УТВЕРЖДАЮ

Директор ООО "МикроКОР"
Головенко В.Б.

"___"_____1999 г.

Модуль управления МК-167-IO-1

для автоматизированных малогабаритных котельных установок АМКУ-МТ2; АМКУ-МТ3

Инструкция по применению
МК3.000.111 Д1

г. Санкт-Петербург
1999г

Содержание

1. Основная информация

1.1. Назначение

Модуль управления (в дальнейшем МУ) предназначен для работы в составе системы управления автоматизированными малогабаритными котельными установками АМКУ-МТ2; АМКУ-МТ3.

1.2. Основные параметры

Модуль управления **МК-167-IO-1** предназначен для установки в блок управления и обеспечивает:

- сбор данных от аналоговых и дискретных датчиков для использования этих данных в управлении;
- прием по последовательному каналу связи управляющих команд;
- формирование управляющих сигналов на исполнительные устройства котельной установки согласно заданному алгоритму.

Данные о датчиках температуры (внешних термопреобразователей сопротивления типа ТМ9210 и ТП9211)и диапазонах измеряемых температур приведены в табл.1.

МУ обеспечивает прием на вход АЦП контроллера аналогового сигнала от датчика вибрации (акселерометр) ДВ1 (ДВ/1 и ДВ/2) с диапазоном изменения напряжения 0–5В. Привязка потенциала входа к земле осуществляется через резистор R1, величиной 20 кОм±5%.

МУ обеспечивает прием на входы АЦП контроллера аналоговых сигналов Ан0 (Ан0/1 и Ан0/2)…. Ан9 (Ан9/1 и Ан9/2) от датчиков давления типа МИДА-ДИ-01П в стандарте 4-20мА.

Таблица 1

Вход	Измеряемый параметр	Тип датчика, НСХ	Рабочий диапазон температур, °С	Суммарная относительная приведенная погрешность преобразования измерительного канала, не более %	Относительн. приведенная погрешность измерения сопротивления датчиков температуры, не более%	Прим.
Т1, ПСН 1	t° масла	50М класс В	0…150	2,0%	1,0%	
Т2, ПСН 2	t° конденс.	50М класс В	0…150	2,0%	1,0%	
Т3, ПСН 3	t° бойлер	50М класс В	0…150	2,0%	1,0%	
Т4, ПСН 4	t° ух. газов	50П класс В	0…200	2,0%	1,0%	
Т5, ПСН 5	t° пит. воды	50П класс В	0…200	2,0%	1,0%	
Т6, ПСН 6	t° пара	50П класс В	0…200	2,0%	1,0%	
Т7, ПСН 7	t° пара	50П класс В	0…400	2,0%	1,0%	
Т8, ПСН 8	t° нар. воздуха	50М класс В	-60…40	1,0%	0,5%	
Т9, ПСН 9	t° хол. воды	50М класс В	0…150	1,0%	0,5%	
Т10, ПСН10	t°гор. воды	50М класс В	0…150	1,0%	0,5%	

Примечания:

1. НСХ – нормированная статическая характеристика по ГОСТ Р 50353-92.
2. Допускается применение ПСН с линейной характеристикой и последующей программной линеаризацией Заказчиком.
3. Датчики температуры подключаются по трех или четырех-проводной схеме.

МУ обеспечивает связь в внешними устройствами по двум последовательным каналам связи: RS232 и гальванически развязанному выходу на CAN-сеть.

МУ обеспечивает прием на входы D0…D11 логических сигналов ТТЛ-уровней.

МУ выдает управляющие сигналы на Вых 0…Вых 7 с максимальным током не менее 15мА для коммутации светодиодов.

Выходы 0…3 снабжены резисторами для подключения светодиодов.

МУ обеспечивает коммутацию по выходам К0…К14 (ключи переменного тока) электромагнитных силовых исполнительных устройств (напряжение питания 220В 50Гц) согласно табл. 2.

Таблица 2

Выходы	Максимальный ток, А	Количество, шт.	Тип нагрузки
К0…К7	0,3	8	Электромагнитные клапаны
К8…К11	0,5	4	Катушки контакторов
К12…К14	1,0	3	Катушки контакторов электродвигателей

Относительная приведенная погрешность преобразования – не более 1%.

Время опроса всех аналоговых каналов (10 термодатчиков + 2 потенциальных входа + 10 токовых аналоговых входов) – не более 1 с.

Гальванической развязки по аналоговым входам не обеспечивается.

МУ в составе блока управления надежно работает при следующих условиях:

- температуре окружающей среды +5...+40 °С
- относительной влажности при +30 ^{0} С не более 95%
- атмосферном давлении 84,0...106,7 кПа (630 ...800 мм рт.ст.)

МУ в составе блока сохраняет работоспособность после хранения и ранспортирования в упакованном виде в диапазоне температур –50…+40°С, влажность 98% при 25°С.

МУ в составе блока управления, упакованного в транспортную тару устойчив к механическим воздействиям в соответствии с ГОСТ 12997-84, п.2.24.3, для группы N2.

1.3. Спецификация

Таблица 3

Технические данные		
Количество аналоговых входных каналов измерения температуры (10-разрядные)		10
Количество аналоговых входных каналов измерения вибрации с диапазоном изменения напряжения 0–5В (12-разрядные)		2
• Количество аналоговых входных каналов от датчиков давления типа МИДА-ДИ-01П в стандарте 4-20мА (12-разрядные)		10
Время выполнения команды - (пересылка типа регистр-регистр)		100 нс
Время опроса всех аналоговых каналов (10 термодатчиков + 2 потенциальных входа + 10 токовых аналоговых входов)		не более 1 с.
Основная приведенная относительная погрешность преобразования по любому каналу		не более 1%
Количество дискретных входных каналов ТТЛ-уровней		12
Количество дискретных выходных каналов		8
Выходные ключи переменного тока напряжение питания 220В 50Гц	0,3А	8
	0,5А	4
	1,0А	3
RS232 и гальванически развязанный выход на CAN-сеть, напряжение изоляции		не менее 500 В
Объем ПЗУ (FLASH		512 Кбайт
Объем ОЗУ		128 Кбайт
Объем внутренней памяти.		2 Кбайта
Количество 16 разрядных таймеров-счетчиков		9
Количество векторов прерываний		56
Скорость передачи по последовательному каналу RS232		до 625 Кбит/с
Скорость передачи по оптоизолированному CAN интерфейсу (спецификация 2.0)		до 1 Мбит/с
Скорость передачи через скоростной синхронный порт		до 5 Мбит/с
Стартовый загрузчик. (Bootstrap loader) Позволяет загружать программу по последовательному порту в ОЗУ контроллера.		-

Обозначение модуля при заказе:

модуль **МК-167-IO-1 -X** – модуль управления

- **C** - коммерческий температурный диапазон (0…70°C);
- **I** - индустриальный темп. диапазон (-40…+80°C).

2. Описание работы модуля

2.1. Состав модуля

В состав МУ интегрированы следующие узлы:

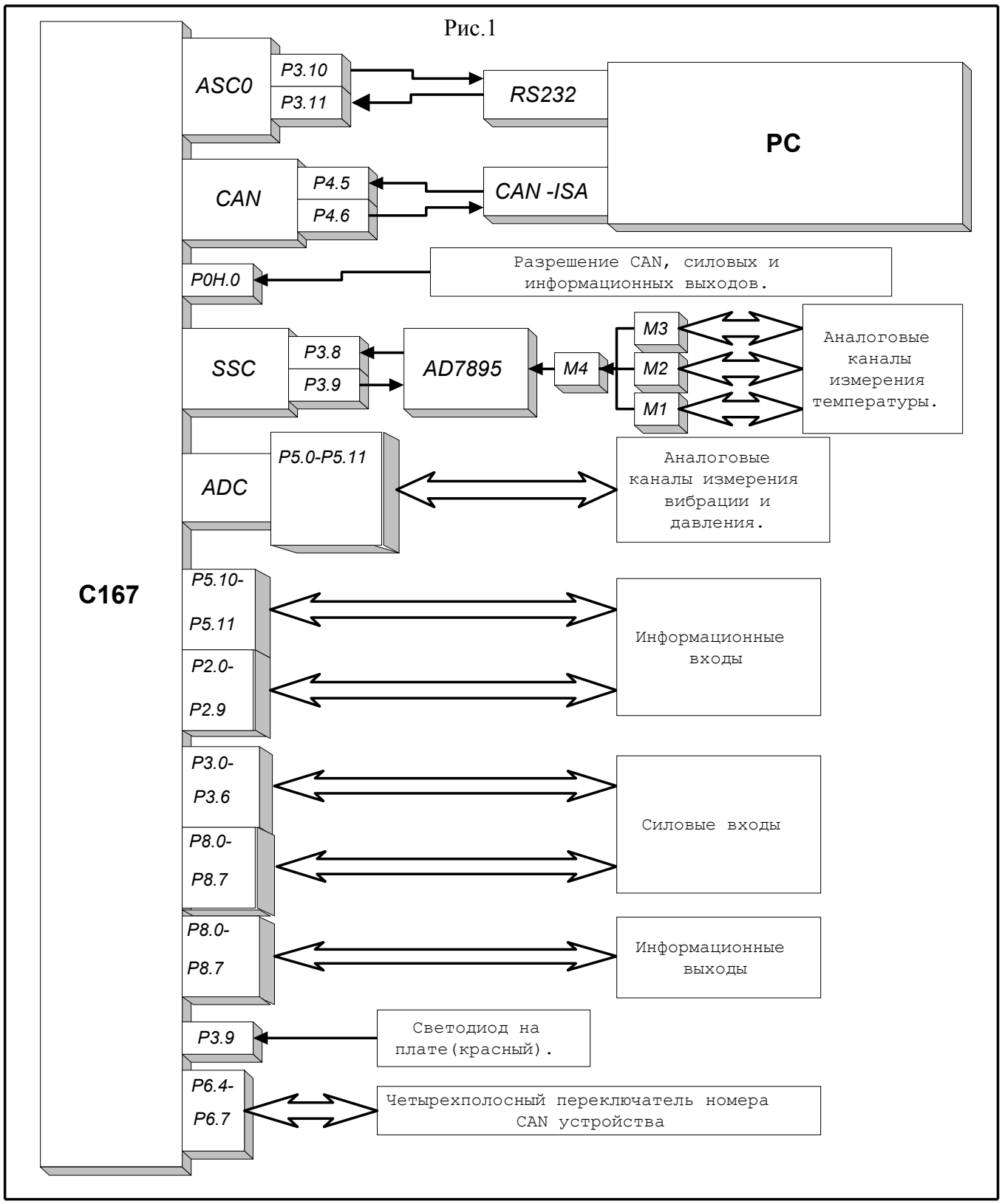
1. узел центрального процессора Siemens C167CR-LM обработки данных;
2. узел сопряжения с датчиками температуры (терморезисторами);
3. узел сопряжения с потенциальными и токовыми аналоговыми датчиками;
4. узел сопряжения с дискретными входами-выходами;
5. узел силовых гальванически развязанных выходов 220В;
6. узел гальванически развязанного CAN-канала;
7. узел питания.

Структурная схема модуля приведена на рис.1.

Обозначения на схеме:

ASC0 – ассинхронно-синхронный последовательный интерфейс
PC – персональный компьютер
CAN – контроллер CAN сети
SSC – скоростной синхронный последовательный интерфейс
AD7895 – микросхема АЦП
M1, M2, M3, M4 – мультиплексоры
Px.y – порты

Структурная схема модуля



2.2. Распределение памяти контроллера.

Пространство памяти разбито на 1024 страницы по 16 Кбайт или 256 сегментов по 64 Кбайт. Страничная адресация (по 16 Кбайт) осуществляется с помощью специальных регистров DPPO-DPP3. Назначение сигналов выборки устройств CS0-CS1:

- Сигнал CSO предназначен для выборки микросхемы ПЗУ.
- Сигнал CS1 предназначен для выборки микросхемы ОЗУ.

Соответствие сигналов выборки устройств (CSO-CS2) регистрам конфигурации микроконтроллера:

- CS0 BUSCON0 .... ПЗУ на плате (128Kb)
- CS1 BUSCON1, ADDRSEL1 .... ОЗУ на плате (512Kb)

2.3. Аналоговые каналы измерения температуры.

Аналоговые каналы измерения температуры реализованы с использованием микросхемы АЦП AD7895AR-3 и четырех мультиплексоров HEF4052BT. Сбор данных с любого из десяти каналов осуществляется по следующему алгоритму:

- Конфигурирование.
- Выбор канала.
- Запуск АЦП.
- Ожидание конца преобразования данных.
- Считывание данных с АЦП и сохранение.

Рассмотрим каждый шаг алгоритма.

1). Конфигурирование осуществляется один раз перед началом работы.

Для выбора канала используются порты P2.12, P2.13, P2.14, P2.15, которые конфигурируются на выход DP2.12=1, DP2.13=1, DP2.14=1, DP2.15=1(после знака “=” указаны константные значения).

Для обмена данными между АЦП и процессором используется скоростной последовательный интерфейс SSC, к которому относятся специальные регистры SSCCON=0xC03F, SSCBR=0x0001 и линии связи P3.8=0, P3.9=1, P3.13=1, которые конфигурируются при помощи портов направления DP3.8=0, DP3.9=1, DP3.13=1.

Для сигнала начала конвертации данных АЦП используется порт P3.12 (активный 0), конфигурируется на выход DP3.12=1. Перед началом работы P3.12=1(неактивное состояние).

2). При помощи мультиплексоров осуществляется выбор сканируемого канала подачей соответствующих сигналов на порты:

- АДР0 – P2.12
- АДР1 – P2.13
- АДР2 – P2.14
- АДР3 – P2.15

Соответствие значений АДР0 - АДР3 выбранному каналу:

Канал	АДР3	АДР2	АДР1	АДР0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	1	0	0	0
8	1	0	0	1
9	1	0	1	0

3). Для запуска АЦП необходимо установить P3.12=0, обратная установка порта в единицу рекомендуется после окончания преобразования для уменьшения помех работе АЦП.

4). Так как сигнал занятости АЦП отсутствует необходимо организовать задержку 10 мксек (например при помощи таймера).

5). После окончания преобразования данные хранятся в АЦП.

Для их съема инициируется обмен по SSC записью в специальный регистр SSCTB произвольного числа. Ожидание окончания передачи по флагу SSCBSY (0- передача окончена) в регистре SSCCON. После окончания передачи данные (12 разрядов) хранятся в регистре SSCRB.

Для получения корректного двухбайтного отрицательного числа старшие 4 разряда заполняются единицами, признаком отрицательности служит 11-ый бит (0 - число положительное, 1 – число отрицательное).

2.4. Аналоговые каналы измерения вибрации и давления.

Каналы измерения вибрации и давления базируются на АЦП входящем во внутреннюю периферию процессора SAF-C167CR-LM. Каналы 0-1 используются для измерения вибрации, 2-11 для измерения давления. Съем данных происходит по алгоритму:

- Конфигурирование.
- Выбор канала.
- Запуск АЦП.
- Ожидание конца преобразования в цифровую форму.
- Считывание данных и сохранение.
- Рассмотрим каждый шаг алгоритма.

1). Конфигурирование осуществляется один раз перед началом работы и сводится только к настройке АЦП. Входами служат порты P5.0-P5.11 которые могут работать только на вход и в настройке не нуждаются.

Специальный регистр конфигурации АЦП ADCON=0 (в данном случае используется режим разового преобразования фиксированного канала).

2). Для выбора канала используются младшие четыре бита регистра ADCON (номер задается двоичным кодом).

3). Конверсия запускается установкой бита ADST=1 регистра ADCON.

4). После установки ADST=1 автоматически устанавливается бит занятости ADBSY регистра ADCON. Сброс бита в 0 сигнализирует об окончании преобразования.

5). Результат находится в регистре ADDAT (биты 0-9), биты 12-16 содержат номер канала.

2.5. Информационные входы.

Модуль управления располагает 12-тью дискретными входами для тумблеров и кнопок. Задействованы порты P2.0-P2.9 и P5.10-P5.11, которые посредством регистров направления портов DP2 и DP5 устанавливаются на вход записью нулей в соответствующие биты.

2.6. Разрешение CAN, силовых и информационных выходов.

Для доступа к CAN, силовым и дискретным выходам необходимо установить P0H.0=0 и DP0H.0=1.

2.7. Информационные выходы.

Для информационных выходов комутации светодиодов служат порты P7.0-P7.7, которые посредством регистра направления DP7=0xFF устанавливаются на выход.

Выключенному состоянию соответствует единица.

2.8. Силовые выходы.

Силовые выходы – 15 ключей переменного тока (220В 50Гц). Для переключения служат порты P3.0-P3.6 и P8.0-P8.7, которые посредством регистров направления портов DP3 и DP8 устанавливаются на выход записью едениц в соответствующие биты.

2.9. Светодиод на плате (красный).

Для использования светодиода устанавливается DP0H.7=0, P0H.7=1- выключенное состояние.

2.10. Четырехполосный переключатель номера CAN устройства.

Для обмена данными по CAN сети между несколькими устройствами, каждое из них должно иметь уникальный идентификатор, который задается при помощи переключателя.
Ипользуются порты P6.4- P6.7, которые конфигурируются на выход через регистр DP6 установкой едениц в соответствующих битах.

2.11. CAN интерфейс.

CAN интерфейс является периферийным устройством процессора SAF-C167CR-LM и соответствует спецификации 2.0.

CAN интерфейс занимает 256 ячеек памяти в первом сегменте, начиная с адреса EFOOh. В этой области располагаются:

- 15 регистров объектов-сообщений,
- регистры конфигурации, контроля и арбитража,
- регистр статуса/контроля (Control/status register 0xEF00),
- регистр прерываний (Interrupt register 0xEF02),
- регистр битов таймера (Bit timing register 0xEF04),
- регистры глобальной (длинной) (Global mask long 0xEF08) и короткой (Global Mask Short 0xEF06) маски,
- маска последнего сообщения (Mask of last message 0xEF0C).

Для приема и передачи используются порты P4.5 и P4.6. Длина сообщения может быть от 1 до 8 байт.

Объекты-сообщения с номерами с 1 по 14 могут работать на передачу и прием, сообщение с номером 15 имеет буферированный режим, то есть можно принять еще одно сообщение до того, как будет прочитано предыдущее.

Конфигурирование CAN сводится к настройке специальных регистров. Например, для обмена данными между двумя устройствами достаточно сконфигурировать два обьекта-сообщения: одно для приема данных, другое для передачи а также параметры для взаимодействия, такие как скорость обмена, арбитраж и т. д. (см. пример в приложении и C167 User Manual).

2.12. Последовательный порт.

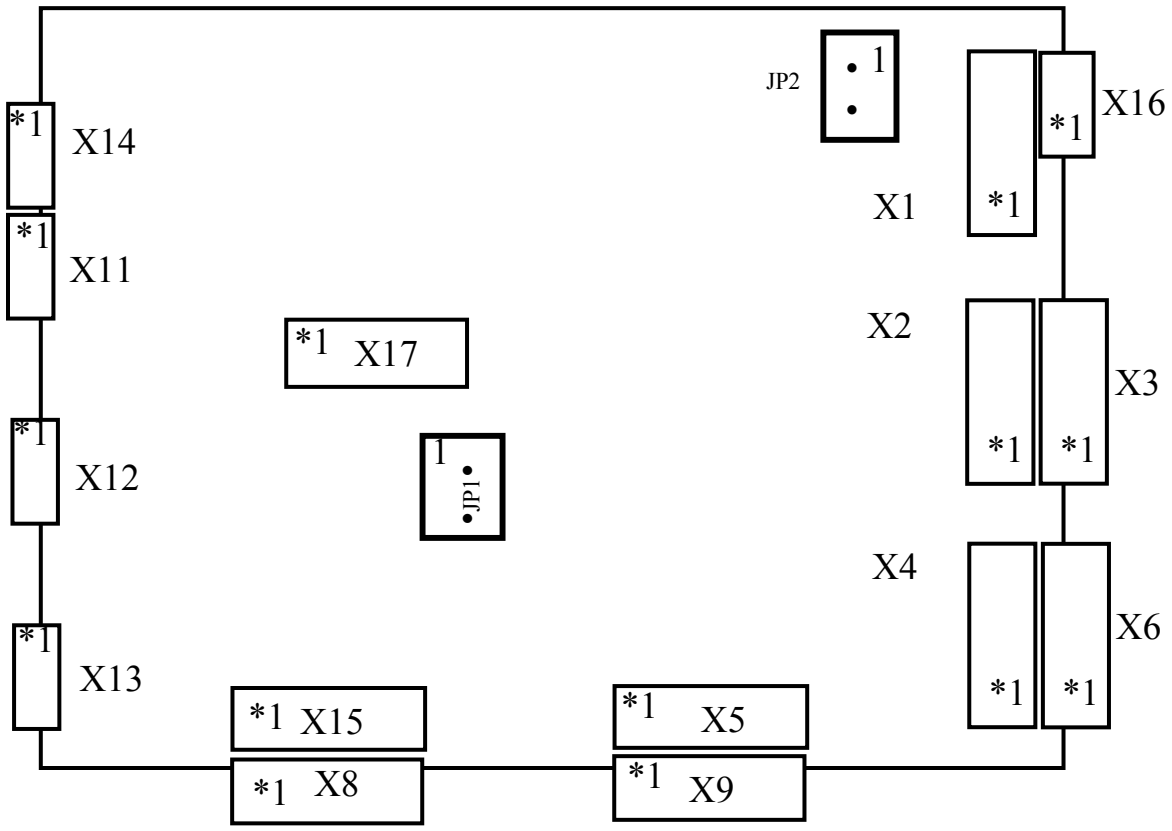
Последовательный порт передачи данных используется для связи с PC в отладочных целях через интерфейс RS232C.

Используется ASC0 входящий в периферию процессора для работы в асинхронном режиме.

При конфигурировании настраиваются порты P3.10=1, P3.11=0 и специальные регистры: S0CON= 0x8011, S0BG=0x40 (данная конфигурация предназначена для работы при скорости 19.2 Кбод, для работы с другими скоростями см. C167 User Manual)

2.13. Описание разъемов МК-167-IO-1

Конструктивно МУ представляет из себя плату размером 267x236x38мм**.**



X1...X6 – разъёмы подключения датчиков температуры и аналоговых датчиков в стандарте 4...20 мА.

X1		X2		X3		X4		X5		X6	
Конт	**Цепь**	**Конт**		**Конт**	**Цепь**	**Конт**	**Цепь**	**Конт**	**Цепь**	**Конт**	**Цепь**
1	AGND	1	AGND	1	AGND	1	AGND	1	AGND	1	AGND
2	Uоп 2	2	Uоп 4	2	Uоп 4	2	AIN I8	2	AIN V0	2	AIN I3
3	T2 C	3	T5 C	3	T8 C	3	AGND	3	AGND	3	AGND
4	T2 A	4	T5 A	4	T8 A	4	AIN I9	4	AIN V1	4	AIN I4
5	Uоп 0	5	Uоп 4	5	Uоп 4	5	AGND	5	AGND	5	AGND
6	T1 C	6	T4 C	6	T7 C	6	-	6	AIN I0	6	AIN I5
7	T1 A	7	T4 A	7	T7 A	7	-	7	AGND	7	AGND
8	Uоп 0	8	Uоп 2	8	Uоп 2	8	Uоп 8	8	AIN I1	8	AIN I6
9	T0 C	9	T3 C	9	T6 C	9	T9 C	9	AGND	9	AGND
10	T0 A	10	T3 A	10	T6 A	10	T9 A	10	AINI2	10	AIN I7

X8, X9 – разъёмы подключения дискретных датчиков

X8		X9	
Конт	**Цепь**	**Конт**	**Цепь**
1	DIN 0	1	DIN 8
2	DIN 1	2	DIN 9
3	DIN 2	3	DIN 10
4	DIN 3	4	DIN 11
5	DIN 4	5	-
6	DIN 5	6	-
7	DIN 6	7	-
8	DIN 7	8	-
9	GND	9	GND
10	GND	10	GND

X11... X13 – разъёмы подключения силовых дискретных выходов

X11		X12		X13	
Конт	**Цепь**	**Конт**	**Цепь**	**Конт**	**Цепь**
1	K0	1	K5	1	K10
2	K1	2	K6	2	K11
3	K2	3	K7	3	K12
4	K3	4	K8	4	K13
5	K4	5	K9	5	K14

X14 – разъём питания 220 В.

Конт	Цепь
1	220V F2
2	220V F
3	220V F
4	220V N
5	220V N

X15 – разъём подключения дискретных выходов на светодиоды

Конт	Цепь	Конт	Цепь
1	LED7	6	LED2
2	LED6	7	LED1
3	LED5	8	LED0
4	LED4	9	Vcc
5	LED3	10	Vcc

X16 – разъём подключения CAN

Конт	Цепь
1	-
2	CAN L
3	GND can
4	CAN H
5	-

X17 – разъём канала RS-232.

Конт	Цепь	Конт	Цепь
1	-	6	-
2	-	7	-
3	RS in	8	-
4	-	9	GND
5	RS out	10	-

Джампер JP1 используется для установки загрузки в режиме Bootstrap

перемычка **1-2** - режим включен.

Джампер JP2 используются для подключения резисторов нагрузки линии CAN:

перемычка **1-2** - линия подгружена.

3. Подготовка к работе с модулем

3.1. Установить на плате контроллера mK_CAN с помощью перемычек и переключателя (в соответствии с руководством по применению контроллера mK_CAN) канал сетевого прерывания CAN и базовый адрес доступа к контроллеру таким образом, чтобы они не вступали в конфликт с ресурсами, используемыми в ПЭВМ, входящей в стенд по проведению испытаний.

Доступны каналы прерываний IRQ5, IRQ7, IRQ10, IRQ11, IRQ12, IRQ15 и базовые адреса 0x120, 0x1B0, 0x220, 0x300.

3.2. Установить в ПЭВМ, входящую в стенд по проведению испытаний, плату CAN контроллера mK_CAN. Монтаж платы контроллера и присоединение сетевого кабеля CAN следует проводить при выключенном питании ПЭВМ.

3.3. Инсталлировать тестовое программное обеспечение с помощью программы-инсталлятора **mK_install**.

В состав тестового программного обеспечения входят:

- Драйвер платы контроллера CAN для Windows 95/98 vichw11.vxd,
- Файл конфигурации платы контроллера CAN для Windows NT 4.0, setup_nt.inf.
- Библиотека поддержки функций связи по сети CAN mK_CAN32.dll,
- Тестовая программа для PC _Mt4H.exe,
- Тестовая программа для модуля Mortex.hex, Mortex-devi.hex,
- Загрузчик программ модуля BSL.exe, boot,
- Файл конфигурации узла сети CAN sm13can.

Для инсталляции программного обеспечения достаточно запустить с дистрибутивного диска программу mK_install.exe и, указав каталог для установки тестового программного обеспечения и тип операционной системы (Windows 95/98 или Windows NT 4.0), нажать на кнопку “Приступить к установке”.

При работе в системе Windows NT 4.0 для завершения процесса установки необходимо зарегистрировать драйвер платы контроллера CAN посредством файла конфигурации setup_nt.inf.

3.4. Подключение модуля к блоку питания, ПЭВМ, испытательному стенду и имитатору произвести в соответствии со схемой электрической подключений (Приложение 1).

Питание модуля производится от сети напряжением 220В.

3.5. Перед подачей питания произведите внешний осмотр модуля **МК-167-IO-1** и убедитесь в отсутствии механических повреждений, пыли, грязи и посторонних предметов; проверьте надежность присоединения кабелей к разъемам.

4. Работа с модулем МК-167-IO-1

4.1. Для работы с модулем следует:

а) Запустить загрузчик программ модуля BSL.exe.

Программа BSL обеспечивает загрузку тестовых программ в оперативную память центрального процессора модуля и их запуск.

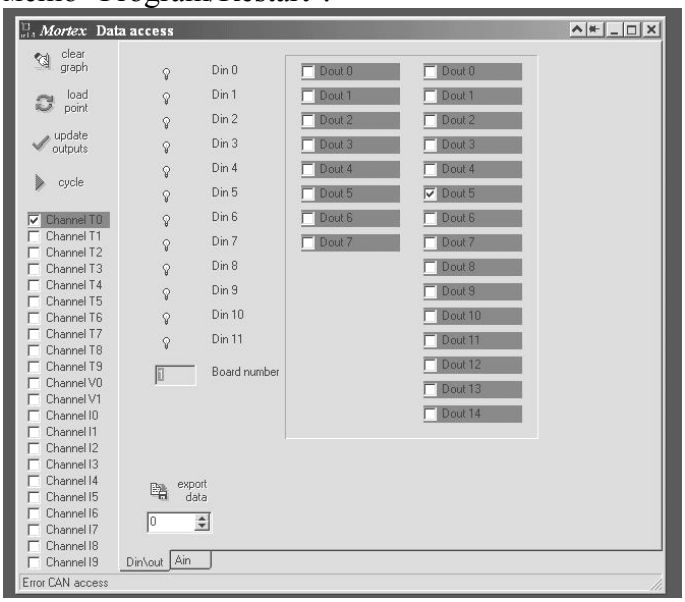
б) С помощью меню "Файл" программы BSL исполнить операцию "Загрузить" задав в появившемся окне диалога имя файла Mortex.hex. После появления сообщения об успешном запуске тестовой программы модуля завершить выполнение программы BSL.

в) Запустить тестовую программу для PC _Mt4H.exe.

Программа _Mt4H обеспечивает:

- Управление платой контроллера CAN mK_CAN,
- Конфигурирование узла сети CAN,
- Дистанционное управление интеллектуальным модулем в ручном режиме,
- Сбор и визуализацию данных от модуля по выбранным каналам,
- Диагностику работоспособности программно-аппаратного комплекса.

г) При необходимости установить на панели “PC virtual” значения “CAN_IRQ” и "CAN_Port" сответствующие выбранным на плате mK_CAN с помощью перемычек и переключателей, а также значение **“Operating board”,** соответствующее выбранному с помощью DIP-перекючателей на плате модуля. Значение “CAN_Spd” для проведения тестовых процедур должно быть равно 250. После внесения изменений в параметрах следует перезапустить программу одновременным нажатием клавиш Ctrl+Alt+Space или через пункт меню "Program/Restart".

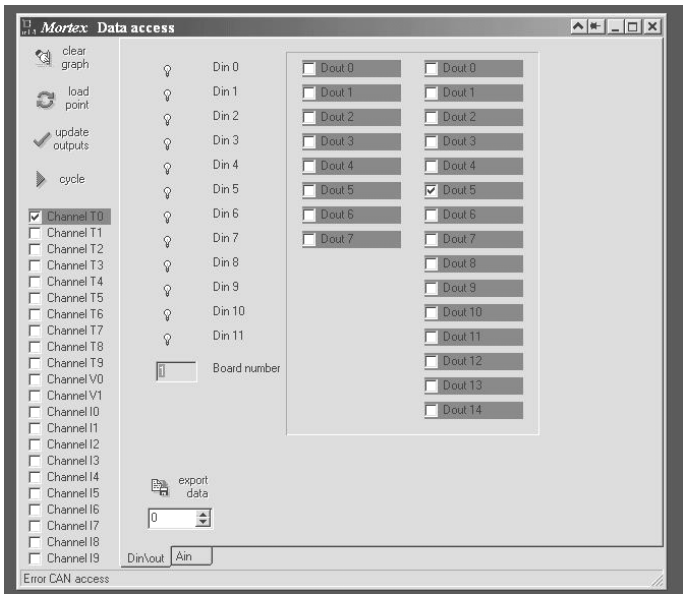


д) После запуска (перезапуска) программы на панели "Debug output" должно появиться сообщение о установлении связи с модулем. В случае сообщения об ошибке следует в первую очередь проверить питание модуля, надёжность кабельных соединений и правильную установку джамперов, переключателей на модуле и плате mK_CAN и параметров программы, убедиться в успешной загрузке тестовой программы в модуль.

е) Для работы с модулем следует перейти в окно "Mortex Data access" нажатием на кнопку "Mortex Direct". В окне доступны две закладки: "Din/out", для визуализации состояния дискретных входов модуля и управления дискретными выходами модуля, а также "Ain" для визуализации на графике любой комбинации аналоговых входов в виде целого числа со знаком.

ж) В левой части окна "Ain" расположены следующие органы управления:

- "clear graph" - для очистки графика аналоговых входов;
- "load point" - для обновления информации о состоянии входов модуля;
- "update outputs" - для обновления состояния дискретных выходов модуля;
- "cycle" - для циклического обновления информации 1 раз в секунду;



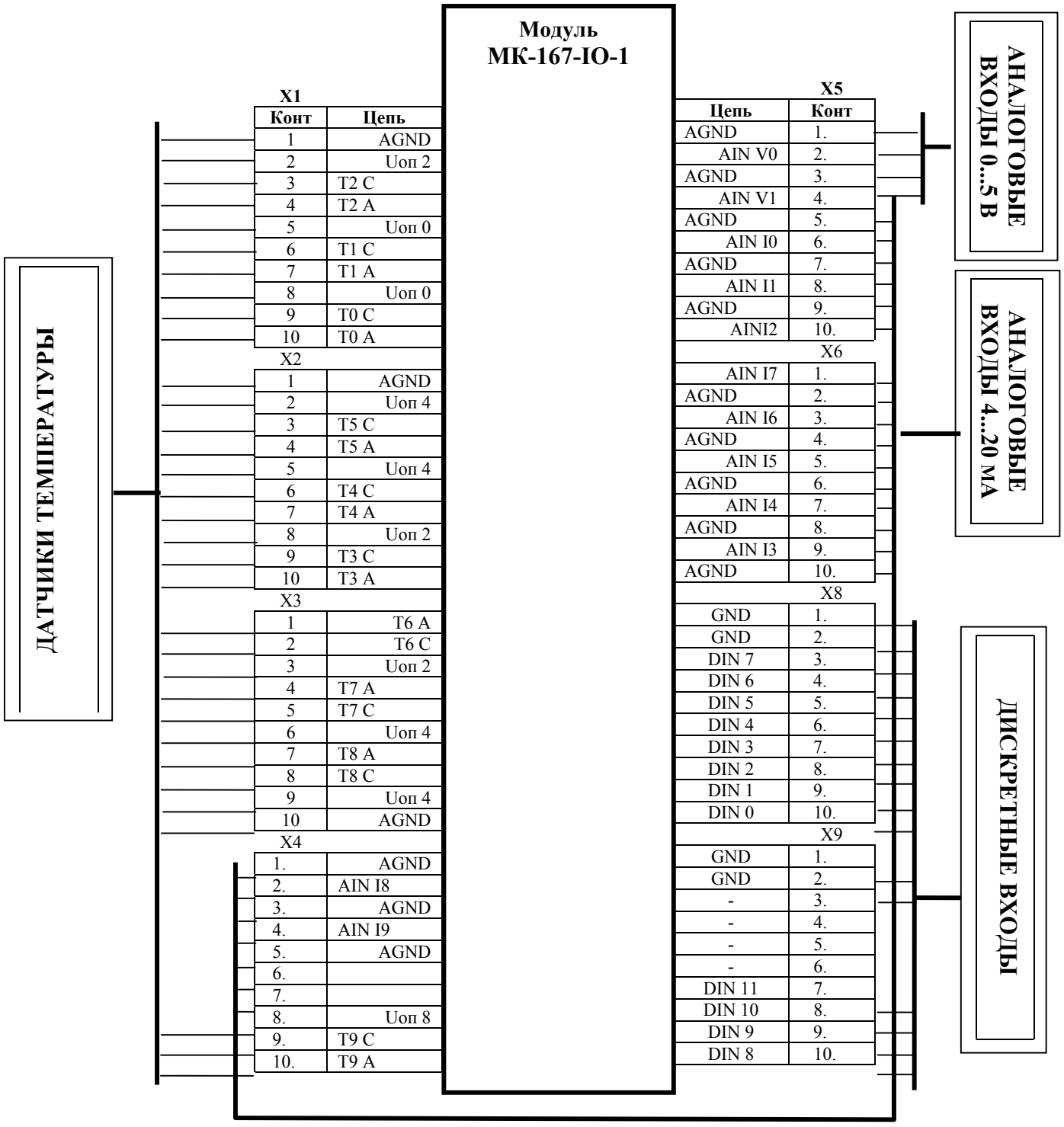
- "channel ..." .. "channel ..." - для управления режимом отображения соответствующих аналоговых входов.

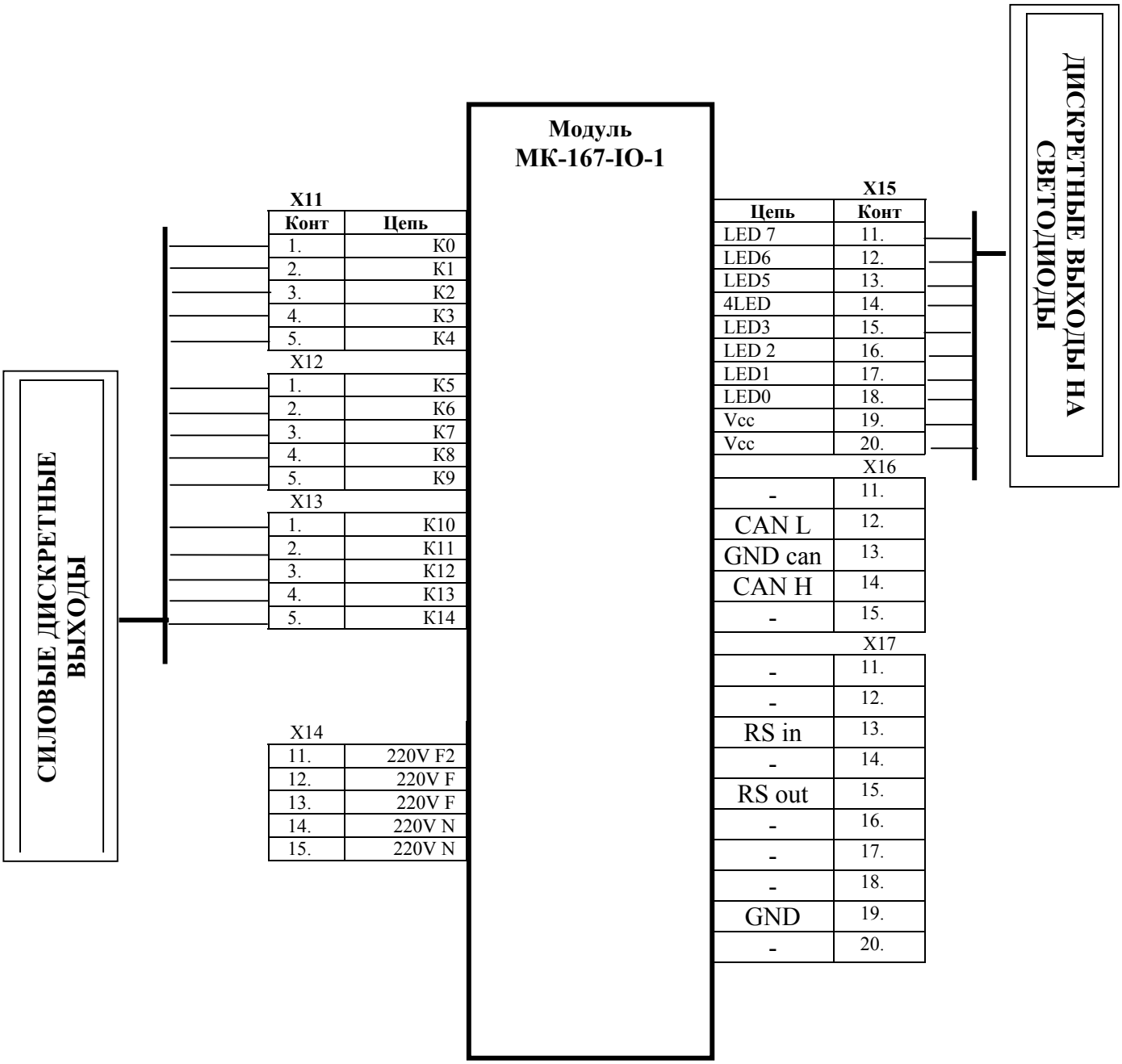
4.2. В процессе аналого-цифрового преобразования измеренная величина входного сигнала преобразуется в код. Максимальное значение кода по любому каналу измерений составляет 2048, минимальное значение – 2047.

Для осуществления работы с сигналами, представленными в виде физических величин напряжения или тока следует произвести тарировку каждого канала модуля в соответствии с той физической величиной, для измерения которой он предназначен.

Приложение 1

Схема электрическая подключений

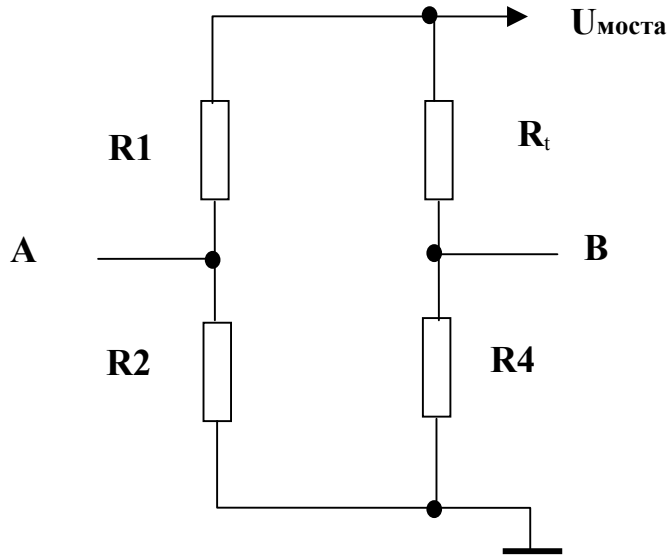



Модуль
МК-167-IO-1
СИЛОВЫЕ ДИСКРЕТНЫЕ ВЫХОДЫ
ДИСКРЕТНЫЕ ВЫХОДЫ НА СВЕТОДИОДЫ
X11
Конт | Цепь
1. K0
2. K1
3. K2
4. K3
5. K4
X12
1. K5
2. K6
3. K7
4. K8
5. K9
X13
1. K10
2. K11
3. K12
4. K13
5. K14
X14
11. 220V F2
12. 220V F
13. 220V F
14. 220V N
15. 220V N
X15
Цепь | Конт
LED 7 11.
LED6 12.
LED5 13.
4LED 14.
LED3 15.
LED 2 16.
LED1 17.
LED0 18.
Vcc 19.
Vcc 20.
X16
- 11.
CAN L 12.
GND can 13.
CAN H 14.
- 15.
X17
- 11.
- 12.
RS in 13.
- 14.
RS out 15.
- 16.
- 17.
- 18.
GND 19.
- 20.

Приложение 2

Методика определения коэффициентов при тарировке каналов измерения температуры

Измерительная схема канала измерения температуры представляет собой мост сопротивлений, в одно из плеч которого включен измерительный элемент (термометр сопротивлений R_t).



При этом

$$R_t = \frac{K1 \bullet R4}{K0 - CODE} - R4$$

где
К0 и **К1** - коэффициенты пропорциональности,
$\mathbf{R_t}$ - сопротивление датчика,
R4 - сопротивление плеча моста,
CODE - значение кода АЦП.

Для определения тарировочных коэфициентов каждого канала основное уравнение преобразуется к виду, линейному по параметрам:

$$(K0-K1)\ R4 + R_t \bullet K0 - R4 \bullet CODE = R_t \bullet CODE$$

Для нахождения трех неизвестных коэффициентов необходимо решить систему уравнений, получаемых по трем измерениям для каждого канала.

№ канала	№ точки	Заданное значение R_t	Измеряемый код АЦП (CODE)
1	1		
	2		
	3		
2	1		
	2		
	3		
...	1		
	2		
	3		
10	1		
	2		
	3		

Пусть $\lambda = (K0-K1)\ R4$,
тогда для трех точек тарировочной характеристики система уравнений примет вид:

$$\begin{cases} \lambda + R_{t1} \cdot K0 - R4 \cdot CODE_1 = R_{t1} \cdot CODE_1 \\ \lambda + R_{t2} \cdot K0 - R4 \cdot CODE_2 = R_{t2} \cdot CODE_2 \\ \lambda + R_{t3} \cdot K0 - R4 \cdot CODE_3 = R_{t3} \cdot CODE_3 \end{cases}$$

Решая систему уравнений для каждого входного канала, необходимо определить значения **K0, K1 и R4 для трех точек тарировочной характеристики канала в виде массива:**

№ канала	**K0**	**K1**	**R4** (в сотых долях Ом)
1			
2			
...			
10			

Приложение 3.

Примеры программного обеспечения работы с модулем

Все описываемые процедуры написаны и проверены при помощи пакета KEIL MicroVision 2.0.

1). Работа с аналоговыми каналами измерения температуры.

```
#include <reg167.h>
#include <intrins.h>

sbit START_CONVERTION=P3^12;                    //бит начала преобразования

//процедура конфигурации SSC
void setup_ssc(void){
SSCCON=0x0;
_bfld_(DP3,0x2300,0x2200);
P3|=0x2300;
SSCCON=0xC03F;                                  //конфигурация SSCCON (16 бит данные в режиме мастера
SSCBR=0x0001;                                   //конфигурация скорости передачи
}

//процедура конфигурация таймера для организации задержки
void setup_t3(void){
T3CON=0x0041;
}

void setup_ports(void){
DP2|=0xF000;                                    //конфигурация адресов мультиплексоров
DP3|=0x1000;                                    //конфигурация бита начала конверсии
}

//процедура съема данных с десяти темпераратурных каналов
//параметр buf – массив для записи конечных данных
void read_chanals(unsigned int *buf){
int i,chanal;
for (i=0;i<10;i++)                              //цикл для десяти каналов
        {
        if (i<7) chanal=i;                      //выбор канала
                else chanal=i+1;
        P2&=~0xF000;                                    //сброс адресов на мультиплексорах
        P2|=_irol_(chanal,12);                          //установка адресов на мультиплексорах
                                        //(выбор канала)
        START_CONVERTION=0;                             //сигнал АЦП начать конверсию
        T3=0xFFF4;                                              //установка таймера для организации //задержки
        while(T3);                                      //задержка 10мксек
        START_CONVERTION=1;                             //сброс сигнала начала конверсии
        SSCTB=0x0;                                              //инициирование обмена по SSC
        while (SSCBSY);                                 //ожидане окончания обмена
        buf[i]=SSCRB;                                   //сохранение результата
        if (SSCRB & 0x0800) buf[i]|=0xF000;     //размножение знака
                        else buf[i]&=~0xF000;
        }                                                       //конец цикла
}

unsigned int b[10];                             //массив для буфера
```

```
//программа сбора данных
void main(void){
setup_ssc();                                    //конфигурирование SSC
setup_t3();                                     //конфигурирование таймера задержки
setup_ports();                                  //конфигурирование портов
while (1){                                      //бесконечный цикл сбора данных
        read_chanals(b);                        //сбор данных
        }
}
```

2) *Работа с аналоговыми каналами* **измерения вибрации и давления**.

```
#include <reg167.h>

//прцедура конфигурации АЦП
void setup_adc(void){
ADCON=0x0;
}

//процедура сбора данных с двенадцати каналов
void read_chanals(unsigned int *buf){
int i;
for (i=0;i<12;i++){                             //цикл сбора данных
        while (ADBSY);                          //ожидание освобождения АЦП
        ADCON&=~0x000F;                         //сброс номера канала
        ADCON|=i;                               //выбор канала
        ADST=1;                                 //сигнал начала конверсии
        while (ADBSY);                          //ожидание конца преобразования
        buf[i]=(ADDAT & 0x03FF);                //сохранение результата
        }
}

unsigned int b[12];                             //массив для буфера

//программа сбора данных
void main(void){
setup_adc();                                    //конфигурация АЦП
while(1){                                       //бесконечный цикл сбора данных
        read_chanals(b);                        //сбор данных
        }
}
```

3). Работа с CAN.

В примере создается и конфигурируется два объекта-сообщения настроенные на прием и соответственно передачу, объекты-сообщения работают в режиме эха, т. е. одно дожидается приема данных, а второе тут же их отсылает.

```
#include <reg167.h>

#define BIT_TIMING          0x3443    //скорость передачи 250000бит/сек

#define ARBITR(id) ((unsigned long)(id) >> 21 & 0x000000ff | \
            (unsigned long)(id) >>  5 & 0x0000ff00 | \
            (unsigned long)(id) << 11 & 0x00ff0000 | \
            (unsigned long)(id) << 27)

#define MSG_CFG(dlc,dir,xtd) ((dlc) << 4 | (dir) << 3 | (xtd) << 2)
```

```
#define CAN_MSG_ID_R (0x008ul << 18)//уникальный идентификатор для принимающего
                                                                    //объекта
#define CAN_MSG_ID_T (0x009ul << 18) //уникальный идентификатор для передающего
                                                                        //объекта

#define LEN_CAN_TIME_MSG 8                          //длна сообщений 8 байт

//биты младшей части control/status регистра
#define CAN_INIT_  1
#define CAN_IE_    2
#define CAN_SIE_   4
#define CAN_EIE_   8
#define CAN_CCE_  64
// биты младшей части control/status регистра
#define CAN_LEC_    7
#define CAN_TXOK_   8
#define CAN_RXOK_  16
#define CAN_EWRN_  64
#define CAN_BOFF_ 128

// Control/Status Register
#define CAN_CTL_STAT   (*((unsigned int volatile sdata *) 0xEF00))
// Управляющая часть Control/Status Register
#define CAN_CTL       (*((unsigned char volatile sdata *) 0xEF00))
/* Статусная часть Control/Status Register
#define CAN_STAT      (*((unsigned char volatile sdata *) 0xEF01))

// Interrupt Register
#define CAN_INTID     (*((unsigned char volatile sdata *) 0xEF02))

// Bit Timing Register
#define CAN_BIT_TIMING (*((unsigned int volatile sdata *) 0xEF04))

// Global Mask Short register
#define CAN_MASK_SHORT (*((unsigned int volatile sdata *) 0xEF06))
// Upper Global Mask Long register
#define CAN_UMASK_LONG (*((unsigned int volatile sdata *) 0xEF08))
// Lower Global Mask Long register
#define CAN_LMASK_LONG (*((unsigned int volatile sdata *) 0xEF0A))

//
#define INTPND_              0x0002u
#define RXIE_                0x0008u
#define TXIE_                0x0020u
#define MSGVAL_              0x0080u
#define NEWDAT_              0x0200u
#define MSGLST_              0x0800u
#define CPUUPD_              0x0800u
#define TXRQ_                0x2000u
#define RMTPND_              0x8000u

//маска для очстки флагов в поле msg_ctl в любом из обьектов
#define INTPND_CLR           (~INTPND_)
#define RXIE_CLR             (~RXIE_)
#define TXIE_CLR             (~TXIE_)
#define MSGVAL_CLR           (~MSGVAL_)
#define NEWDAT_CLR           (~NEWDAT_)
```

```
#define MSGLST_CLR            (~MSGLST_)
#define CPUUPD_CLR            (~CPUUPD_)
#define TXRQ_CLR              (~TXRQ_)
#define RMTPND_CLR            (~RMTPND_)
//маска для очстки флагов в поле msg_ctl в любом из обьектов
#define INTPND_SET            (~(INTPND_ >> 1))
#define RXIE_SET              (~(RXIE_ >> 1))
#define TXIE_SET              (~(TXIE_ >> 1))
#define MSGVAL_SET            (~(MSGVAL_ >> 1))
#define NEWDAT_SET            (~(NEWDAT_ >> 1))
#define MSGLST_SET            (~(MSGLST_ >> 1))
#define CPUUPD_SET            (~(CPUUPD_ >> 1))
#define TXRQ_SET              (~(TXRQ_ >> 1))
#define RMTPND_SET            (~(RMTPND_ >> 1))

//маска поле msg_ctl в любом из обьектов
#define XTD_MASK              0x0004
#define DIR_MASK              0x0008

//структура для CAN обьекта
struct can_obj {
  unsigned int  msg_ctl;   // Message Control
  unsigned long arbitr;    // Arbitration
  unsigned char msg_cfg;   // Message Configuration
  unsigned char msg[8];    // Message Data 0 .. 7
  unsigned char dummy;     // Reserved Byte
};

//настройка структуры на нужный адрес
#define CAN_MSGOBJ ((struct can_obj volatile sdata *) 0xEF00)

//флаги натравлений
#define CANDIR_RECEIVE 0
#define CANDIR_TRANSMIT 1

// используемые обьекты
#define OBJ_T 1                       //для передачи используем обьект номер 1
#define OBJ_R 2                       //для приема используем обьект номер 2


//конфигурация CAN
void setup_can(void)
{
// очистка всех объектов перед началом работы
   int object_number;
   CAN_CTL_STAT = CAN_INIT_ | CAN_CCE_;
   CAN_BIT_TIMING = BIT_TIMING;
   for (object_number = 1; object_number <= 15; object_number++) {
      CAN_MSGOBJ[object_number].msg_ctl = MSGVAL_CLR;
                        }
//настройка масок
   CAN_MASK_SHORT = 0xffff;
   CAN_UMASK_LONG = 0xffff;
   CAN_LMASK_LONG = 0xf8ff;

//конфигурирование принимающего объекта
   CAN_MSGOBJ[OBJ_R].msg_ctl = MSGVAL_CLR;
```

```
  CAN_MSGOBJ[OBJ_R].arbitr = ARBITR(CAN_MSG_ID_R);
  CAN_MSGOBJ[OBJ_R].msg_cfg = MSG_CFG(LEN_CAN_TIME_MSG, CANDIR_RECEIVE, 0);
  CAN_MSGOBJ[OBJ_R].msg_ctl =INTPND_CLR & RXIE_CLR & TXIE_CLR & MSGVAL_SET &
      NEWDAT_CLR & MSGLST_CLR & TXRQ_CLR & RMTPND_CLR;

          //конфигурирование передающего объекта
  CAN_MSGOBJ[OBJ_T].msg_ctl = MSGVAL_CLR;
  CAN_MSGOBJ[OBJ_T].arbitr = ARBITR(CAN_MSG_ID_T);
  CAN_MSGOBJ[OBJ_T].msg_cfg = MSG_CFG(LEN_CAN_TIME_MSG, CANDIR_TRANSMIT, 0);
  CAN_MSGOBJ[OBJ_T].msg_ctl =INTPND_CLR & RXIE_CLR & TXIE_CLR & MSGVAL_SET &
      NEWDAT_CLR & CPUUPD_SET & TXRQ_CLR & RMTPND_CLR;

  CAN_CTL_STAT = CAN_IE_ | CAN_EIE_;          //разрешение работы CAN
}


//процедура ожидания прихода сообщения
//msg – буфер для входящего сообщения
void wait_message(unsigned char *msg){
unsigned char i;
CAN_MSGOBJ[OBJ_R].msg_ctl=NEWDAT_CLR;       //сброс флага прихода сообщения
while (!(CAN_MSGOBJ[OBJ_R].msg_ctl & NEWDAT_)); //ожидание прихода сообщения
for (i=0;i<LEN_CAN_TIME_MSG;i++)
          msg[i]=CAN_MSGOBJ[OBJ_R].msg[i];          //копирование данных из объекта-
                                                    //          сообщения в буфер данных
}


//процедура передачи сообщения
//msg – буфер для исходящего сообщения
void transmit_message(unsigned char *msg){
unsigned char i;
CAN_MSGOBJ[OBJ_T].msg_ctl=NEWDAT_SET & CPUUPD_SET; //установка флагов данных и
                                                    //изменения данных
for (i=0;i<LEN_CAN_TIME_MSG;i++)                    //копирование данных из буфера
          CAN_MSGOBJ[OBJ_T].msg[i]=msg[i];  //данных в объект-сообщение
CAN_MSGOBJ[OBJ_T].msg_ctl = CPUUPD_CLR;     //сброс флага изменения данных
CAN_MSGOBJ[OBJ_T].msg_ctl =TXRQ_SET;                //установка флага запроса на //передачу
}

//процедура конфигурирования портов
void setup_ports(void){
DP0H|=0x01; P0H&=0xFE;                      //разрешение CAN
}

unsigned char b[8];                                 //массив для приема и передачи данных

//прграмма принимает данные в один объект и отправляет их через другой
void main(void){
setup_ports();                              //конфигурирование портов
setup_can();                                //конфигурирование CAN
while(1){                                   //бесконечный цикл
wait_message(b);                            //ожидание сообщения
transmit_message(b);                        //отправление полученного сообщения
}
}
```

Габаритный чертеж