МИНИСТЕРСТВО ПУТЕЙ СООБЩЕНИЯ РФ
Департамент кадров и учебных заведений

САМАРСКИЙ ИНСТИТУТ ИНЖЕНЕРОВ ЖЕЛЕНОДОРОЖНОГО ТРАНСПОРТА

Кафедра “Телекоммуникации на железнодорожном транспорте”

УСТРОЙСТВА ВВОДА-ВЫВОДА АНАЛОГОВЫХ И ДИСКРЕТНЫХ СИГНАЛОВ ДЛЯ КОМПЬЮТЕРНЫХ СИСТЕМ

Методические указания для студентов специальностей
“Информационные системы и технологии” и
“Автоматика, телемеханика и связь на железнодорожном транспорте”

Составители В.А. Засов
Н.Н. Савченков

Самара – 2002

УДК 681.327
З……

Устройства ввода-вывода аналоговых и дискретных сигналов для компьютерных систем. Методические указания для студентов специальностей “Информационные системы и технологии” и “Автоматика, телемеханика и связь на железнодорожном транспорте”. – Самара, СамИИТ, 2002. - __с.

Утверждено на заседании кафедры “Телекоммуникации на железнодорожном транспорте” 18 марта 2002 г., протокол № 7.

Печатается по решению редакционно-издательского совета Самарского института инженеров железнодорожного транспорта.

В методических указаниях рассматриваются функции, характеристики и структурные схемы устройств ввода-вывода аналоговых и дискретных сигналов для компьютерных систем управления. Описывается программный интерфейс и библиотека функций, на основе которой удобно производить разработку драйверов устройств. Для многих конкретных задач приведены с подробными комментариями примеры программных драйверов.

Предназначены для студентов специальностей “Информационные системы и технологии” и “Автоматика, телемеханика и связь на железнодорожном транспорте” для выполнения практических и лабораторных работ, курсовых и дипломных проектов. Методические указания могут быть также полезны студентам других специальностей, изучающим компьютерные информационно-управляющие системы.

Составители Валерий Анатольевич Засов
Николай Николаевич Савченков

Рецензенты: к.т.н., доцент В.И. Качур
(НПЦ “Информационные и транспортные системы”)
к.т.н., доцент кафедры АТС Н.Е. Федоров
(СамИИТ)

Подписано в печать
Тираж Заказ №

СОДЕРЖАНИЕ

Стр.

1. Функции и основные характеристики устройств ввода-вывода сигналов (УВВС) для компьютерных систем управления**……**
   - **1.1. Функции и классификация УВВС……..……..**
   - **1.2. Основные характеристики и выбор УВВС…….…..**

2. Способы подключения устройств ввода-вывода сигналов к компьютерным системам**……..……..……**
   - **2.1. Компьютерные шины, используемые для подключения УВВС……**
   - **2.2. Подключение централизованных УВВС.....….....…….....**
   - **2.3. Подключение распределенных УВВС.....…….....**

3. Структурная схема типовой платы централизованного устройства ввода-вывода сигналов**.....…….....**

4. Организация программного интерфейса устройств ввода-вывода сигналов**.....…….....**
   - **4.1. Уровни управления платами УВВС.....…….....**
   - **4.2. Форматы данных и модели памяти.....…….....**

5. Описание библиотеки функций и примеры ее использования **.....…….....**
   - **5.1. Функции конфигурации.....…….....…...**
   - **5.2. Функции задания временных параметров АЦП..…….....….....**
   - **5.3. Функции ввода по аналого-цифровым каналам..…….....….....**
   - **5.4. Функции вывода по цифро-аналоговым каналам..…….....….....**
   - **5.5. Функции ввода-вывода по цифровым каналам.....…….....**
   - **5.6. Функции ввода по аналого-цифровым каналам с использованием прерываний.....…….....….....**
   - **5.7. Функции ввода-вывода по аналого-цифровым каналам с использованием прямого доступа к памяти.....….....…….....**

6. Практические и лабораторные работы по вводу-выводу аналоговых и дискретных сигналов с примерами программ**.....….....……....**
   - **6.1. Изучение интерфейса и основных возможностей программы “Осциллоскоп”.....….....……....**
   - **6.2. Организация ввода-вывода аналоговых сигналов в программном режиме……**
   - **6.3. Организация ввода-вывода аналоговых сигналов использованием прерываний…….**
   - **6.4. Организация ввода-вывода аналоговых сигналов в режиме прямого доступа к памяти……**
   - **6.5. Организация ввода-вывода дискретных сигналов**

Библиографический список**…….……**

ФУНКЦИИ И ОСНОВНЫЕ ХАРАКТЕРИСТИКИ УСТРОЙСТВ ВВОДА-ВЫВОДА СИГНАЛОВ (УВВС) ДЛЯ КОМПЬЮТЕРНЫХ СИСТЕМ УПРАВЛЕНИЯ

1.1. Функции и классификация УВВС

1.1.1. Компьютерные системы различного назначения - управляющие технологическими процессами, информационно-измерительные, сбора данных, диагностические - подключаются к объектам автоматизации с помощью *устройств связи с объектами (УСО)*. В состав УСО входят датчики, исполнительные устройства, нормирующие преобразователи и преобразователи формы информации, устройства коммутации сигналов и управления нагрузками, а также интерфейсные схемы для сопряжения с шинами компьютера.

Датчики преобразуют с заданной точностью физические величины или параметры объектов (например, температуру, давление, ускорение, скорость, перемещение и др.) в электрические сигналы напряжения или тока, которые могут быть аналоговыми, дискретными, импульсными. Далее нормирующими преобразователями эти сигналы усиливаются (приводятся к стандартным диапазонам), фильтруются, гальванически изолируются, линеаризуются. Устройства коммутации выполняют мультиплексирование входных сигналов. Преобразователи формы информации осуществляют аналого-цифровое и цифро-аналоговое преобразование сигналов, преобразование частоты в цифровой код и обратно и т.п. С помощью интерфейсных схем производится ввод или вывод цифровых сигналов по принятым протоколам через ту или иную шины компьютера. Устройства управления нагрузками - реле, транзисторные усилители и ключи - формируют выходные сигналы заданной мощности для исполнительных устройств, управляющих объектами и процессами.

Датчики (первичные преобразователи) и исполнительные устройства находятся непосредственно на объекте. Нормирующие преобразователи и преобразователи формы информации, устройства коммутации сигналов и управления нагрузками, а также интерфейсные схемы интегрируются на специальных электронных модулях (платах), которые называются *устройствами ввода-вывода сигналов (УВВС)*. Нередко с целью обеспечения гибкости и возможности работы УВВС с различными датчиками и исполнительными устройствами нормирующие преобразователи и устройства управления нагрузками реализуют в виде отдельных самостоятельных модулей (модулей нормализации и управления нагрузками), конструктивно вынесенных за пределы УВВС. Это позволяет, изменяя номенклатуру модулей нормализации и управления нагрузками, использовать одно и тоже УВВС для решения разных задач управления и контроля.

Таким образом, с функциональной точки зрения в УСО можно выделить три компоненты:

1 датчики и исполнительные устройства (Д и ИУ);

2 модули нормализации и управления нагрузками (Н и УН);

3 устройства ввода-вывода сигналов (УВВС).

Конструкции датчиков и исполнительных устройств различного назначения, принципы работы и схемы модулей нормализации и управления нагрузками подробно рассмотрены в многочисленных источниках, например, /1,2,3,4/. Поэтому в методических указаниях основное внимание уделено изучению технических и программных средств УВВС, методам подключения этих устройств к компьютеру и практическому применению для решения задач железнодорожного транспорта.

УВВС является системообразующей компонентой УСО, обеспечивающей информационное взаимодействие датчиков, исполнительных устройств и компьютера. Структурная схема и алгоритм работы УВВС определяют структуру и алгоритм работы УСО.

1.1.2. В зависимости от топологии структурной схемы все УВВС (и соответственно все УСО) разделяются на две группы - *централизованные УВВС и распределенные УВВС*
.

Централизованные УВВС используются для создания несложных автономных компьютерных систем управления небольшими объектами, поэтому такие УВВС иногда называют локальными. Структурная схема централизованных УСО и УВВС изображена на рис.1.1. В таких устройствах каждый датчик или

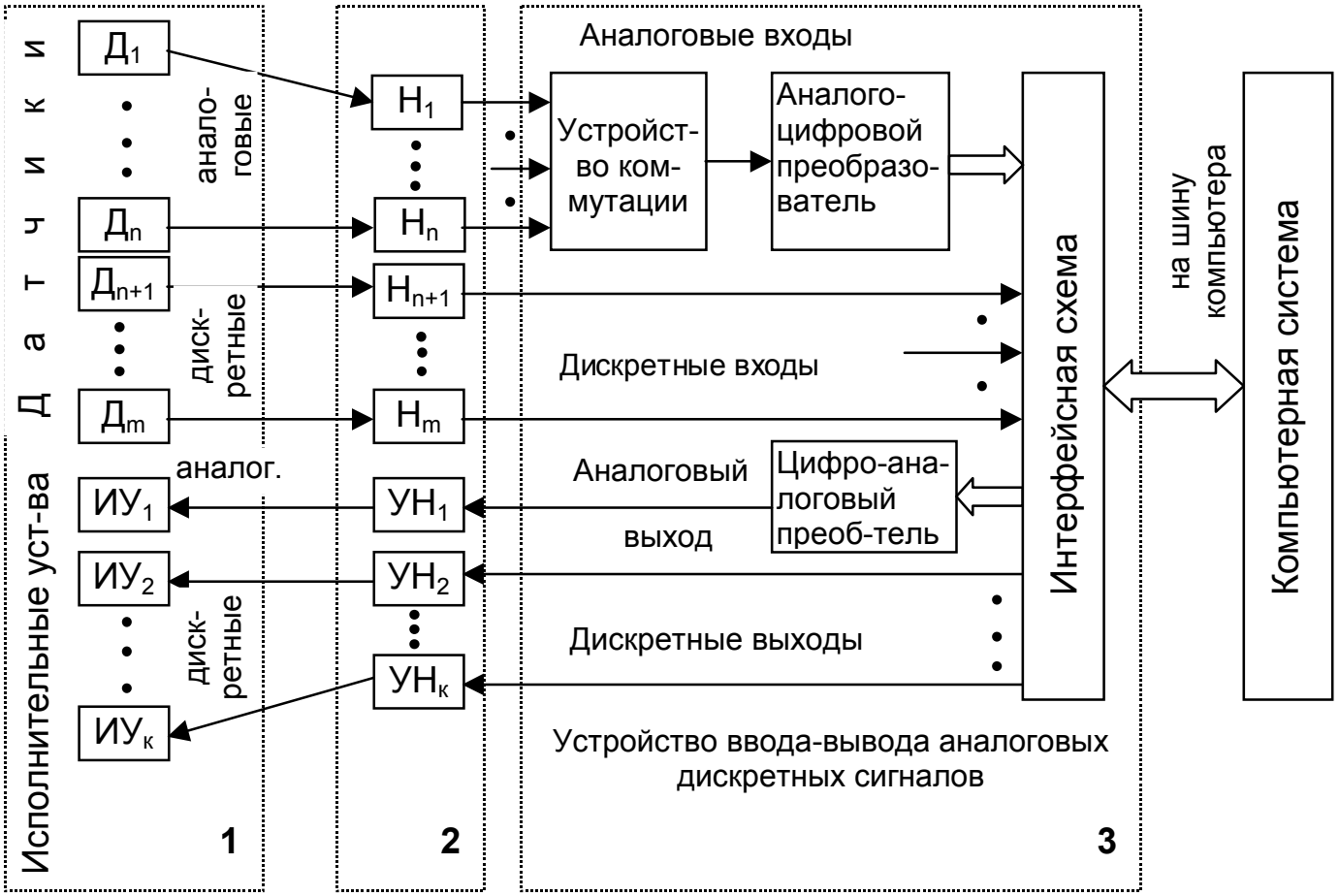


Рис.1.1. Структурная схема централизованных УСО и УВВС

исполнительное устройство подключаются к УВВС компьютерной системы с помощью индивидуальной линии связи, т.е. по радиальной структурной схеме. Учитывая высокие стоимости медных линий связи и работ по их прокладке, применение централизованных УВВС для сопряжения с протяженными и сложными объектами экономически нецелесообразно. Централизованные УВВС, как правило, подключаются с помощью шинной интерфейсной схемы к системной шине расширения (либо к одной из внешних шин) компьютера, поэтому эти УВВС размещают в корпусе компьютера или рядом с ним. Обычно управление работой УВВС этой группы производится компьютером, поэтому наличие собственных процессоров у этих УВВС не является обязательным, хотя современные УВВС нередко содержат встроенные цифровые процессоры сигналов (ЦПС) или программируемые логические интегральные схемы (ПЛИС).

Распределенные УВВС используются для создания компьютерных систем

управления сложными и протяженными (сотни – тысячи метров) в пространстве объектами. Например, на железнодорожном транспорте это системы управления сортировочными горками, диспетчерские центры управления движением проездов, энергоснабжением и т.п. Распределенное УВВС, структурная схема которого изображена на рис.1.2., реализуется на основе локальных УВВС, объединяемых специальными промышленными коммуникационными сетями – полевыми шинами (Fieldbus). Сети Fieldbus являются разновидностью

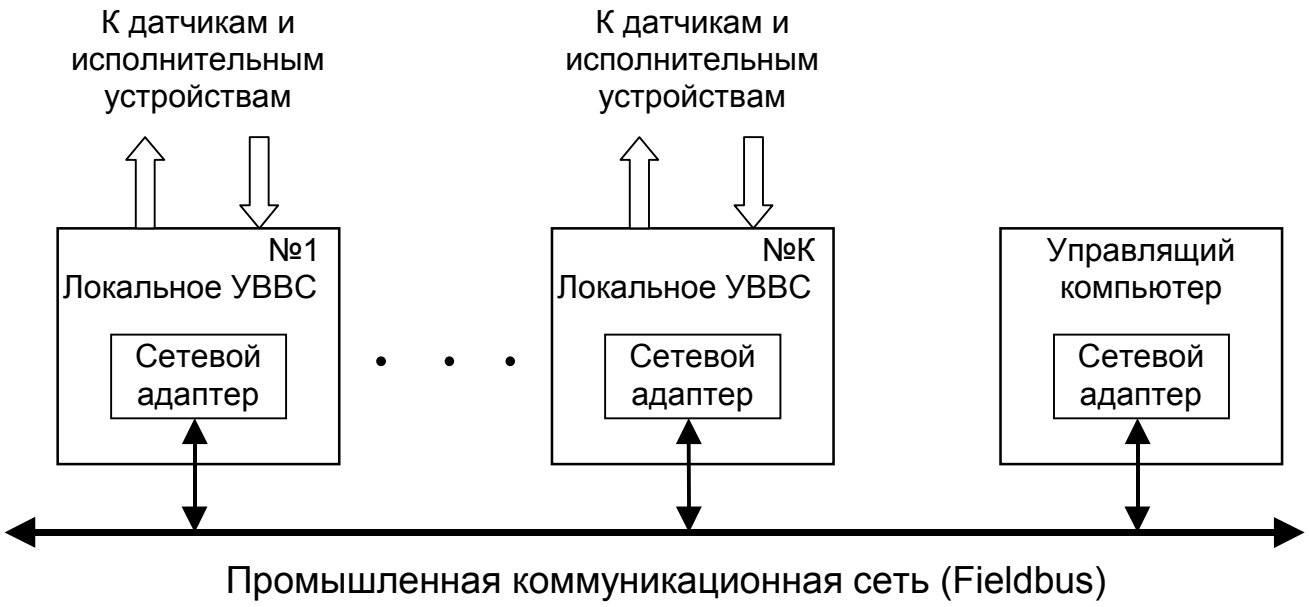


Рис.1.2. Структурная схема распределенного УВВС

коммуникационных сетей общего назначения, предназначенной для работы в промышленных условиях. Их отличительными особенностями являются высокая помехоустойчивость, предсказуемость времени доставки информации, невысокая стоимость и возможность самовосстановления в случае нештатных ситуаций. Примерами промышленных сетей являются сети Modbus, Profibus, Industrial Ethernet, Interbus, DeviceNet, CANopen, Foundation Fieldbus, LONWorks, ASI /5,6/. Структурные схемы локальных и централизованных УВВС во многом подобны, но имеется и ряд отличий. В локальных УВВС распределенных систем вместо шинных интерфейсных схем применяются сетевые адаптеры, с помощью которых УВВС подключаются к промышленной локальной сети. Такое подключение позволяет расположить локальные УВВС рядом с объектом управления. Каждое из локальных УВВС имеют малое число входов и выходов, к которым подключаются только близко расположенные датчики и исполнительные устройства, что сокращает стоимости линий связи и работ по их прокладке. С другой стороны из-за удаленности локальных УВВС от компьютера, централизованное оперативное управление их работой практически трудно осуществимо. Кроме того каждое из локальных УВВС должно поддерживать тот или иной сетевой протокол обмена, принятый в распределенном УВВС. Поэтому каждое локальное УВВС обязательно содержит встроенный процессор, выполняющий функции управления УВВС и сетевой поддержки. В современных локальных УВВС на встроенный процессор дополнительно возлагают некоторые функции управления объектом, выполняющиеся в жестком реальном времени. Таким образом, современные распределенные УВВС трансформируются в распределенные системы управления, которые по сравнению с централизованными системами являются более производительными, гибкими, легко

масштабируемыми и надежными.

1.2.3. На практике возможны случаи одновременного использования в компьютерной системе УВВС двух рассмотренных групп: близко расположенные датчики и исполнительные устройства подключаются с помощью централизованных УВВС, а удаленные – с помощью распределенных УВВС.

1.2. Основные характеристики и выбор УВВС

1.2.1. **Набор характеристик УВВС можно разделить на следующие группы, каждая из которых описывает те или иные средства УВВС: системные, аппаратные, программные, конструктивные.**

Характеристики системного обеспечения определяют:

- тип УВВС – централизованное или распределенное;
- вид шины компьютера (для централизованных УВВС), к которой подключается устройство;
- вид промышленной сети (для распределенных УВВС), к которой подключается устройство;
- состав специальных системных средств для счета реального времени, парирования сбоев, для горячей замены и т.д;
- возможность работы в режимах прямого доступа к памяти (DMA) или прямого управления шиной (Bus mastering).

Характеристики аппаратного обеспечения определяют:

- тип входных и выходных сигналов (аналоговые, дискретные, частотные и др.) и их параметры (величина амплитуды, мощность, частотный диапазон и т.д.);
- количество входных и выходных каналов;
- тип аналоговых входов (дифференциальный или с общим проводом);
- точность АЦП и ЦАП, определяемая числом разрядов преобразователей и их нелинейностью;
- время АЦ и ЦА преобразования, апертурное время и максимальная частота выборки АЦП;
- схема запуска АЦП (внешним сигналом, внутренним программно управляемым таймером и др.);
- максимальные скорости ввод–вывода сигналов;
- наличие и емкость внутреннего буфера памяти.

Характеристики программного обеспечения определяют:

- тип операционной системы;
- наличие готовых драйверов для УВВС;
- наличие библиотеки готовых функций в поставляемом ПО для проектирования собственных драйверов;
- наличие специальной графической среды для разработки ПО для УВВС.

Общие и конструктивные характеристики определяют:

- требования к электропитанию;
- тип конструкции и размеры;
- условия эксплуатации: диапазон рабочих температур, относительная влажность воздуха, предельные вибрационные и ударные нагрузки;
- степень защиты IP;

- средняя наработка на отказ;
- набор типовых аксессуаров (кабелей, шлейфов, клеммных плат и т.п.), необходимых для подключения УВВС.

1.2.2. Выбор УВВС определяется назначением, областью применения и критериями эффективности (быстродействием, сложностью, стоимостью, энергопотреблением, надежностью и т.д.), обоснованными для создаваемой компьютерной системы управления.

Хотя выбор УВВС в значительной степени определяется спецификой приложений, выделим ряд общих свойств, которые должны иметь УВВС, интегрируемые в современные системы автоматизации.

Это соответствие распространенным стандартам (в том числе стандартам качества ISO9000) и типизация применяемых решений, открытость архитектуры, возможность масштабирования, тиражируемость устройств.

В настоящее время разнообразные и качественные УВВС выпускаются десятками российских /7/(L-Card, Fastwel, НПО МЕРА, ICOS) и зарубежных (Advantech, Adlink, Octagon Systems, PEP Modular Computers, INOVA Computers) производителей, поэтому собственная разработка этих устройств целесообразна только при тщательном технико-экономическом обосновании. Использование производимых в промышленных масштабах УВВС, удовлетворяющим вышеуказанным свойствам, позволяет существенно сократить себестоимость и сроки создания систем, значительно повысить их качество. Поэтому главной инженерной задачей при создании автоматизированных систем является не разработка УВВС, а их правильный выбор и квалифицированное применение.

2. СПОСОБЫ ПОДКЛЮЧЕНИЯ УСТРОЙСТВ ВВОДА-ВЫВОДА СИГНАЛОВ К КОМПЬЮТЕРНЫМ СИСТЕМАМ

2.1. Компьютерные шины, используемые для подключения УВВС

2.1.1. Для интеграции УВВС в компьютерные системы используются различные компьютерные шины (интерфейсы). Способ подключения УВВС определяется выбором того или иного типа шины, который задает протокол обмена, аппаратный интерфейс и конструктивные параметры устройств. Система компьютерных шин приведена на структурной схеме на рис.2.1 /8/. Из многочисленных шин, приведенных на этой схеме, для подключения УВВС обычно применяют *системные шины расширения* и *шины внешних периферийных устройств.*

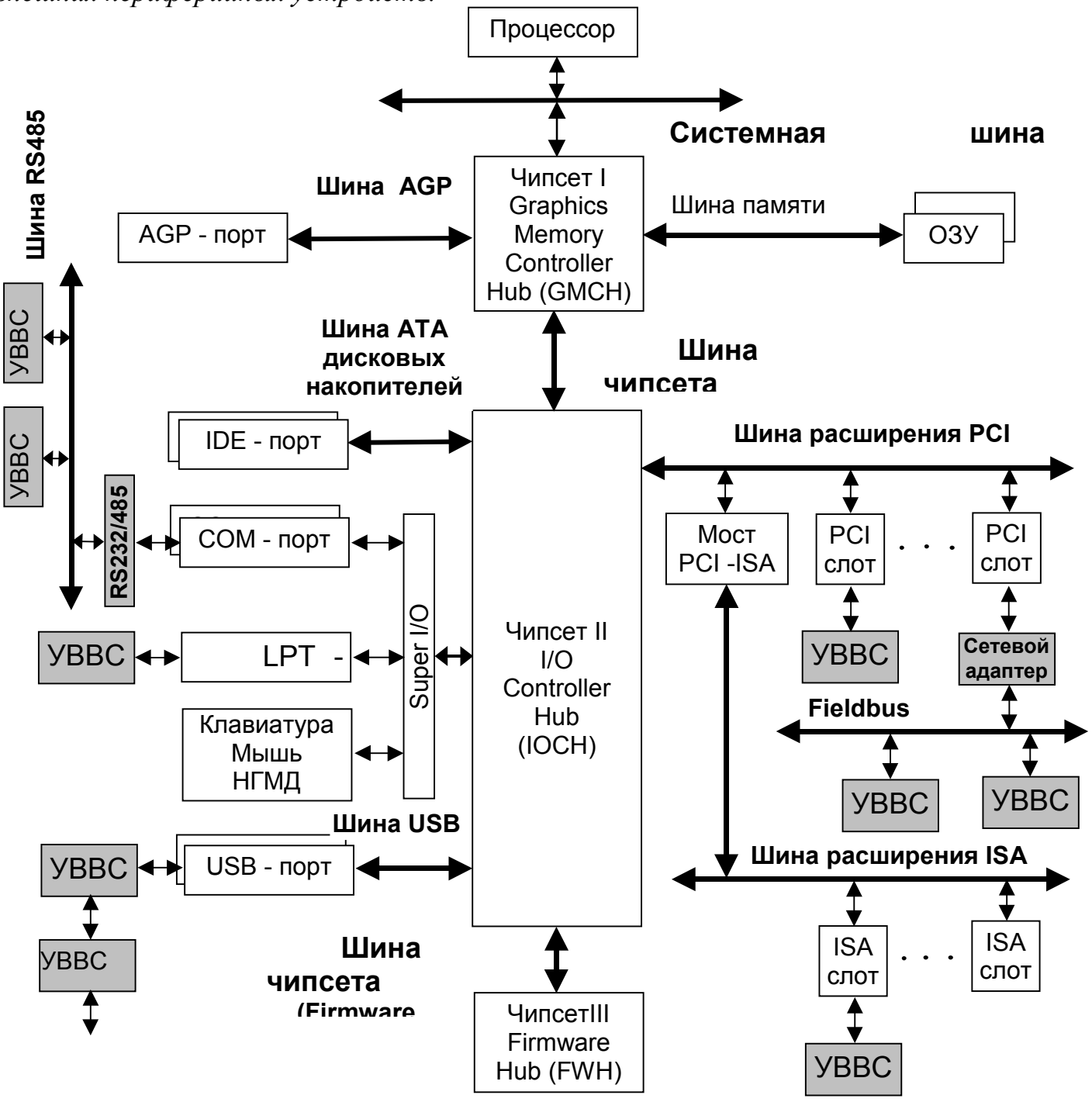


Рис.2.1. Система компьютерных шин и способы подключения УВВС

2.1.2. *Системные шины расширения* предназначены для подключения к системной плате компьютера адаптеров (контроллеров) различных устройств, расширяющих возможности базовой модели компьютера. Системные шины расширения и их слоты (разъемы) размещаются на системной плате компьютера, поэтому дополнительные адаптеры (контроллеры) устанавливаются внутри корпуса компьютера, вовне выводятся только их внешние разъемы. Наиболее распространенными системными шинами расширения являются шины ISA и PCI.

Шина ISA (Industry Standard Architecture) была специально разработана для персональных компьютеров типа IBM PC и применяется с первых моделей таких компьютеров. Это параллельная шина с раздельными линиями адреса и данных. Обмен осуществляется 8- или 16-разрядными данными. Максимальный объем адресуемой памяти – 16 Мбайт (24 адресные линии). Максимальное адресное пространство для устройств ввода-вывода 64 Кбайт (16 адресных линий), хотя все практически выпускаемые адаптеры (платы) расширения используют только 10 адресных линий (1 Кбайт). В распоряжение абонентов шины выделено 11 линий запросов на прерывания и 6 каналов прямого доступа к памяти. Допускается захват и управление шиной другими активными (кроме процессора) устройствами – Bus-Masters. Шина имеет невысокое быстродействие – ее тактовая частота всего 8 МГц, т.е. теоретическая скорость обмена не превышает 16 Мбайт/сек. Конструктивно слот ISA выполнен в виде двух щелевых разъемов с 62 и 36 контактами. В 8-разрядной версии ISA используется только 62 – контактный разъем, а в 16-разрядной версии – оба разъема. Назначение контактов разъемов и размеры типовых плат расширения приведены в многочисленных источниках, например, /9/. В настоящее время область применения шины ISA в офисных компьютерах постоянно сужается вследствие вытеснения ее более современной шиной PCI. Зато в промышленных компьютерах и контроллерах эта дешевая и надежная шина используется по-прежнему широко.

На базе шины ISA разработана более совершенная *шина EISA (Extended ISA).* Конструктивное исполнение EISA обеспечивает совместимость с ней и обычных ISA–адаптеров. Шина EISA имеет 32 линии для передачи данных, 32 адресных линии, что позволяет адресовать до 4 Гбайт памяти. Тактовая частота шины по-прежнему равна 8 МГц, но благодаря специально предусмотренному пакетному режиму передачи предельная скорость обмена достигает 33 Мбайт/сек. Для арбитража выделены особые линии, индивидуальные для каждого разъема, число которых может достигать 8.

Шина PCI (Peripheral Component Interconnect) является наиболее распространенной и популярной системной шиной расширения в современных компьютерах различного назначения. Это синхронная мультиплексированная шина, обеспечивающая передачу 32-разрядных кодов адресов и данных с тактовой частотой 33 МГц. Таким образом, скорость обмена для типовых версий шины (например, версии 2.1.) достигает 132 Мбайт/сек. Новые версии шины PCI допускают тактовую частоту 66 МГц, а формат данных – 64 бит, что позволяет получать скорости обмена 264 и 528 Мбайт/сек. Высокие скорости обмена шины достигаются также благодаря введенному пакетному режиму передачи. Каждый пакет начинается циклом адреса, за которым следует один или несколько циклов данных. Исключение передачи цикла адреса для каждого цикла данных в пакетном режиме позволяет сократить время передачи и, следовательно, увеличить скорость обмена. На одной шине PCI может быть установлено не более четырех слотов: 124-контактных для 32-разрядных данных или 188-контактных для 64-разрядных данных /9/. Для увеличения числа слотов шины PCI или подключения ее к другим шинам применяются мосты шины PCI (PCI Bridge).

2.1.3.В *промышленных компьютерах* и *контроллерах,* работающих в условиях больших перепадов температур (от –40 до +85 град.С), ударов (до 20g), вибраций (до 5g), электромагнитных помех и агрессивной окружающей среды, часто используются *промышленные системные шины расширения*, являющиеся функциональными аналогами шин офисных компьютеров, но имеющие отличия по конструктивным и электрическим параметрам /10/.

Промышленные системные шины расширения позволяют увеличить число подключаемых модулей, имеют штырьковые разъемы, обеспечивающее более жесткую и надежную установку плат (модулей). Конструкция разъемов позволяет производить замену плат без отключения электропитания ("горячую" замену). Промышленные шины более компактны и имеют повышенную помехозащищенность от электрических и магнитных полей. Например, шина ISA-8 имеет промышленный аналог STD-8, шина EISA – STD-32, шина PCI – Compact PCI, а для информационно-измерительных систем – PXI.

2.1.4.Современные высокопроизводительные отказоустойчивые системы управления реального времени с высоким коэффициентом готовности (0,99999) для жестких условий эксплуатации имеют магистрально-модульную архитектуру. Такие системы создаются на основе специальных магистрально-модульных шин, среди которых самой распространенной является шина VMEbus.

Шина VMEbus (VERSAmodule Eurocard) – асинхронная 32 – разрядная шина с раздельными 32 линиями адреса и 32 линиями данных на объединительной панели /11/. В настоящее время разработана 64 – разрядная версия этой шины, в которой линии адреса и данных мультиплексируются. Шина поддерживает скорость передачи до 500 Мбайт/сек в блочном режиме, установку до 21 модуля механического стандарта Евромеханика (3U,6U,9U /10/) и режим "горячей" замены.

2.1.5.Для построения небольших встраиваемых систем используются *мезонинные* /9,12/ или стекируемые шины, к числу которых относятся, например, шины PC/104 и PC/104-Plus, которые является соответственно функциональными аналогами шин ISA-16 и PCI. Благодаря двухстороннему разъему мезонинных шин PC/104 и PC/104-Plus, контроллер собирается в виде стопки из нескольких плат (вторая плата вставляется в шинный разъем первой, третья – в шинный разъем второй т. д.). Платы (мезонинные модули) скрепляются несущими стоечками, и получается простой и удобный конструктив.

Другой способ установки мезонинных модулей предусматривает размещение 2-4 модулей на специальной плате-носителе (кросс-плате), выполненной в стандарте ISA, PCI, CompactPCI, VMEbus. Для этой цели используются малогабаритные мезонинные модули типа IP или PMC /12/.

2.1.6.В *мобильных компьютерах* в качестве системной шины расширения используется шины, регламентируемые стандартом PC Card, который ранее назывался PCMCIA (Personal Computer Memory Card International Association). Эта шина – PC Card - является аналогом шины ISA и предназначена для подключения малогабаритных карт флэш-памяти, модемов и УВВС, выполненных в стандарте PC Card. В 1995 г. для мобильных интерфейсов принята более совершенная спецификация Card Bus. Шина Card Bus 68–контактная, 32-битная, с тактовой частотой 33 МГц поддерживает практически такой же протокол обмена, что и PCI, но с некоторыми упрощениями /9/.

2.1.7.*Шины внешних периферийных устройств* предназначены для подключения к системной плате разнообразного внешнего периферийного оборудования, расширяющего функциональные возможности компьютера. К типовому периферийному

оборудованию, располагаемому вне системного блока, относятся, например, принтер, внешний модем, клавиатура, мышь, сканер, плоттер и т.д. Внешние интерфейсы представляют собой группу портов (параллельных и последовательных), реализуемых специальным контроллером портов ввода-вывода (Super I/O).

Параллельный интерфейс реализуется *LPT-портом* и определяется стандартом IEEE 1284, который задает стандартный (SPP), улучшенный (EPP) и расширенный (ECP) режимы работы интерфейса /9/. Этот стандарт определяет типы используемых разъемов (например, DB-25), характеристики интерфейсных кабелей, содержащих от 18-25 проводников длиной обычно не более 2 метров, а также электрические параметры передаваемых сигналов. Скорость обмена по параллельному интерфейсу достигает 2 Мбайт/сек. Наиболее распространенным применением LPT-портов является подключение принтеров, сканеров, внешних накопителей Iomega Zip Drive. Режимы EPP и ECP позволяют использовать параллельный интерфейс для ввода – вывода цифровых и дискретных сигналов, что часто применяется в промышленных компьютерах для сопряжения с управляемыми объектами, технологическими клавиатурами, индикаторами и будет подробно рассмотрено ниже.

Последовательный интерфейс реализуется *COM-портом* и на физическом уровне определяется стандартом RS-232C /9/. Последовательный интерфейс используется для подключения удаленных до 15 метров устройств и обеспечивает лишь невысокую скорость обмена – до 115200 бит/сек. К COM-порту RS-232C можно подключить только одно устройство. Лучшие параметры имеют родственные последовательные интерфейсы RS-422A и RS-485, широко применяемые в промышленной автоматике. Например, магистральный интерфейс RS-485 обеспечивает подключение до 256 устройств, удаленных до 1200 метров при скорости обмена сотни Кбит/сек.

Универсальная последовательная шина USB (Universal Serial Bus), стандарт на которую был разработан в 1996 г., предназначена для унификации подключения к компьютеру различных внешних периферийных устройств, телефонии и бытовой электроники. USB-порт призван заменить многочисленные внешние порты (LPT, COM, GAME, клавиатуры, мыши и т.д.) и упростить эксплуатацию компьютера. Четырехпроводная шина USB допускает подключение до 127 устройств и обеспечивает скорость обмена до нескольких сотен Мбит/сек на расстоянии до 25 м /9/.

Высокопроизводительная последовательная шина стандарта IEEE 1394 (Fire Wire) предназначена для подключения к компьютеру наряду с типовыми периферийными устройствами и цифровых аудио- видеоустройств – фото- и видеокамер, приемников кабельного и спутникового телевидения, акустических систем и т.п. Шестипроводная шина IEEE 1394 обеспечивает цифровую связь до 63 устройств, соединенных по любой топологии, при суммарной длине кабеля не превышающей 72 м. Эта последовательная шина допускает высокую скорость обмена, близкую параллельной шине PCI, асинхронный и изохронный режимы передачи, не требует для подключения устройств дополнительной аппаратуры, имеет низкую цену компонентов и кабеля /9/.

2.2. Подключение централизованных УВВС

2.2.1. Централизованные УВВС обычно подключаются к системным шинам расширения и магистрально-модульным шинам компьютерных систем. Для офисных и критически офисных компьютеров это шины ISA и PCI, для промышленных компьютеров с пассивными объединительными панелями (passive back-plane) это шины STD и CompactPCI, для магистрально-модульных систем – шины VMEbus

(значительно реже MULTIBUS-I и II). Для встраиваемых малогабаритных систем управления это шины PC/104 и PC/104-Plus, для портативных мобильных компьютеров (note-book, palm top, PDA) – шины стандарта PC Card.

УВВС, устанавливаемые на шину ISA или ее промышленные аналоги, обеспечивают скорости ввода-вывода сигналов не более одного Мбайт/сек. Для обеспечения скоростей ввода-вывода до сотен Мбайт/сек используют более дорогие УВВС, устанавливаемые в шины PCI (CompactPCI) и VMEbus.

Спецификации шин и поддерживающих их стандартов определяют для УВВС конструкцию разъема, размеры плат УВВС, параметры сигналов интерфейсов и протокол обмена при вводе-выводе сигналов.

Платы УВВС устанавливаются внутри корпуса компьютерной системы управления в слоты расширения системной платы или в разъемы пассивной объединительной панели. Внешние разъемы (типа DB, IDC, SCSI-II, BNC и др.) для подключения сигналов датчиков или вывода сигналов на исполнительные устройства обычно располагаются на платах с тыльной стороны корпуса. Требования к уровням помехозащищенности и стоимости ограничивают длину радиальных линий связи УВВС с объектами единицами метров.

При сопряжении с объектами, измерительные цепи и устройства управления, нагрузками которых используют небольшие значения токов (единицы А) и напряжений (десятки В), модули нормализации и управления нагрузками размещают на специальных платах, также устанавливаемых внутри компьютерных корпусов. В противном случае (токи десятки А, напряжения сотни В) в целях безопасности эти модули размещают вне компьютерной системы в отдельных корпусах с обязательным применением гальванической изоляции.

Типовые размеры производимых, например, фирмой Advantech в промышленных масштабах многофункциональных плат устройств ввода-вывода аналоговых и дискретных сигналов следующие: для шины ISA – (185 х 100)мм, для шины PCI – (175 х 100)мм. Платы для промышленных шин CompactPCI и VMEbus выполняются в механическом стандарте Евромеханика, который определяет, в частности, следующие размеры: (160 х 100)мм – для формата 3U, (160 х 233)мм – для формата 6U.

Малогабаритные платы УВВС для шины PC/104 имеют размеры (96 х 90)мм, для шины ISA/8 (формат микроPC) – (124 х 114)мм, типа PC Card – (54 х 85,5)мм.

Таким образом, пользователь имеет возможность выбрать УВВС, наиболее удовлетворяющее требованиям конкретной задачи.

2.2.2. Для ввода и вывода небольшого числа дискретных сигналов нередко используется параллельный (LPT) порт в различных режимах - стандартном (SPP), улучшенном (EPP) и расширенном (ECP) /9,13/. Например, в режиме EPP 8 линий порта можно использовать как линии ввода-вывода, 5 линий – как линии вывода, а 4 линии – как линии ввода сигналов TTL уровней.

2.2.3. Заметим, выше были приведены шины, которые можно использовать для установки плат УВСС в IBM PC совместимые компьютеры. В компьютерах других типов используются иные шины, несовместимые с вышерассмотренными. Например, в компьютерах Macintosh применяется 32-разрядная 10-МГц системная шина расширения NuBus, для которой фирмой National Instruments производятся платы УВВС с набором драйверов для процессоров Power PC, поддерживаемых операционной системой MacOS. В России в задачах АСУ ТП в подавляющем большинстве приложений используются IBM PC совместимые компьютеры, поэтому УВВС для этого типа компьютеров наиболее распространены и доступны.

2.3. Подключение распределенных УВВС

2.3.1.Способ подключения распределенных УВВС определяется типом применяемой промышленной коммуникационной сети.

При небольших скоростях обмена (десятки Кбит/сек), малом числе (единицы – десятки) несложных локальных УВВС успешно используются дешевые промышленные коммуникационные сети на основе интерфейса RS-485 /14/. Управляющий компьютер подключается к такой коммуникационной сети посредством последовательного (COM) порта через простой преобразователь интерфейсов RS-232/RS-485. Примерами распределенных УВВС могут служить системы на базе локальных устройств серии ADAM-4000 фирмы Advantech или устройств серии I-7000.

Для лабораторных и инструментальных компьютерных систем разработаны УВВС, подключаемые к порту USB.

Для создания сложных распределенных УВВС, требующих больших скоростей передачи информации, используются более развитые промышленные коммуникационные сети Modbus, Profibus, Industrial Ethernet, Interbus, DeviceNet, CANopen, Foundation Fieldbus, LONWorks, ASI /5,6/. В этом случае включение локальных УВВС и управляющего компьютера в сеть осуществляется посредством сетевых адаптеров для того или иного типа сети. Сетевой же адаптер управляющего компьютера обычно устанавливается на системную шину расширения – ISA или PCI. Такие системы создаются, например, на базе устройств серии ADAM-5000 фирмы Advantech или WAGO I/O фирмы WAGO.

3. СТРУКТУРНАЯ СХЕМА ТИПОВОЙ ПЛАТЫ ЦЕНТРАЛИЗОВАННОГО УСТРОЙСТВА ВВОДА-ВЫВОДА СИГНАЛОВ

Для получения практических навыков работы с УВВС на учебных занятиях используется плата L-305 российской фирмы L-CARD. Эта многофункциональная широко доступная плата представляет собой централизованное устройство ввода-вывода аналоговых и дискретных сигналов, подключаемое к шине ISA персонального компьютера. Структурная схема этой платы изображена на рис.3.1.

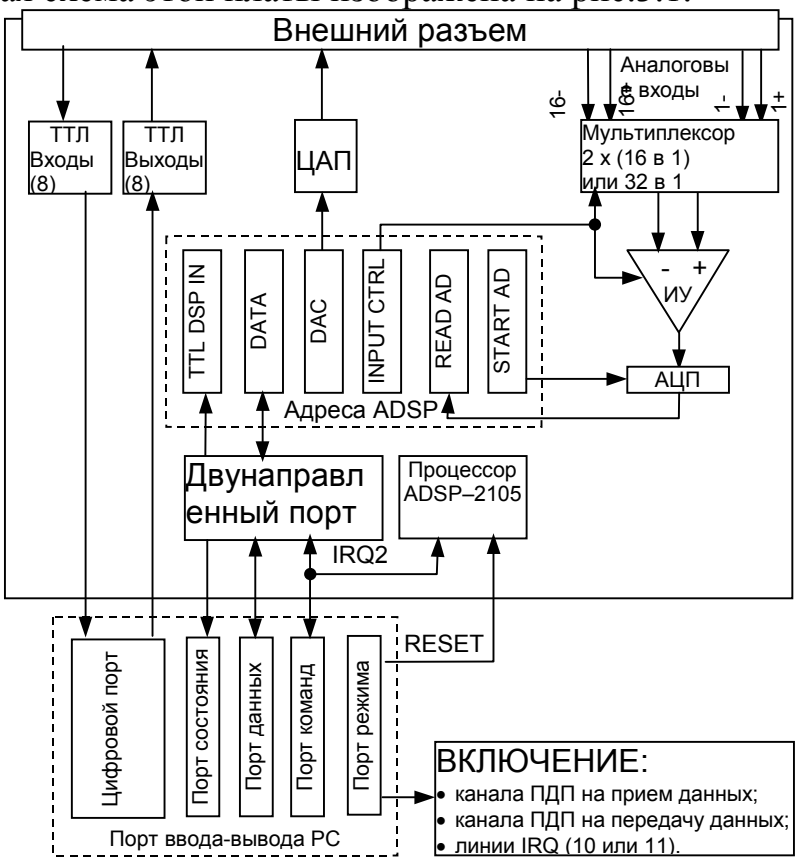


Рис. 3.1. Структурная схема платы L-305

На плате имеется один АЦП, на вход которого через аналоговый мультиплексор и измерительный усилитель может быть подключен один из 16 дифференциальных или один из 32 с общей землей аналоговых каналов с внешнего разъема платы. Разрядность АЦП 12 бит, время преобразования 1,7 мксек, максимальная частота преобразования 300 кГц, входное сопротивление – более 1 МОм. Диапазон входного сигнала может задаваться +/-5,12 В, +/-2,56 В,
+/-1,024 В.

Ввод аналоговых сигналов можно осуществлять в трех режимах: в программном режиме, в режиме прямого доступа к памяти и в режиме генерации прерываний IRQ.

Входы АЦП защищены от напряжений, превышающих +/- 5,12 В. При включенном питании компьютера входная защита выдерживает +/- 20 В, при выключенном питании - +/- 10 В.

На плате установлен один 12-битовый ЦАП, выходное напряжение которого изменяется в диапазоне +/-5,12 В, а время установления составляет 10 мксек.

Принципы работы и схемы аналоговых мультиплексоров, АЦП и ЦАП, их основные характеристики подробно рассмотрены в /2/.

Плата L-305 обеспечивает асинхронный ввод и вывод 8 цифровых сигналов

уровней TTL.

Модули нормализации и управления нагрузками не входят в состав платы и подключаются извне, как дополнительные устройства.

Управление платой и обмен данными осуществляется через двунаправленный порт данных с автоматически устанавливаемыми и сбрасываемыми битами готовности.

Управление аналоговым вводом и выводом, частотой преобразования и т. п. осуществляется сигнальным процессором ADSP-2105. При помощи порта режима и порта данных в процессор можно загрузить программу, которая будет осуществлять требуемые алгоритмы ввода - вывода. Фирменный драйвер Lbios работает по принципу команд, когда управление происходит по двум линиям: через порт команд программе в процессоре передаётся номер команды, а через порт данных осуществляется передача параметров и данных. При записи числа в порт команд, число записывается в порт данных и одновременно с этим в процессоре ADSP генерируется прерывание IRQ2. Всеми периферийными устройствами на плате управляет процессор ADSP через собственное пространство адресов ввода - вывода. Компьютеру доступен только регистр данных, через который идет обмен данными между компьютером и процессором ADSP. Â êîìïëåêò ïîñòàâêè ïëàòû âõîäèò ïðîãðàììû äëÿ ïðîöåññîðà ADSP-2105, позволяющие осуществить ввод-вывод аналоговых сигналов в различных режимах.

4. ОРГАНИЗАЦИЯ ПРОГРАММНОГО ИНТЕРФЕЙСА УСТРОЙСТВ ВВОДА-ВЫВОДА СИГНАЛОВ

4.1. Уровни управления платами УВВС

4.1.1. Для каждого УВВС имеется программное обеспечение и руководство по программированию, позволяющее работать с ним на трех уровнях.

При программировании на низком уровне (языки С, Ассемблер) доступ к устройствам осуществляется непосредственно через порты и регистры /15/, что обеспечивает гибкость и эффективность, но требует очень больших затрат на разработку.

При программировании на уровне драйверов возможно использование для работы с устройством готовых функций, обеспечиваемых драйверами. Вызов этих функций производится из программ, написанных на языках высокого уровня. Как правило, для плат УВВС набор готовых функций поставляется в виде библиотек подпрограмм, реализующих для упрощения программирования плат множество разнообразных функций. Готовая библиотека подпрограмм позволяет использовать практически все возможности платы УВВС, не вдаваясь в тонкости программирования на уровне Ассемблера и портов ввода-вывода. Организация вызова функций библиотеки, например, из программы на языке Turbo C, подробно описана в /16/, а из программ на языке Turbo Pascal – в /17/.

Наконец, при работе на уровне специальных пакетов приложений возможно создание рабочих программ для УВВС с использованием специальных графических сред разработки (например, LabView, UltraLogic, язык функциональных блоковых диаграмм FBD и т.п.). В этом случае разработка алгоритмов управления и программных интерфейсов производится без написания программного кода в классическом смысле. Последний способ обеспечивает минимальное время на разработку программного обеспечения, но стоимость таких сред разработки является высокой.

Разработку программного обеспечения для рассмотренной платы L-305 в дальнейшем будем производить средствами второго уровня на основе библиотеки функций ввода-вывода /18/.

4.1.2. Смысловое содержание терминов, используемых в дальнейшем для описания устройств платы L-305, приведено в табл.4.1.

Табл.4.1

Термины, используемые для описания устройств платы L-305

Название	Содержание
Rate	Интервал ввода в микросекундах
Nch	Число каналов для многоканального ввода
Data	Указатель на целочисленный массив для данных
N point	Число отсчетов для одноканального ввода и число кадров для многоканального ввода (под кадром понимается последовательность вводимых отсчетов по Nch каналам)
Channel	Номер канала при одноканальном вводе
Channels	Указатель на целочисленный массив с номерами каналов для многоканального ввода
Usil	Коэффициент усиления

4.2. Форматы данных и модели памяти

4.2.1. Данные, считанные с АЦП, представлены в формате знакового целого двухбайтного числа (-2048 <= N < 2048). При этом максимальное напряжение, соответствующее установленному входному диапазону, равно +2047, а минимальное -2048.

Например:

int i, i=ADCHAN(0);

значение переменной *i* будет лежать в пределах от -2048 до +2047.

Ñîîòâåòñòâèå êîäà ÀÖÏ âõîäíîìó íàïðÿæåíèþ çàäàåòñÿ òàáë.4.2.,

Табл.4.2

Код	Напряжение
+204	+MAX Вольт
0	0 Вольт
-2048	-MAX Вольт

где MAX - значение установленного диапазона для канала АЦП, может быть установлено от 1.024 В до 5.12 В.

4.2.2. Коды, выдаваемые на ЦАП, связаны с устанавливаемым выходным напряжением ЦАП в соответствии с табл.4.3.:

Табл.4.3

Код	Напряжение
+204	+5,12 Вольт
0	0 Вольт
-2048	-5,12 Вольт

4.2.3. На плате АЦП имеется 16 дифференциальных аналоговых каналов или 32 канала с общей землей (тип подключения устанавливается при помощи соответствующей перемычки). В функциях ввода с АЦП задается такой параметр, как номер канала АЦП или последовательность номеров каналов для многоканального ввода. С номером канала связан также коэффициент усиления, т. е. для каждого канала можно установить индивидуальный коэффициент усиления. Формат номера канала задается табл.4.4.

Табл.4.4

Бит	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
Им	C5	X	X	X	X	X	U2	U1	X	X	X	X	C4	C3	C2	C1

Поля C5 и C4-C1 формируют пятибитный номер канала (первому каналу соответствует число 0, 32-му число 31). При дифференциальном подключении бит C5 всегда должен быть равен 0, при подключении с общей землёй бит C5 является старшим битом номера канала АЦП. Поля U2 и U1 этой таблицы определяют коэффициент усиления канала в соответствие с табл.4.5.

Табл.4.5

Бит U2	Бит U1	Усиление
0	0	1
0	1	2
1	0	5
1	1	резерв

Например, при выставленном на плате перемычками диапазоне ±5.12 В и усилении 5 диапазон входного сигнала будет ±1.024 В, а при усилении 1 диапазон будет ±5.12 В.

4.2.4. Все функции библиотеки предполагают использование дальних указателей типа FAR как для вызова процедур, так и для передачи адресов данных, поэтому

компиляцию программы на языке С желательно проводить в модели памяти LARGE. В этой модели по умолчанию используются дальние вызовы процедур и дальние указатели на данные.

4.2.5. Для использования в разрабатываемой программе библиотеки функций платы L-305 необходимо выполнить следующие действия:

1 создать файл проекта для языка Си(.prj);
2 добавить в него файл Lbiosdrv.obj;
3 добавить в него файл с программой, которая будет использовать функции драйвера;
4 добавить в начало программы строку #include "function.h" (при этом файл function.h, содержащий описания функций библиотеки должен находиться в той же директории, что и файл проекта).

Компилировать программу нужно для модели памяти LARGE.

4.2.6. Для запуска откомпилированной программы под ОС Windows 9x необходимо создать командный файл, в котором сначала загружается драйвер, затем — сама программа. Загрузка драйвера осуществляется следующим образом: lh loadbios lbios11,

где loadbios — программа-загрузчик;

lbios11 — код драйвера платы.

Командный файл должен находиться в одной директории с загрузчиком драйвера, сама программа — в директории, прописанной в путях поиска ОС (переменная PATH).

5.ОПИСАНИЕ БИБЛИОТЕКИ ФУНКЦИЙ И ПРИМЕРЫ ЕЕ ИСПОЛЬЗОВАНИЯ

5.1. Функции конфигурации

Функции конфигурации предназначены для уведомления драйвера (библиотеки функций для платы) об установленной с помощью перемычек конфигурации платы. При помощи перемычек можно изменить базовый адрес платы в пространстве ввода-вывода компьютера, и изменить номер линии прерывания, которое может генерироваться платой. По умолчанию драйвер настроен на базовый адрес 0õ300 и на линию прерывания IRQ 11, при изменении этих параметров необходимо сообщить об этом драйверу, вызвав описываемые ниже функции с соответствующими параметрами.

5.1.1. *Установка базового адреса*

void SETBASEADDRESS(int address);

Назначение. Устанавливает новый базовый адрес в пространстве PC для функций драйвера, переменная address должна быть равна одному из значений 0õ300, 0õ310, 0õ330, 0õ340 в соответствии с установленными на плате перемычками.

Параметры: *address* -> значение базового адреса платы в компьютере.

5.1.2. *Установка номера прерывания*

void INTR_SETUP(int irq_number);

Назначение. Сообщает драйверу о выбранном номере прерывания (IRQ10 или IRQ11) на плате. По умолчанию на плате установлен номер прерывания IRQ11 и драйвер изначально предполагает, что номер прерывания равен IRQ11.

Параметры:

irq_number:

- *irq_number* = 0 -> IRQ10
- *irq_number* = 1 -> IRQ11.

5.1.3. *Проверка наличия платы в компьютере*

int PLATA_TEST();

Назначение. Проводит диагностику наличия платы в компьютере. Возвращает нулевое значение в случае успешного тестирования присутствия платы с загруженным драйвером Lbios и ненулевое значение в противном случае. Ненулевое значение означает, что либо плата отсутствует в компьютере либо не был загружен драйвер LBIOS, либо значение базового адреса платы не соответствует установленному через функцию SETBASEADDRESS().

Возвращаемое значение:

- 0 плата обнаружена
- 1 плата не обнаружена.

5.2. Функции задания временных параметров АЦП

На плате АЦП имеется один встроенный в процессор счетчик-таймер, при помощи которого можно осуществлять синхронный ввод-вывод с аналоговых каналов. Таймер запускается установленным на плате кварцевым генератором, что обеспечивает точную синхронизацию процессов ввода-вывода. На плате установлен кварц 10 МГц, при этом минимальный интервал, с которым программируется таймер составляет соответственно 100 нс.

При многоканальном вводе иногда возникает необходимость контролировать межканальную задержку. По умолчанию программа настраивается на 4 мксек, для работы с меньшим межканальным интервалом необходимо переустановить межканальную задержку.

Таймер управляется двумя регистрами: базовым 16-битным регистром, в котором хранится интервал ввода и 8-битным регистром масштабирования таймера, при помощи которого можно замедлять процессы ввода. Например, при установленном коэффициенте масштабирования равным единице, максимальный интервал ввода равен 0xffff*100нс*1=6553,5 мксек, при коэффициенте масштабирования равным 100 максимальный интервал ввода составляет 0xffff*100нс*100=655,350 мсек.

Во всех функциях драйвера, где передается параметр интервал ввода (Rate), параметр "Rate" записывается в базовый регистр таймера. Для переустановки регистра масштаба используется специальная функция изменения масштаба.

5.2.1. *Изменение масштаба таймера*

void SET_TIMER_SCALE(int value);

Назначение. По умолчанию при вводе-выводе массивов временной интервал задается в микросекундах (т. е. масштаб равен 10). Данная функция предназначена для изменения масштаба задания интервала ввода/вывода.

Параметры: параметр ***value ->*** 8 битное число, определяющее масштаб ввода в сотнях наносекунд (минимальное значение - 1. максимальное значение 256):

Например:

- value = 1 интервал ввода/вывода задается в сотнях наносекунд
- ***value* =10 интервал ввода/вывода задается в микросекундах (установлено по умолчанию)**
- value = 100 интервал ввода/вывода задается в десятках микросекунд

Пример:

```
main()
{
int ch=0, i, Rate= 10;
int Data[100];
//установим временной масштаб в 10 мкс
SET_TIMER_SCALE( 100);
//введем 100 отсчетов с интервалом 10мкс*Rate=100 мкс
//т, е. общее время ввода составит 100отсчетов*100мкс=
// 10 мс.
STREAM (Data, 100, ch, Rate);
}
```

5.2.2. *Установка межканальной задержки*

void SET_INTER_DELAY(int delay_value);

Назначение. Устанавливает межканальную задержку для функций многоканального ввода с АЦП. Межканальная задержка также изменяется при вызове функции SET_ADSP_SPEED(). При этом задержка переустанавливается на 4 мкс.

Параметры: устанавливаемая межканальная задержка в наносекундах равна delay_value*200нс + 1800нс (после инициализации платы задержка устанавливается равной 4 мкс).

Пример:

```
main()
{
int i, Data[100], Channels[16]={0,1,2}, Rate=20, Nch=3;
// установим задержку на 2 мкс
```

```
SET_INTER_DELAY(1);
// осуществим трехканальный ввод десяти кадров на
// межканальной задержке 2 мкс
SOFT(Data, 10, Channels, Nch, Rate);
}
```

5.3. Функции ввода по аналого-цифровым каналам в программном режиме

Функции данного раздела служат для обеспечения ввода аналоговых сигналов в компьютер в различных программных режимах: в асинхронном режиме и в режиме синхронизации от таймера.

5.3.1. *Установка номера канала АЦП*

void SETCHANNEL(int Channel);

Назначение. Устанавливает канал АЦП Channel для последующего ввода функцией однократного асинхронного ввода SAMPLE().

Параметры: *Channel ->* номер канала АЦП (см. формат номера канала)

Пример:

```
main()
{
int i;
SETCHANNEL(0);
// установим первый канал
i=SAMPLE();
// введем значение с канала
cprintf("Код АЦП = %d", i);
// выведем его на экран
}
```

5.3.2. *Однократный асинхронный ввод с АЦП*

int SAMPLE()

Назначение. Осуществляет аналого-цифровое преобразование с канала АЦП, предварительно установленного при помощи функции SETCHANNEL(). Данной функцией удобно пользоваться для асинхронного одноканального ввода, когда не требуется переустанавливать номер канала.

Ïàðàìåòðû îòñóòñòâóþò.

5.3.3. *Однократный ввод с переустановкой канала АЦП*

int ADCHAN(int Channel);

Назначение. Устанавливает заданный канал АЦП и осуществляет аналого-цифровое преобразование. Данная функция удобна для осуществления асинхронного ввода с разных каналов АЦП. Возвращает результат преобразования по каналу 'Channel'.

Параметры: номер канала АЦП.

Пример:

```
main()
{
// введем значение с первого канала и выведем преобразованное
// значение кода в вольтах
cprintf("\n\r Напряжение на первом канале = %f", ADCHAN(0)*5.12/2048.);
}
```

5.3.4. *Однократный ввод последовательности каналов АЦП*

void KADR (int *Data, int *Channels, int Nch);

Назначение. Вводит Nch отсчетов с каналов АЦП, указанных в массиве Channels,

в массив Data. Например, если требуется ввести по одному значению с трех каналов АЦП, то можно сформировать целочисленный массив Channels из 3 элементов, при этом первый будет равен коду для первого канала, второй для следующего и т.д.

Параметры:

1. *Data* - целочисленный массив, в который будут помещены результаты ввода с Nch каналов;
2. *Channels* - целочисленный массив с номерами каналов;
3. *Nch* - число каналов.

Пример:

```
main()
{
int i, Nch=3, Channels[16]={0, 1, 2}, Data[16];
// введем первые три канала
KADR (Data, Channels, Nch);
// нарисуем на экране их значения в кодах АЦП (-2048..2047)
for(i=0, i < 3; i++) cprintf("\n\r Канал N%d = %d", i+1, Data[i]);
}
```

5.3.5. *Одноканальный ввод с синхронизацией от таймера*

void STREAM (int *Data, int Npoint, int Channel, int Rate);

Назначение. Данная функция осуществляет считывание последовательности отсчетов с заданного канала АЦП с интервалом *Rate* между отсчетами.

Параметры:

1. *Data* - целочисленный массив, в который будут помещены вводимые отсчеты;
2. *Npoint* - число вводимых отсчетов (1 <= Npoint <= 32768);
3. *Channel* - номер канала АЦП, по которому будут вводится данные;
4. *Rate* - интервал ввода (1 <= Rate <= 0xffff).

Пример:

```
main()
{
int i, Data[100], Npoint=100, Channet=0, Rate=10;
// введем 100 значений по первому каналу АЦП с интервалом ввода 10 мкс
STREAM(Data, Npoint, Channel, Rate);
//выведем на экран первые 10 введенных значений
for(i=0, i < 10, i++) cprintf("\n\rValue[%2d]=%5d", i+1, Data[i]);
 }
```

5.3.6. *Многоканальный ввод с синхронизацией от таймера*

void SOFT(int *Data, int NPoint, int *Channels, int Nch, int Rate);

Назначение. Функция SOFT() осуществляет ввод NPoint кадров с Nch аналоговых каналов, номера которых передаются в целочисленном массиве Channels с интервалом Rate. Под кадром подразумевается ряд отсчетов, состоящий из результатов последовательного опроса первых Nch каналов, указанных в массиве ‘Channels'.

Параметры:

Data - целочисленный массив, в который будут помещены вводимые отсчеты;

Npoint - число вводимых кадров (1 <= Npoint*Nch <= 32768);

Channels - целочисленный массив с номерами каналов АЦП, по которым будут вводится данные;

Nch - число каналов;

Rate - интервал ввода (1 <= Rate <= 0xffrf).

Пример:

```
main()
{
int i, j, Data[100], Npoint=20, Channels[16]= {0,1,2,3};
int Nch=4, Rate=50;
//ввод 20 кадров (20*4=80 отсчетов) по первым четырем
// каналам с интервалом 50 мкс между кадрами
SOFT(Data, Npoint, Channels, Nch, Rate);
// выведем первые два кадра
for(t=0, t < 2, i++) for(j=0, cprintf("\n\r Кадр=%2d" ,i+1); j < Nch; j++)
        cprintf ("Канал #%2d=%5d, “, j+1, Data[i*Nch+j]);
}
```

5.3.7. *Синхронизация ввода с АЦП*

void SYNCHRO_MODE(int SMode, int TtlMask, int AdChannel, int AdValue);

Назначение. Для решения некоторых задач требуется начинать процесс ввода с АЦП только после какого-либо события: превышения уровня на одном из каналов АЦП, после изменения уровня на цифровой линий и т.п. Для решения подобных задач служит описываемая функция. Она позволяет устанавливать режим синхронизации на все функции синхронного ввода с АЦП.

Параметры:

SMode - тип синхронизации;

- ***SMode* = 0 -> синхронизация старта выключена (по умолчанию).** При этом ввод начинается непосредственно после передачи последнего параметра ввода процессору ADSP;
- *SMode* = 1 -> синхронизация старта по каналу АЦП. При синхронизации по каналу АЦП программа дожидается заданного превышения порогового значения "AdValue" на заданном канале АЦП 'AdChannel' и, после достижения такого значения, начинает вводить данные (первый ввод произойдет примерно через 5 мкс после достижения порогового значения);
- *SMode* = 2 -> покадровая синхронизация по цифровому биту. При покадровой синхронизации передача данных с АЦП в компьютер производится не через FIFO, поэтому использование этого режима на медленных компьютерах может давать сбои (пропуск отсчетов). При покадровой синхронизации плата дожидается отрицательного перепада длительностью не менее 20 нс на цифровой ТТЛ линии ‘Внешний запуск’ (линия 42 на разъеме платы). После перепада плата вводит один кадр (Nch аналоговых каналов), после чего плата опять переходит в режим ожидания перепада на линии 'Внешний запуск’. В этом случае параметр 'интервал ввода’ не имеет смысла;
- *SMode* = 3 -> синхронизация старта по цифровому биту. При этом плата один раз дожидается отрицательного перепада длительностью не менее 20 нс на цифровой ТТЛ линии ‘Внешний запуск' (линия 42 на разъеме платы), после чего запускается ввод данных с заданным интервалом ввода 'Rate'.

TtlMask зависит от SMode.

В режиме синхронизации по каналу АЦП параметр *TtlMask* определяет, ожидается ли переход через пороговое значение “снизу вверх” или “сверху вниз”

- *TtlMask=1:* начало ввода начнется после того, как по указанному каналу АЦП код с АЦП (от -2048 до 2047) превысит пороговое значение (т.е. для порога 1000 это значения > 1000 (1001, 1002 и т.д.); для порога -1000 — это -999, -998 и т.д.);
- *TtlMask=0:* начало ввода начнется после того, как по указанному каналу АЦП код (от -2048 до 2047) станет меньше порогового значения.

В режиме синхронизации по цифровому биту параметр TtlMask не имеет смысла.

Параметр *AdChannel* определяет номер аналогового канала, по которому производится синхронизация.

Параметр *AdValue* определяет пороговое значение при синхронизации по аналоговому каналу.

Пример:

```
//В данном примере ввод начинается только после того, как на втором канале АЦП
//превысится установленное пороговое значение (во избежание зависания пороговое //значение установлено равным -1000).

main()
{
int, TtlMask, SynchroAdChannel, AdPorog, SMode, SMode=1;
int Data[100], NPoint=20, Nch=4, Channels[16]={0,1,2,3};
int Rate=10;

// ждем превышения
TtlMask=1;

// второй канал синхронизации
SynchroAdChannel= 1;

// установим пороговое значение
AdPorog=-1000;

//переустановим режим синхронизации
SYNCHRO_MODE(Smode, TtlMask, SynchroAdChannel, AdPorog);

// введем данные с синхронизацией
SOFT(Data, NPoint, Channels, Nch, Rate);

// выключим режим синхронизации
SYNCHRO_MODE(0, 0, 1, 0);
}
```

5.4. Функции вывода по цифро-аналоговым каналам

На плате установлен один цифро-аналоговый преобразователь (ЦАП), при помощи которого можно управлять внешними устройствами, генерировать сигналы произвольной формы и т.п.

5.4.1. *Режим вывода на ЦАП*

void SET_DA_NUMBER(int Mode, int DaNumber);

Назначение. Устанавливает режим вывода на ЦАП для функций синхронного ввода с АЦП.

Параметры:

1. *Mode* включает одновременный вывод на ЦАП в функциях синхронного ввода с АЦП. *Mode* =1 включает режим вывода на ЦАП одновременно со вводом с АЦП (в режиме *Mode=1* во время выполнения функций DMAALL() любые значения, записываемые в порт данных платы, передаются на ЦАП). *Mode=0* выключает режим одновременного вывода на ЦАП.
2. *DaNumber* – зарезервирован.

5.4.2. *Асинхронный вывод на ЦАП*

void OUTDA(int Code);

Назначение. Устанавливает выходное напряжение на ЦАПе в соответствии с кодом Code.

Параметры: Code — выводимый на ЦАП код (см. описание форматов данных).

5.4.3. ***Асинхронный вывод на ЦАП одновременно с вводом с АЦП***

void STREAM_OUTDA(int Code);

Назначение. Устанавливает выходное напряжение на ЦАПе в соответствии с кодом Code во время выполнения функций DMAONE, DMAALL без прерывания работы этих функций.

Параметры: Code — выводимый на ЦАП код (см. описание форматов данных).

Пример:

```
main()
{
int i, *Data, Chan=0, NPoint=100, Rate=10, Dmm=0;

// установим указатель на старшую страницу памяти
Data=(int far *) 0ð90000000,

// установим режим фонового вывода на ЦАП
SET_DA_NUMBER(1,0);

// запустим ПДП на ввод
DMAONE(Data, NPoint, Chan.Rate, Dmm);

// выведем на ЦАП минимальное напряжение
 STREAM_OUTDA(-2048);

// выведем на ЦАП 0
STREAM_OUTDA(0);

// выведем на ЦАП максимальное напряжение
STREAM_OUTDA(2047);

//дождемся завершения ввода
while(!DMA_TEST());

// выключим плату и контроллер ПДП
 DMA_OFF();
// отключим режим фонового вывода на ЦАП
SET_DA_NUMBER(0,0);
}
```

5.4.4. ***Синхронный одноканальный вывод на ЦАП***

void DASTREAM(int Rate, int NPoint, int *Data);

Назначение. Выводит NPoint отсчетов из массива Data на ЦАП с интервалом Rate.

***Параметры*:**

1. *Rate* - интервал ввода;
2. *Npoint* - число выводимых отсчетов;
3. *Data* - массив с выводимыми отсчетами.

Пример:

```
#include <math.h>
main()
{
int i, Data[1000], Rate=10, Npoint=1000;
```

```
// генерируем синусоидальный сигнал
for(i=0; i < Npoint; i++) Data[i]=2000*sin(i*M_PI/80 );

// установим первый ЦАП
SET_DA_NUMBER(0, 0);

// выведем его на первый ЦАП
DASTREAM(Rate, NPoint, Data);
}
```

5.5. Функции ввода-вывода по цифровым каналам

5.5.1. *Чтение восьми младших цифровых линий*

int INPBYTE_305();

Назначение. Возвращает состояние 8-и внешних цифровых ТТЛ линий.

***Ïàðàìåòðû*:** îòñóòñòâóþò.

Пример:

```
main()
{
int i,
// введем 8 линий
i=INPBYTE_305();
}
```

5.5.2. *Вывод на восемь цифровых линий*

void OUTBYTE_305(int Code);

Назначение. Функция установки 8-и цифровых линий

Параметры: Code — устанавливаемый код.

5.6. Функции ввода по аналого-цифровым каналам с использованием прерываний

Ввод по прерываниям используется, как правило, при анализе процессов на сравнительно медленных частотах ввода (до 10 кГц). Общая идеология ввода обычно сводится к следующему: АЦП вводит во внутреннюю память платы УВВС ряд отсчетов с заданных каналов и после этого генерирует прерывание в компьютер, в котором предварительно должен быть загружен драйвер-обработчик используемого платой прерывания, который считывает из платы введенный ряд отсчетов.

5.6.1. *Генерирование прерываний без ввода с АЦП*

void INIT_SIMPLE_INTR (int Rate, interrupt *Vector);

Назначение. Программирует контроллер прерываний и плату для работы в режиме генерации установленного прерывания. После вызова процедуры INIT_SIMPLE_INTR() плата генерирует прерывания с интервалом Rate мкс, которое обрабатывается драйвером-обработчиком, адрес которого передается в параметре Vector. Обработчик должен позаботиться о сбросе как контроллера прерываний компьютера, так и контроллера прерываний платы АЦП.

Параметры:

1. *Rate* — интервал генерирования прерываний;
2. *Vector* — адрес обработчика прерываний.

Пример:

```
// Драйвер - обработчик прерывания IRQ от платы АЦП при каждом вызове приращивает на //
```

единицу статическую переменную и выводит её значение на экран.

```
// обработчик прерывания
void interrupt E_Interrupt(PARM)
{
static unsigned int cnt;

// выведем значение статической переменной
cprintf("\n\r Счетчик = %5u", cnt++);

// сбросим флаг прерывания на плате АЦП и в компьютере
RESET_IRQ();
}

// основная функция
main()
{
// установим интервал между прерываниями 10000 мкс и
// адрес обработчика E_Interrupt
INIT_SIMPLE_INTR( 10000, E_Interrupt);

// ждем пока не нажата клавиша
while(!kbhit());

// запретим прерывания от платы и восстановим
// контроллер прерываний
STOP_INTR();
}
```

5.6.2. *Прерывания с одноканальным вводом с АЦП*

void STREAM_INTR (int Rate, interrupt *Vector, int Channel);

Назначение. Программирует контроллер прерываний и плату для работы в режиме ввода с заданным интервалом и последующим генерированием прерывания. Драйвер - обработчик должен позаботиться о сбросе как контроллера прерываний компьютера, так и контроллера прерываний платы АЦП. Для чтения введенного отсчета с АЦП можно использовать функцию READ_DATA().

Параметры:

1. *Rate* — интервал генерирования прерываний;
2. *Vector* — адрес обработчика прерываний;
3. *Channel* — номер канала АЦП.

Пример:

```
// Драйвер - обработчик прерывания от платы
// при каждом вызове вводит отсчет с АЦП и выводит его на экран.

// глобальная переменная
int GlobalVar=0;

// обработчик прерывания
void interrupt E_Interrupt(PARM)
{
int i;

if(!GlobalVar)
{
```

```
// введем отсчет с АЦП
i=READ_DATA();

// отобразим его на экран
cprintf("\n\r Код с АЦП = %5d", i);
}

// сбросим флаг прерывания на плате АЦП и в компьютере
RESET_IRQ();
}

// основная функция
main()
{
int Channel=0;
GlobalVar=0;

// установим интервал между прерываниями 10000 мкс и
// адрес обработчика E_Interrupt
INIT_SIMPLE_INTR {10000, E_Interrupt, Channel);

// ждем пока не нажата клавиша
while(!kbhit());

// сообщим обработчику, что мы собираемся выключить прерывания
GlobalVar=1;

// запретим прерывания от платы и восстановим контроллер прерываний
STOP_INTR();
}
```

5.6.3. ***Прерывания с многоканальным вводом с АЦП***

void SOFT_INTR (int Rate, interrupt *Vector, int *Channels, int Nch);

Назначение. Программирует контроллер прерываний и плату для работы в режиме многоканального ввода с АЦП с заданным интервалом ввода и последующим генерированием прерывания. Драйвер - обработчик должен позаботиться о сбросе как контроллера прерываний компьютера, так и контроллера прерываний платы. Обработчик должен считать заказанный ряд отсчетов с каналов АЦП при помощи функции READ_DATA().

Параметры:

1. *Rate* — интервал генерирования прерываний;
2. *Vector* — адрес обработчика прерываний;
3. *Channels* — целочисленный массив с номерами каналов АЦП;
4. *Nch* — число вводимых каналов АЦП.

5.6.4. ***Сброс флага прерываний***

void RESET_IRQ(void);

Назначение. Обработчик прерывания должен перед выходом сбросить регистр прерывания на плате и в компьютере. Для этого достаточно вызвать функцию RESET_IRQ().

Ïàðàìåòðû: îòñóòñòâóþò

5.6.5. ***Чтение отсчетов из обработчиков прерывания***

int READ_DATA ();

Назначение. При генерировании прерывания в режиме одноканального ввода с АЦП плата производит преобразование с АЦП и записывает результат в регистр данных, после чего генерирует прерывание. Функция READ_DATA() осуществляет чтение регистра данных. При многоканальном вводе по прерываниям плата производит ввод кадра во внутреннюю память, после чего генерирует прерывание. Для чтения результата кадрового ввода нужно вызвать функцию READ_DATA() 'Nch' раз, при этом первый вызов вернет значение первого канала, а последний вызов вернет значение, полученное по последнему каналу.

Ïàðàìåòðû: îòñóòñòâóþò.

5.6.6. ***Выключение прерываний***

void STOP_INTR(void);

Назначение. Выключает режим генерации прерываний на плате и восстанавливает контроллер прерываний PC. Данную функцию необходимо вызывать после завершения использования платы в режиме генерирования прерываний.

Внимание! При вызове данной функции будет сгенерировано одно прерывание, вызванное отключением платы от линии прерывания. Для избежания зависания программы из-за этого "ложного" прерывания в примерах используется переменная *GlobalVar,* при помощи которой основная программа предупреждает обработчик прерываний о том, что будет сгенерировано "ложное" прерывание, которое не надо будет обрабатывать.

Параметры: отсутствуют.

5.7.Функции ввода-вывода по аналого-цифровым каналам с использованием прямого доступа к памяти

5.7.1.Канал ПДП является удобным средством для реализации алгоритмов работы в реальном масштабе времени на больших частотах (до 300 кГц), когда требуется одновременно с вводом данных осуществлять их обработку, визуализацию, запись на винчестер и т.п. В режиме ввода-вывода по каналам ПДП контроллеру ПДП необходимо только указать область памяти, в которую будет осуществляться ввод или из которой будет осуществляться вывод, и контроллер ПДП сам в фоновом для центрального процессора режиме начнет заполнять указанную область памяти, при этом имеется возможность отслеживать счетчик используемого канала ПДП, который может использоваться для определения того, какая часть буфера ввода заполнена данными.

Наличие режима автоинициализации контроллера ПДП позволяет контроллеру входить в вечный цикл заполнения указанной области памяти по принципу кольцевого буфера. После заполнения буфера контроллер ПДП сам без остановки и пропуска отсчетов продолжает вводить снова с начала буфера и т.д. При этом появляется очень удобная возможность организовывать следующий алгоритм работы в режиме реального времени.

1. Инициализация платы АЦП и контроллера ПДП и перевод ПДП в режим автоинициализации.
2. Ожидание заполнения первой половины буфера памяти.
3. Обработка первой половины буфера (пока контроллер ПДП заполняет вторую половину, можно осуществлять обработку первой, не заботясь о вводе с АЦП, поскольку ввод автоматически осуществляется контроллером ПДП во вторую часть буфера).
4. Ожидание заполнения второй половины буфера ПДП.

5. Обработка второй половины (тем временем ПДП автоматически продолжает заполнять буфер сначала).
6. Переход к пункту 2.

Приведенный алгоритм позволяет, например, организовать фоновую запись вводимых данных на жесткий диск даже при большой частоте ввода (до 300 кГц).

5.7.2. Единственным недостатком работы с каналом ПДП является выделение памяти под буфера для ввода-вывода данных. Фирма IBM оставила для совместимости очень неудобные ограничения на формирование базового адреса для буфера ввода по ПДП. Для правильной установки базового адреса необходимо представлять организацию памяти PC с точки зрения контроллера ПДП. Для контроллера ПДП вся память разбивается на последовательные блоки по 128 К, при этом в стандартном режиме контроллер ПДП может записывать данные только в один из этих блоков. Если после записи очередного отсчета адрес следующего попадает на границу страницы, то контроллер переходит на начало текущей страницы, а не на начало следующей. Простым способом выделения памяти является запись в указатель непосредственного адреса самой старшей девятой страницы с адресом SEG=8, OFFS=0x8000 Buffer=(int *) 0ð90000000. При этом, если в программе требуется осуществлять динамическое распределение памяти, то возможно предусмотреть проверку возвращаемых указателей на отсутствие наложения на последнюю страницу памяти.

5.7.3. ***Одноканальный ввод в режиме ПДП***

void DMAONE(int *Data, int NPoint, int Chan, int Rate, int Dmm);

Назначение. Инициализирует контроллер ПДП и плату АЦП в режим одноканального ввода с канала *Chan* числа *Npoint* отсчетов в массив *Data.* После выхода из функции плата начинает вводить данные по каналу ПДП. Для проверки завершения ввода и доступа к счетчику контроллера ПДП, позволяющего узнать, сколько данных было введено, используются функции DMA_TEST() или DMA_COUNTER().

Параметры:

1. *Data* — целочисленный массив для данных;
2. *NPoint* — число вводимых отсчетов (1 <= NPoint <= 32768);
3. *Chan* — номер канала АЦП;
4. *Rate* — интервал ввода (от 1 до 0xffff);
5. Dmm — параметр, включающий режим автоинициализации. В случае, если режим автоинициализации будет включен, после ввода *NPoint* отсчетов плата вновь продолжит ввод в буфер, начиная с адреса *Data,* до тех пор, пока канал ПДП и плата не будут остановлены при помощи функции DMA_OFF().

Dmm = 0 — режим автоинициализации выключен;

Dmm = *1* — режим автоинициализации включен.

Пример:

```
main()
{
int i, *Data, Chan=0, NPoint=100, Rate=10, Dmm=0;

// установим указатель на старшую страницу памяти
Data=(int far *) 0ð90000000;

// запустим ПДП на ввод
DMAONE(Data, NPoint, Chan, Rate, Dmm);

//дождемся завершения ввода
```

```
while(!DMA_TEST());

// выключим плату и контроллер ПДП
DMA_OFF();
}
```

5.7.4. *Многоканальный ввод в режиме ПДП*

void DMAALL(int *Data, int NPoint, int *Channels, int Nch, int Rate, int Dmm);

Назначение. Инициализирует контроллер ПДП и плату АЦП в режим многоканального ввода с Nch каналов, указанных в массиве Channels, числа NPoint кадров в массив Data. После выхода из функции плата начинает вводить данные по каналу ПДП. Для проверки завершения ввода и доступа к счетчику контроллера ПДП, позволяющего узнать, сколько данных было введено, используются функции DMA_TEST() или DMA_COUNTER().

Параметры:

1. *Data* — целочисленный массив для данных;
2. *Npoint* — число вводимых кадров (1 <= NPoint*Nch<= 0xffff);
3. *Channels* — целочисленный массив, содержащий номера каналов АЦП, с которых будет производится ввод;
4. *Nch* — число вводимых каналов;
5. *Rate* — интервал ввода (от 1 до 0xffff);
6. *Dmm* — параметр, включающий режим автоинициализации. В случае, если режим автоинициализации будет включен, после ввода *NPoint* кадров плата вновь продолжит ввод в буфер, начиная с адреса *Data,* до тех пор, пока канал ПДП и плата не будут остановлены при помощи функции DMA_OFF().

Пример:

```
//Ввод по ПДП по четырем каналам
main()
{
int i, *Data, Channels[16]={0,1,2,3}, Npoint =100, Rate=20;
int Dmm=0, Nch=4;

// установим указатель на старшую страницу памяти
Data=(int far *) 0ð90000000,

// запустим ПДП на ввод 4 каналов АЦП с интервалом
// 20 мкс между кадрами
DMAALL(Data, NPoint, Channels, Nch, Rate, Dmm);

//дождемся завершения ввода
while(!DMA_TEST());

// выключим плату и контроллер ПДП
DMA_OFF();
}
```

5.7.5. *Одновременный ввод с АЦП и вывод на ЦАП*

void DMA_ALL_DA(int *AdData, int AdNkadr, int *Channels, int Nch, int Rate, int AdDmm, int *DaData, int DaNkadr, int DaDmm, int DaNumber);

Назначение. Функция покадрового считывания ‘AdNkadr’ кадров с аналоговых каналов в буфер ОЗУ по адресу AdData с первых 'Nch' каналов АЦП в массиве Channels с интервалом 'Rate' мкс между кадрами использует канал ПДП с одновременным выводом на ЦАП 'DaNumber' массива DaData размером DaNkadr по каналу ПДП.

Переменная 'AdDmm' определяет режим работы контроллера ПДП компьютера. Она должна быть равна 0 для однократной передачи заказанного блока данных и 1 для циклического ввода. Аналогично переменная 'DaDmm' определяет режим вывода на ЦАП (однократный или циклический).

Параметры:

1. *AdData* — адрес буфера, в который будут помещаться вводимые данные с АЦП;
2. *AdNkadr* — число вводимых кадров;
3. *Channels* — целочисленный массив с номерами каналов АЦП;
4. *Nch* — число каналов АЦП;
5. *Rate* — интервал ввода-вывода;
6. *AdDmm* — включение автоинициализации канала ПДП АЦП;
7. *DaData* — адрес массива, из которого будут считываться данные, выводимые на ЦАП;
8. *DaNkadr* — размер массива, выводимого на ЦАП;
9. *DaDmm* — включение автоинициализации канала ПДП ЦАП;
10. *DaNumber* — зарезервирован.

Пример:

```
//В примере осуществляется ввод с четырех каналов АЦП одновременно с генерированием
//синусоидального сигнала на выходе ЦАПа с интервалом 100 мкс
#include <math.h>
main()
{
int I, *Data, Npoint=500, Channels[16]={0,1,2,3};
int Nch=4, Rate=100, DaNKadr=1000, DaNumber=0;
int AdDmm=1, DaDmm=1;

// установим указатель на старшую страницу памяти
Data=(int far *) 0б90000000;

// генерируем синусоидальный сигнал
for(I=0, I < 1000, I++) Data[I]=2000.*sin(I*M_PI/80. );

// запустим ввод с АЦП одновременно с выводом на ЦАП
DMA_ALL_DA(Data+1000, Npoint, Channels, Nch, Rate, AdDmm, Data, DaNKadr, DaDmm,
DaNumber);

// ждем нажатия на клавишу
while (!kbhit());
// выключим контроллер ПДП
DMA_OFF();
}
```

5.7.6. Синхронный вывод на ЦАП по каналу ПДП

void DADMASTREAM(int *Data, int Npoint, int Rate, int Dmm);

Назначение. Выводит массив отсчетов из массива Data на установленный ЦАП по каналу ПДП.

Параметры:

1. Data - целочисленный массив для данных;
2. Npoint - число выводимых отсчетов (1 <= NPoint <= 32768);
3. Rate - интервал вывода (от 1 до Oxffff);
4. Dmm - параметр, включающий режим автоинициализации. Если режим

автоинициализации будет включен, после вывода NPoint отсчетов плата вновь продолжит вывод из буфера, начиная с адреса Data до тех пор, пока канал ПДП и плата не будут остановлены при помощи функции DMA_OFF().

Пример:

```
main()
{
int i, *Data, Chan=0, NPoint=10000, Rate=10, Dmm=1;

//установим указатель на старшую страницу памяти
Data=(int far *) 0б90000000;

// генерируем синусоидальный сигнал
for(i=0; i < Npoint; i++) Data[i]=2000.*sin(i*M_PI/80.);

// установим первый ЦАП
SET_DA_NUMBER(0, 0);

// запустим ПДП на вывод на ЦАП
DADMASTREAM (Data, Npoint, Chan, Rate, Dmm);

// поскольку DMM = 1 ПДП будет выводить вечно на ЦАП
// пока его не остановят
// выводим до тех пор, пока не будет нажата клавиша
while (!kbhit());

// выключим плату и контроллер ПДП
DMA OFF();
}
```

5.7.7. *Функция выключения контроллера ПДП*

void DMA_OFF(void);

Назначение. Функция для выключения контроллера ПДП. Ее необходимо вызывать, когда работа с контроллером ПДП завершена. Если контроллер был переведен в режим ввода с автоинициализацией и не был выключен, то после выхода из программы, скорее всего, произойдет зависание компьютера.

5.7.8. *Функция проверки завершения ввода по ПДП*

int DMA_TEST(void);

Назначение. Возвращает ноль, если ввод по ПДП не завершен и единицу — в противном случае. Данную функцию можно применять только при вводе с выключенным режимом автоинициализации. При выключенном режиме автоинициализации счетчик канала ПДП устанавливается в -1 после завершения ввода, функция DMA_TEST() проверяет значение счетчика на -1 и, если оно не равно -1, то возвращает ноль. При включенном режиме автоинициализации счетчик ПДП никогда не становится равным -1, поэтому в этом режиме функция DMA_TEST() будет всегда возвращать ноль.

5.7.9.*Проверка базового адреса буфера для ПДП*

int DMA_ADDRESS_TEST(int *Data, int NPoint);

Назначение. Функция возвращает ноль, если буфер с базовым адресом Data и размером NPoint слов не пересекает страницу ПДП.

Параметры:

1. *Data* — проверяемый базовый адрес буфера;
2. *Npoint* — размер буфера.

Пример:

```
main()
{
int *Data;

// установим указатель на старшую страницу памяти
Data=(int far *) 0õ90000000,

// 32000 слов должны "влезть" в старшую страницу
i=DMA_ADDRESS_TEST(Data, 32000);
cprintf("\n\r Тест на 32000=%d", i);

// 64000 слов не должны "влезть" в старшую страницу
i=DMA_ADDRESS_TEST(Data, 64000);
cprintf("\n\r Тест на 64000=%d", i);
}
```

5.7.10. *Доступ к счетчику канала ПДП*

int DMA_COUNTER(void);

Назначение. Возвращает число слов, которое осталось ввести по ПДП (при однократном вводе после завершения ввода счетчик ПДП установится в -1). Первоначальное содержимое счетчика перед вводом первого слова равно числу передаваемых слов минус один.

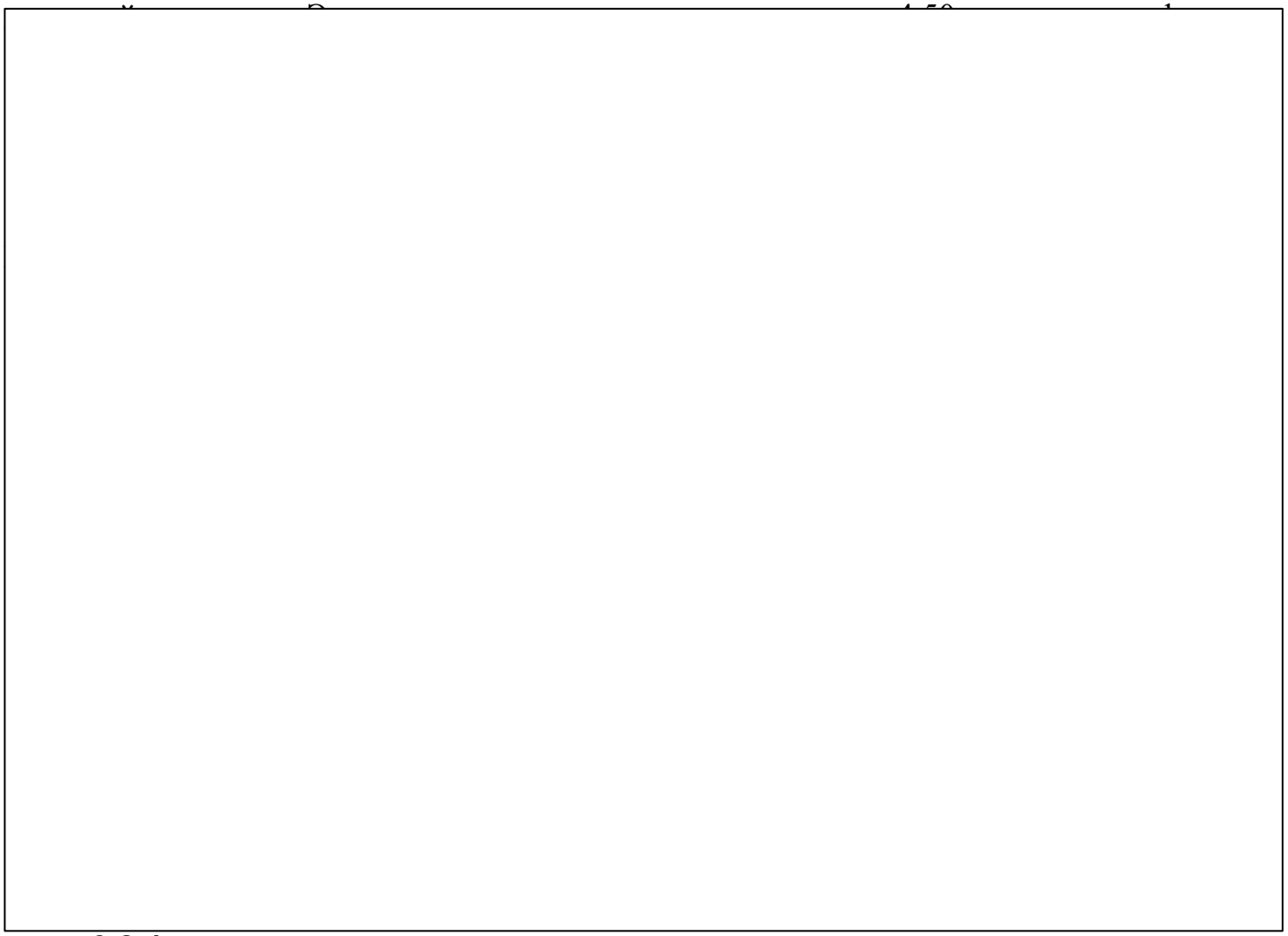
6. ПРАКТИЧЕСКИЕ И ЛАБОРАТОРНЫЕ РАБОТЫ ПО ВВОДУ-ВЫВОДУ АНАЛОГОВЫХ И ДИСКРЕТНЫХ СИГНАЛОВ С ПРИМЕРАМИ ПРОГРАММ

6.1. Изучение интерфейса и основных возможностей программы “Осциллоскоп”

6.1.1. ***Назначение программы.*** Программа “Осциллоскоп” предназначена для работы с аналого-цифровыми платами, выпускаемыми фирмой L-CARD. Программа позволяет осуществлять ввод/вывод аналоговых сигналов и их отображение на экране монитора, производить тестирование плат при наличии тестовой заглушки, сохранять введенные сигналы в файл, а также осуществлять спектральный анализ вводимых сигналов и печать графиков практически на любом принтере.

Программа “Осциллоскоп” и плата УВВС типа L-305 позволяют создать на базе персонального компьютера виртуальный цифровой осциллограф с широкими функциональными возможностями. Пример экранной формы этой программы представлен на рис.6.1.

Для работы программы необходимо наличие в компьютере 600 Кбайт свободной



6.2.1. Разработать программу, осуществляющую ввод сигналов через АЦП в программном режиме с отображением сигналов на экране. Сигналы подаются с внешнего генератора. Номера каналов АЦП и режим синхронизации задаются преподавателем.

Осуществить ввод сигналов с отображением их на экране с помощью программы “Осциллоскоп”.

6.2.2. Разработать программу, осуществляющую вывод сигнала через ЦАП в программном режиме с последующим отображением его на осциллографе. Форма сигнала задается преподавателем.

6.2.3. Разработать программу, осуществляющую вывод сигнала на ЦАП, с последующим вводом его через АЦП в программном режиме и отображением на экране. Форма сигнала и номер канала АЦП задаются преподавателем.

6.2.4. ***Пример программы:***

```
// Демонстрационная программа формирует синусоиду на выходе
// ЦАП, затем вводит этот сигнал через АЦП и отображает на экране

#include <function.h>
#include <conio.h>
#include <stdio.h>
#include <iostream.h>
#include <dos.h>
#include <graphics.h>
#include <math.h>

// Количество точек
#define NPoint    100
// Верхняя граница интервала
#define maxU      6*M_PI
// Нижняя граница интервала
#define minU    0

int
  gd = DETECT, gm, errorcode, y0, maxx;
float
  y, x, dx, di, i;

void main()
{
  INTR_SETUP(0);  // Установка номера прерывания платы
  if (PLATA_TEST())  // Если ошибка при проверке платы…
  {
     cout << "\nПлата не найдена или не загружен ее драйвер";
      exit(1);

  }
  else cout << "\nOK";
  initgraph(&gd, &gm, ""); // Инициализация графического режима
  x = 0;
// Настройка автомасштабирования
  maxx = getmaxx();
  y0 = getmaxy() / 2;
  dx = maxx / NPoint;
```

```
  di = (maxU - minU) / NPoint;

  while(!kbhit())
  {
     for(i = minU; i <= maxU; i+=di)
     {
        SET_DA_NUMBER(1, 0);
        OUTDA((int)2000*sin(i));
        delay(1);
        SETCHANNEL(1);
        y = y0 - SAMPLE() / 2048. * y0;
        lineto((int)x, (int)y);
        delay(100);
        x+=dx;
        if (x > maxx)
        {
            cleardevice();
            moveto(0, (int)y);
            x = 0;

        }
     if (kbhit()) break;

     }

  }
  closegraph();

}
```

6.3.Организация ввода-вывода аналоговых сигналов с использованием прерываний

6.3.1.Разработать программу, осуществляющую ввод сигналов через АЦП с заданным интервалом ввода и последующим генерированием прерываний с отображением вводимых сигналов на экране. Сигналы на АЦП подаются с внешнего генератора.

Осуществить ввод сигналов с отображением их на экране с помощью программы “Осциллоскоп”.

6.3.2. ***Пример программы:***

```
// Драйвер обработчик прерывания IRQ от платы АЦП
// при каждом вызове вводит ряд отсчетов с АЦП и выводит их на экран

// глобальная переменная
int GlobalVar=0, Nch=4;

// обработчик прерывания
void interrupt E_Interrupt(PARM)
{
```

```
 int i, j;

if(!GlobalVar)
for(i=0; i < Nch; i++)
{
// введем отсчет с АЦП
j=READ_DATA();

// отобразим его на экран
cprintf("\n\r Отсчет # %5d cАЦП = %5d", i+1, j);
}

// сбросим флаг прерывания на плате АЦП и в компьютере
RESET_IRQ();
}

// основная функция
main()
{
int Channels[16]={0,1,2,3};

// установим интервал между прерываниями 10000 мкс и
//адрес обработчика E_Interrupt
SOFT_SIMPLE_INTR( 10000, E_Interrupt, Channels, Nch );

// ждем пока не нажата клавиша
while(!kbhit());

// сообщим драйверу обработчику, что мы собираемся
// выключить прерывания
GlobalVar=1;

// запретим прерывания от платы и восстановим
// контроллер прерываний
STOP_INTR(),
}
```

6.4. Организация ввода вывода аналоговых сигналов в режиме прямого доступа к памяти

6.4.1. Разработать программу, осуществляющую ввод сигнала через АЦП в режиме ПДП с отображением его на экране. Сигнал подается с внешнего генератора.

Осуществить ввод сигналов с отображением их на экране с помощью программы “Осциллоскоп”.

6.4.2. Разработать программу, осуществляющую вывод сигнала через ЦАП в режиме ПДП с отображением этого сигнала на экране соседнего компьютера.

6.4.3. Разработать программу, осуществляющую вывод сигнала через ЦАП в режиме ПДП с последующим отображением его на осциллографе. Форма сигнала задается преподавателем.

6.4.5.*Пример программы:*

```
//Данный пример осуществляет ввод и запись на диск в реальном масштабе
//времени 16 каналов платы АЦП в течении 4 секунд на частоте ввода 5 кГц
//(интервал ввода 200 мкс)

main()
{
int i, *Data, Channels[16], NPoint=2000, Rate=200;
int Dmm=1, counter, Nch=16;
FILE *fp;

// откроем файл (желательно на нефрагментированном диске)
fp=fopen("test.dat", "wb");
if(fp==NULL) exit(1);

// установим указатель на старшую страницу памяти
Data=(int far *) 0õ90000000,

// запустим ПДП на ввод 16 каналов АЦП с интервалом
// 200 мкс между кадрами
DMAALL(Data, NPoint, Channels, Nch, Rate, Dmm);

// вводим 2000*10*200мкс=4секунды
for(counter=0, counter < 10, counter++)
{
// ждем заполнения первой половины буфера
while(DMA_COUNTER() > 1000*16);

// сохраним первую половину в файл
fwrite(Data, 1000*16, 2, fp);

// ждем заполнения второй половины буфера
while(DMA_COUNTER() < 1000*16);

// запишем в файл
fwrite(Data+1000*16, 1000*16,2, fp);
}

// выключим плату и контроллер ПДП
DMA_OFF();
fclose(fp);
}
```

6.5. Организация ввода-вывода дискретных сигналов

6.5.1.С помощью тестовой заглушки проверить работу каналов ввода-вывода дискретных (цифровых) сигналов.

6.5.2.Разработать программы, осуществляющие следующий обмен:

на одном компьютере вывод 8-разрядных цифровых сигналов;
на соседнем — ввод этих сигналов и отображение их на экране монитора.

6.5.3. ***Пример программы:***

// Данная программа осуществляет вывод восьми нулей и затем восьми единиц

```
main()
{
// выведем все нули
OUTBYTE_305(0);

// выведем все единицы
OUTBYTE_305(0xff);
}
```

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Сопряжение датчиков и устройств ввода данных с компьютерами IBM PC / Под ред. У. Томпкинса, Дж. Уэбстера. – М.: Мир, 1992. – 592 с.
2. Васин Н.Н., Мохонько В.П. Системы сбора информации на железнодорожном транспорте: Учебное пособие. – Самара: СамИИТ, 2001. – 120 с.
3. Краус М., Кучбах Э., Вошни О.-Г. Сбор данных в управляющих вычислительных системах. – М.: Мир, 1987. – 294 с.
4. Адаптивные системы сбора и передачи аналоговой информации. Основы теории / А.Н. Дядюнов, Ю.А. Онищенко, А.И. Сенин. – М.: Машиностроение, 1988. – 288 с.
5. Гусев С.А. Краткий экскурс в историю промышленных сетей //Современные технологии автоматизации. – 2000. - №4. – С.78-84.
6. Гэри А. Минтчел. Ethernet в управлении производственными процессами //Мир компьютерной автоматизации. – 2000. - №3. – С.48-52.
7. Аристова И.И., Корнеева А.И. Промышленные программно-аппаратные средства на отечественном рынке АСУТП. – М.: НАУЧТЕХЛИТИЗДАТ, 2001. – 402 с.
8. Засов В.А. Основы микропроцессорной техники: Учебное пособие. – Самара: СамИИТ, 2001. – 215 с.
9. Гук М. Аппаратные средства IBM PC: Энциклопедия. - 2-е изд. – СПб.: Питер, 2001. – 928 с.
10. Сорокин С.А. IBM PC в промышленности //Современные технологии автоматизации. – 1996. - №1. – С.6-15.
11. Свиридов В. Современные интегрированные системы. Шины и объединительные магистрали. 20 лет VME bus //Мир компьютерной автоматизации. – 2001. - №4. – С.11-18.
12. Сысоев А.Д. Мезонины: что сегодня? //Мир компьютерной автоматизации. – 2001. - №4. – С.19-24.
13. Разработка устройств сопряжения для персонального компьютера типа IBM PC: Практическое пособие / Ю.В. Новиков, О.А. Калашников, С.Э. Гуляев. Под ред. Ю.В. Новикова. – М.: ЭКОМ, 1997. – 224 с.
14. Локотков А. Интерфейсы последовательной передачи данных. Стандарты EIA RS-422A/RS-485 //Современные технологии автоматизации. – 1997. - №3. – С.110-119.
15. Кулаков В. Программирование на аппаратном уровне. Специальный справочник. – СПб: Питер, 2001. – 496 с.
16. Белецкий Я. Турбо Си++: Новая разработка: Учебное пособие. – М.: Машиностроение, 1994. – 400 с.
17. Ан П. Сопряжение ПК с внешними устройствами. – М.:ДМК Пресс, 2001. – 320 с.
18. Плата L-305. Техническое описание и инструкция по эксплуатации. – Фирма L-CARD, 1998. – 49 с.

ДОПОЛНИТЕЛЬНАЯ ИНФОРМАЦИЯ

Более подробную техническую информацию по многочисленным конкретным УВВС можно получить, посетив, например, следующие сайты: www.prosoft.ru
www.rtsoft.ru
www.icos.ru
www.cta.ru
www.ni.com/russia
www.lcard.ru
www.mka.ru