— — — ФИРМА НИППЕЛЬ ПРЕДСТАВЛЯЕТ — — —

Описание модуля часов "Nippel Clock Card"

для ЭВМ Агат 9

Руководство пользователя и программиста

Автор: Голов А.А.

ВНИМАНИЕ

Перед установкой модуля часов внимательно ознакомьтесь с инструкцией по эксплуатации. Помните, что в изделии используется малопотребляющая КМОП микросхема, "боящаяся" статических зарядов, а постоянно находящиеся под напряжением узлы платы могут стать причиной выхода из строя и других элементов при случайном замыкании контактных площадок.

ОСНОВНЫЕ ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ

Модуль "Nippel Clock Card" позволяет вести счет текущего времени в часах, минутах, секундах и даты в виде числа, месяца и года, а также дня недели. Автоматически учитываются число дней в месяце и високосные года.

Благодаря наличию на плате литиевого элемента питания МЛ2325, часы никогда не прерывают своего хода. Они автоматически переходят на автономное питание в момент выключения ЭВМ и на сетевое питание в момент ее включения. Время непрерывного хода часов от нового элемента МЛ2325 составляет не менее 12 месяцев.

Текущее время и дата могут быть свободно считаны и переустановлены программным способом.

Кроме обычных функций часов и календаря, модуль "Nippel Clock Card" предоставляет в распоряжение пользователя 50 ячеек энерго-независимой памяти и три вида прерываний:

1. Прерывание, происходящее каждую секунду непосредственно после очередной смены информации в часах.

2. Циклические прерывания от внутреннего программируемого делителя, с одной из частот: 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096 и 8192 Гц.

3. Прерванние от "будильника" по значению предварительно установленного времени.

Установку и замену элемента питания пользователь может без особого труда производить самостоятельно, воспользовавшись отверткой.

На плате часов установлен подстроечный конденсатор, позволяющий подстраивать точность хода часов.

Автор разработки модуля: В.Березутский.

ИНСТРУКЦИЯ ПО ЭКСПЛУАТАЦИИ

1. Меры предосторожности.

Чтобы гарантировать долгую и надежную работу модуля "Nippel Clock Card", Вам нужно соблюдать несколько не слишком затруднительных правил.

Беритесь руками только за торцы платы и не касайтесь токоведущих дорожек, чтобы случайно накопившийся на Вас статический заряд не вывел из строя микросхему MC146818 (КР512ВИ1).

Не кладите плату с установленным элементом питания на токоведущие поверхности, так как короткое замыкание может привести к выходу из строя как элемента питания, так и других элементов схемы.

Хранение модуля без элемента питания лучше всего производить в металлических или металлизированных упаковках, либо завернутым в электропроводную металлическую фольгу. Если же на плате остается элемент питания, то лучшим способом хранения будет картонная коробка с прочными (непрокалываемыми) картонными прокладками, отделяющими друг от друга несколько плат. Ни в коем случае не применяйте упаковки из полистирола и оргстекла, которые являются хорошим накопителем статических зарядов.

2. Установка элемента питания.

Элемент питания МЛ 2325 фиксируется на плате металлической пластинкой, удерживаемой двумя стягивающими винтами. Элемент устанавливается положительным электродом вверх – сторона, на которой написано его название, там же находится и знак "+". Перед тем, как взять плату в руки, снимите с себя возможный статический заряд, например, касаясь металлической части батареи отопления.

Прихватите металлическую пластинку двумя винтами, чтобы тем не менее элемент питания свободно проходил под ней. После чего, удерживая его двумя пальцами и вращая поочередно два винта, слегка прижмите элемент. Проверьте, не возникло ли перекоса пластинки в одну из сторон. После чего можете осуществить окончательную фиксацию.

Помните, что пластинка должна лишь слегка прижимать элемент МЛ 2325, который не допускает деформации корпуса, приводящей к выходу его из строя. Так как потребление тока от элемента питания ничтожно мало, качество электрического контакта не очень важно, механическая же прочность крепления достигается уже при очень малой силе стягивания винтов, что можно проверить, попытавшись сдвинуть элемент питания в сторону.

3. Установка модуля в ЭВМ.

Установку и изъятие модуля из ЭВМ производите только при отключенном питании. Знайте, что наибольшая вероятность отказа модуля или ЭВМ возникает именно при изъятии или установке плат в необесточенную машину.

Откройте крышку Вашей ЭВМ Агат 9 и осмотрите, какие разъемы у нее не заняты. Согласно принятой терминологии, разъемы у Агата 9 нумеруются слева направо (если смотреть с фронтальной стороны корпуса), начиная с первого и так далее до шестого.

В принципе, "Nippel Clock Card" может быть установлена Вами в любой свободный разъем машины, но наиболее удобным для Вас, видимо, будет установка ее в первый (самый левый) разъем ЭВМ, который у Вас, скорее всего, свободен, так как другие, разработанные производителем Агата 9 модули, не способны функционировать в первом разъеме, и он, как правило, пустует.

У модуля "Nippel Clock Card" нет никакого ключа, не позволяющего установить его в неверное положение, поэтому будьте предельно внимательны, устанавливая плату в машину. Есть общее правило; элементы, находящиеся на плате, должны располагаться справа (вид с фронтальной стороны), как и на всех остальных платах ЭВМ. При установке платы удерживайте ее двумя указательными пальцами за боковые торцы и, надавливая двумя большими пальцами на верхний торец платы, немного покачивая вдоль разъема, углубите ее до упора.

Еще раз проверьте, правильно ли Вы установили модуль – детали справа,

элемент питания ближе к тыльной стороне машины и, только убедившись в правильности, можете подать питание на ЭВМ.

4. Проверка модуля и установка времени.

На имеющемся в комплекте с модулем "Nippel Clock Card" магнитном диске, кроме данной инструкции, имеется программа для тестирования и демонстрации возможностей модуля. На диске Вы найдете два совершенно одинаковых объектных файла типов "B" и "K", называющихся соответственно BTESTCLOCK и KTESTCLOCK.

Запустите программу TestClock; это надо сделать сразу после установки элемента питания, иначе потребление тока модулем возрастет в 15 раз и этого элемента надолго не хватит. Если Вы еще не установили часы в машину, то получите уведомление о том, что плата часов не найдена, и программа прекратит выполнение. Если же Вы все сделали правильно, а плата тем не менее не найдена, то, значит, она находится в совершенно не рабочем состоянии.

Когда плата найдена, на экране появляется информация о текущем времени, дате, содержимом "будильника", которые при первом после установки элемента питания запуске будут в случайном состоянии.

Обратите первоочередное внимание на две надписи: "Часы на X разъеме" и "Память (не) в порядке". Номер разъема в первой надписи должен соответствовать тому, в который Вы установили модуль, а память должна быть, естественно, "в порядке".

Часы в данной программе могут находиться в двух режимах: счета или установки времени. Признаком режима установки времени служит наличие курсора желтого цвета на одной из альтернатив: "Время", "Дата" и т.д. Переключение режима производится клавишей "ВВОД".

Если информация на Ваших часах не соответствует действительности, то установите показания часов, для чего в первую очередь переведите их в режим установки. В режиме установки часы стоят, поэтому, переводя в режим счета, Вы одновременно их и запускаете.

Переводя большой желтый курсор стрелками вверх/вниз, Вы выбираете параметр для изменения, причем на всех режимах, кроме дня недели, Вы увидите еще и маленький белый курсорчик, указывающий на изменяемую цифру, перемещаемый стрелками влево/вправо. Нажимая на цифровые клавиши, Вы можете установить новое показание любого из параметров. День недели задается цифрами от 1 до 7, соответственно от понедельника до воскресенья. Для проверки работы будильника, введите в часах и минутах букву "X", а в секундах любое число, например 25. Наличие буквы "X" в будильнике показывает, что эти цифры не учитываются при сравнении времени. В нашем случае это означает, что будильник будет срабатывать на каждой 25 секунде.

Помните, что соблюдение правильности вводимой информации полностью ложится на Вас. При неверно введенной информации, часы могут вести счет непредсказуемым образом.

Переведите часы в режим счета нажатием клавиши "ВВОД". Часы должны начать ход с установленного Вами времени. Отображение показаний производится непосредственно по секундным прерываниям, при неисправности которых никаких изменений времени не будет. При срабатывании будильника справа от его значения появляется надпись "Сработал", исчезающая через десять секунд.

Для проверки циклических прерываний от делителя частоты используется нижняя часть экрана, где Вы увидите бегущую строку, перемещающуюся с частотой циклического прерывания. Пользуясь клавишами "F1" и "F2", Вы можете выбирать другую частоту прерываний, и тем самым менять скорость. Текущая частота прерываний обозначается желтым инверсным курсором. Естественно, отсутствие бегущей строки свидетельствует о неисправности платы.

Когда Вы полностью убедитесь в работе платы при включенной ЭВМ, проверьте ее работу от внутреннего элемента питания, для чего выключите машину, и, включив опять через минуту, вновь запустите программу TestClock, и проверьте показания часов.

РУКОВОДСТВО ДЛЯ ПРОГРАММИСТА

В руководстве для программиста мы расскажем о внутренней архитектуре модуля "Nippel Clock Card", способах доступа и управления им, опишем все соглашения нормального функционирования контроллера, а также приведем рекомендации и примеры программного обеспечения и поддержки некоторых функций модуля. Кроме того, мы приведем текст – пример драйвера для использования в программах на языке ассемблера и комментарии к программе "PrimerClk", демонстрирующей способы использования часов в программах на BASIC. Вы также найдете на диске поставки исходный текст программы "TestClk".

1. Внутренняя структура модуля.

В качестве основы модуля "Nippel Clock Card" использована микросхема контроллера часов MC146818 фирмы Motorola, Inc., широко используемая в зарубежных разработках (например, вы можете встретить ее в контроллере часов IBM AT), либо ее аналог КР512ВИ1, производства минского НПО "Интеграл".

С точки зрения программы, микросхема представляет из себя 64 ячейки памяти, в большинство из которых можно производить как запись, так и чтение информации, кроме того, модуль вырабатывает прерывания IRQ (по вектору $FFFE–$FFFF). Первые 14 ячеек ($00–$0D) используются непосредственно для работы часов, остальные 50 ($0E–$3F) могут свободно использоваться как энергонезависимая память машины, для хранения разных признаков и режимов (эта свобода не является полной, т.к. по принятым соглашениям большинство ячеек зарезервировано).

Для обращения к ячейкам из программ выделены два адреса в области Device Select, т.е. области $C0X0–$C0XF. Чтобы согласовать возможное размещение на одной плате нескольких полезных устройств, таких как контроллеры принтера, мыши и предлагаемых часов, были выбраны непересекающиеся адреса обращения ко всем этим устройствам. Как известно,

контроллер принтера использует адреса $C0X0-$C0X3, часам же зарезервировали адреса $C0X6 для размещения адреса ячейки, $C0X7 для чтения/записи данных.

В таблице 1 представлено распределение системных ячеек часов и указано их назначение:

Таблица 1

Адрес	Данные
$00	Секунды
$01	Секунды ("будильник")
$02	Минуты
$03	Минуты ("будильник")
$04	Часы
$05	Часы ("будильник")
$06	День недели
$07	Число
$08	Месяц
$09	Год
$0A	Регистр A
$0B	Регистр B
$0C	Регистр C
$0D	Регистр D

Четыре ячейки $0A-$0D, так называемые регистры A-D, предназначены для управления и контроля работы микросхемы. Каждый бит, либо группа битов этих регистров, имеет свое уникальное назначение. В таблице 2 приведено условное обозначение этих битов:

Таблица 2

Адрес	D7	D6	D5	D4	D3	D2	D1	D0
$0A	UIP[1]	DV2	DV1	DV0	RS3	RS2	RS1	RS0
$0B	SET	PIE	AIE	UIE	SQWE	DM	24/12	DSE
$0C	IRQF[1]	PF[1]	AF[1]	UF[1]	0[1]	0[1]	0[1]	0[1]
$0D	VRT[1]	0[1]	0[1]	0[1]	0[1]	0[1]	0[1]	0[1]

Примечание. Из разрядов, помеченных[1], процессор может только считывать информацию.

Рассмотрим назначение каждого разряда в отдельности:

Регистр A:

UIP – Единица в этом разряде означает, что происходит или начнется менее чем через 244 мкс цикл обновления информации о времени. Длительность цикла обновления 1984 мкс при тактовой частоте 32768 Гц или 248 мкс при других тактовых частотах. UIP можно считывать, на него не действуют сигнал СБРОС. Записав единицу в разряд SET регистра B, можно запретить обновление и тем самым сбросить UIP.

DV0–DV2 – Устанавливают режим работы внутреннего делителя частоты в соответствие с используемой тактовой частотой (см. таблицу 3). В "Nippel Clock Card" используется тактовая частота 32768 Гц.

RS0–RS3 – Устанавливают частоту циклических прерываний (таблица 4).

Таблица 3

DV2	DV1	DV0	Тактовая частота, Гц
0	0	0	4 194 304
0	0	1	1 048 576
0	1	0	32 768
1	1	1	Сброс делителя

Таблица 4

RS3 – RS0	Тактовая частота, Гц	
	4 194 304 или 1 048 576	32 768
	Частота прерываний	Частота прерываний
$0	—	—
$1	32 768	256
$2	16 384	128
$3	8 192	8 192
$4	4 096	4 096
$5	2 048	2 048
$6	1 024	1 024
$7	512	512
$8	256	256
$9	128	128
$A	64	64
$B	32	32
$C	16	16
$D	8	8
$E	4	4
$F	2	2

Регистр B:

SET – Если в этом разряде записан ноль, то каждую секунду выполняется цикл обновления информации о текущем времени и сравнении текущего времени с заданным временем срабатывания "будильника". Единица в этом разряде запрещает обновление, давая возможность записать в регистры времени, календаря и будильника новые значения.

PIE – Единица в этом разряде разрешает прерывания с периодом, задаваемым разрядами RS3-RS0 регистра A. Сбрасывается при подаче сигнала СБРОС.

AIE – Единица в этом разряде разрешает прерывания от будильника. Сбрасывается при подаче сигнала СБРОС.

UIE – Единица в этом разряде разрешает прерывания по окончании цикла обновления. Сбрасывается при подаче сигнала СБРОС.

SQWE – Разрешение выдачи сигнала на выход SQW. Сбрасывается при подаче сигнала СБРОС. В "Nippel Clock Card" не используется.

DM – Единица в этом разряде означает, что данные о времени и дате представлены в двоичном виде, ноль – в двоично-десятичном. Значение этого разряда нельзя изменять без повторной записи начальных значений в ячейки времени и календаря.

24/12 – Устанавливает 24-часовой (1) или 12-часовой (0) режим счета времени. В 12-часовом режиме время после полудня (PM) отмечается единицей в старшем разряде регистра часов (адрес $04).

DSE – Разрешение автоматического перехода на летнее время, по правилам принятым в США. Когда в этом разряде записана единица, то в последнее воскресенье апреля после 01 час 59 мин 59 сек автоматически устанавливается 03 час 00 мин 00 сек, а в последнее воскресенье октября после 01 час 59 мин 59 сек следует 01 час 00 мин 00 сек.

Регистр C

IRQF – Флаг запроса прерываний. Устанавливается в единицу, когда сработало хотя бы одно из разрешенных в данный момент прерываний. Другими словами, это означает, что прерывание произошло именно из "Nippel Clock Card". Флаг сбрасывается после чтения регистра C или сигналом СБРОС.

PF – Устанавливается в единицу сигналом от внутреннего делителя частоты, выбранным в соответствии с разрядами RS3-RS0 регистра A. Флаг сбрасывается после чтения регистра C или сигналом СБРОС.

AF – Устанавливается в единицу при совпадении текущего времени с временем, указанным в регистрах будильника. Флаг сбрасывается после чтения регистра C или сигналом СБРОС.

UF – Устанавливается в единицу после окончания цикла обновления (ежесекундно). Флаг сбрасывается после чтения регистра C или сигналом СБРОС.

Регистр D

VRT – В этом разряде устанавливается ноль при низком уровне на входе PS, входе сбоя питания. Единица устанавливается только считыванием регистра D. В "Nippel Clock Card" не используется.

2. Основные вопросы программирования.

Вы, наверное, уже поняли, что часы могут работать в нескольких разных режимах, поэтому для осуществления совместимости разных программ требуется соблюдать некоторые соглашения по представлению и обработке информации в часах.

Итак, в модуле "Nippel Clock Card" используется кварцевый генератор на 32768 Гц (биты DV2-DV0 должны иметь значение :010), данные о времени и дате представляются в двоичном виде (бит DM=1), интервал счета 24 часа (бит 24/12=1), автоматический переход на летнее время не используется (бит DSE=0).

Информация в счетчиках представлена просто двоичными числами. Секунды

и минуты могут принимать значения от 0 до 59 ($0-$3B), часы от 0 до 23 ($0-$17), день недели от 1 до 7 ($1-$7), число от 1 до 28..31 ($1-$1C..$1F), в зависимости от месяца и года, месяц от 1 до 12 ($1-$C), год от 0 до 99 ($0-$63). Если в одну из ячеек будильника записан код с двумя старшими разрядами равными 1 ($C0-$FF), эта ячейка переходит в безразличное состояние и не учитывается при сравнении с текущим временем.

Как уже было сказано выше, для доступа к ячейкам микросхемы MC146818 выделены два адреса - $C0X6 для установки адреса и $C0X7 для чтения/записи данных. Адреса управления формируются по стандартным правилам - т.е. первому разъему соответствуют адреса $C096, $C097, второму $C0A6, $C0A7, третьему $C0B6, $C0B7 и т.д. Адрес в $C0X6 нужно устанавливать перед каждым обращением к $C0X7, так как после чтения/записи он теряется. Так как у микросхемы MC146818 имеются только 64 ячейки памяти, два старших бита восьмиразрядного адреса ячейки просто игнорируются, поэтому, например, адреса $0E и $CE аналогичны.

Перед чтением информации о времени нужно проанализировать состояние бита UIP регистра A, и если он покажет, что сейчас идет обновление информации, то читать информацию нельзя, так как ее правильность не гарантируется (это правило не относится к регистрам A-D). Процедура чтения всей нужной информации должна уложиться в 244 мкс, потому что именно за это время до начала обновления бит UIP устанавливается в единицу.

Все изменения в ячейках текущего времени и даты нужно производить, предварительно остановив счет времени, для чего в бит SET регистра B нужно записать 1. Следует отметить, что остановка обновления информации не прерывает работы делителя частоты, отмеряющего секундный интервал. Это означает, что, во-первых, останавливать обновление нужно, дождавшись конца текущего цикла обновления, как при чтении информации, во-вторых, изменения в календаре не будут в этом случае нарушать правильность отсчета секунд, и, в-третьих, для исключения элемента случайности в отсчете первого интервала после коррекции секунд, нужно сбрасывать

делитель записав в биты DV0-DV2 значения :111 и после снова :010.

Наиболее удобным способом считывания информации о текущем времени и дате является использование встроенного прерывания, происходящего ежесекундно, в конце каждого цикла обновления (естественно удобство обеспечивается только если логика работы программы не противоречит использованию прерываний). Это прерывание разрешается установкой бита UIE регистра B в единицу и стробируется установкой в единицу бита UF регистра C.

После прочтения регистра C все четыре бита - флага, содержащихся в нем, автоматически сбрасываются в ноль. Это значит, что первой (после сохранения регистров и CLD) командой программы обработки прерывания должна быть операция чтения регистра C и сохранения его в специальной ячейке, после чего можно, анализируя биты, принимать соответствующие меры. Нужно помнить, что ситуации, когда срабатывают одновременно два или три прерывания, не только вполне вероятны, но и обычны (т.к. все изменения в часах происходят синхронно с изменениями в делителе), что требует обязательной проверки и обработки всех битов-признаков, конечно, только в том случае, если Вы их вообще собирались обрабатывать.

Так как во время обработки прерывания бит I процессора (маска прерывания) устанавливается в единицу, программу обработки прерывания уже ничто не прервет. Тем не менее, если в это время придет сигнал прерывания, он будет запомнен в соответствующем бите-признаке и прервет процессор непосредственно после выполнения команды RTI. Важно только соблюдать, чтобы время выполнения программы обработки прерывания не оказалось больше времени периода соответствующего прерывания, так как в этом случае программа зависнет. По той же причине регистр C должен быть прочитан хотя бы раз программой обработки прерывания. Если программа не совершит чтения регистра C, то машина зависнет, так как не будут сброшены флаги, и, выполнив команду RTI, процессор сразу же опять прервется.

Например, в программе "TestClock" для реализации такого опасного прервания как 8192 Гц используется операция повторного чтения регистра C

непосредственно перед командой RTI, чтобы предотвратить зависание, т.к. время периода этого прерывания (122 мкс) недостаточно для перевывода бегущей строки. Конечно, некоторые из запросов не будут обработаны, и реальная частота вызовов будет меньше 8192 Гц, но прерывания с такой частотой реализованы в этой программе исключительно для проверки всех режимов работы модуля, поэтому реальная частота не имеет значения. Более подробно с этим приемом и другими Вы сможете ознакомиться из исходного текста программы "TestClock".

3. Ассемблерные процедуры – примеры управления часами

В конце этой главы приведен исходный текст драйвера часов "DClock", для связи с которым требуются три однобайтовых переменных CLPARM1–CLPARM3 (описаны непосредственно за ORGом). Все операции происходят через точку входа "DCLOCK". В регистре X должен находиться номер вызываемой команды (0–6,$FF). Как работают команды, нетрудно понять из головных комментариев к ним.

Драйвер часов, прежде всего, должен определить разъем, на котором находится модуль. Как уже отмечалось выше, на одном разъеме могут быть установлены несколько устройств с разнесенными адресами доступа, поэтому процедуры поиска и идентификации лучше всего оформлять отдельно для каждого искомого устройства, со своим циклом просмотра разъемов. При таком решении программа поиска устройств принимает вид группы вызовов подпрограмм, каждая из которых определяет номер разъема конкретного устройства, либо его (устройства) отсутствие, что делает программу элегантной, имеющей структурный вид и легко понимаемой.

В приведенном ниже драйвере часов используется специальная команда $FF (стандартный номер функции инициализации для драйверов фирмы Ниппель), инициализирующая драйвер и производящая в том числе поиск разъема контроллера. Эта команда должна быть вызвана при выполнении стартовых процедур программы, в противном случае драйвер будет работать как при

отсутствии часов.

Алгоритм поиска контроллера основан на попытке записи в ячейки $E и $F, часов значений 1 и 2 соответсвенно с последующей проверкой и записи значений 3 и 4 туда же, также с последующей проверкой. Кроме того, программа предварительно сохраняет содержимое ячеек $E и $F и восстанавливает после проверки текущего разъема. По результатам проверки делаются соответствующие выводы: либо номер разъема фиксируется, либо производится переход к следующему. Если контроллер не найден, номер разъема будет установлен равным 0 и подпрограммы управления часами выполняться не будут, чтобы не происходили "неприятные" передергивания портов других плат, а читаться по запросам будут буферированные данные.

Кроме поиска модуля подпрограмма инициализации устанавливает соответствующие стандарту фирмы Ниппель режимы работы часов и сбрасывает флаги разрешения прерываний.

Чтобы узнать текущее время или дату, нужно выполнить команду фиксации значений (0) и запросами 1-3 прочитать время, дату или день недели.

Возникает резонный вопрос, зачем запрос времени или даты делается двухшаговым, через фиксацию значений? На то есть две причины, во-первых, такой способ позволяет доступать ко всей информации быстрее, т.к. не требуются повторные обращения к контроллеру, и исключается соответственно лишнее время ожидания готовности. Во-вторых, и это главная причина, такой способ исключает возникновение серьезных ошибок датирования. В самом деле, например, Вам нужно маркировать файлы текущим временем и датой, и Вы запрашиваете сначала время, допустим, это будет 23 часа 59 минут 59 секунд, помещаете его в заголовок файла, а затем запрашиваете дату, и как раз в этот момент началось обновление информации, и к моменту прихода готовности уже наступят следующие сутки, в результате чего Ваш файл "помолодеет" сразу на сутки, или вернее, перенесется во времени на сутки вперед. В предложенном варианте такого никогда не случится, т.к., зафиксировав сначала параметры, Вы можете размещать их в любой последовательности в любое время, хоть через несколько секунд.

Так как сигнал готовности исчезает за 244 мкс до начала обновления информации, у нас есть только 244 мкс для того, чтобы успеть считать всю нужную нам информацию. Подпрограмма FIXPAR производит это за 240 мкс, а значит, выполняет все предъявляемые временные критерии. В остальном, методы доступа к часам можно легко выяснить из текста драйвера, ниже мы обсудим только детали, не отраженные в драйвере, по причине неиспользования им режима прерываний.

Прежде всего, мы отметим один из важнейших моментов в программировании операций над информацией в часах. Если Вы используете прерывания от часов (или от других устройств, в программах обработки которых используется обращение к часам), Вы должны в обязательном порядке в начале операции по доступу к любым ячейкам часов запрещать прерывания. Эта необходимость вызвана растянутостью операций доступа к ячейкам часов на несколько команд, в течение которых вполне может произойти прерывание. Например, если Вы занесли в $C0X6 адрес и в этот момент придет прерывание от "Nippel Clock Card", программа обработки прерывания волей-неволей произведет чтение хотя бы регистра C, испортив тем самым установленный ранее Вашей программой адрес. Поэтому, например, для чтения/записи байтов из ячеек могут быть рекомендованы приведенные ниже две процедуры RDBYTE и WRBYTE.

```
*--------------------------*
* Прочитать байт из часов *
*--------------------------*
ADDRESS  DS  1            Текущий адрес ячейки КР512ВИ1

RDBYTE   PHP              Сохранить состояние флага 'I'
         SEI              Запретить прерывания
         LDX SLOT         - Установить адрес
         LDA ADDRESS
         STA $C086,X      - ;
         LDA $C087,X      Прочитать данные
         PLP              Восстановить состояние флага 'I'
         RTS
```

```
*-------------------------*
*  Записать байт в часы   *
*-------------------------*
WRBYTE   PHP              Сохранить состояние флага 'I'
         SEI              Запретить прерывания
         PHA
         LDX SLOT         - Установить адрес
         LDA ADDRESS
         STA $C086,X      - ;
         PLA              - Записать байт
         STA $C087,X      - ;
         PLP
         RTS
```

Обе эти процедуры используют ячейку ADDRESS, в которую предварительно нужно занести текущий адрес. Если в программе обработки прерываний использовать те же процедуры RDBYTE и WRBYTE для доступа к ячейкам, то необходимо позаботиться о сохранении содержимого ячейки ADDRESS на время обработки прерывния.

Обратите внимание, что в начале обеих процедур запоминается текущее состояние регистра признаков и устанавливается маска прерываний. Это гарантирует, что между установкой адреса и чтением/записью данных не произойдет прерывание и не будет испорчен установленный Вами адрес. Тем не менее, эти процедуры не гарантируют правильность работы во всех случаях. Например, если Вы производите модификацию отдельных битов и пришедшее между чтением старого и записью нового значения прерывание произведет изменение своей части информации в этом же регистре, то записываемая Вами информация будет неверной, так как не будет содержать измененной программой обработки прерывания части. В этом случае нужно заключать в скобки "PHP SEI ... PLP" весь программный код, производящий модификацию:

```
*-------------------------------------------------*
* Устанавливаем разрешение циклических прерываний *
*-------------------------------------------------*
SETPIE   PHP             Сохранить состояние флага 'I'
         SEI             Запретить прерывания
         LDA #$B         - Установить адрес ячейки
         STA ADDRESS     - ;
         JSR RDBYTE      Прочитать ячейку $0B
         ORA #:01000000  Установить бит PIE в единицу
         JSR WRBYTE      Записать в ячейку $0B
         PLP
         RTS
```

Как мы уже отмечали выше, режим автоматического перехода на летнее время не используется в модуле "Nippel Clock Card" из-за несовпадения его правил с принятыми в свое время в СССР. Кроме того, в США первым (1) днем недели считается воскресенье, а последним (7) суббота. У нас же под единицей подразумевается понедельник, под семеркой воскресенье.

Нужно заметить, что отсутствие автоматического перехода на летнее время, в принципе, можно компенсировать, реализовав его программно. Для этого зарезервирована ячейка памяти $0E, способная принимать два значения, определяющих тип текущего времени. Если в ней содержится $55 - сейчас зимнее время, если $AA - летнее. Если же в ней какое-либо другое число, то это значит, что тип текущего времени не определен (часы не были установлены и т.д.) и соответственно требует его определения.

Драйвер часов после фиксации времени и запроса типа текущего времени (ячейка $E) должен вычислить момент перехода с текущего времени на альтернативное (т.е. с летнего на зимнее или наоборот), по принятым у нас правилам и проверить, не перейден ли этот рубеж текущим временем, если такое случилось, нужно изменить тип времени и скорректировать само время в соответствующую сторону.

Так или иначе, но автор этих строк, а равно и приведенных здесь программ, пока не реализовал этот механизм в своем драйвере, т.к. он показался мне сложным в реализации (в основном имеется ввиду алгоритм нахождения момента перехода с одного времени на другое), и излишне громоздким для того, чтобы выполнять его при каждом запросе. Тем не

менее, есть еще способ выполнять этот алгоритм только при инициализации драйвера, но работающих по ночам людей это не спасет от ошибок во времяисчислении. Так или иначе, вопрос о реализации такого механизма будет рассмотрен автором и возможно включен в состав драйвера часов программы "Disk Manager".

Помимо ячейки $0E, зарезервированными для работы "Nippel OS" считаются все ячейки до $2F включительно, т.е. свободными для использования можно считать 16 ячеек от $30 до $3F. Тем не менее, нужно быть осторожным в их использовании, так как нерешаемой впрямую проблемой остается согласование использования этих ячеек из разных программ.

Мы рассказали вам об основных приемах написания программ на языке ассемблера для управления часами и надеемся, что приведенной информации будет достаточно для написания надежных и качественных программ.

В конце главы хотелось бы отметить еще одну немаловажную особеннность функционирования часов, вернее не часов, а ЭВМ Агат 9. Из-за ошибки разработчика, нельзя включать одновременно внутренние прерывания ЭВМ Агат 9 (обращаться по адресу $C040) и разрешать прерывания от "Nippel Clock Card", что приведет к логическим конфликтам (контроллер часов спроектирован с учетом этого дефекта и защищает себя и Агат 9 от электрических конфликтов на шине ЭВМ) и сбоям в работе программы. Можно сказать, что обращение по адресу $C020 (выключение внутренних прерываний) должно быть первой командой перед включением прерываний из часов. Отсюда следует, что нет возможности использовать прерывание NMI (50Гц синхронно с кадровой разверткой) и прерывания IRQ, исходящие от "Nippel Clock Card".

```
 SBTL 'Драйвер часов'
*************************************************************
*                                                           *
*       Драйвер ввода/корректировки текущего времени        *
*                                                           *
*       Пример управления часами "Nippel Clock Card"        *
*                                                           *
*              из программ на языке ассемблера.            *
* _________________________________________________________ *
*                                                           *
*    (C) 1993, Ниппель, by ALV Software, Автор: А.Голов     *
*                                                           *
*************************************************************
 ORG $2000
*===========================================================*
CLPARM1 EQU $2F0 Первый параметр
CLPARM2 EQU $2F1 Второй параметр
CLPARM3 EQU $2F2 Третий параметр

*===========================================================*
*            Вызов драйвера управления часами               *
* --------------------------------------------------------- *
*  Вход: X - команда драйверу управления часами             *
*    0 - фиксация текущей информации                        *
*    1 - запрос информации о времени                        *
*    2 - запрос информации о дате                           *
*    3 - запрос информации о дне недели                     *
*    4 - установка нового времени                           *
*    5 - установка новой даты                               *
*    6 - установка нового дня недели                        *
*  $FF - инициализация драйвера часов                       *
*  ---------------------------------------------------      *
*  Выход: В соответствии с командой                         *
*===========================================================*
DCLOCK  CPX #$FF         Инициализация
        BEQ DCLKINT      = Выполнение инициализации
        CPX #7
        BCS DCLOCKE      = Нет такой команды
        TXA              - Вызов подпрограммы
        ASL A
        TAX
        LDA DCLOCKA,X
        PHA
        LDA DCLOCKA+1,X
        PHA
DCLOCKE RTS              - ;

*--------------------------------------*
* Список подпрограмм исполнения команд *
*--------------------------------------*
DCLOCKA DDB FIXPAR-1  Фиксация информации
        DDB INQTIM-1  Запрос времени
        DDB INQDAT-1  Запрос даты
        DDB INQDAY-1  Запрос дня недели
        DDB SETTIM-1  Установка времени
        DDB SETDAT-1  Установка даты
        DDB SETDAY-1  Установка дня недели
```

```
*--------------------------------*
*  Инициализация драйвера часов  *
*--------------------------------*
DCLKINT  LDA #$10
DCLKINT0 STA CSLOT
         JSR DCLKINV    Проверить на этом разъеме
         BCS DCLKINT1   = Здесь нет часов
         LDY #$A        - Инициализировать регистр A
         LDA #:00101111 Тип кварца и прерывания 2 Гц.
         JSR CWRBYTE    - ;
         LDY #$B        - Инициализировать регистр B
         LDA #:00000110 Данные в двоичном, ход 24 часа
         JSR CWRBYTE    прерывания запрещены - ;
         JMP FIXPAR     = Зафиксировать время
DCLKINT1 LDA CSLOT      - Перейти к следующему разъему
         CLC
         ADC #$10
         CMP #$70
         BCC DCLKINT0   - ;
         LDA #0         - Признак отсутствия часов
         STA CSLOT      - ;
         RTS

*------------------------------*
*  Проверка текущего разъема  *
*------------------------------*
DCLKINV  LDY #$E       - Сохранить ячейки $E и $F
         JSR CRDBYTE   Прочитать байт
         STA CWORK0
         INY
         JSR CRDBYTE   Прочитать байт
         STA CWORK1    - ;
         LDA #2        - Записать #2 в ячейку $F
         JSR CWRBYTE   Записать байт - ;
         DEY           - Записать #1 в ячейку $E
         LDA #1
         JSR CWRBYTE   Записать байт - ;
         JSR CRDBYTE   - Проверить записалось ли
         CMP #1
         BNE DCLKINVR = Не записалось
         INY
         JSR CRDBYTE   Прочитать байт
         CMP #2
         BNE DCLKINVR = Не записалось - ;
         LDA #4        - Записать #4 в ячейку $F
         JSR CWRBYTE   Записать байт
         DEY           - Записать #3 в ячейку $E
         LDA #3
         JSR CWRBYTE   Записать байт - ;
         JSR CRDBYTE   - Проверить записалось ли
         CMP #3
         BNE DCLKINVR = Не записалось
         INY
         JSR CRDBYTE   Прочитать байт
         CMP #4
         BNE DCLKINVR = Не записалось
         CLC           -- Часы найдены
         DFB @44       Пропуск SECa
DCLKINVR SEC           -- Часов нет
DCLKINVE LDY #$E       - Востановить ячейки $E и $F
         LDA CWORK0
```

```
          JSR CWRBYTE  Записать байт
          INY
          LDA CWORK1
          JMP CWRBYTE  Записать байт - ; -- ;

*-----------------------------------*
*  Фиксация текущего времени/даты   *
* --------------------------------- *
*  Вход: -//-                       *
*  --------------------------------- *
*  Выход: информация зафиксирована  *
*-----------------------------------*
FIXPAR  LDA CSLOT
        BEQ FIXPARE     = Часов нет
FIXPAR0 LDY #$A         - Прочитать бит готовности
        JSR CRDBYTE
        AND #:10000000 'UIP'
        BNE FIXPAR0     = Информация не готова
        LDY #4          - Часы
        JSR CRDBYTE     Прочитать байт
        STA CFIXDCLK    - ;
        LDY #2          - Минуты
        JSR CRDBYTE     Прочитать байт
        STA CFIXDMIN    - ;
        LDY #0          - Секунды
        JSR CRDBYTE     Прочитать байт
        STA CFIXDSEC    - ;
        LDY #9          - Год
        JSR CRDBYTE     Прочитать байт
        STA CFIXDYER    - ;
        LDY #8          - Месяц
        JSR CRDBYTE     Прочитать байт
        STA CFIXDMON    - ;
        LDY #7          - Число
        JSR CRDBYTE     Прочитать байт
        STA CFIXDNUM    - ;
        LDY #6          - День недели
        JSR CRDBYTE     Прочитать байт
        STA CFIXDDAY    - ;
FIXPARE RTS

*-------------------------------------*
*        Запрос текущего времени      *
* ----------------------------------- *
*  Вход: зафиксированное ранее время  *
*  ----------------------------------  *
*  Выход: CLPARM1 - час               *
*          CLPARM2 - минута           *
*          CLPARM3 - секунда          *
*-------------------------------------*
INQTIM LDA CFIXDCLK - Час
       STA CLPARM1  - ;
       LDA CFIXDMIN - Минута
       STA CLPARM2  - ;
       LDA CFIXDSEC - Секунда
       STA CLPARM3  - ;
       RTS
```

```
*-------------------------------------*
*         Запрос текущей даты         *
* ----------------------------------- *
*  Вход: зафиксированная ранее дата   *
*  -------------------------------    *
*  Выход: CLPARM1 - год               *
*          CLPARM2 - месяц            *
*          CLPARM3 - число            *
*-------------------------------------*
INQDAT LDA CFIXDYER - Год
       STA CLPARM1  - ;
       LDA CFIXDMON - Месяц
       STA CLPARM2  - ;
       LDA CFIXDNUM - Число
       STA CLPARM3  - ;
       RTS

*-----------------------------------------*
*         Запрос текущего дня недели      *
* --------------------------------------- *
*  Вход: зафиксированный ранее день недели *
*  ------------------------------------   *
*  Выход: CLPARM1 - день недели           *
*-----------------------------------------*
INQDAY LDA CFIXDDAY
       STA CLPARM1
       RTS

*---------------------------------*
*    Установка нового времени     *
* ------------------------------- *
*  Вход: CLPARM1 - час            *
*         CLPARM2 - минута        *
*         CLPARM3 - секунда       *
*  ----------------------------   *
*  Выход: установлено             *
*---------------------------------*
SETTIM  LDA CSLOT
        BEQ SETTIME     = Часов нет
        JSR CSTOP       Остановить часы
        LDA CLPARM1     - Час
        STA CFIXDCLK
        LDY #4
        JSR CWRBYTE     Записать байт - ;
        LDA CLPARM2     - Минута
        STA CFIXDMIN
        LDY #2
        JSR CWRBYTE     Записать байт - ;
        LDA CLPARM3     - Секунда
        STA CFIXDSEC
        LDY #0
        JSR CWRBYTE     Записать байт - ;
        LDY #$A         - Сброс делителя отсчета секунды
        JSR CRDBYTE     Прочитать байт
        AND #:10001111
        ORA #:01110000
        LDY #$A         Регистр A
        JSR CWRBYTE     Записать байт
        LDY #$A         Регистр A
        JSR CRDBYTE     Прочитать байт
        AND #:10001111
```

```
         ORA #:00100000
         LDY #$A          Регистр A
         JSR CWRBYTE      Записать байт
         JSR CPUSK        Запустить часы
SETTIME RTS

*---------------------------------*
*       Установка новой даты      *
* ------------------------------- *
*  Вход: CLPARM1 - год            *
*         CLPARM2 - месяц         *
*         CLPARM3 - число         *
*  ------------------------------ *
*  Выход: установлено             *
*---------------------------------*
SETDAT  LDA CSLOT
         BEQ SETDATE    = Часов нет
         JSR CSTOP      Остановить часы
         LDA CLPARM1    - Год
         STA CFIXDYER
         LDY #9
         JSR CWRBYTE    Записать байт - ;
         LDA CLPARM2    - Месяц
         STA CFIXDMON
         LDY #8
         JSR CWRBYTE    Записать байт - ;
         LDA CLPARM3    - Число
         STA CFIXDNUM
         LDY #7
         JSR CWRBYTE    Записать байт - ;
         JSR CPUSK      Запустить часы
SETDATE RTS

*---------------------------------*
*    Установка нового дня недели  *
* ------------------------------- *
*  Вход: CLPARM1 - день недели    *
*  ------------------------------ *
*  Выход: установлено             *
*---------------------------------*
SETDAY  LDA CSLOT
         BEQ SETDAYE    = Часов нет
         JSR CSTOP      Остановить часы
         LDA CLPARM1    - День недели
         STA CFIXDDAY
         LDY #6
         JSR CWRBYTE    Записать байт - ;
         JSR CPUSK      Запустить часы
SETDAYE RTS

*-------------------------*
*  Остановить ход часов   *
*-------------------------*
CSTOP LDY #$A
       JSR CRDBYTE
       AND #:10000000
       BNE CSTOP        = Ожидание готовности
       LDY #$B
       JSR CRDBYTE      Прочитать байт
       ORA #:10000000
       JMP CWRBYTE      Записать байт
```

```
*------------------------*
*  Запустить ход часов  *
*------------------------*
CPUSK LDY #$B
      JSR CRDBYTE     Прочитать байт
      AND #:01111111
      JMP CWRBYTE     Записать байт

*----------------------------*
*  Чтение байта по адресу Y  *
*----------------------------*
CRDBYTE LDX CSLOT   Текущий разъем
        TYA
        STA $C086,X Порт адреса
        LDA $C087,X Порт данных
        RTS

*----------------------------*
*  Запись данных по адресу Y  *
*----------------------------*
CWRBYTE PHA
        LDX CSLOT   Текущий разъем
        TYA
        STA $C086,X Порт адреса
        PLA
        STA $C087,X Порт данных
        RTS

*-----------------------------------*
* Текущая информация о времени/дате *
*-----------------------------------*
CFIXDCLK DFB 23 Час
CFIXDMIN DFB 30 Минута
CFIXDSEC DFB 00 Секунда
CFIXDYER DFB 69 Год
CFIXDMON DFB 07 Месяц
CFIXDNUM DFB 30 Число
CFIXDDAY DFB 3  День недели
CSLOT    DFB 0  Разъем контроллера *$10
CWORK0   DFB 0  Рабочие:
CWORK1   DFB 0

*=========================================================*
```

4. BASIC процедуры – примеры управления часами

На диске, входящем в поставку, имеется программа "PrimerClk", написанная на BASICe, предназначенная для программистов с недостаточным знанием программирования на языке ассемблера, хотя для программирования часов некоторый уровень знаний о работе процессора необходим.

Так как при работе с контроллером часов имеются времязависимые участки, работа с контроллером не может быть реализованной полностью на BASICe. Поэтому центром программы является набор подпрограмм, написанных на языке ассемблера и скомпиллированных в объектный код. Исходный текст подпрограмм приведен в конце главы.

Для работы всех подпрограмм требуется предварительно найти разъем модуля (помноженный на $10) и разместить в ячейке CSLOT. Присваивание значений именам ячеек и подпрограмм производится в подпрограмме "Инициализация", программы "PrimerClk".

Двоичная подпрограмма "FIXPAR" фиксирует текущее время и дату в ячейках:

CFIXDCLK – час

CFIXDMIN – минута

CFIXDSEC – секунда

CFIXDYER – год

CFIXDMON – месяц

CFIXDNUM – число

CFIXDDAY – день недели

которые можно считывать с помощью функции PEEK и обрабатывать по собственному усмотрению, например, выводить на экран.

Двоичная подпрограмма "CSTOP" останавливает ход часов для того, чтобы Вы могли произвести корректировку текущей информации. Для обратного запуска часов достаточно средств BASICa. Как это делается, будет рассказано ниже.

Две последние двоичные подпрограммы "CRDBYTE" и "CWRBYTE" предназначены для чтения/записи (соответственно) управляющей информации.

Для их работы используются две ячейки – "RWADRESS" для размещения адреса ячейки часов и "RWBYTE" для считанного или размещения записываемого байта для подпрограмм "CRDBYTE" и "CWRBYTE" соответственно.

Рассмотрим подробно работу программы "PrimerClk". Обратите внимание, что основной цикл программы состоит из набора вызовов подпрограмм (строки 112–119).

1. Подпрограмма "Вывод шапки" комментариев не требует и не относится к сути рассматриваемого вопроса.

2. Подпрограмма "Инициализация" (строки 274–280) состоит из присваиваний значений адресным константам, требующимся для обращения к ячейкам и двоичным подпрограммам. Здесь Вы можете увидеть соответствие подпрограмм и переменных в исходном тексте и их адресов (все наименовния в программе на BASICe сохранены). Для хранения адресов использованы целые переменные, поэтому все имена оканчиваются на "%".

3. Подпрограмма "Пересылка подпрограмм" (строки 257–268) содержит команды записи в соответствующие ячейки кодов скомпиллированных подпрограмм. Так как в BASICe невозможно гарантировано выделить участок памяти для размещения специальной программы, мы использовали хвостовую часть входного буфера строки $270–$2FF. Это накладывает определенные ограничения на использование подпрограмм, в частности, нужно повторно вызывать подпрограмму "Пересылка подпрограмм" после ввода чего-либо оператором INPUT. Как, например, это сделано в строке 137 подпрограммы "Корректировка текущего времени" (строки 133–151).

4. Подпрограмма "Поиск часов" (строки 216–252) производит поиск контроллера часов по алгоритму, используемому и в драйвере на языке ассемблера. Сначала в переменную ANSWER% заносится 1 (True–Истина) и сохраняется содержимое ячеек $E и $F в переменных WORK0% и WORK1% соответственно. Затем в эти ячейки заносятся значения 1 и 2 и проверяется, записалось ли. По результату, если не записалось, в переменную ANSWER% заносится значение 0 (False–Ложь), в те же ячейки ($E и $F) заносятся значения 3 и 4 и также проверяются. В конце значения

ячеек $E и $F восстанавливаются и проверяется переменная ANSWER%. Если в ней истина, то разъем найден и записывается в ячейку "CSLOT" оператором POKE (строка 220). Если нет, то осуществляется переход к следующему разъему.

5. Подпрограмма "Тест на наличие часов" (строки 285-286) проверяет содержимое переменной SLOT%, в которую записывается 0, если часы не будут найдены. Эта подпрограмма либо завершает свою работу по оператору RETURN, либо выдает сообщение, что плата не найдена, и заканчивает работу по оператору END.

6. Подпрограмма "Вывод меню", также как и первая, комментариев не требует и не относится к сути рассматриваемого вопроса.

7. Подрограмма "Опрос клавиатуры" (строки 179-184) ожидает нажатия клавиши на клавиатуре и постоянно выводит на экран текущее время и дату с помощью подпрограммы "Вывод текущего времени и даты" (строки 200-210). В строке 200 она вызывает двоичную подпрограмму "FIXPAR", после чего из соответствующих ячеек, список которых был приведен выше, достает значения и выводит их.

8. Подпрограмма "Обработка клавиши" (строки 125-174) осуществляет переход на подпрограммы коррекции времени и даты, которые практически идентичны и различаются только адресами ячеек часов, хранящих время и дату.

Сначала оператором INPUT запрашиваются новые значения времени или даты, затем вызывается подпрограмма "Пересылка подпрограмм", чтобы восстановить возможно запорченный код двоичных подпрограмм. Далее следует вызов подпрограммы "CSTOP", останавливающей ход часов, как это положено делать перед коррекцией. После чего с помощью двоичной подпрограммы "CWRBYTE" новые значения размещаются в ячейках платы часов. Последний фрагмент "Запуск хода часов" требует пояснения. Логически нужно сбросить в 0 старший бит ячейки часов $0B, но т.к., подобных операций BASIC не поддерживает, это делается через вычитание константы $80 из считанного из ячейки $0B в переменную WORK0% значения. Предварительно проверяется, не

был ли бит уже сброшен, и если да, то чтобы не исказить значение переменной, к ней предварительно добавляется $80. После чего измененное значение переменной записывается в ячейку $0B двоичной подпрограммой "CWRBYTE"

По мнению автора, приведенной информации вполне достаточно для реализации всех потребных функций, кроме использующих прерывания. Последние в среде BASIC будут работать непредсказуемым образом и могут приводить к зависаниям программы.

```
*********************************************************
*                                                       *
*         Подпрограммы обслуживания часов из BASIC      *
*                                                       *
*  ---------------------------------------------------  *
*   (C) 1993, Ниппель, by ALV Software, Автор: А.Голов  *
*                                                       *
*********************************************************
 ORG $270
*=======================================================*
*              Фиксация текущего времени/даты           *
*=======================================================*
FIXPAR  LDA CSLOT
        BEQ FIXPARE      = Часов нет
FIXPAR0 LDY #$A          - Прочитать бит готовности
        JSR RDBYTE
        AND #:10000000 'UIP'
        BNE FIXPAR0      = Информация не готова
        LDY #4           - Часы
        JSR RDBYTE       Прочитать байт
        STA CFIXDCLK     - ;
        LDY #2           - Минуты
        JSR RDBYTE       Прочитать байт
        STA CFIXDMIN     - ;
        LDY #0           - Секунды
        JSR RDBYTE       Прочитать байт
        STA CFIXDSEC     - ;
        LDY #9           - Год
        JSR RDBYTE       Прочитать байт
        STA CFIXDYER     - ;
        LDY #8           - Месяц
        JSR RDBYTE       Прочитать байт
        STA CFIXDMON     - ;
        LDY #7           - Число
        JSR RDBYTE       Прочитать байт
        STA CFIXDNUM     - ;
        LDY #6           - День недели
        JSR RDBYTE       Прочитать байт
        STA CFIXDDAY     - ;
FIXPARE RTS

*=========================*
*  Остановить ход часов   *
*=========================*
CSTOP LDY #$A
      JSR RDBYTE
      AND #:10000000
      BNE CSTOP       = Ожидание готовности
      LDY #$B
      JSR RDBYTE      Прочитать байт
      ORA #:10000000
      JMP WRBYTE      Записать байт

*====================================*
*  Чтение данных по адресу RWADRESS  *
*====================================*
CRDBYTE LDY RWADRESS
        JSR RDBYTE    Прочитать байт
        STA RWBYTE    Прочитанный байт
        RTS
```

```
*====================================*
*  Запись данных по адресы RWADRESS  *
*====================================*
CWRBYTE LDY RWADRESS Адрес байта
        LDA RWBYTE   Записываемый байт
        JMP WRBYTE   Записать байт

*----------------------------*
*  Чтение байта по адресу Y  *
*----------------------------*
RDBYTE LDX CSLOT   Текущий разъем
       TYA
       STA $C086,X Порт адреса
       LDA $C087,X Порт данных
       RTS

*----------------------------*
*  Запись данных по адресу Y  *
*----------------------------*
WRBYTE PHA
       LDX CSLOT   Текущий разъем
       TYA
       STA $C086,X Порт адреса
       PLA
       STA $C087,X Порт данных
       RTS

       DS  1

*----------------------------------*
* Текущая информация о времени/дате *
*----------------------------------*
RWADRESS DFB $3F Адрес для чтения/записи
RWBYTE   DFB 0   Прочитанный/записываемый байт
CFIXDCLK DFB 23  Час
CFIXDMIN DFB 30  Минута
CFIXDSEC DFB 00  Секунда
CFIXDYER DFB 69  Год
CFIXDMON DFB 07  Месяц
CFIXDNUM DFB 30  Число
CFIXDDAY DFB 3   День недели
CSLOT    DFB 0   Разъем контроллера *$10

*=========================================================*
```