

## ПРЕДВАРИТЕЛЬНАЯ ИНФОРМАЦИЯ

ВСЯ ИНФОРМАЦИЯ В ЭТОМ РУКОВОДСТВЕ ПРЕДВАРИТЕЛЬНАЯ И МОЖЕТ БЫТЬ ИЗМЕНЕНА.

# TeleServ – Fan controller and LCD/4xButton/RS232 Terminal

(Версия 1.3 Март 2006)

2004 ОП "Индустриальный Компьютер"®

## Введение

Этот продукт следует подсоединять и настраивать только специализированному персоналу и только после изучения данного руководства!

Это руководство соответствует реализации программы версии 1.3. Номер версии реализации программы указан на стикере, который прикреплен к микроконтроллеру, находящемуся на плате устройства, и индицируется на экране ЖКИ в конце информационной бегущей строки, когда устройство находится в ждущем режиме. Этот номер также может быть получен по последовательному каналу в ответ на соответствующую поданую команду.

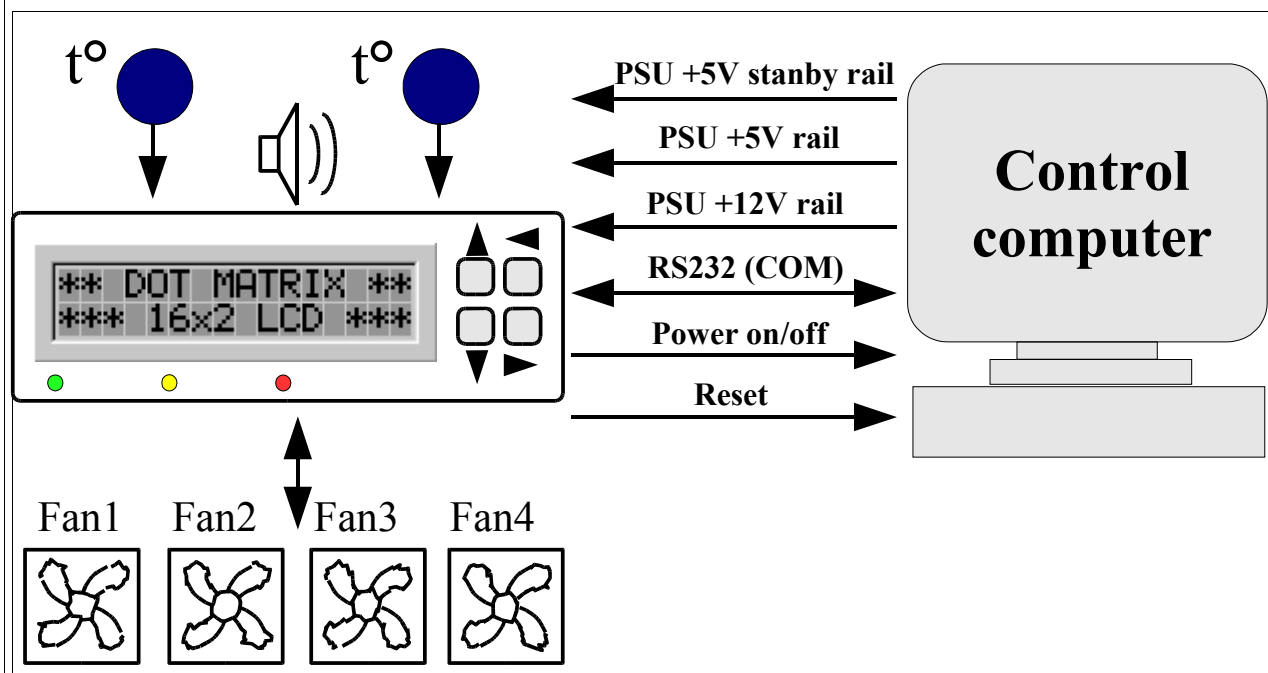
Все зарегистрированные торговые марки являются собственностью соответствующих владельцев.

## Описание

**TeleServ – Fan controller and LCD/4xButton/RS232 Terminal** (далее - контроллер) – это программно-аппаратное устройство, реализующее RS232

терминал, позволяющий связаться через COM-порт с компьютером. Для организации диалога в программах пользователя в контроллере имеется 4 кнопки и 16x2 LCD (16-знаков x 2-строки ЖКИ). Контроллер также реализует функции WatchDog (сторожевого) таймера, мониторинга температуры, напряжений, частот вращения вентиляторов. Контроллер плавно управляет частотами вращения вентиляторов в зависимости от температуры по используемому таблицы алгоритму, выступая автономным контроллером системы охлаждения. Контроллер обеспечивает раннюю диагностику неисправностей вентиляторов, продлевает срок их службы, снижает потребляемую вентиляторами мощность и производимый шум, повышает надежность работы электронного оборудования за счет полного мониторинга системы охлаждения и сигнализации ее неудовлетворительной работы. Использование контроллера в системе избыточного охлаждения позволяет создать отказоустойчивое оборудование длительного срока эксплуатации.

Кроме того, контроллер эмулирует кнопки включения/выключения и сброса компьютера, защищенные PIN-кодом от несанкционированного управления компьютером.



Возможности контроллера включают:

- Фиксированные параметры последовательной связи 9600, 8 бит, без контроля четности, 1 стоп-бит.
- RTS/CTS аппаратное управление потоком последовательной связи.
- 16x2 LCD (16-знаков x 2-строки ЖКИ).
- 4 кнопки.
- Эмулятор кнопок включения и сброса компьютера, защищенные PIN-кодом.
- Управляемый счетчик, который действует как WatchDog(сторожевой) таймер для подсоединенного компьютера.
- Контроллер-регулятор частоты вращения 4-х вентиляторов постоянного тока 12В. Контроль и плавное управление в функции температуры по алгоритму использующему таблицы с переменными параметрами. Автономный мониторинг системы охлаждения с индикацией результатов на экран ЖКИ и в последовательную линию.
- 3 канала измерения напряжения (шины +5В, +12В и варьируемого напряжения на вентиляторах)
- 2 канала измерения температуры (используются вынесенные термисторы)
- Звуковая и световая сигнализация (зуммер и "красный/аварийный" светодиод)
- EEPROM для хранения данных, таблиц и кодов конфигурации

Отображение информации на экране ЖКИ контроллера осуществляется в двух основных режимах – в режиме “монитор” и в режиме “терминал”. И в первом и во втором случае контроллер осуществляет непрерывный контроль и управление вентиляторами. Но, отображая данные в режиме “монитор”, контроллер выводит данные на экран ЖКИ собственной программой. А в режиме “терминал” на экран ЖКИ поступают данные, направленные из последовательной линии (компьютера). Контроллер сохраняет коды символов для отображения (8 битные) в памяти данных (RAM), преобразует их в символьный растр пиксельной матрицы 5x7 и отображает символы на экране ЖКИ. ЖКИ включает ROM знакогенератор (неизменяемый) который производит 160 различных символьных растров пиксельных матриц 5x7 (см. таблицу справа). Устройство также включает RAM генератор символов (64 байт) в котором пользователь может определить до восьми дополнительных символьных растров пиксельных матриц 5x8 так, как это требует приложение. Контроллер использует 4 кнопки: - “Вверх”, “Вниз”, “Влево”, “Вправо” для ввода команд пользователя. В режиме “монитор” контроллер сканирует кнопки, но коды нажатых кнопок в последовательную линию не передает. В режиме “терминал” контроллер сканирует кнопки и передает

коды нажатых кнопок в последовательную линию.

При обнаружении одновременно двух нажатых кнопок: {"Вниз" + "Влево"}, {"Вниз" + "Вправо"}, {"Вверх" + "Вправо"}, {"Вверх" + "Влево"} - виртуальных горячих кнопок, контроллер обрабатывает соответствующие меню защищенные PIN-кодом. “Виртуальные горячие кнопки” обрабатываются обоих режимах - “терминал” и “монитор”.

Поток входных данных последовательной линии синтаксически анализируется, декодируется и либо выполняется как последовательность команд, либо отображается на экране ЖКИ (только в режиме “терминал”).

В режиме “монитор”, терминал возвращает эхо на поступающие по последовательной линии байты. Таким образом подключившись простым TTY-терминалом по СОМ-порту к “монитору” можно осуществить диалог с программой “монитора”, просмотреть на экране компьютера значения напряжений, частот вращения вентиляторов, провести диагностику вентиляторов и пр. Для этого не нужно устанавливать никаких программ на компьютере т.к. простой TTY-терминал обычно входит в комплект предустановленных программ всех операционных систем.

Char. code	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
xxxx0000	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
xxxx0001	!	"	#	\$	%	&	'	(	)	*	+	,	-	.	/	:
xxxx0010	"	2	B	R	b	r	Г	И	У	Х	Ф	В	В	В	В	В
xxxx0011	#	3	C	S	c	s	Г	И	У	Х	Ф	В	В	В	В	В
xxxx0100	\$	4	D	T	d	t	Г	И	У	Х	Ф	В	В	В	В	В
xxxx0101	%	5	E	U	e	u	Г	И	У	Х	Ф	В	В	В	В	В
xxxx0110	&	6	F	V	f	v	Г	И	У	Х	Ф	В	В	В	В	В
xxxx0111	'	7	G	W	g	w	Г	И	У	Х	Ф	В	В	В	В	В
xxxx1000	(	8	H	X	h	x	Г	И	У	Х	Ф	В	В	В	В	В
xxxx1001	)	9	I	Y	i	y	Г	И	У	Х	Ф	В	В	В	В	В
xxxx1010	*	:	J	Z	j	z	Г	И	У	Х	Ф	В	В	В	В	В
xxxx1011	+	;	K	[	k	[	Г	И	У	Х	Ф	В	В	В	В	В
xxxx1100	,	<	L	¥	l	¥	Г	И	У	Х	Ф	В	В	В	В	В
xxxx1101	-	=	M	]	m	]	Г	И	У	Х	Ф	В	В	В	В	В
xxxx1110	.	>	N	^	n	^	Г	И	У	Х	Ф	В	В	В	В	В
xxxx1111	/	?	O	_	o	_	Г	И	У	Х	Ф	В	В	В	В	В

# Подключение

Схема подключения (см. ниже) показывает распиновку и структуру коннекторов контроллера. **ВНИМАНИЕ!** Контроллер питается от шины +5Vsb (+5В дежурного режима). Преобразователь питания вентиляторов питается от шины +12V. Шина +5V служит только для измерения.

J1 (4-х пиновый коннектор с поляризатором) должен быть подключен к блоку питания компьютера. Следует обратить внимание, что стандартный ATX блок питания, подключенный к материнской плате компьютера, не содержит свободного вывода шины +5Vsb. Поэтому необходимо “врезаться” в эту шину и сделать отвод с коннектором на конце, который подсоединяется к J1 для питания контроллера. Контроллер потребляет 0,1А по шине +5Vsb. Подключение к шинам +12V и +5V сложностей не имеет.

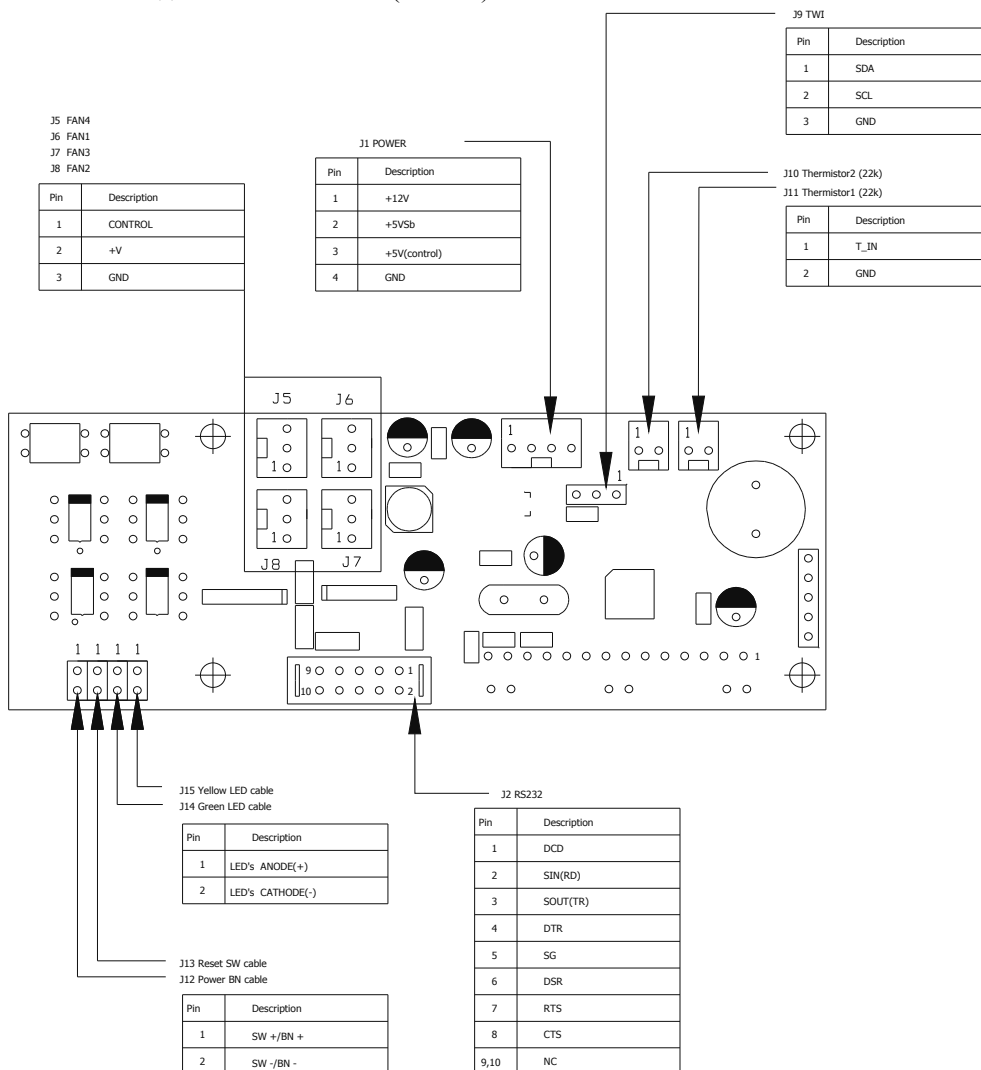
J2 – это 10-ти пиновая колодка с поляризатором для подключения по последовательной линии (RS-232) к

COM-порту компьютера. Контроллер использует RTS/CTS аппаратное управление потоком последовательной линии. Однако, сигнал линии RTS, контроллер игнорирует, считая, что компьютер всегда готов принять данные. Поэтому реально для связи с компьютером требуется четырехпроводная линия связи - SG(сигнальная “земля”), SIN(RD) (последовательный ввод/прием данных), SOUT(TR) (последовательный вывод/передача данных), CTS (разрешение контроллером передать данные компьютеру или готовность принять от компьютера).

J3 и J4 – это служебные коннекторы. Подключение к этим коннекторам категорически **запрещается!**

J5, J6, J7 и J8 – коннекторы, к которым подключаются 3-х пиновые вентиляторы (третий вывод – сигнал тахо-генератора) постоянного тока 12В.

Коннектор J9 зарезервирован для следующих версий и для версии 1.3 он используется только для установки значений EEPROM, действующих по умолчанию.



J10 и J11 – коннекторы подключения к выносным (0,4м) датчикам температуры (термисторам).

J12 и J13 следует подключить к “Front Panel” коннектору материнской платы компьютера, к которому обычно подключаются кнопки “Питание” и “Сброс”, расположенные на передней панели компьютера. Т.е. контроллер выполняет функции этих кнопок с той лишь разницей, что при подключении требуется соблюсти полярность. Контроллер может быть подключен вместо или параллельно кнопкам “Питание” и “Сброс”.

J14 и J15 следует подключить к “Front Panel” коннектору материнской платы компьютера, к которому обычно подключаются светодиоды “Питание” и “HDD activity” расположенные на передней панели компьютера. При этом роль этих светодиодов будут выполнять зеленый и желтый светодиоды контроллера. Впрочем, эти коннекторы можно и не подключать, используя имеющуюся в корпусе компьютера аналогичную индикацию.

## Режимы работы

Контроллер может работать в трех основных режимах:

- *Дежурный режим.* Питание +5В и +12В не подано; первой строкой экрана ЖКИ пробегает бегущая строка; во второй строке постоянно присутствует надпись: “STANDBY MODE”.
- *Режим “монитор”.* На экран ЖКИ циклически выводятся данные мониторинга питающих напряжений и системы охлаждения. Отображение данных осуществляется программой контроллера автономно без участия компьютера и последовательной линии.
- *Режим “терминал”.* На экран ЖКИ поступают данные, направленные из последовательной линии (компьютера).

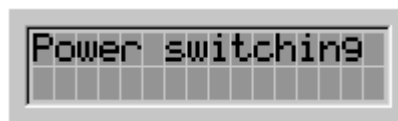
### “Дежурный” режим

В дежурном режиме контроллер фактически ожидает одновременного нажатия двух кнопок {“Вниз” + “Влево”} с последующим входом в меню “виртуальной горячей кнопки” управления питанием и появления напряжения +5В, означающего выход из дежурного режима. Ни управления и контроля

вентиляторами, ни вывода на экран ЖКИ какой-либо информации, кроме презентующей бегущей строки контроллер не осуществляет. Для того чтобы включить компьютер и выйти из “дежурного” режима нужно одновременно нажать две кнопки {“Вниз” + “Влево”} и войти в меню виртуальной горячей кнопки управления питанием. На экране ЖКИ появится приглашение ввести PIN-код:



Нажатием кнопок “Вверх” и “Вниз” поочередно вводятся все четыре позиции (цифры) кода. Переход к следующей позиции и возврат к предыдущей осуществляется кнопками “Вправо” и “Влево”. Нажатие кнопки “Вправо”, когда курсор находится в пятой позиции (знак “?”), завершает ввод кода. Контроллер анализирует введенный код и в случае совпадения замыкает контакты SW+/BN и SW-/BN коннектора J12 между собой на время не менее 0,1сек. Удержание кнопки “Вправо” продлевает это время до момента отпускания. На время замыкания контактов коннектора J12 на экране ЖКИ отображается следующее:



Если при “нажатии” виртуальной горячей кнопки управления питанием напряжение на шине +5В не появилось, то контроллер продолжает работать в “ждущем” режиме.

### Режим “монитор”

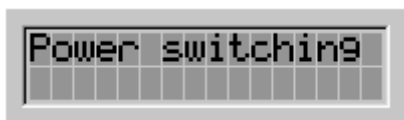
Контроллер переключается в этот режим при подаче напряжения на шину +5В. Контроллер входит в режим “монитор” независимо от того, подано ли напряжение на шину +5В при “нажатии” виртуальной горячей кнопки включения питания или без такого “нажатия”. Контроллер находится в режиме “монитор” только при поданном на шину +5В напряжении не менее 4В. Если напряжение на шине +5В падает ниже 4В, контроллер переключается в “дежурный” режим.

Для того чтобы выключить компьютер и перейти из режима “монитор” (также как и из режима “терминал”) в “дежурный” режим, нужно одновременно нажать две кнопки {“Вниз” + “Влево”} и войти в меню виртуальной горячей

кнопки управления питанием. На экране ЖКИ появится приглашение ввести PIN-код:



Нажатием кнопок “Вверх” и “Вниз” поочередно вводятся все четыре позиции (цифры) кода. Переход к следующей позиции и возврат к предыдущей осуществляется кнопками “Вправо” и “Влево”. Нажатие кнопки “Вправо”, когда курсор находится в пятой позиции (знак “?”), завершает ввод кода. Контроллер анализирует введенный код и в случае совпадения замыкает контакты SW+/BN и SW-/BN коннектора J12 между собой на время не менее 0,1сек. Удержание кнопки “Вправо” продлевает это время до момента отпускания. На время замыкания контактов коннектора J12 на экране ЖКИ отображается следующее:



Длительно удерживая кнопку “Вправо” можно добиться выключения компьютера в случае “залипания” логики управления питанием материнской платы.

Процедура выключения компьютера в режиме “монитор” с помощью виртуальной горячей кнопки управления питанием аналогична процедуре включения компьютера описанной в разделе “дежурный” режим. В режим “монитор” контроллер можно перевести также и подачей соответствующей команды от компьютера по последовательной линии, если перед этим контроллер находился в режиме “терминал”. В режиме “монитор” (также как и в режиме “терминал”) контроллер выполняет программу мониторинга температуры, напряжений, плавно управляет и контролирует частоты вращения вентиляторов в зависимости от температуры по алгоритму использующему таблицы, выступая автономным контроллером системы охлаждения. Но только в режиме “монитор” контроллер циклически выводит данные на экран ЖКИ, которые были получены (измерены) программой контроллера (частоты вращения вентиляторов, температуру, напряжения) независимо от команд полученных по последовательной линии, а в режиме “терминал” информация на экране ЖКИ выводится только по командам последовательной линии.

## Режим “терминал”

В этот режим контроллер можно перевести только подачей соответствующей команды от компьютера по последовательной линии, если перед этим контроллер находился в режиме “монитор”. В режиме “терминал” (также как и в режиме “монитор”) контроллер может находиться только при поданном на шину +5В напряжении не менее 4В. Если напряжение на шине +5В падает ниже 4В, контроллер переходит в “дежурный” режим. Процедура выключения компьютера в режиме “терминал” полностью совпадает с аналогичной процедурой режима “монитор” и “дежурного” режима. В режиме “терминал” (также как и в режиме “монитор”) контроллер выполняет программу WatchDog (сторожевого) таймера, мониторинга температуры, напряжений, плавно управляет и контролирует частоты вращения вентиляторов в зависимости от температуры по алгоритму использующему таблицы, выступая автономным контроллером системы охлаждения. Но в режиме “терминал” информация на экране ЖКИ выводится только по командам последовательной линии, а в режиме “монитор” контроллер циклически выводит данные на экран ЖКИ, которые были получены (измерены) программой контроллера (частоты вращения вентиляторов, температуру, напряжения) независимо от команд полученных по последовательной линии. Еще одно отличие работы контроллера в режиме “терминал” - контроллер сканирует кнопки и передает коды нажатых кнопок в последовательную линию, а в режиме “монитор” контроллер сканирует кнопки ожидая одновременного нажатия кнопок {“Вниз” + “Влево”}, {“Вниз” + “Вправо”}, {“Вверх” + “Вправо”}, {“Вверх” + “Влево”}, но коды нажатых кнопок в последовательную линию не передает.

## Управление вентиляторами

Работа контроллера по управлению вентиляторами состоит в постоянном (3 раза в секунду) измерении и анализе температуры воздушного потока и частот вращения присоединенных вентиляторов, а также в выработке напряжения, питающего вентиляторы, согласно предварительно составленной таблице. К контроллеру можно произвольно подключать до 4-х вентиляторов постоянного тока номинального напряжения 12В с тремя проводами (третий провод – сигнал генератора импульсов вращения). Количество присоединенных вентиляторов и их параметры контроллер может определить самостоятельно специально вызываемой процедурой. Подробнее эта процедура будет описана далее.

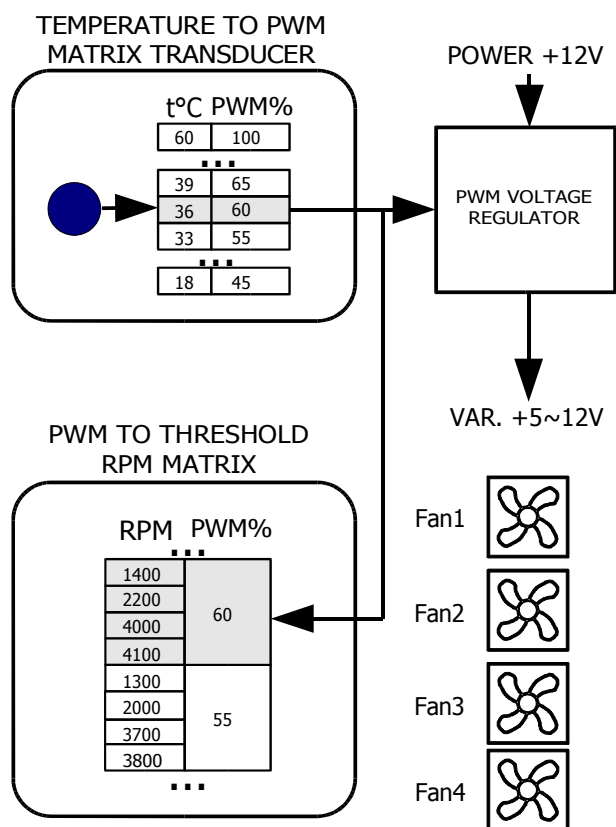
Диапазон измеряемой температуры от 18 до 60°C

разбит на интервалы по 3°C. Каждому интервалу температуры дано в соответствие значение PWM. Значение PWM для интервала температур можно изменять, тем самым формируя регулировочную характеристику по потребностям. Содержание этой таблицы, прописанное изготовителем и действующее по умолчанию приведено ниже:

T°C	<18	<21	<24	<27	<30	<33	<36	<39	<42	<45	<48	<51	<54	<57	<60
PWM%	45	45	45	45	50	55	60	65	70	75	80	85	90	95	100

бы один из вентиляторов будет иметь частоту вращения ниже заданной, то контроллер определяет это, как аварийную ситуацию, вырабатывает звуковой и световой сигнал и задает значение PWM 100%. Таким образом осуществляется ранняя диагностика состояния работы вентиляторов, когда способность к вращению (и охлаждению) вентилятора еще не

Из приведенного ниже рисунка поясняющего управление вентиляторами видно как измеренное значение температуры (36°C) адресует строку таблицы с величиной PWM равным 60%.



утрачена, а сигнализация об неудовлетворительной работе уже имеется. При избыточном количестве и производительности подключенных вентиляторов, ранняя диагностика позволяет существенно повысить надежность работы всей системы охлаждения, а снижение оборотов вентиляторов на низких температурах – продлить срок службы и снизить производимый шум самого ненадежного и самого шумного элемента современной электроники – вентилятора. Обе таблицы находятся в энергонезависимой памяти и имеется несколько способов их корректировки при потребности. Можно индивидуально настраивать контроллер (таблицы) как на зависимость PWM от температуры формируя собственную характеристику управления, так и на реакцию присоединенных вентиляторов в конкретных условиях применения.

Величина PWM задает уровень напряжения питания вентиляторов и в конечном итоге частоту их вращения.

Измеренные значения частот вращения вентиляторов сравниваются с табличными (для каждого вентилятора отдельно). Нормально работающие вентиляторы должны иметь частоты вращения не ниже табличных. В приведенном примере измеренное значение температуры (36°C) и величина PWM 60% определяют предельные значения частот вращения равные 1400, 2200, 4000 и 4100 для соответственно 1-го, 2-го, 3-го и 4-го вентиляторов. В случае, если хотя

## Процедура тестирования присоединенных вентиляторов

Эта процедура может быть вызвана, только если контроллер находится в режиме “монитор”, т.е. когда подано напряжение и +5В и +12В. В “ждущем” режиме и в режиме “терминал” вызов процедуры игнорируется контроллером. Процедура может быть инициирована как посылкой соответствующей команды от компьютера по последовательной линии, так и нажатием соответствующих кнопок на контроллере. Поведение процедуры не зависит от способа вызова. Для того чтобы вызвать процедуру нажатием кнопок, нужно одновременно нажать две кнопки {“Вниз” + ”Вправо”} и войти в меню виртуальной горячей кнопки вызова процедуры тестирования. На экране ЖКИ появится приглашение ввести PIN-код:



Нажатием кнопок “Вверх” и “Вниз” поочередно вводятся все четыре позиции (цифры) кода. Переход к

следующей позиции и возврат к предыдущей осуществляется кнопками “Вправо” и “Влево”. Нажатие кнопки “Вправо”, когда курсор находится в пятой позиции (знак “?”), завершает ввод кода. Контроллер анализирует введенный код и в случае совпадения начинает выполнение процедуры тестирования присоединенных вентиляторов, подавая каждые 3 сек. тестовое напряжение, начиная с максимального и заканчивая минимальным уровнем, изменяя при этом значение PWM со 100% до 40% с шагом 5%. Т.к. процедура начинается с значения PWM 100%, то на экране PWM появляется изображение:



Затем все 12 значений PWM от 95% до 40%. Результатом работы процедуры является обновленная таблица граничных частот вращения для реально подключенных к контроллеру вентиляторов. Процедура длится около 1 минуты. В течении всего этого времени контроллер следит за питающими напряжениями и поведением вентиляторов. Контроллер определяет количество подключенных вентиляторов и их номинальные частоты вращения для всех значений PWM и соответствующих напряжений питающих вентиляторы. Если во время проведения тестирования произойдет аварийная (ошибочная) ситуация, или будет обнаружено некорректное поведение вентиляторов, то в таблицу граничных частот вращения будут записаны значения действующие по умолчанию (прописанные изготовителем при производстве). Содержимое таблицы значений граничных частот вращения вентиляторов действующих по умолчанию приведено ниже:

PWM% \ RPM	100	95	90	85	80	75	70	65	60	55	50	45	40
<b>Fan1</b>	1900	1800	1700	1600	1500	1400	1300	1200	1100	1000	900	800	700
<b>Fan2</b>	1900	1800	1700	1600	1500	1400	1300	1200	1100	1000	900	800	700
<b>Fan3</b>	1900	1800	1700	1600	1500	1400	1300	1200	1100	1000	900	800	700
<b>Fan4</b>	1900	1800	1700	1600	1500	1400	1300	1200	1100	1000	900	800	700

фиксированными параметрами: 9600бит/с, 8-бит, без контроля четности, 1 стоп-бит. Инициатором обмена как правило выступает компьютер. Компьютер может посылать контроллеру команды и данные для отображения на экране ЖКИ. На большинство посланных с компьютера команд контроллер отвечает блоком данных. Контроллер реализует подобие ANSI/VT100 терминала подмножеством команд/escape-последовательностей с расширением для работы с WatchDog таймером, каналами измерения температуры и напряжения и др. Символы принятые по последовательной линии попадают во входной буфер объемом 23 байта. Строка с 24 и более байтами будет обрезана. Контроллер начинает синтаксический разбор входной строки после получения символа возврата каретки CR(0x0d), перевода строки LF(0x0a) или приема полного числа символов (23)(заполнения входного буфера). Параметры, такие как позиции курсора, передаются символами ASCII, т.е. 12 представляет символ “1” и символ “2”, а не один байт со значением 12. Символы чувствительны к регистру, поэтому Esc[ @W и Esc [ @w – это разные команды. В режиме “терминал” контроллер посылает коды нажатых кнопок без предварительного запроса со стороны компьютера, поэтому программа компьютера должна анализировать эти коды и поддерживать диалог. Перевод контроллера в режим процедуры тестирования вентиляторов также иницирует обмен по линии последовательной связи, но при этом инициатором такого обмена может выступать и компьютер, пославший соответствующую команду, и контроллер, если оператор вызвал процедуру нажатием соответствующих кнопок. Команды, коды кнопок и примеры программ использующие обмен по последовательной линии связи описаны далее в отдельных разделах.

## Линия последовательной связи

Контроллер, подключенный к компьютеру линией последовательной связи, действует как терминальное устройство, обмениваясь информацией по линии с

## Виртуальная кнопка “сброс” и WatchDog таймер

Одновременное нажатие кнопок {“Вверх” + “Вправо”} в режимах “монитор” и “терминал” позволяет войти в меню виртуальной горячей кнопки

“сброс”. На экране ЖКИ появится приглашение ввести PIN-код:



Нажатием кнопок “Вверх” и “Вниз” поочередно вводятся все четыре позиции (цифры) кода. Переход к следующей позиции и возврат к предыдущей осуществляется кнопками “Вправо” и “Влево”. Нажатие кнопки “Вправо”, когда курсор находится в пятой позиции (знак “?”), завершает ввод кода. Контроллер анализирует введенный код и в случае совпадения замыкает контакты SW+/BN и SW-/BN коннектора J13 между собой на время не менее 0,1сек. Удержание кнопки “Вправо” продлевает это время до момента отпускания. На время замыкания контактов коннектора J12 на экране ЖКИ отображается следующее:



Удерживая кнопку “Вправо” можно добиться длительного замыкания контактов и длительного состояния “сброс” для надежного сброса компьютера. Еще один способ замыкания контактов SW+/BN и SW-/BN коннектора J13 между собой для выработки сигнала “сброс” – это использование WatchDog таймера. При включении контроллера WatchDog таймер выключен. Для того чтобы его включить, требуется по последовательной линии послать команду “Esc[WW” с требуемым значением счетчика таймаута W – от 0 до 255 - число тиков (каждый тик равен 0,3 сек). Программа-драйвер должна периодически устанавливать таймаут для избежания “гонга” и последующего замыкания контактов “сброса”. Подробнее работа с WatchDog таймером описана в следующих разделах.

## Смена PIN-кода

При изготовлении в контроллер записан PIN-код 7268. PIN-код хранится в EEPROM, поэтому может быть легко изменен. Одновременное нажатие кнопок {“Вверх” + “Влево”} во всех режимах работы контроллера позволяет войти в меню смены PIN-кода. На экране ЖКИ появится приглашение ввести текущий PIN-код:



Затем нужно ввести новый PIN-код:



И еще раз повторить новый PIN-код:



В случае если повторный ввод PIN-кода не соответствует первоначальному, новый PIN-код не активируется и продолжает действовать старый (текущий) PIN-код. Хорошо запомните новый PIN-код. В случае потери PIN-кода, можно вернуть PIN-код, действующий по умолчанию (7268), установив все таблицы значений, действующих по умолчанию (восстановление EEPROM).

## Установка значений EEPROM действующих по умолчанию

Выполнить эту процедуру можно только в случае замыкания контактов 1 и 2 (установки перемычки) на коннекторе J9. ВНИМАНИЕ! Требуется полное снятие всех напряжений с контроллера до установки перемычки! Для этого нужно выдернуть вилку с блока питания для прекращения подачи высокого напряжения снятия напряжения дежурного режима 5В (+5Vsb). После этого, выдержав паузу 5-10 секунд, нужно установить перемычку и подать напряжение дежурного режима 5В (вставить вилку высокого напряжения в блок питания). На экране ЖКИ появится надпись:



Появление этой надписи означает, что таблица значений PWM для интервалов температур и таблица значений граничных частот вращения вентиляторов прописаны значениями действующими по умолчанию. Кроме того, в контроллер записан PIN-код 7268.

## Описание команд принимаемых контроллером по последовательной линии в режиме "Терминал"

Замечание: символы в командах чувствительны к регистру и состоят из ASCII кодов. Многие команды требуют числовых или символьных аргументов. В данном документе эти аргументы обозначаются подчеркиванием. Так, выражение "Esc[F $\underline{E}$ @ $\underline{P}$ = $\underline{R}$ " означает, что  $\underline{E}$ ,  $\underline{P}$  и  $\underline{R}$  нужно заменить десятичными числами (например, "Esc[F1@40=1000").

### Установить Видимость Курсора – Esc [ C C

Использование этой команды позволяет управлять видимостью курсора. Посылка последовательности "Esc[C0" делает курсор невидимым. Команда с любым значением  $\underline{C}$  не равным символу 0 (ноль) включает мигающий курсор.

### Получить Номер Версии Реализации Программы – Esc [ c

Строка "V1.3" - номер версии реализации программы посылается контроллером в ответ на команду "Esc[c".

### Очистить Экран – Esc [ e

Эта команда очищает экран ЖКИ и устанавливает курсор в начальную позицию (0,0).

### Установить Граничное Значение Частоты Вращения (RPM) для Выбранного Вентилятора и Значения PWM – Esc [ F $\underline{F}$ @ $\underline{P}$ = $\underline{R}$

Новое граничное значение частоты вращения  $\underline{R}$  (в оборотах в минуту – 0~25000) для вентилятора  $\underline{F}$  (допустимые значения от 1 до 4) записывается в таблицу EEPROM для выбранного значения широтно-импульсной модуляции  $\underline{P}$  (допустимые значения от 40% до 100%). Контроллер проверяет параметры команды на допустимые значения. В случае несоответствия – команда игнорируется. Контроллер также, в случае необходимости, округляет  $\underline{R}$  до ближайшего меньшего круглого значения с шагом 100об/мин., а  $\underline{P}$  с шагом 5%. Например, посылка команды "Esc[F1@42=1055" установит новое граничное значение 1000об./мин. (округление числа 1055 до 1000) частоты вращения вентилятора 1 для значения широтно-импульсной модуляции 40% (округление числа 42 до 40).

### Получить Граничное Значение Частоты Вращения (RPM) для Выбранного Вентилятора и Значения PWM – Esc [ f $\underline{F}$ @ $\underline{P}$

Граничное значение частоты вращения (в оборотах в минуту) для вентилятора  $\underline{F}$  (допустимые значения от 1 до 4) считывается из таблицы EEPROM для выбранного значения широтно-импульсной модуляции  $\underline{P}$  (допустимые значения от 40% до 100%). Контроллер проверяет параметры команды на допустимые значения. В случае несоответствия – команда игнорируется. Контроллер также, в случае необходимости, округляет  $\underline{P}$  до ближайшего меньшего круглого значения с шагом 5%. Например, в ответ на команду "Esc[f2@99" контроллер передаст в последовательную линию граничное табличное значение частоты вращения вентилятора 2 для значения широтно-импульсной модуляции 95% (округление числа 99 до 95).

### Загрузить Растр Пиксельной Матрицы Символа – Esc [ g $\underline{C}$ = {Char[8]}

Для символов с кодами ASCII от 0 до 7 можно задать изображение символа на экране ЖКИ. Растр пиксельной матрицы 5x8 определяется последовательностью 8-ми символов завершающей команду "Esc[g $\underline{C}$ =", т.е. следующей после символа '='. ASCII коды последовательности 8-ми символов интерпретируются контроллером, как коды загружаемые в RAM генератора символов (пиксельную матрицу). Загрузка пиксельной матрицы 5x8 начинается с верхней строки. Первый ASCII код последовательности 8-ми символов загружается в верхнюю строку, второй код – во вторую строку сверху и так далее, до последнего восьмого кода. Младший бит кода отображается в правой части знакоместа. Значащими для пиксельной матрицы являются только 5 младших бита кода ASCII. Остальные (старшие) биты игнорируются и могут принимать любое значение. Например, команда "Esc[g7=nQabdh?@" загрузит растр пиксельной матрицы с изображением числа 2 для символа с кодом ASCII 7. После загрузки пиксельной матрицы, контроллер устанавливает курсор экрана ЖКИ в начальную позицию (0,0).

Offset	Char	Data	Hex/Bin
====	===	=====	
		D7-----D0	
0	n	6e	x x x 0 1 1 1 0 ***
1	Q	51	x x x 1 0 0 0 1 * *
2	a	61	x x x 0 0 0 0 1 *
3	b	62	x x x 0 0 0 1 0 *
4	d	64	x x x 0 0 1 0 0 *
5	h	68	x x x 0 1 0 0 0 *
6	?	3f	x x x 1 1 1 1 1 *****
7	@	40	x x x 0 0 0 0 0 (cursor line)

Как видно из приведенного выше рисунка, каждый байт данных строки  $\underline{C}$  представляет одну строку

пикселей для символа, причем D0 является крайним правым пикселем, а D4 является крайним левым (3 старших бита не отображаются). Первый байт строки для символа является верхней строкой, а 8-ой - нижней. (9-й накладывается на подчеркивание курсора). '1' в позиции пиксела - это черный пиксел на экране ЖКИ.

### Переключить Контроллер в Режим “Монитор” – Esc [ m

После получения этой команды, контроллер переходит в режим “монитор” и посылает в последовательную линию соответствующее сообщение и приглашение к диалогу - “\n\rMonitor mode\n\r>”. Символы \n и \r соответственно символы возврата каретки и перевода строки.

### Установить Значение PWM для Выбранного Интервала Температуры – Esc [ P P @ T

Новое значение PWM равно P (допустимые значения широтно-импульсной модуляции P от 40% до 100%) для интервала температуры T (допустимые значения от 18 до 60 градусв Цельсия) записывается в таблицу EEPROM. Контроллер проверяет параметры команды на допустимые значения. В случае несоответствия – команда игнорируется. Контроллер также, в случае необходимости, округляет P до ближайшего меньшего круглого значения с шагом 5%, а T с шагом 3°C. Например, посылка команды "Esc[P63@38" установит новое значение PWM равно 60% (округление числа 63 до 60) для интервала температуры 36 (округление числа 38 до 36, для ряда температур начиная с 18 до 60 с шагом 3).

### Получить Значение PWM для Выбранного Интервала Температуры – Esc [ p @ T

Значение PWM считывается из таблицы EEPROM для выбранного значения интервала температуры T (допустимые значения от 18 до 60 градусв Цельсия). Контроллер проверяет параметры команды на допустимые значения интервала температуры (от 18 до 60). В случае несоответствия – команда игнорируется. Контроллер также, в случае необходимости, округляет T до ближайшего меньшего значения с шагом 3°C для ряда температур начиная с 18 до 60. Например, в ответ на команду "Esc[p@20" контроллер передаст в последовательную линию табличное значение PWM для интервала температуры 18°C (округление числа 20 до 18 для ряда температур начиная с 18 до 60 с шагом 3).

### Получить Текущее Значение Частоты Вращения Вентилятора – Esc [ r F

Текущее значение частоты вращения (обороты в

минуту), измеренное контроллером для выбранного вентилятора F (0 или 1) посылается в последовательную линию в ответ на принятую команду.

### Отобразить Сивол с Указанным Кодом ASCII на Экране ЖКИ – Esc [ s C

В текущей позиции корсора отображается символ с кодом ASCII C, который вводится как десятичный параметр команды. Например, команда "Esc[s48" выведет на экран ЖКИ изображение символа '0' (ноль), т.к. коду ASCII 48 соответствует цифра ноль.

### Получить Текущее Значение Температуры – Esc [ t C

Текущее значение температуры, измеренное контроллером для выбранного канала C (0 или 1) посылается в последовательную линию в ответ на принятую команду.

### Получить Текущее Значение Напряжения – Esc [ v C

Текущее значение напряжения (в милливольтгах), измеренное контроллером для выбранного канала C (0 – соответственно +5V, 1 - соответственно +12V и 2 – напряжение на вентиляторах) посылается в последовательную линию в ответ на принятую команду.

### Получить Текущее Значение Счетчика Таймаута – Esc [ w

Текущее значение счетчика таймаута WatchDog таймера (в тиках, каждый тик равен 0,3 сек) посылается в последовательную линию в ответ на принятую команду.

### Установить Значение Счетчика Таймаута – Esc [ W W

В счетчик таймаута WatchDog таймера (в тиках, каждый тик равен 0,3 сек) записывается новое значение W – число секунд от 0 до 255. Например посылка команды "Esc[W25" установит таймаут 25сек.

### Установить Тукущую Позицию Курсора – Esc [ L ; C N

Первый параметр команды L – означает номер строки (нумерация начинается с 0), а второй - C – означает столбец (нумерация начинается с 0). Т.е. для установки курсора в левый верхний угол следует послать команду "Esc[0;0H".

## Перечень Кодов Кнопок Посылаемых Контроллером по Последовательной Линии в Режиме “Терминал”

### Коды Нажатия

“Вверх”	– U
“Вниз”	– D
“Влево”	– L
“Вправо”	– R

### Коды Отпускания

“Вверх”	– u
“Вниз”	– d
“Влево”	– l
“Вправо”	– r

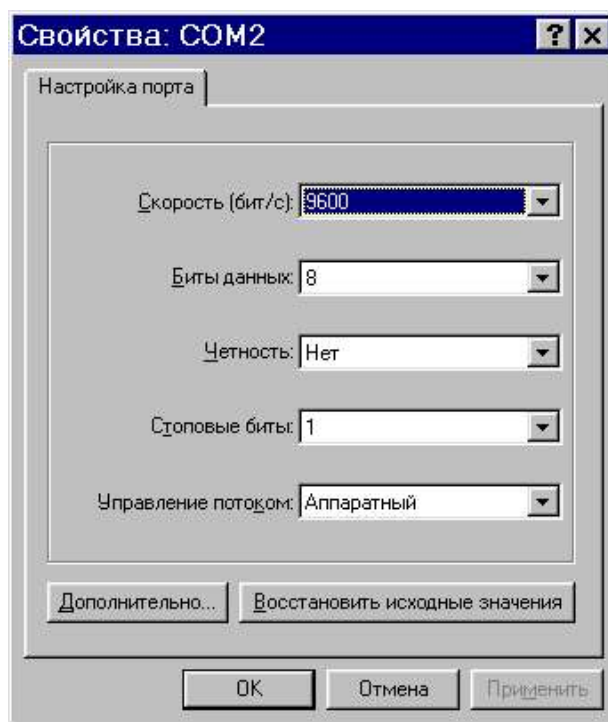
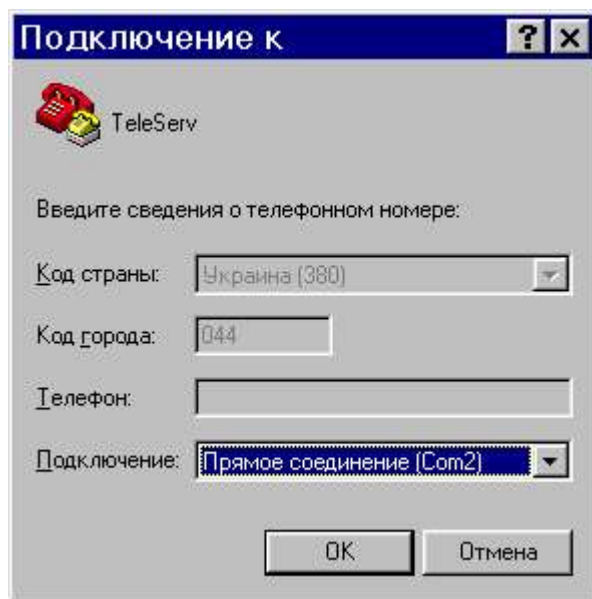
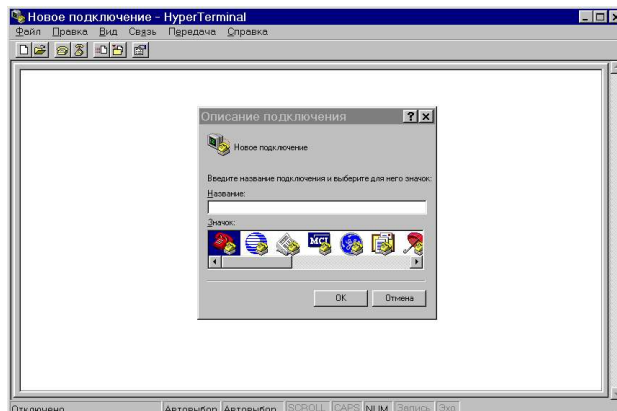
### Коды “Отпускания” Виртуальных Кнопок

После “отпускания” виртуальных кнопок, экран ЖКИ будет содержать информацию оставшуюся от диалога виртуальной кнопки. Для того, чтобы программа управляющего компьютера могла определить это и восстановить экран, ЖКИ контроллер посылает коды “отпускания” виртуальных кнопок.

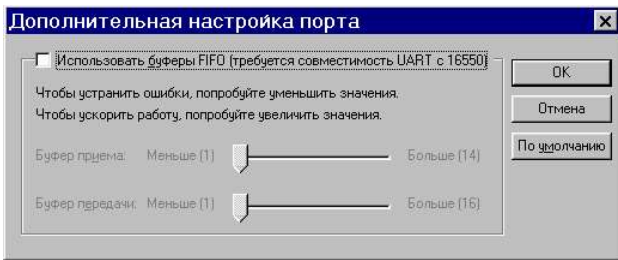
“PIN-код”	– p	- PIN-код изменен.
“Не PIN-код”	– e	- неправильно введен PIN-код.

## Использование Контроллера в Режиме “Монитор” в среде MS Windows (на примере Windows 98)

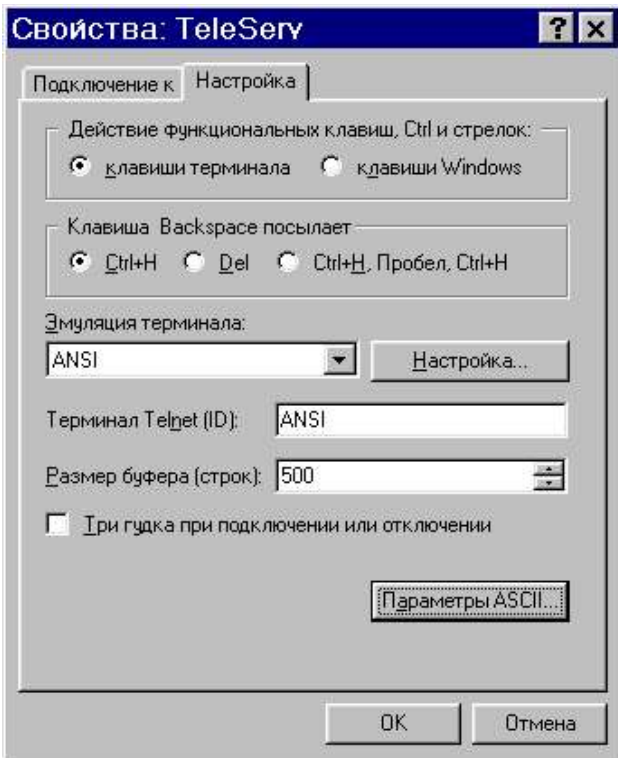
Предположим компьютер имеет свободный порт COM2, который мы задействуем для соединения с контроллером линией последовательной связи с параметрами: 9600бит/с, 8-бит, без контроля четности, 1 стоп-бит. Диалог на компьютере будем вести с помощью входящей в комплект ОС MS Windows программы *Hyper Terminal*. Для этого открываем папку *Hyper Terminal* (нажатием диалоговой кнопки “Пуск” с последующим движением по меню <Программы> <Стандартные> <Связь>). И запускаем программу *Hypertrm.exe*. На экране появляется диалоговое окно, в котором в названии подключения введем *TeleServ* (название выбирается произвольно), затем выберем прямое подключение через последовательный порт COM2 и назначим его свойства:



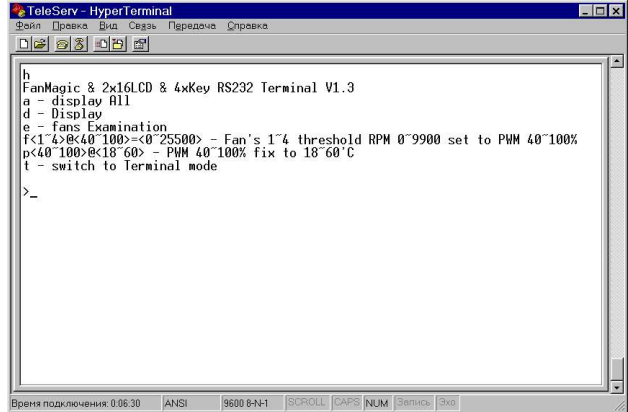
В дополнительных настройках порта укажем:



Теперь требуется настроить терминал. Нажатием кнопки **Файл** на главном меню окна программы **Hyper Terminal** вызываем выпадающее меню в котором выбираем пункт **Свойства**. В появившемся диалоговом окне **Свойства: TeleServ** выбираем закладку **Настройка** и устанавливаем следующие свойства:

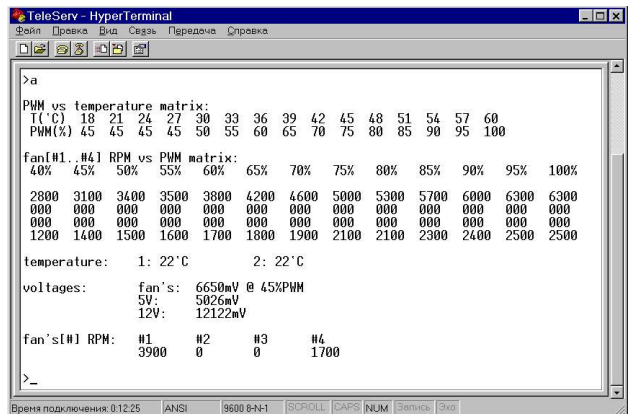


После настройки порта и инициализации терминала все готово для диалога с контроллером. Теперь все набранные на клавиатуре (терминала) команды будут подаваться непосредственно на контроллер **TeleServ** через последовательный порт, а весь поток данных от контроллера будет отображаться в окне терминала. В чистом окне терминала введем первую команду **h** (**Help**) - нажимаем на клавиатуре клавиши **h** и **Enter** (ввод). В ответ – контроллер выводит сведения о версии программного обеспечения и перечень команд в режиме “монитор”. Если Вы видите этот перечень, то все настройки выполнены правильно и можно перейти к изучению команд:



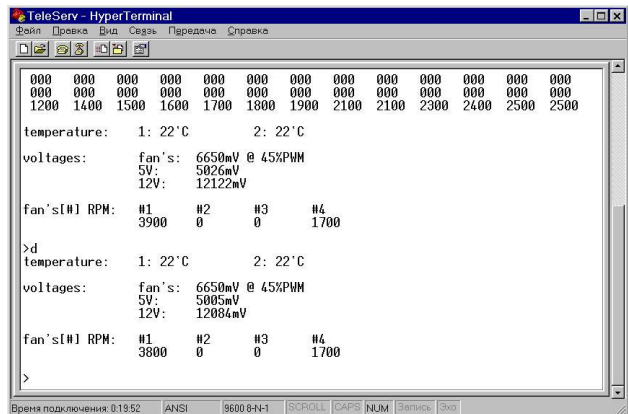
**-Команда a (display all / отобразить всю информацию)**

Нажимаем на клавиатуре клавиши **a** и **Enter** (ввод). В ответ – контроллер выводит содержимое таблицы значений PWM для всех интервалов температур, содержимое таблицы значений граничных частот вращения вентиляторов, текущие значения температуры, напряжений и частот вращения вентиляторов на момент подачи команды:



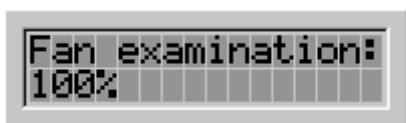
**-Команда d (display / отобразить информацию)**

Нажимаем на клавиатуре клавиши **d** и **Enter** (ввод). В ответ – контроллер выводит текущие значения температуры, напряжений и частот вращения вентиляторов на момент подачи команды :



### -Команда e (fans examination / тестирование присоединенных вентиляторов)

Нажимаем на клавиатуре клавиши **e** и **Enter** (ввод). В ответ – контроллер начинает выполнение процедуры тестирования присоединенных вентиляторов, подавая каждые 3 сек. тестовое напряжение, начиная с максимального и заканчивая минимальным уровнем, изменяя при этом значение PWM со 100% до 40% с шагом 5%. В окне программы **Hyper Terminal** последовательно отображаются все 13 шагов процедуры тестирования с указанием измеренных значений частот вращения и напряжений питающих вентиляторы. Параллельно на экране ЖКИ контроллера появляется изображение:



Затем все 12 значений PWM от 95% до 40%. Результатом работы процедуры является обновленная таблица граничных частот вращения для реально подключенных к контроллеру вентиляторов. Процедура длится около 1 минуты. В течении всего этого времени контроллер следит за питающими напряжениями и поведением вентиляторов. Контроллер определяет количество подключенных вентиляторов и их номинальные частоты вращения для всех значений PWM и соответствующих напряжений питающих вентиляторы. Если во время проведения тестирования произойдет аварийная (ошибочная) ситуация, или будет обнаружено некорректное поведение вентиляторов, то в таблицу граничных частот вращения будут записаны значения действующие по умолчанию (прописанные изготовителем при производстве). Скриншот закончившейся процедуры приведен ниже:

```

TeleServ - HyperTerminal
Файл Правка Вид Сервис Передача Справка
3800 0 0 1700
>e
Fan examination:
PWM fan voltage fan's RPM
% mV #1 #2 #3 #4
100 11799 7200 000 000 2700
95 11780 7200 000 000 2700
90 11248 6900 000 000 2700
85 10602 6500 000 000 2500
80 10013 6100 000 000 2500
75 9367 5800 000 000 2300
70 8797 5400 000 000 2200
65 8265 5000 000 000 2000
60 7638 4500 000 000 1900
55 6992 4300 000 000 1800
50 7144 4100 000 000 1800
45 6726 3800 000 000 1700
40 6118 3500 000 000 1500
threshold: 800 000 000 300
Matrix have updated in EEPROM
>_
Время подключения: 1.2317 ANSI 9600 8-N-1 SCROLL CAPS NUM Backsc Stop

```

Из приведенного примера видно, что к контроллеру подключены два вентилятора с номинальными частотами вращения 7200 и 2700 об/мин. (для PWM 100%). При значении PWM 40% напряжение

питающее вентиляторы снижается до 6,118В. Частоты вращения вентиляторов при этом снижаются до 3500 и 1500 об/мин. соответственно. Контроллер вычислил и записал в EEPROM граничные частоты вращения исходя из того, что для первого вентилятора граничные значение ниже номинальных на 800, а для второго на 300 об/мин. Эти величины 800 и 300 об/мин. контроллер рассчитывает сам, по собственному алгоритму.

### -Команда f (Fan's threshold RPM set to PWM / назначить значение граничной частоты вращения вентилятора для выбранного значения PWM)

В приведенном скриншоте работы процедуры тестирования присоединенных вентиляторов для вентилятора под номером 4, для значения PWM 40% указано значение частоты вращения 1500 об/мин. В таблице EEPROM соответственно записано граничное значение частоты вращения 1500-300=1200 об/мин. В этом можно убедиться с помощью команды **a** (в Вашем случае значения будут зависеть от свойств присоединенных вентиляторов). Для того, чтобы изменить это граничное значение на 1100 об/мин. вводится команда:

**f4@40=1100** и **Enter** (ввод).

Аналогично, для вентилятора под номером 1, для значения PWM 60% можно изменить граничное значение частоты вращения на 4300 об/мин. вводится команда:

**f1@60=4300** и **Enter** (ввод).

Правильность назначения граничных частот вращения удобно контролировать, периодически выполняя команду **a**.

### -Команда p (PWM fix to temperature / назначить значение PWM для выбранного интервала температур)

В нашем примере скриншот для команды **a** отображает действующую по умолчанию таблицу значений PWM для интервала температур, со значением PWM для интервала от 18 до 21°C равным 45%. Для того, чтобы изменить регулировочную характеристику до значения PWM 40%, и понизить частоты вращения для интервала температур от 18 до 21°C требуется ввести команду:

**p40@18** и **Enter** (ввод).

Аналогично, для того, чтобы изменить регулировочную характеристику до значения PWM 45%, и понизить частоты вращения для интервала температур от 30 до 33°C требуется ввести команду: **p45@30** и **Enter** (ввод).

Правильность назначения значений PWM для интервалов температур удобно контролировать периодически выполняя команду **a**.

*-Команда **t** (switch to terminal mode / переключить контроллер в режим "терминал")*

Служит для переключения контроллера в режим "терминал" программой запускаемой на компьютере. Можно переключить контроллер и "вручную" с тем, чтобы продолжить диалог с помощью программы *Hyper Terminal*, но этот будет диалог очень недружественным и скорее имеет смысл только для целей изучения команд контроллера в режиме "терминал". Назначение режима "терминал" - работа со специализированным программным обеспечением написанным специально для контроллера *TeleServ*.

## Использование Контроллера в Режиме "Монитор" в среде Linux

Для демонстрации функциональных возможностей контроллера в ОС Linux, можно воспользоваться терминальной программой *minicom*. *Minicom* - это программа с дружественным интерфейсом, которая позволяет читать данные из последовательного порта и писать в этот порт (COM-порт, в терминологии Windows). Эта программа входит в состав практически любого дистрибутива. Если соответствующий пакет не установлен в системе по умолчанию, установите его. Для настройки всех необходимых параметров, запустите ее из под пользователя *root*:

```
minicom -s
```

Перед вами появиться конфигурационное меню.



Если надписи в меню оказались не читаемыми необходимо установить английскую локаль, это можно сделать следующей командой:

```
export LANG=en_US
```

Навигация по меню производится с помощью клавиш "Up", "Down" при выборе одного из пунктов меню появляется окно диалога установки параметров, изменение которых выполняется нажатием соответствующей буквы латинского алфавита, указанной возле соответствующего пункта. В окне диалога **Serial port setup** необходимо изменить **Serial Device** на соответствующее устройство к которому подключен контроллер (в обычной конфигурации /dev/ttyS0 или /dev/ttyS1), и в **Bps/Par/Bits** установить скорость передачи данных через последовательный порт (**Speed**) равную 9600, аппаратное управление потоками (**Hardware Flow Control**) установить в **Yes**, после всех приведенных манипуляций окно диалога должно иметь вид как показано на рисунке:

```
A - Serial Device      : /dev/ttyS0
B - Lockfile Location : /var/lock
C - Callin Program    :
D - Callout Program   :
E - Bps/Par/Bits     : 9600 8N1
F - Hardware Flow Control : Yes
G - Software Flow Control : No

Change which setting? █
```

В окне диалога **Modem and dialing**, нужно удалить строчку инициализации (параметр **Init string**) и сброса модема (параметр **Reset string**). После завершения настройки не обходимо перейти в конфигурационное меню и выбрать пункт **Save setup as dfl** - этим мы сохраним все измененные настройки в файле */etc/minirc.dfl*. Чтобы выйти из конфигурационного меню и перейти непосредственно в терминал *minicom*, выберите пункт **Exit**. Теперь все набранные на клавиатуре (терминала) команды будут подаваться непосредственно на контроллер *TeleServ* через последовательный порт, а весь поток данных от контроллера будет отображаться в окне терминала. В окне терминала введем первую команду **h** (**Help**) - нажимаем на клавиатуре клавиши **h** и **Enter** (ввод). В ответ - контроллер выводит сведения о версии программного обеспечения и перечень команд в режиме "монитор" (аналогично рассмотренному в предыдущем параграфе случаю работы в среде MS Windows). Также как и в среде MS Windows, можно вести диалог набором поддерживаемых команд **a**, **d**, **e**, **f**, **p**, которые рассматривались в предыдущем параграфе. Реакция контроллера на введенные команды будет абсолютно аналогичной т.к. диалог ведется программным обеспечением контроллера и не зависит от операционной системы. Для выхода из программы *minicom* нажмите **Ctrl+A** затем **Q**. Для работы с контроллером *TeleServ* с помощью программы *minicom* лучше не использовать учетную запись администратора, а завести пользователя у которого были бы привилегии для записи в */dev/ttyS0* и */dev/ttyS1*. Обычно подобное достигается путем добавления нового пользователя в группу *dialout*.

## Примеры Использования Контроллера в Режиме “Терминал”

Примеры использования возможностей контроллера в режиме “терминал” реализованы при помощи скриптов на языке Perl. Работа этих скриптов проверялась в ОС SuSE Linux, но они также должны заработать в любой другой Unix подобной операционной системе. Функции интерфейса с контроллером, вынесены в отдельный модуль *TeleServ.pm*. Необходимо переписать этот файл в

какой-нибудь каталог, и соответственно определить переменную окружения PERLLIB. Проще всего - это добавить в файл /etc/profile строчку:

```
PERLLIB="название вашего каталога"
export PERLLIB
```

Можно указать каталог поиска модулей непосредственно в самом скрипте, добавив в его начало директиву:

```
use lib "название вашего каталога"
```

Такой путь предпочтительнее если скрипт запускается в окружение web-сервера Apache в виде CGI программы (это будет описано позже). Текст модуля *TeleServ.pm* приведен ниже:

```
package TeleServ;

use strict;
use vars qw(@ISA @EXPORT $VERSION $KEYBUF $DELAY);
use Fcntl qw(:DEFAULT :flock);
use Carp;
use Time::HiRes qw(usleep);

use Exporter;
@ISA = qw(Exporter);
@EXPORT = qw(tsOpen tsRead tsReadAll tsWrite tsCommand tsTemperature tsVoltage tsRpm tsWatchDog
             tsPwm tsVersion tsKey tsWaitKey tsPrint tsSetTerminalMode tsCursor);
$VERSION = '0.01';

$KEYBUF = "";
$DELAY = 0.1;

#Open device TeleServ (the controller)
sub tsOpen {
    my $name = shift;
    $name = "/dev/ttyS0" unless ($name);
    sysopen(TELESERV, $name, O_RDWR, 0666) || croak "Can't open $name";
    flock(TELESERV, LOCK_EX);
}

#Read string from the controller
sub tsRead {
    my $rin;
    my $str = "";
    vec($rin = "", fileno(TELESERV), 1) = 1;
    sysread(TELESERV, $str, 32) if (select($rin, undef, undef, $DELAY) > 0);
    return $str;
}

#Read all strings from the controller
sub tsReadAll {
    my ($rin, $str);
    while (1) {
        vec($rin = "", fileno(TELESERV), 1) = 1;
        last if (select($rin, undef, undef, $DELAY) <= 0);
        sysread(TELESERV, $str, 32);
    }
    return $str;
}
}
```

```
#Write string to the controller
sub tsWrite {
    my $str = shift;
    return unless ($str);
    syswrite(TELESERV, $str);
}

#Send command to the controller (valid for "terminal" mode)
sub tsCommand {
    my $comm = shift;
    tsWrite(chr(0x1b)."[".$comm."\n");
}

#Send some commands to the controller (parameters should be placed in array)
sub tsGroupCommand {
    my ($comm, @in) = @_ ;
    my @out = ();
    foreach (@in) {
        tsCommand($comm.$_);
        push(@out, tsRead());
    }
    return @out;
}

#Get temperature array
sub tsTemperature {
    return tsGroupCommand("t", qw(0 1));
}

#Get voltage array
sub tsVoltage {
    return tsGroupCommand("v", qw(0 1 2));
}

#Get fan rotation speed array (RPM)
sub tsRpms {
    return tsGroupCommand("r", qw(0 1 2 3));
}

#Get / Set value of counter of WatchDog timer
sub tsWatchDog {
    my $watch_dog = shift;
    if ($watch_dog ne "") {
        $watch_dog = int($watch_dog / 0.3);
        $watch_dog = 255 if $watch_dog > 255;
        tsCommand("W$watch_dog");
        usleep(100000);
        tsRead();
    }
    tsCommand("w");
    usleep(100000);
    my $curr_val = tsRead();
    return (0.3 * $curr_val);
}
```

```
#Get PWM value for indicated temperature
sub tsPwm {
  my $t = shift;
  tsCommand("p@".$t);
  return tsRead();
}

#Get the controller release number
sub tsVersion {
  tsCommand("c");
  return tsRead();
}

#Get key
sub tsKey {
  my $str = $KEYBUF.tsRead();
  if($str && $str =~ /([LIUuDdRr]{1})(.*)/ {
    $str = $1; $KEYBUF = $2;
  } else {
    $str = "";
  }
  return $str;
}

#Wait until key press. Return key code
sub tsWaitKey {
  my $key;
  while(($key = tsKey) eq "") {
    sleep 1;
  }
  return $key;
}

#Print string on LCD screen. Start on begin of either line 1 or 2
sub tsPrint {
  my ($str, $pos) = @_ ;
  $pos = 0 if (!$pos || (($pos != 0) && ($pos != 1)));
  tsCursor($pos, 0);
  usleep(50000);
  $str = (length($str) > 16)?substr($str, 0, 16):$str." "x(16 - length($str));
  tsWrite($str."\n");
}

#Close device
sub tsClose {
  close(TELESERV);
}

#Switch to "terminal" mode
sub tsSetTerminalMode {
  tsWrite("\t\n");
  tsReadAll;
  tsCommand("e");
}
```

```
#Set cursor position
sub tsCursor {
  my ($x, $y) = @_ ;
  tsCommand($x.";"$.Sy."H");
}
1;
```

Первый пример использования контролера **TeleServ** - удаленный мониторинг. Информацию о текущих значениях частот вращения вентиляторов, значений напряжения и температуры удаленного хоста можно получать в окно браузера вашего компьютера. Для этого на удаленный хост с подключенным контроллером устанавливается и запускается web-сервер и скрипт **monitoring.cgi** - обеспечивающий такой сервис. Этот скрипт проверялся с

запустить web браузер и набрать в адресной строке локальный URL скрипта, например:

<http://127.0.0.1/cgi-bin/monitoring>

на экране появиться страница со следующей таблицей:

Если хост подключен к сети, то в адресной строке вашего браузера необходимо ввести URL скрипта и наблюдать такую же картинку, но уже находясь на расстоянии от контролируемого хоста. Скрипт

## TeleServ V1.3

Mar. 2, 2006

14:40:55

Temperature	
1	22°C
2	21°C

PWM	
45%	

Voltage	
Fan's	7828 mV
+5V	4870 mV
+12V	12236 mV

Fun's	
1	0 RPM
2	0 RPM
3	0 RPM
4	1900 RPM

WatchDog	
0	

установленным на контролируемый (удаленный) хост web сервер Apache. В настройках Apache должно быть разрешено исполнение CGI программ (за подробностями обратитесь к документации к web серверу). Скрипт **monitoring.cgi** нужно скопировать в каталог, где находятся CGI программы контролируемого хоста. Для проверки можно

автоматически обновляет информацию на экране браузера. Частота обновления 3сек задается переменной `$delay_refresh` и может быть изменена.

Текст скрипта **monitoring.cgi** приведен ниже:

```
#!/usr/bin/perl -w

#use lib "";
use CGI qw(:standard);
use POSIX;
use TeleServ;

#Set 3 sec delay for page refresh
$delay_refresh = 3;
#TeleServ controller serial line port
$device_name = "/dev/ttyS1";

tsOpen($device_name);
tsSetTerminalMode;

#Get measured values from the controller
$version = tsVersion();
($temp_1, $temp_2) = tsTemperature();
$pwmm = tsPwm($temp_1);
($vol_5, $vol_12, $vol_f) = tsVoltage();
($fun_1, $fun_2, $fun_3, $fun_4) = tsRpms();
$swatch_dog = tsWatchDog();

#Get data and time
POSIX::setlocale( &POSIX::LC_TIME, "en_US" );
$curr_time = POSIX::strftime("%H:%M:%S", localtime());
$curr_date = POSIX::strftime("%b. %e, %Y", localtime());

#HTML page refresh
$java_script=<<<END;
function load()
{\nsetTimeout( "refresh()", $delay_refresh*1000 );\n}
function refresh()
{\nwindow.location.href = unescape(window.location.pathname);\n}
END

print header();
print start_html(
  -title=>"TeleServ $version",
  -script=>$java_script,
  -onload=>"load()"
);

print <<EOF;
<div align="center"><h2>TeleServ $version</h2>
<p>${curr_date}<br>${curr_time}</p></div>
<table width="30%" border="0" cellpadding="5" align="center">
<tr>
<td>
  <table width="100%" border="1" cellspacing="0" cellpadding="0">
  <tr>
    <td align="center" colspan="2">Temperature</td>
  </tr>

```

```

        <tr>
            <td align="center">1</td>
            <td align="center">${temp_1}&#176;C</td>
        </tr>
        <tr>
            <td align="center">2</td>
            <td align="center">${temp_1}&#176;C</td>
        </tr>
    </table>
</td>
</tr>
<tr>
<td>
        <table width="100%" border="1" cellspacing="0" cellpadding="0">
            <tr>
                <td align="center">PWM</td>
            </tr>
            <tr>
                <td align="center">${pwm}%</td>
            </tr>
        </table>
</td>
</tr>
<tr>
<td>
        <table width="100%" border="1" cellspacing="0" cellpadding="0">
            <tr>
                <td align="center" colspan="2">Voltage</td>
            </tr>
            <tr>
                <td align="center" width="25%">Fan's</td>
                <td align="center">${vol_f} mV</td>
            </tr>
            <tr>
                <td align="center" width="25%">+5V</td>
                <td align="center">${vol_5} mV</td>
            </tr>
            <tr>
                <td align="center" width="25%">+12V</td>
                <td align="center">${vol_12} mV</td>
            </tr>
        </table>
</td>
</tr>
<tr>
<td>
        <table width="100%" border="1" cellspacing="0" cellpadding="0">
            <tr>
                <td align="center" colspan="2">Fun's</td>
            </tr>
            <tr>
                <td align="center" width="25%">1</td>
                <td align="center">${fun_1} RPM</td>
            </tr>
        </table>

```

```

        <tr>
            <td align="center" width="25%">2</td>
            <td align="center">${fun_2} RPM</td>
        </tr>
        <tr>
            <td align="center" width="25%">3</td>
            <td align="center">${fun_3} RPM</td>
        </tr>
        <tr>
            <td align="center" width="25%">4</td>
            <td align="center">${fun_4} RPM</td>
        </tr>
    </table>
</td>
</tr>
<tr>
<td>
        <table width="100%" border="1" cellspacing="0" cellpadding="0">
            <tr>
                <td align="center">WatchDog</td>
            </tr>
            <tr>
                <td align="center">${watch_dog}</td>
            </tr>
        </table>
</td>
</tr>
</table>
EOF
print end_html();

```

Второй пример – программа, использующая WatchDog таймер контроллера. WatchDog таймер предназначен для борьбы с остановкой “зависшего” компьютера. Как правило компьютер почти никогда не “зависает”. Но если он “завис” и “остановился”, то обычно на месте или никого нет, чтобы нажать кнопку “сброс”, или никто не знает где именно находится “зависший” компьютер, поскольку обычно с ним нет проблемы. Именно для этих целей и служит WatchDog таймер – автоматически перезапустить “зависший” компьютер. При включении контроллера WatchDog таймер выключен и для того чтобы его включить требуется по последовательной линии послать команду "Esc[WW" с требуемым значением счетчика таймаута W (число тиков от 0 до 255, каждый тик равен 0,3 сек). Для того, чтобы выключить таймер, требуется установить таймаут=0 - "Esc[W0". Управлением загрузкой счетчика и контролем его содержимого занимается специальная программа-драйвер. Для непрерывной работы компьютера программа-драйвер должна периодически устанавливать таймаут для избежания “гонга” и последующего замыкания контактов “сброс”. Если компьютер “залип”, то программа-драйвер перестает посылать “свежие” команды установки таймаута и WatchDog таймер “ударит в

гонг” - замыкание контактов “сброс”. WatchDog таймер реализован так, что он “ударяет” только один раз. Это позволяет избежать повторяющихся “ударов” во время проверки файловой системы, которая вероятней всего следует после сброса (или при первом включении контроллера/компьютера). После того как компьютер снова “поднимается”, программа-драйвер должна себя повторно разрешить (послать команду Esc[WW с требуемым значением таймаута). WatchDog таймер гарантирует, что система всегда способна выполнять программы. Он не гарантирует, что приложение все еще выполняется и отзывается. Для того, чтобы проверить эти две вещи можно использовать *crontab entry* для Linux/UNIX или другие программы. Можно быть уверенным в исполнении главной программы, до тех пор пока работает *crontab*, поскольку WatchDog таймер обеспечивает это. Например, можно создать скрипт, который автоматически запускается с помощью *cronjob* и загружает веб-страничку с некоторого вебсервера каждые 50 секунд, но нужно понимать, что вебсервер может быть полностью загружен множеством запросов и поэтому это нормально, что не может ответить скрипту. В нашем примере мы создали скрипт *watchdog.cgi* для интерактивного изучения возможностей WatchDog таймера. Этот

скрипт, также как и первый пример, представляет собой CGI программу с web интерфейсом и его также можно использовать для удаленного хоста. Для установки скрипта воспользуйтесь указаниями приведенными для первого примера (скрипт *monitoring.cgi*, который приведен выше). Для изучения WatchDog таймера на локальном компьютере наберите в адресной строке вашего браузера URL скрипта:

<http://127.0.0.1/cgi-bin/watchdog>

на экране появиться страница со следующей таблицей :

Current value of WatchDog timer (sec): 0

Refresh time (sec)	<input type="text"/>
WatchDog timeout (sec)	<input type="text" value="0"/>
Update time (sec)	<input type="text"/>

start stop

В поле **Refresh time** укажите время автоматического обновления информации на экране браузера. Комфортное время 2-3 секунды. В поле **WatchDog timeout** укажите начальное значение счетчика (например 50 секунд). В поле **Update time** укажите время обновления начального значения счетчика (например 30 секунд). Клик на кнопку **start** запускает страницу на непрерывное обновление (с периодом указанным в **Refresh time**), “взводит” счетчик таймера в начальное значение и позволяет наблюдать, как периодически значение счетчика уменьшается, а затем снова устанавливается в начальное значение. Желательно не подключать коннектор J13 контроллера к коннектору “сброс” на материнской плате, когда подбираются различные начальные значения счетчика и время обновления. Иначе всякий раз, когда значение счетчика будет обнуляться, компьютер будет идти на перезагрузку. Если хост подключен к сети, то в адресной строке вашего браузера необходимо ввести URL скрипта и наблюдать такую же картинку, но уже находясь на расстоянии от контролируемого хоста.

Текст скрипта *watchdog.cgi* приведен ниже:

```
#!/usr/bin/perl -w

#use lib "";
use TeleServ;
use CGI qw(:standard);

sub isInt {
    my $str = shift;
    return ($str =~ /^d+$/)?$str:"";
}

$device_name = "/dev/ttyS1";

tsOpen($device_name);
tsSetTerminalMode;
$version = tsVersion();

%val = ();
%cookie = cookie('watchdog');
@true_keys = qw(update wd wdtick wdtickfirst);

#Check cookie
foreach (@true_keys) {
    $val{$_} = isInt($cookie{$_});
}
```

```

if (param()) {
  my $tmp;
  if( param('start') ) {
    $val{update} = isInt(param('update'));
    $val{wd} = isInt(param('wd'));
    $val{wdtick} = 0 if ($val{wdtickfirst} = isInt(param('wdtick')));
  }
  #Stop refresh if button "stop" pressed or invalid values are entered
  if( param('stop') || ($val{update} eq "") || ($val{wd} eq "") || ($val{wdtickfirst} eq ""))
  {
    foreach (keys %val) {
      $val{$_} = "";
    }
    $watch_dog = tsWatchDog(0);
    goto EXIT;
  }
}

$val{wd} = 0 unless ($val{wd});

if ($val{wdtick} ne "") {
  $val{wdtick}--;
  if ($val{wdtick} < 0) {
    #Number of refresh cycles when WatchDog timeout should be uploaded
    $val{wdtick} = int($val{wdtickfirst}/$val{update});
    $watch_dog = tsWatchDog($val{wd});
    goto EXIT;
  }
}
$watch_dog = tsWatchDog;
EXIT:

$cookie = cookie( -name=>'watchdog',
                  -value=>\"%val );

print header(-cookie=>$cookie);

$html_name = "TeleServ $version";

if ($val{update}) {
#java script that should refresh web page
  $java_script=<<<JAVA;
function load()
{\setTimeout( "refresh()", $val{update}*1000 );\n}
function refresh()
{\nwindow.location.href = unescape(window.location.pathname);\n}
JAVA
  print start_html(
    -title=>$html_name,
    -script=>$java_script,
    -onload=>'load()'
  );
} else {
  print start_html(
    -title=>$html_name
  );
}

```

```

print "Current value of WatchDog timer in ticks (sec): $watch_dog<br>";

print start_form();
print table( {-border=>undef},
  Tr( {-align=>CENTER}, td( ["Refresh time (sec)", textfield(-name=>'update', -default=>$val{update})])),
  Tr( {-align=>CENTER}, td( ["WatchDog timeout (sec)", textfield(-name=>'wd', -default=>$val{wd})])),
  Tr( {-align=>CENTER}, td( ["Update time (sec)", textfield(-name=>'wdtick', -default=>$val{wdtickfirst})]))
);
print submit(-name=>'start');
print submit(-name=>'stop');
print end_form();

print end_html();

```

Третий пример – скрипт *mp3play.pl* - управление простым проигрывателем MP3 файлов. Управление осуществляется с помощью кнопок контролера, а вся нужная информация о проигрываемой композиции отображается на дисплее контроллера *TeleServ* в виде бегущей строки. Для проигрывания MP3 композиций используется проигрыватель `mpg123` ([www.mpg123.de](http://www.mpg123.de)). Установите его в папку `/usr/bin`. Для работы с MP3 тегами нам также будет необходим модуль `MP3::Tag`. Его можно взять на сервере CPAN ([www.cpan.org](http://www.cpan.org)), просто наберите в командной строке (из-под учетной записи root):

```
perl -MCPAN -e 'install(MP3::Tag);'
```

Замечание, [CPAN.pm documentation](#) содержит более полные инструкции, как пользоваться этим удобным средством. Теперь можно запустить проигрыватель, указав при этом проигрываемый каталог с музыкой, например:

```
./mp3play /home/Music
```

Нажатие клавиши “Вверх” на контроллере осуществляет переход на следующую композицию, “Вниз” - на предыдущую, “Вправо” - остановит проигрывание.

Текст скрипта *mp3play.pl* приведен ниже:

```

#!/usr/bin/perl -w

use lib "";
use MP3::Tag;
use File::Find;
use Time::HiRes qw(usleep);
use Getopt::Std;
use TeleServ;

$prog_name = $0;
$player_name = "/usr/bin/mpg123";
@mp3files = ();

sub getid3 {
  my $filename = shift;
  my $mp3 = MP3::Tag->new($filename);
  my @out_id3 = $mp3->autoinfo();
  return @out_id3;
}

sub formatid3 {
  my @param = @_;
  my @formats = qw(t n p a c y g);
  return if (scalar @param != 8);
  my $format_str = pop @param;
  for(my $i = 0; $i < 7; $i++) {
    $format_str =~ s/%%$formats[$i]/$param[$i]/g;
  }
  return $format_str;
}

```

```

sub wanted {
    /^.*\.[mM][pP]3\z/s &&
    push(@mp3files, $File::Find::name);
}

# Return string array to scroll on LCD
sub scroll_arr {
    my ($str, $num) = @_;
    my @out_arr;
    return $str if (length($str) <= 16);
    my $len = length($str) - $num;
    for( $i = 0; $i <= $len; $i++ ) {
        $out_arr[$i] = substr($str, $i, $num);
    }
    return @out_arr;
}

sub child_kill {
    my $p = shift;
    $SIG{CLD} = IGNORE;
    kill(KILL, $p) if ($p);
}

sub usage {
    print "Usage: $prog_name -d <device_name> -f <format_string> dir_name\n";
    print "\tRead man page for more detail\n";
    exit 0;
}

getopts("d:f:h", \%opts);

$opts{d} = "/dev/ttyS0" unless($opts{d});
$opts{f} = "%t" unless($opts{f});
usage() unless($opt{h} || $ARGV[0] || -d $ARGV[0]);

File::Find::find({wanted => \&wanted}, $ARGV[0]);

tsOpen($opts{d});
tsSetTerminalMode;

$mp3_ind = 0;

while (1) {
    $name_file = $mp3files[$mp3_ind];
    $print_str="Unsupported" unless ($print_str = formatid3(getid3($name_file), $opts{f}));
    @scroll = scroll_arr($print_str, 16);

    #Run player (fork)
    if( $pid = open(CHILD,"|") ) {
        #Set $proc_exit when player is stopped
        $proc_exit = 0;
        $SIG{CLD} = sub { $proc_exit = 1; };
        #direction of scroll on LCD
        $scr_course = 1;
        $scr_ind = 0;
    }
}

```

```
while (1) {
    #one position scroll on LCD
    if( $scr_course ) {
        if ($scr_ind >= scalar(@scroll) - 1) {
            $scr_course = 0;
        } else {
            $scr_ind++;
        }
    } else {
        if ($scr_ind <= 0) {
            $scr_course = 1;
        } else {
            $scr_ind--;
        }
    }
    tsPrint($scroll[$scr_ind]);
    usleep 400000;
    #Next or Previous file playing
    $key = tsKey();
    if ($proc_exit || ($key eq "u")) {
        $mp3_ind++;
        $mp3_ind = 0 if( $mp3_ind >= scalar(@mp3files) );
        last;
    } elsif ($key eq "d") {
        $mp3_ind--;
        $mp3_ind = scalar(@mp3files) - 1 if( $mp3_ind < 0 );
        last;
    } elsif ($key eq "r") {
        tsPrint("Stop");
        child_kill $pid;
        while(tsWaitKey() ne "r") {}
        goto EXIT;
    }
}
child_kill $pid;
EXIT:
close CHILD;
} else {
    exec $player_name, $name_file;
}
}
```