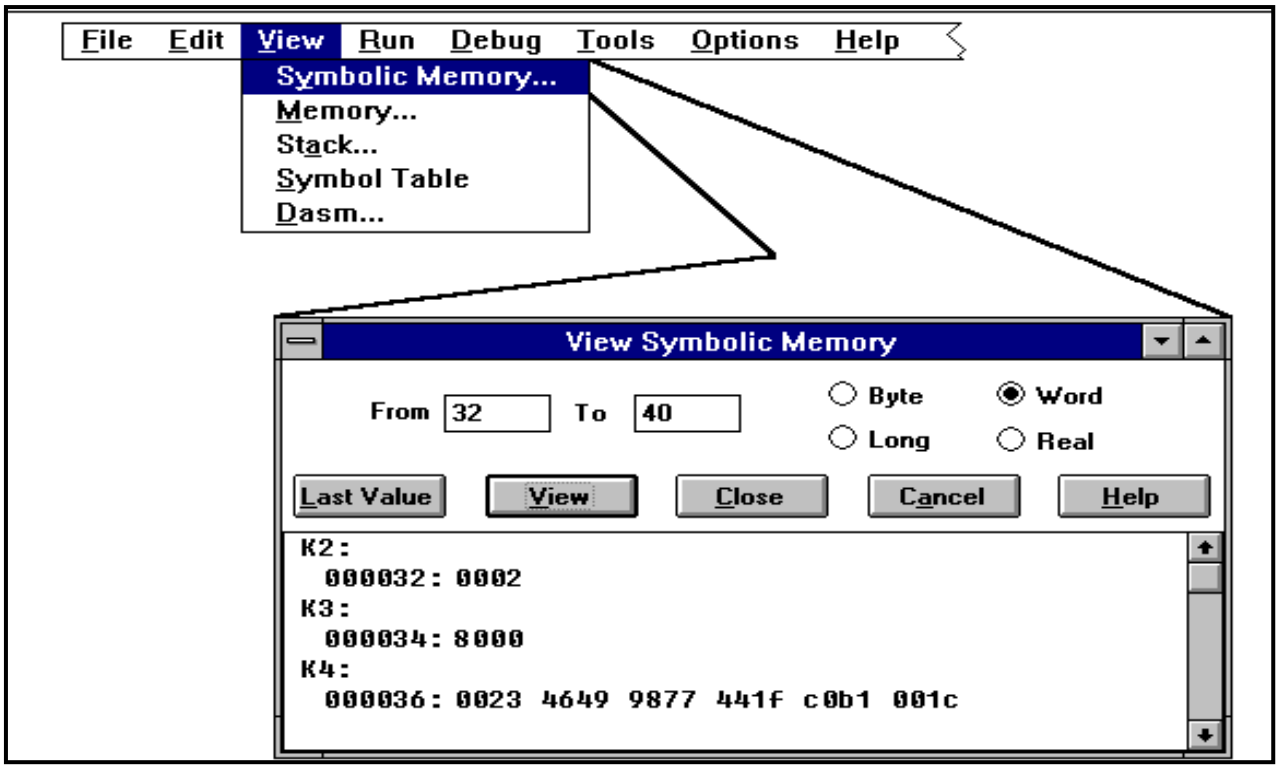
Верхняя граница дизассемблируемого участка памяти может задаваться двумя способами: значением адреса верхней границы, задаваемого в окне ***To***, либо количеством дизассемблируемых команд в окне ***For***. Какой именно способ задания верхней границы используется в текущий момент, определяется тем, какая из двух кнопок слева от надписей ***To*** и ***For*** активна. Переключение этих кнопок производится "мышью".

После задания границ дизассемблируемого участка нажмите клавишу ***Dasm*** и в рабочем поле окна *Дизассемблера* появится последовательность ассемблерных мнемоник, соответствующих машинным кодам из заданного участка памяти.

Внимание: объем дизассемблируемого участка памяти определяется характеристиками Вашего компьютера, и в том случае, если этот участок слишком велик, в окне *Дизассемблера* будут представлены мнемоники лишь какого-то числа последних команд из рассматриваемого участка памяти. Если какой-то код не соответствует ни одному из разрешенных кодов машинных команд, и, следовательно, не может быть дизассемблирован, *Монитор* выдаст сообщение об ошибке и прервет дизассемблирование перед этим кодом.

3.4. Просмотр содержимого памяти

Для просмотра памяти служат команды ***Symbolic Memory*** и ***Memory*** из меню ***View.*** В том случае, если Вы хотите просмотреть участок памяти вместе с символическими метками (а это возможно, если Ваша программа отассемблирована с опцией ***$debug***), следует выбрать ***Symbolic Memory***. На экране появится окно, в котором необходимо указать границы просматриваемого участка памяти и формат представления данных ***(BYTE, WORD, LONG или REAL).*** После нажатия кнопки ***View*** в рабочем поле окна появятся данные из указанного участка памяти в требуемом формате.



Внимание: выводятся лишь те символические метки, которые выравнены по границам выбранного типа данных (например, если задан тип отображаемых данных ***WORD***, то выводятся только те метки, которые расположены по четным адресам).

Кнопка ***Last Value*** восстанавливает значения адресов ***From*** и ***To*** и тип представления данных, использованные до их последней коррекции в текущем сеансе работы.

Команда ***Memory*** меню ***View*** позволяет просматривать содержимое памяти в виде последовательности байтов и соответствующих им ASCII-символов. После выбора этого пункта в появившемся окне следует задать границы просматриваемого участка памяти ***From*** и ***To***, и нажать кнопку ***View.*** В рабочем поле окна появится выбранный блок данных в виде трех колонок:

первая (левая) - содержит адреса первых байтов строки;

вторая - строки байтов рассматриваемого блока в 16-ричном формате (в строке не более 16 байт);

третья - строки ASCII-символов соответствующих байтов; в том случае, если байт не имеет символьного представления, в третьей колонке ставится прочерк.

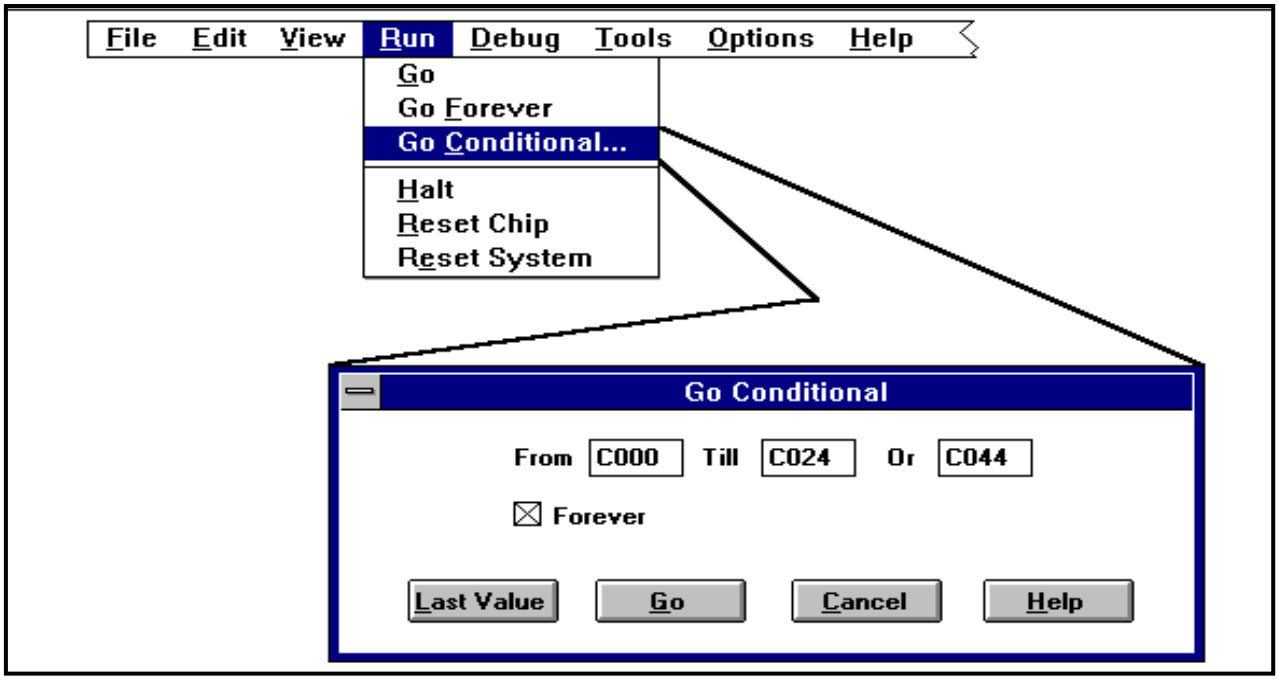
Чтобы восстановить предыдущие значения границ блока ***From*** и ***To,*** нажмите кнопку ***Last Value***.

Действие команды ***Stack*** аналогично действию команды ***Memory***, но границы просматриваемого участка памяти ***From*** и ***To*** задаются относительно текущего значения указателя стека ***SP.***

И, наконец, команда ***Symbol Table*** отображает содержимое текущей таблицы символических обозначений. Эта таблица генерируется управляющим компьютером при загрузке объектного файла (в случае, если трансляция осуществлялась с опцией ***$debug***).

3.5. Исполнение программ

Для запуска программ на исполнение служат команды меню ***Run***.



По команде ***Go*** происходит запуск программы, загруженной в память оценочного модуля. Выполнение программы начинается с текущего значения программного счетчика. В процессе исполнения кода в строке состояния в окнах ***PC***, ***SP*** и ***PSW*** отображается неопределенное состояние ***"XXXX",*** а в окне состояния процессора ***Processor State -*** **"Runing"**.

При достижении точки останова выполнение модулем пользовательской программы прерывается. В строке состояния в окнах ***PC*** и ***PSW*** отображается адрес точки останова и значение слова состояния пользовательской программы, в окне ***SP*** - значение указателя стека пользовательской программы. Повторный вызов команды ***Go*** запустит пользовательскую программу с текущего состояния счетчика команд ***PC***.

Замечание: команда ***Go*** дублируется кнопкой ***Go*** в клавишной строке.

Внимание: в исходном состоянии память модуля заполнена байтами с 16-ричным значением **00h** или **0FFh**. Байт **00h** представляет собой машинный код команды ***NOP*** (нет операции), а байт **0FFh** - код команды ***RST,*** которая производит сброс микроконтроллера. При исполнении модулем команды ***RST*** (как и при нажатии кнопки сброса на плате модуля) выполнение программы прерывается и система

сбрасывается в исходное состояние. Поэтому в отлаживаемой программе рекомендуется предусмотреть меры против перехода программы на выполнение команд, следующих за последней командой программы, например, организовав в этом месте "бесконечный" цикл (динамический останов) или используя режим отладки с остановом в контрольной точке, адрес которой равен адресу команды, следующей за последней командой программы.

Пункт меню ***Run - Go Forever*** позволяет запустить программу на исполнение с игнорированием всех установленных точек останова.

Следующая команда этого же меню ***Run - Go Conditional*** обеспечивает выполнение фрагмента программы, ограничиваемой заданными стартовым и конечным адресами. Стартовый адрес устанавливается в окне ***From;*** в следующих двух окнах задаются два возможных конечных адреса - при достижении любого из них исполнение пользовательской программы будет приостановлено. Для запуска программы используется кнопка ***Go.*** Для восстановления значений границ исполняемого фрагмента, используемых в предыдущем сеансе работы, служит кнопка ***Last Value.***

Остановить выполнение пользовательской программы можно с помощью команды ***Halt*** из меню ***Run*** (эта функция дублируется в клавишной строке). При этом происходит обновление информации в окнах ***View Symbolic Memory***, ***View Memory***, ***View Stack*** (если они открыты), а управление работой процессора передается ***iRISM***-программе.

При выборе пункта ***Reset Chip iRISM***-программа выполнит команду ***RST***. При этом произойдет сброс процессора, однако содержимое памяти модуля не изменится.

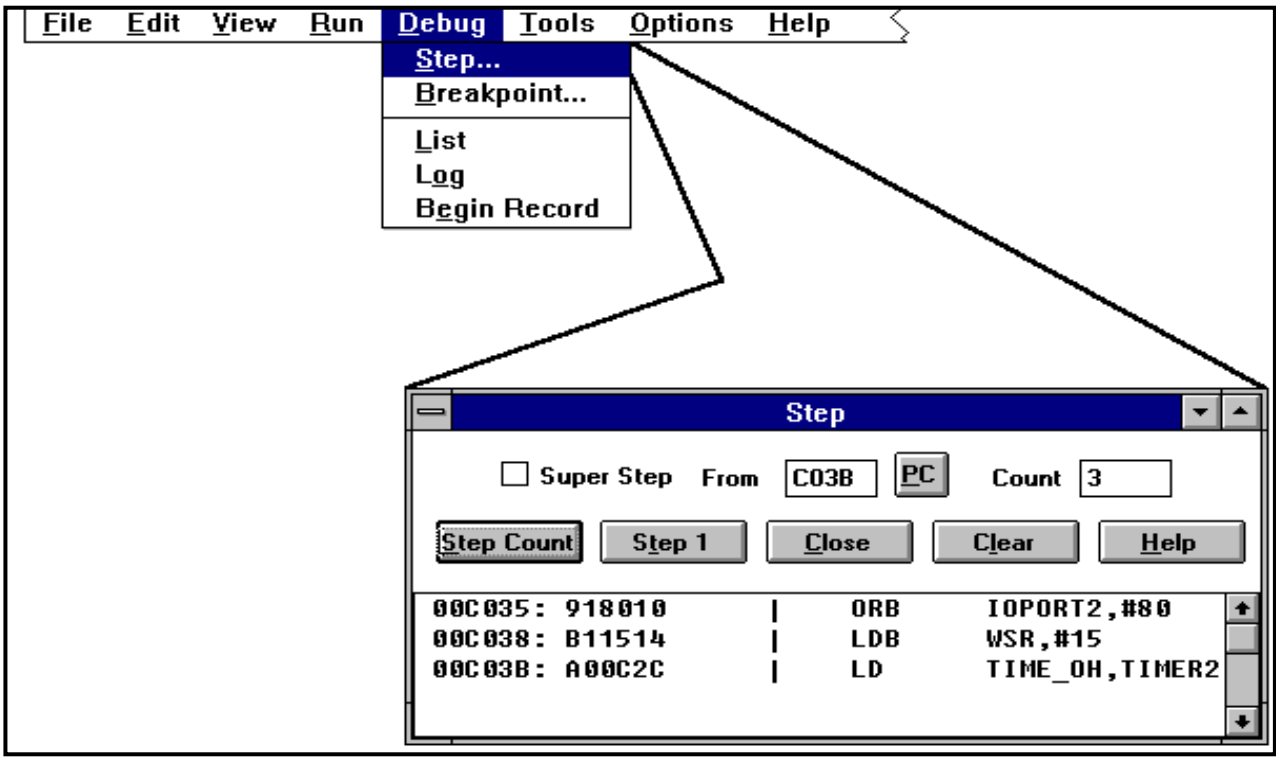
Команда ***Reset System*** в настоящей версии *Монитора* полностью идентична команде ***Reset Chip***.

3.6. Пошаговое исполнение программ

В меню ***Debug*** представлены основные функции отладки программ***.***

Команда ***Step*** позволяет отлаживать программу в пошаговом режиме: при нажатии кнопки ***Step1*** выполняется одна команда программы, после этого в окнах меню ***View*** и в строке состояния можно проконтролировать результат выполнения команды - изменились ли значения переменных Вашей программы и содержимое регистров процессора. Это помогает лучше понять, что же на самом деле делает Ваша программа. В рабочем поле окна ***Step*** при этом отображаются мнемоники выполненных команд и мнемоника команды, которая будет выполнена при очередном нажатии кнопки ***Step***.

Внимание: В пошаговом режиме *Монитор* блокирует прерывания, даже если пользовательская программа их разрешила. Поэтому пошаговое исполнение программ происходит не в реальном времени и этот метод отладки бесполезен при отладке программ, работающих с событиями реального времени - в таких случаях нужно использовать внутрисхемный эмулятор.



Разновидность пошагового режима - с опцией ***Super Step*** - не подавляет прерываний. Для перехода в этот режим отладки поставьте "мышью" крестик в окне слева от надписи ***Super Step.*** Вызовы подпрограмм обработки прерывания будут при этом исполняться как отдельные неделимые инструкции. Это позволяет отрабатывать прерывания по мере их появления с сохранением возможности трассировки тела главной программы. Однако система все-таки работает не в реальном времени и узнать, как производится обслуживание прерываний можно только после завершения соответствующих подпрограмм их обработки. В некоторых случаях лучшим способом отладки может оказаться использование команды ***Go*** с расстановкой точек останова и переход к пошаговому режиму при достижении программой интересующей Вас точки.

Кнопка ***Step Count*** инициирует пошаговое исполнение последовательности команд, число которых задается в окне ***Count***.

По окончании каждого шага в окне ***PC*** строки состояния отображается адрес следующей команды. При исполнении команд в режимах ***Step*** или ***Step Count*** в окно состояния процессора выводится сообщение ***"STEPPING"***, а по окончании исполнения команд - ***"STOPPED"***.

3.7. Особенности реализации пошагового режима работы

Для реализации пошагового режима *Монитор* совместно с ***iRISM***-программой использует инструкцию ***TRAP*** *(программное прерывание)*. Перед тем, как запустить на выполнение очередную команду пользовательской программы, *Монитор* определяет все подпоследовательности команд, которые может вызвать эта команда, и помещает по начальным адресам этих подпоследовательностей код команды ***TRAP.*** Поэтому отлаживаемая программа исполняется до тех пор, пока не будут

достигнуты точки вызова программного прерывания. При выполнении команды ***TRAP*** вызывается ***iRISM-***программа, которая совместно с *Монитором* исследует состояния регистров и ячеек памяти, восстанавливает коды программы пользователя, "поверх" которых были записаны команды ***TRAP***, определяет новые подпоследовательности, которые могут быть вызваны следующей командой из программы пользователя, устанавливает по их начальным адресам коды команды ***TRAP***, а затем ожидает от пользователя следующей команды ***Step***. Во время останова Вы можете модифицировать содержимое ячеек памяти и регистров процессора с помощью команд меню ***Edit***, анализировать состояние переменных Вашей программы и т.д.

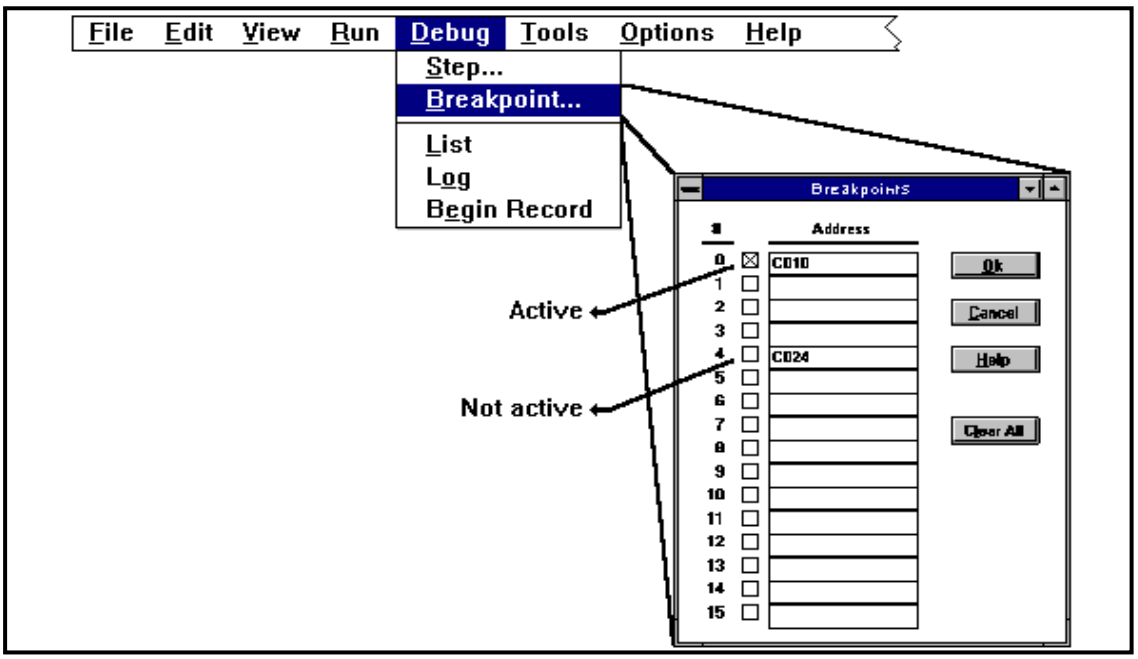
Если следующая команда из программы пользователя - команда условного перехода, то определяются две возможные подпоследовательности команд, и, соответственно, устанавливаются две команды ***TRAP***. Во всех остальных случаях устанавливается лишь одна команда ***TRAP***.

Режим ***Super Step*** также реализуется с использованием команды ***TRAP***, но отработка команды ***CALL*** производится отличным от обычного пошагового режима способом. В обычном пошаговом режиме *Монитор* помещает команду ***TRAP*** в ту ячейку, управление которой передает команда ***CALL***. В режиме ***Super Step*** *Монитор* устанавливает ***TRAP*** на место той команды, которая находиться следом за командой ***CALL***. *Монитор* подавляет прерывания, сбрасывая бит ***I*** глобального разрешения прерываний микроконтроллера перед исполнением команды ***Step1*** (значение этого бита, заданное пользователем, при этом сохраняется в зарезервированных ячейках памяти оценочного модуля). Для гарантии того, что команды, чье исполнение может модифицировать значение бита ***I***, не нарушат работу отладочной системы, некоторые команды ***(PUSHF, POPF, PUSHA, POPA, DI, EI)*** имитируются *Монитором* программно, еще до того, как они выполнятся процессором. Во избежание блокировки процессора команда ***IDLPD*** процессоров 80C196KC/KD в пошаговом режиме **не выполняется**, а вместо нее процессор исполняет команду ***NOP***. В режиме отладки ***Go*** и в пошаговом режиме с опцией ***Super Step*** процессор оценочного модуля исполняет **все** команды пользовательской программы.

3.8. Использование точек останова

Для задания и активизации точек останова служит команда ***Breakpoint*** меню ***Debug***. *Монитор* поддерживает до 16-ти точек останова.

Внимание: установка точки останова на адрес, который не соответствует первому байту команды, приведет к непредсказуемым ошибкам при исполнении программы.



При вызове пункта ***Breakpoint*** на экране появляется окно с таблицей для определения до 16-ти возможных точек останова. Чтобы установить точку останова, наберите в соответствующей графе таблицы адрес команды, перед исполнением которой Вы хотите остановить выполнение программы, и активизируйте эту точку "щелчком" “мыши" по квадратику, расположенному слева от соответствующего окна задания адреса. Активны лишь отмеченные таким образом точки останова.

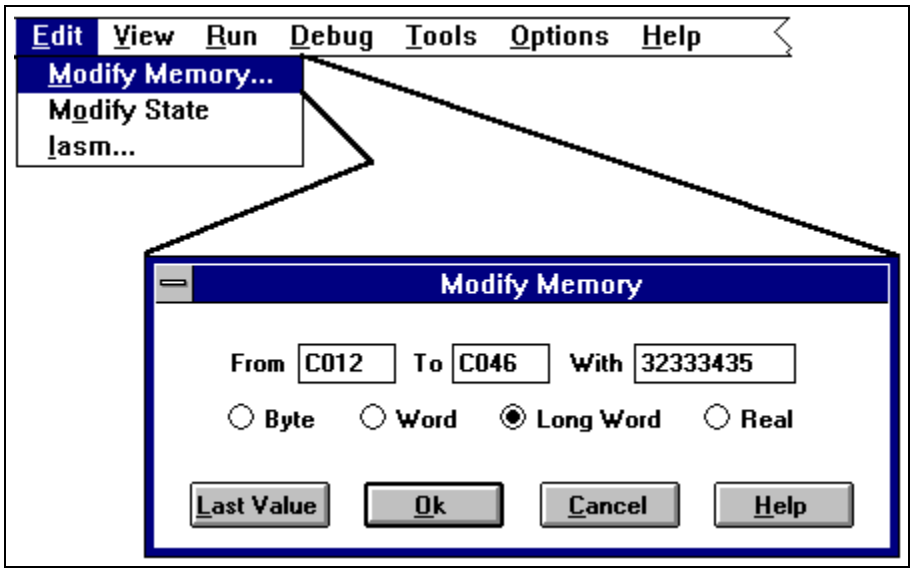
Реализация режима отладки с остановом в контрольных точках идентична реализации пошагового режима отладки: в память модуля по адресам, соответствующим адресам точек останова, записывается команда ***TRAP***. Пользовательская программа, запущенная командой ***Go***, исполняется до достижения ею команды ***TRAP***. После этого произойдет программное прерывание, управление передается ***iRISM-***программе и *Монитору*, обновляются окна группы ***View***, а пользователь получает возможность модифицировать содержимое ячеек памяти оценочного модуля и регистров процессора. При повторном вызове команды ***Go*** *Монитор* восстанавливает в памяти модуля код команды пользовательской программы, вместо которой была записана команда ***TRAP,*** и записывает команду ***TRAP*** по адресу первого байта следующей команды. После этого выполняется команда, на первом байте которой была установлена точка останова, и команда ***TRAP***. Обрабатывая ее, *Монитор* восстанавливает точку останова и код команды из программы пользователя, на место которой была записана эта команда. Затем управление снова передается программе пользователя до тех пор, пока не встретится очередная точка останова.

Внимание: если коды Вашей программы находятся в *EPROM* или *FLASH*-памяти, Вы не можете использовать пошаговый режим работы и точки останова, т.к. для реализации этих режимов Монитору необходимо вставлять в код программы пользователя свои команды.

При приостановке выполнения программы по точке останова периферийные устройства микроконтроллера, такие, как АЦП, последовательный порт, ШИМ, таймеры, сервер периферийных транзакций и блок высокоскоростного ввода-вывода, продолжают работать.

3.9. Изменение содержимого ячеек памяти и специальных регистров микроконтроллера

Для модификации ячеек памяти и регистров процессора служат команды меню ***Edit***.



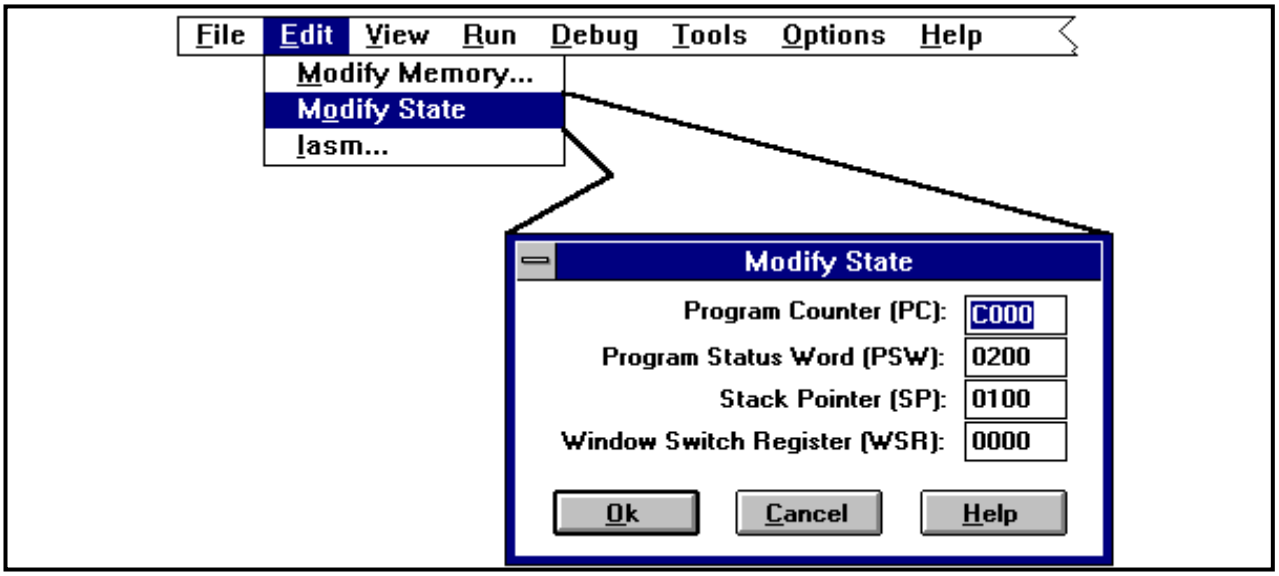
С помощью команды ***Modify Memory*** можно заполнить блок ячеек памяти модуля определенным значением. При вызове этой команды на экране появится окно, в котором необходимо указать нижнюю (***From***) и верхнюю (***To***) границы модифицируемого блока памяти, тип данных, которыми Вы хотите заполнить блок (***BYTE***, ***WORD***, ***LONG*** или ***REAL***), и значение, которым необходимо заполнить заданную область памяти (***With***).

После нажатия кнопки ***Ok*** происходит заполнение памяти и обновление открытых окон меню ***View***.

Внимание: вводимые Вами данные *Монитор* воспринимает как 16-ричные числа.

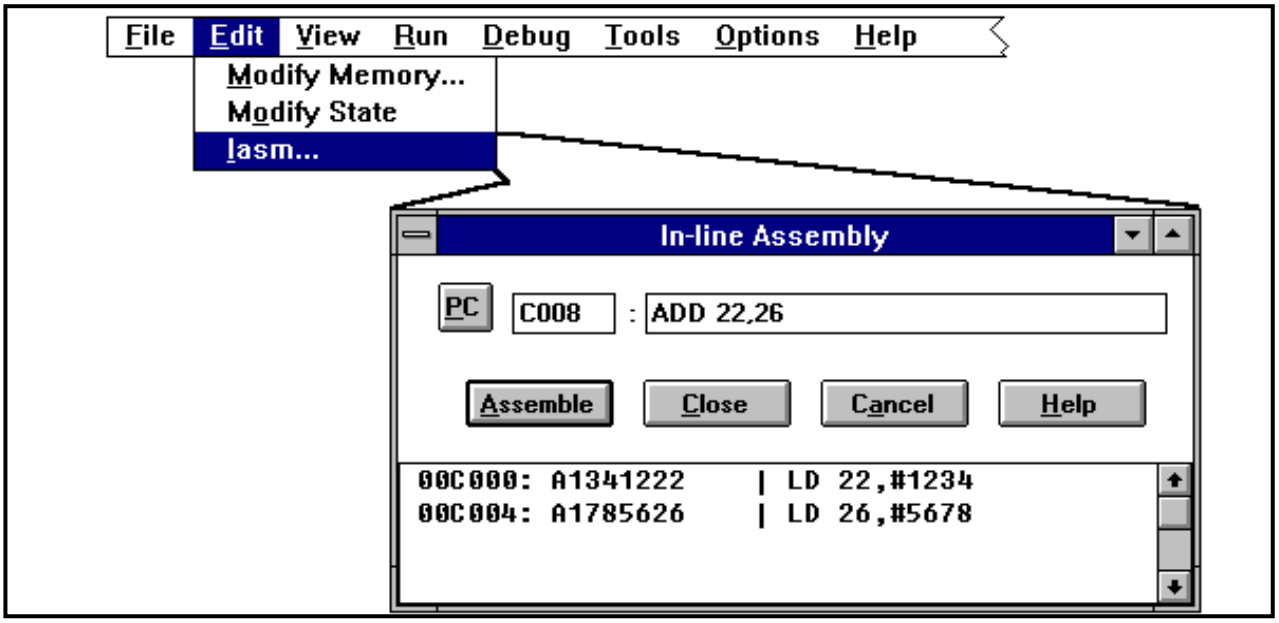
С помощью кнопки ***Last Value*** можно восстановить предыдущие значения ***From***, ***To***, ***With***.

Команда ***Modify State*** вызывает окна с текущим значением программного счетчика (***PC***), слова состояния программы (***PSW***), указателя стека (***SP***), регистра выбора окон (***WSR***). Содержимое этих регистров, управляющих работой процессора, может быть изменено непосредственно в окнах.



3.10. Встроенный ассемблер

Монитор содержит встроенный построчный ассемблер, который может быть вызван командой ***Iasm*** из меню ***Edit***. Использование этого ассемблера позволяет делать вставки небольших фрагментов программ непосредственно в память модуля. Таким образом может производиться коррекция пользовательских программ, не прерывая сеанса тестирования. Отметим, что встроенный ассемблер не является заменителем настоящей программы-ассемблера, такой как ***ASM96***, однако полезен при отладке программ. Команда ***Iasm*** может быть вызвана независимо от того, запущена программа пользователя или нет, однако очевидно, что пытаться модифицировать программу во время ее исполнения - вещь довольно рискованная. Чтобы запустить процесс ассемблирования и модификации памяти оценочного модуля, введите в левое окно адрес, по которому должна располагаться команда, в следующее окно - мнемоническое обозначение команды и нажмите кнопку ***Assemble***. После исполнения этой команды произойдет обновление окна *Дизассемблера* (если оно открыто).



Встроенный ассемблер допускает использование символических имен. Символическое имя должно предваряться одним из следующих символов: точка ***(.)***, двоеточие ***(:)***, двойные кавычки ***(")*** или вертикальная черта ***(|)***.

Пример:

```
MUL   .TEMP,.REG_1,.REG_2
LD    .REG_1,#.INIT_VAL
```

3.11.Сохранение содержимого памяти оценочного модуля на диске

Команда ***Save Memory*** из меню ***File*** позволяет сохранять фрагмент памяти отладочного модуля в виде объектного файла. После того, как Вы зададите границы сохраняемого фрагмента (***From*** и ***To***), укажите в появившемся окне устройство, каталог и имя сохраняемого файла.

Внимание: символьная таблица в записываемом файле не сохраняется.

3.12. Другие возможности *Монитора*

При возникновении проблем в процессе отладки программ Вы можете получить необходимую дополнительную информацию об особенностях тех или иных блоков или функций микроконтроллера с помощью команд раздела ***Tools*** главного меню. Эти команды - ***ModelBUILDER*** и ***ApBUILDER*** - дублируются в клавишной строке кнопками, которыми обеспечивается быстрый вызов соответствующих программ:

С помощью команд ***List***, ***Log***, ***Begin record*** раздела ***Debug*** главного меню можно вести протокол сеанса работы в *Мониторе* с сохранением его на диске. Предварительно необходимо отметить, в каком из форматов - текстовым, "логическом" или в обоих форматах Вы желаете вести протокол. Для этого достаточно щелкнуть "мышью" по соответствующей строке меню ***Debug***. Функция ведения протокола сохранит Ваши команды в порядке их инициализации, точно так же будут сохранены реакции *Монитора* на эти команды. Перед началом протоколирования можно указать имена файлов, в которые предполагается записывать протокол - это делается при вызове пунктов ***List File Name*** или ***Log File Name*** в меню ***Options***. Начало протоколирования инициируется "щелчком" по строке ***Begin Record***. Не забудьте в конце сеанса работы закрыть файл протокола "щелчком" по появившейся в том же меню строке ***End Record***. Управлять началом/окончанием ведения протокола можно с помощью кнопки клавишной строки.

Приложение

В этом разделе приводится краткий обзор возможностей и характеристик оценочных модулей, используемых при отладке приложений в среде ***Debug Monitor***.

Корпорация ***Intel*** предлагает разработчикам два основных типа модулей: ***Demo Board*** (демонстрационные модули), которые иногда называют также ***Development Board*** (модули проектирования) и ***Evaluation Board*** (оценочные модули). И те, и другие поддерживают процесс разработки систем и позволяют не только отлаживать создаваемые программные и аппаратные средства в среде ***Debug Monitor***, но и оценивать их эффективность в конкретном приложении с помощью системы моделирования ***ModelBUILDER***.

1. Демонстрационный модуль (Target Board)

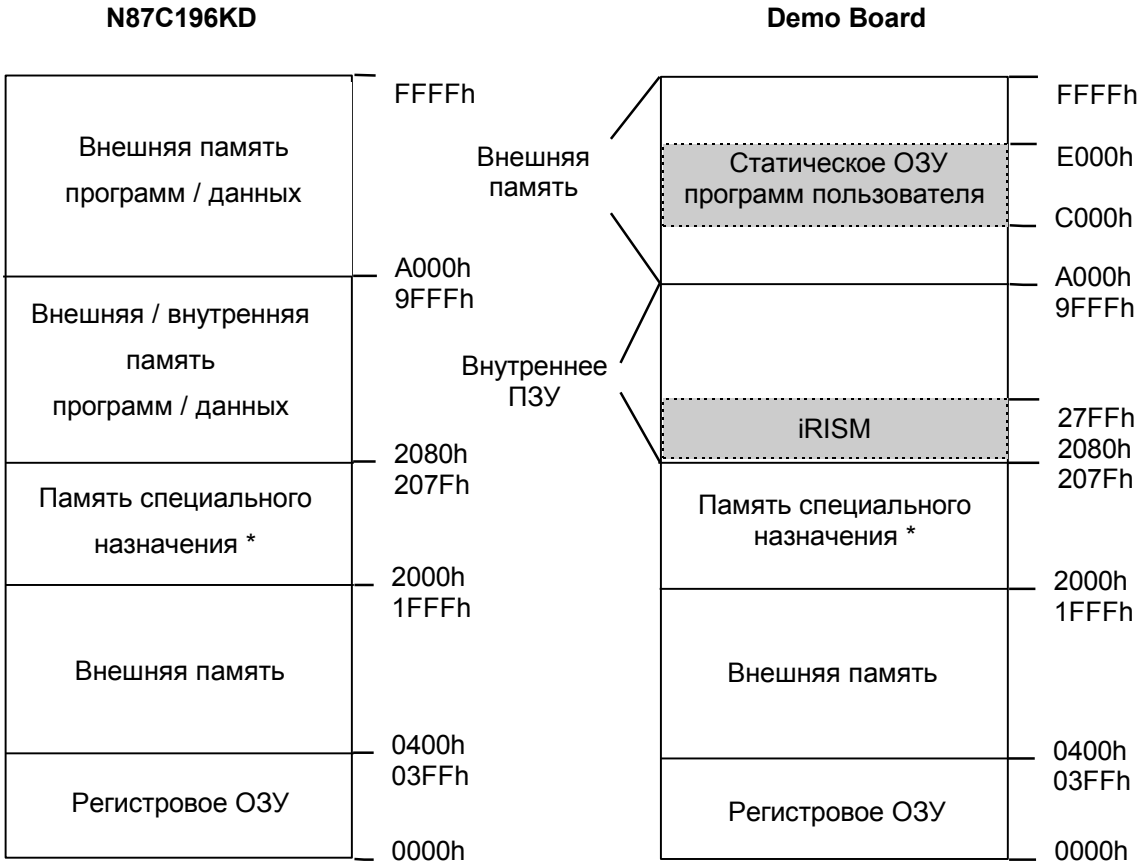
Как следует из названия, главное назначение этого модуля - познакомить разработчика с новым для него типом микроконтроллера семейства ***MCS-196***, продемонстрировать его отличительные особенности. Вполне разумное требование к такого рода изделиям - низкая стоимость, которая достигается за счет предельно малых аппаратных затрат. Действительно, взглянув на принципиальную схему модуля **DVMCS96R**, Вы увидите, что она содержит минимальное количество компонентов: микроконтроллер **N87C196KD** (со встроенным ПЗУ), микросхему статического ОЗУ, светодиодную матрицу и компоненты обвязки ("защелка" адреса, согласователи уровней, подтягивающие резисторы и т.п.).

Но, как известно, за все приходится платить - часть ресурсов микроконтроллера используется для поддержки функций отладки, и ,следовательно, недоступна для разработчика. Эти ограничения крайне важно знать до начала работы, поэтому рассмотрим их подробней.

Управляющая ***iRISM***-программа “зашита” во внутреннее ПЗУ микроконтроллера - именно ей передается управление модулем при включении питания. Эта программа занимает всего лишь несколько сот байт, но все ПЗУ занимает 32KB, которые, естественно, не доступны пользователю. На рисунке приводятся карты памяти микроконтроллера и демонстрационного модуля.

Как видно из рисунка, программа пользователя объемом до 8-ми KB может быть размещена в статическом ОЗУ начиная с адреса **0C000h**. Именно с этого адреса должна начинаться Ваша программа, если Вы хотите запустить ее на выполнение на демонстрационном модуле!

Следует помнить и о том, что вектора прерываний также расположены в ПЗУ, и при обслуживании прерываний управление передается по **фиксированным** адресам. Ниже приводится таблица, содержащая эти адреса.



**)Память специального назначения включает таблицы векторов прерываний, байт конфигурации кристалла и др. (подробности смотри в соответствующем руководстве пользователя).*

Вектор прерывания	Номер прерывания	Адрес ячейки вектора процессора	Содержимое вектора в ПЗУ модуля	Содержимое вектора PTS в ПЗУ модуля
timer_overflow	0	2000h	D000h	360h
ad_done	1	2002h	D020h	368h
hsi_data	2	2004h	D040h	370h
hso_event	3	2006h	D060h	378h
hsi_zero	4	2008h	D080h	380h
software_timer	5	200Ah	D0A0h	388h
serial_port	6	200Ch	D0C0h	390h
external_int	7	200Eh	D0E0h	398h
serial_txd	8	2030h	D100h	3A0h
serial_rxd	9	2032h	*	3A8h
hsi_entry_4	10	2034h	D140h	3B0h
timer2_capture	11	2036h	D160h	3b8h
timer2_overflow	12	2038h	D180h	3C0h
external_int	13	203Ah	D1A0h	3C8h
hsi_fifo_full	14	203Ch	D1C0h	3D0h
NMI	15	203Eh	*	-
TRAP		2010h	*	-
illegal_Opcode		2012h	RET	-

***) Эти прерывания зарезервированы для использования iRISM-программой.**

Например, при обслуживании прерывания ***ad_done*** (вектор которого находится в ПЗУ микроконтроллера по адресу **02002h**) произойдет передача управления по адресу **0D020h**. Вектор следующего прерывания hsi***_data*** указывает на адрес **0D040h**. Таким образом, под программу обслуживания прерывания ***ad_done*** отводится всего 32 байта. Обычно этого объема памяти достаточно лишь для размещения “шапки” программы обработки прерывания - команд передачи управления ее “главной” части, а саму “главную” часть поместить где-нибудь в другом месте ОЗУ программ пользователя.

В том случае, если обслуживание прерывания ***ad_done*** ведется через сервер периферийных транзакций ***(PTS), PTS*** будет извлекать свои команды из ячеек **368h-36Fh**. Поскольку максимальная длина блока команд ***PTS*** не превышает 8-ми байт, отведенной области вполне достаточно для размещения этого блока.

Кроме упомянутых, необходимо учитывать еще ряд ограничений, которые перечисляются ниже общим списком без дополнительных пояснений:

1. Стартовый адрес программы пользователя должен быть равен 0C000h.

2. Программы и данные пользователя могут размещаться по адресам от 0C000h до 0E000h при использовании ОЗУ модуля. При подключении внешнего ОЗУ программы и данные пользователя могут размещаться по адресам от 0A000h до 0FFFFh.

3. На области памяти от 0D000h до 0D1DFh и от 360h до 3D8h накладываются ограничения, если программа пользователя использует прерывания (см. выше).

4. Шестнадцать байт регистрового ЗУ 0E0h-0EFh зарезервированы iRISM-программой.

5. Вход немаскируемого прерывания NMI и прерывание NMI зарезервированы iRISM-программой.

6. Команда TRAP зарезервирована iRISM-программой.

7. Вывод порта P2.5 используется для связи модуля с управляющим компьютером.

8. Выводы TxD и RxD используются для связи с управляющим компьютером.

9. Прерывание приемника последовательного порта зарезервировано iRISM-программой.

10. Два слова из стека пользователя зарезервированы iRISM-программой.

2. Оценочный модуль (Evaluation Board)

Основное назначение оценочных модулей - дать возможность пользователю проектировать, тестировать и отлаживать программы еще до того, как приложение, для которого пишутся эти программы, появится “в железе”. Естественно, все те многочисленные ограничения, которые накладывают на проектируемую программу демонстрационные модули, в данном случае нежелательны. Плата за сокращение накладываемых ограничений - существенное усложнение аппаратной части модуля и **iRISM**-программы. Рассматриваемый здесь оценочный модуль **EV80C196KC** содержит: микроконтроллер **N87C196KC16**, 2 микросхемы ПЗУ с **iRISM**-программой, 3 розетки для микросхем памяти пользователя, программируемый контроллер последовательного порта **I82510** и все необходимые компоненты обвязки.

Применение программируемой логической матрицы для операций переопределения карты памяти и внешней микросхемы для связи с компьютером верхнего уровня сняло почти все ограничения, свойственные демонстрационным модулям. Остались лишь несколько:

1. Программы и данные пользователя могут размещаться в ОЗУ модуля емкостью до 24 Кбайт по адресам от 2000h до 7FFFh. Для расширения этих границ можно подключить дополнительное ОЗУ.

2. Участок памяти 2014h-202Fh зарезервирован iRISM-программой.

3. Участок памяти 1E00h-1EFFh зарезервирован для контроллера последовательного порта.

4. Пользователь не должен изменять вектор программного прерывания *TRAP* (размещается по адресу 2010h) и вектор немаскируемого прерывания *NMI* (размещается по адресу 203Eh).

5. Команда *TRAP* зарезервирована iRISM-программой.

6. Вход *NMI* используется для связи с управляющим компьютером.

7. Восемь ячеек регистрового ЗУ 30h-38h зарезервированы iRISM-программой.

8. Два слова из стека пользователя зарезервированы iRISM-программой.

Литература:

1. 196 Microcontroller Family Fact Sheet
2. 8XC196KC/8XC196KD User's Manual
3. EV80C196KC Microcontroller Evaluation Board User's Manual
4. 196KD-20 Microcontroller Target Board User's Manual

При разработке пособия использованы также гипертекстовые руководства, входящие в состав систем:

1. ApBuilder V1.21
2. ModelBuilder
3. Debug Monitor