Министерство образования и науки Российской Федерации

Федеральное агентство по образованию

ГОУ ВПО

Череповецкий государственный университет

Институт информационных технологий

Кафедра автоматизации и систем управления

Расчетно-пояснительная записка к курсовой работе

**Микропроцессорная система дрессировочного стана**

по дисциплине

Проектирование микропроцессорных систем

Выполнили: студентки группы 1АП-41

Новожилова Е.В., Лушкина Н.С.

Научный руководитель: к. т. н. Харахнин К.А.

Череповец 2010 г.

Содержание

Введение

1. Технологическая часть

2. Аппаратные средства микропроцессорной системы

3. Описание технических средств для реализации проекта

4. Описание программных средств для реализации проекта

5. Программа для работы микропроцессорного контроллера на языке ассемблер

6. Программа для персонального компьютера на языке с++ в среде Borland C++Builder 6

7. Графическая часть:

А. Электрическая принципиальная схема микропроцессорной системы

Б. Алгоритмы работы системы

В. Окно программы для персонального компьютера

Заключение

Список литературы

Введение

В ходе курсового проекта разрабатывается микропроцессорная система для измерения относительной деформации полосы на дрессировочном стане.

Дрессировочный стан — прокатный стан, служащий для отделки (дрессировки) тонких металлических листов и ленты посредством холодиной прокатки с малыми обжатиями (около 2—3%). Характерной особенностью дрессировочных станов является их несколько облегчённая конструкция и меньшая, по сравнению с другими станами холодной прокатки, мощность привода. Дрессировочные станы конструируются большей частью как одноклетьевые, двухвалковые или четырёхвалковые. Станы последнего типа отличаются лучшими условиями работы подшипников валков и меньшим расходом энергии. По обеим сторонам станов для дрессировки ленты устанавливаются моталки для намотки ленты. Для дрессировки листов часто строятся многоклетьевые дрессировочные станы линейного типа. Такой стан представляет, по существу, несколько станов, установленных рядом и работающих на одном приводе; каждый из этих станов — одноклетьевой или многоклетьевой. В последнем случае клети расположены последовательно, одна за другой. Процесс дрессировки аналогичен процессу холодной прокатки, его осуществляют в один пропуск с меньшими обжатиями и без подачи эмульсии или технологической смазки. При этом валки профилируют подогревом с помощью специальных горелок.

При дрессировке, однако, практически нельзя применять обжатия больших степеней, так как в противном случае пластические свойства материала из-за упрочнения падают. При дрессировке повышается твердость и снижается вязкость металла. Излишняя дрессировка вызывает после деформирования склонность стали к старению, а это приводит к ухудшению физико-механических свойств металла. При дрессировке предел прочности стали повышается очень незначительно, твердость несколько возрастает, а относительное удлинение уменьшается. Что касается условного предела текучести, то его изменение при дрессировке носит сложный характер. Так, для малоуглеродистых сталей предел текучести при степени деформации от 05 до 12 % уменьшается, а при дальнейшем увеличении степени деформации начинает возрастать. При дрессировке с обжатием порядка 0% в поверхностных слоях полосы происходит отрыв дислокаций, благодаря чему предел текучести при растяжении соответственно уменьшается. При дрессировке электротехнических сталей подача смазки позволяет увеличить обжатие до 14% (вместо 5 - 6% при отсутствии смазки) без перегрузки стана и осуществить промышленное производство этих сталей с однократной дрессировкой. Станы для дрессировки стали имеют одну или две клети: кварто-кварто или кварто-дуо. При дрессировке на двухклетевых непрерывных станах суммарная деформация делится на два обжатия. Обжатие в первой клети всегда больше, а с помощью второго (меньшего по величине) обжатия осуществляется правка полосы и уменьшается ее разнотолщинность.

1. Технологическая часть

Схема процесса дрессировки представленна на рис.1.

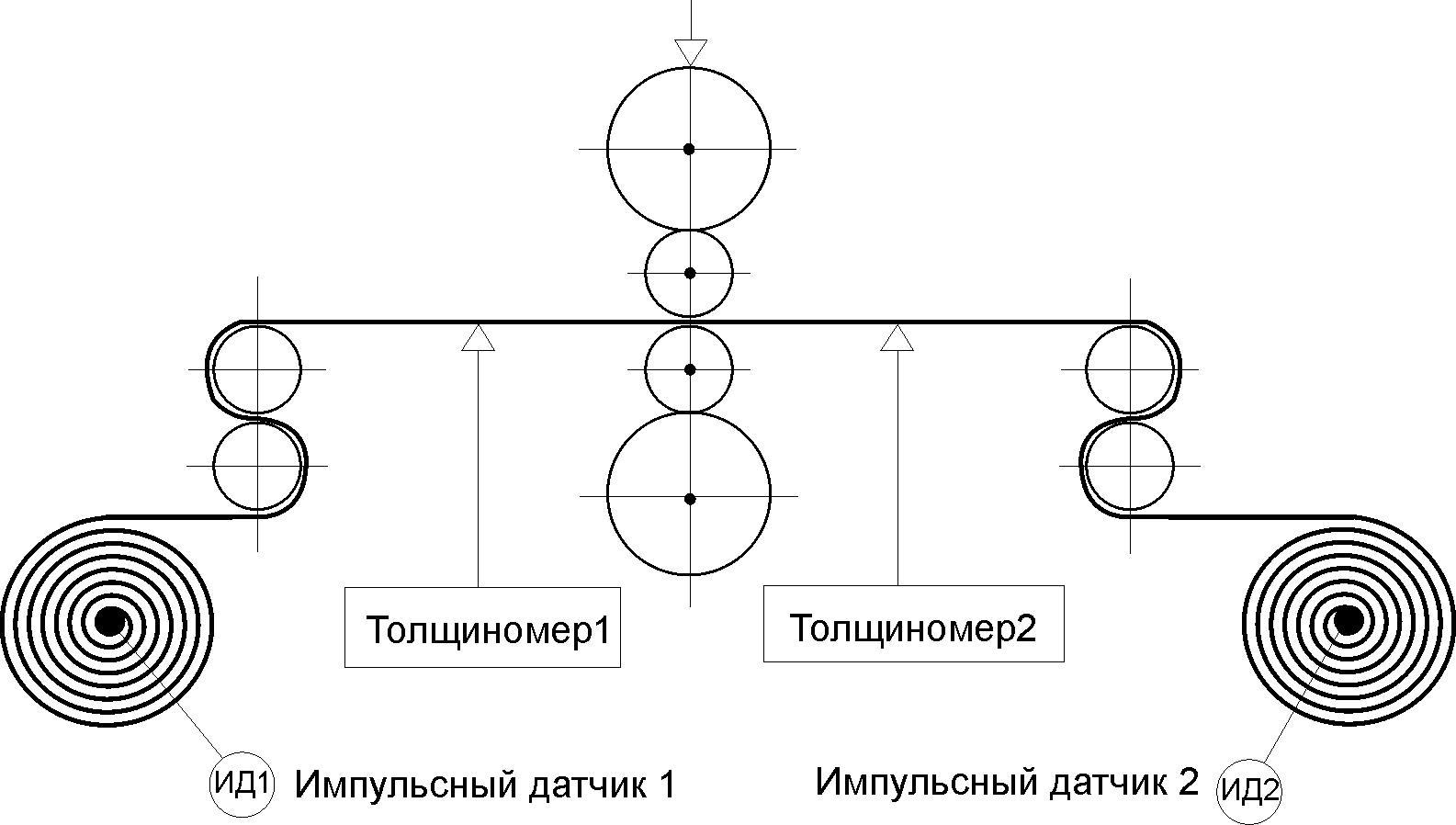


Рис.1. Схема технологического процесса

Дрессировку металла производят после его термической обработки. Отожженный лист поступает на дрессировочный стан, на котором производится прокатка с малым обжатием, порядка 0,5-3%. Дрессировка необходима для того, чтобы получить стальной лист, относительно мягкий внутри и твёрдый на поверхности, пригодный для глубокой штамповки, и, кроме того, произвести окончательную отделку поверхности холоднокатаного отожженного листа. Для дрессировки служат специальные дрессировочные станы, нереверсивные, мало отличающиеся в конструктивном отношении от обычных станов холодной прокатки.

Цель дрессировки является улучшение поверхности стали, имеющей после горячей прокатки и травления много неровностей, а также для создания поверхностного упрочнения (наклепа), что имеет существенное значение для листов, подвергаемых в дальнейшем холодной штамповке, так как при недостаточно жесткой поверхности металла на ней могут появиться линии сдвига (следы от пересечения плоскостей сдвигов с внешней поверхностью листа).

В процессе дрессировки деформация листа должна быть незначительной, и необходимо проводить четкий контроль относительной деформации полосы на дрессировочном стане. Расчет этого параметра можно провести следующим образом:

,



где - количество импульсов на выходе клети, полученных от импульсного датчика, полученных с импульсного датчика, - количество импульсов, полученных с импульсного датчика на входе.



Для измерения толщины используется радиоизотопный толщинометр, формирующий на своем выходе аналоговое напряжение в диапазоне 0-2,5В. Сигналы с датчиков подаются на соответствующие входы микропроцессорной системы (SDK), которая обрабатывает полученные данные и отправляет их по последовательному порту в управляющий компьютер, который производит необходимые расчеты.

2. Аппаратные средства микропроцессорной системы

В качестве микропроцессорной системы для проведения проектных работ используется учебный стенд SDK1.1 на базе микроконтроллера ADuC842 с 12-разрядными АЦП и ЦАП, встроенным высокоскоростным МК с ЭРПЗУ (FLASH) на 62 кб. Структурная схема стенда представлена на рис. 2.

Учебный лабораторный комплекс SDK-1.1 предназначен для освоения архитектуры и методов проектирования:

• Систем на базе микропроцессоров и однокристальных микроЭВМ:

• Встраиваемых контроллеров и систем сбора данных:

• Периферийных блоков вычислительных систем:

• Подсистем ввода-вывода встраиваемых систем.

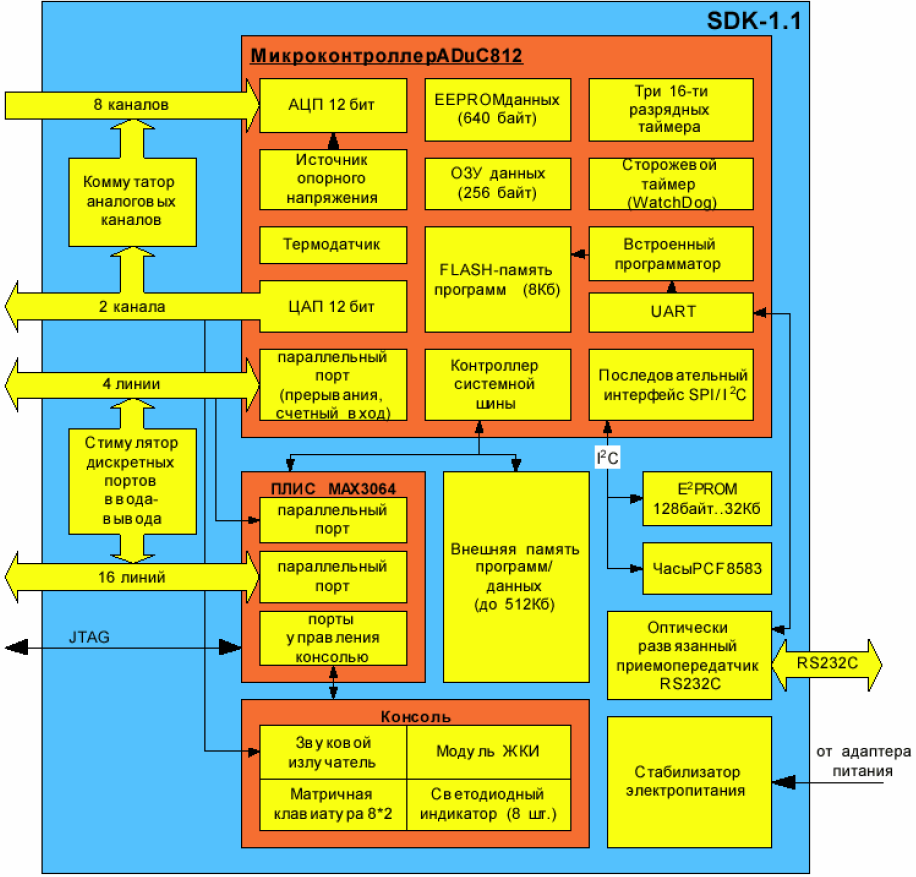


Рис 2. Структурная схема стенда

**Описание архитектуры стенда**

Основу лабораторного комплекса микропроцессорный стенд SDK-1.1 на базе процессора ADuC812 или ADuC842.

**Аппаратные блоки стенда:**

* вычислительное ядро на основе 8-разрядного процессора ADuC812 (ADuC842), FLASH-памяти и внешнего ОЗУ объемом до 512Кб. Процессорное ядро является клоном ядра Intel MCS51.
* COM-порт для связи с ПК (интерфейс RS232C);
* жидкокристаллический индикатор для вывода текста с поддержкой русского алфавита (2 строки по 16 символов);
* программируемая логическая интегральная схема (ПЛИС) семейства MAX3000A фирмы Altera. В очень упрощенном виде ПЛИС представляет собой набор макроячеек и механизм для организации связи между ними. Микросхема EPM3064A содержит 64 макроячейки. В стенде SDK-1.1 MAX3064A используется как расширитель портов ввода-вывода. Микросхема MAX3064A подключена к внешней шине ADuC812
* порт дискретного ввода-вывода, предназначенный для ввода и вывода информации, представленной в двоичном виде. Сигнал на входе или выходе дискретного порта может принимать значение логического нуля или единицы. В SDK-1.1 дискретные порты выведены на разъем J3. Эти порты можно использовать для подключения модулей SDX или каких- либо других внешних устройств. Кроме этого, к дискретным входам-выходам подключены DIP переключатели, позволяющие задавать фиксированные значения сигналов на входах.
* аналоговый порт ввода на базе 8-миканального 12-тиразрядного АЦП;
* аналоговый порт вывода на основе двух 12-тиразрядных ЦАП;
* EEPROM-память емкостью 640 байт;
* второй блок EEPROM-памяти емкостью до 32 Кб, подключенный к вычислителю через интерфейс I2C;
* три 16-тиразрядных таймера-счетчика с внешними счетными входами (возможностью подачи сигналов через переключатели стенда) и блоком захвата/сравнения для измерения параметров и/или формирования дискретных сигналов;
* сторожевой таймер;
* линейку из 8 сигнальных светодиодов;
* акустический пьезокерамический излучатель;
* матричная клавиатура на 16 клавиш организована в виде матрицы 4x4. Доступ к колонкам и рядам организован как чтение/запись определенного байта внешней памяти (4 бита соответствуют 4 колонкам, другие 4 бита - рядам). При нажатии на кнопку, происходит изменение значения сигнала на входе соответствующего ряда с единицы на ноль
* часы/календарь с возможностью подключения внешней батареи питания.

**Распределение памяти**

Стандартная для архитектуры 8051 структура внутренней памяти представлена четырьмя банками по 8 регистров общего назначения (диапазоны адресов 00h-07h, 08h-0Fh, 10h-17h, 18h-1Fh), битовым сегментом (20h-2Fh), свободным участком 30h-7Fh, областью размещения **SFR** (регистров специального назначения) 80h-FFh, доступной при прямой адресации, и свободной областью 80h-FFh, доступной при косвенной адресации.

Внешняя память SDK-1.1 разбита на следующие области: AduC812 Flash/EE, SRAM,MAX.

**Flash/EE.** Это область, в которой располагается таблица векторов прерываний и резидентный загрузчик файлов в формате HEX в память SRAM.

**SRAM.** Статическая память SRAM в SDK-1.1 имеет страничную организацию (максимум 8 страниц по 64 К) и условно разделяется на две области. Первая занимает младшие 64 Кбайт (страница 0) и доступна для выборки команд микроконтроллером ADuC812. Таким образом, программы могут располагаться только в этих младших 64 К адресного пространства. Остальные страницы доступны только для размещения данных. Для адресации ячейки памяти определенной страницы необходимо записать номер страницы в регистр специального назначения DPP ADuC812 (адрес 84h).

**MAX.** В младших адресах восьмой страницы адресного пространства (080000h- 080007h) располагается 8 ячеек-регистров ПЛИС MAX8064 (MAX8128). Эта область предназначена для взаимодействия с периферийными устройствами стенда.

**Сопряжение с ПК**

Сопряжение стенда с компьютером, необходимое для программирования микроконтроллера стенда и передачи данных между ПК и SDK, осуществляется с помощью последовательного порта (RS-232).

Требования к ПК:

* IBM-совместимый компьютер с наличием последовательного СОМ-порта.
* Операционная система: Windows 95/98/ME/2000/XP/Vista.

3. Описание технических средств для реализации проекта

Следует отметить, что основное внимание в данном курсовом проекте уделено разработке программного обеспечения для микропроцессорного учебного стенда SDK и ПК, а технологический процесс моделируется упрощенно с помощью имеющихся на стенде устройств.

Так, например, в качестве датчика толщинометра использован цифро-аналоговый преобразователь, замкнутый на аналого-цифровом преобразователе. На выходе ЦАП формируется напряжение ≈1.1 В, после преобразований в АЦП и передаче значения через последовательный порт в расчетах в программе на С++ используется значение ≈1, которое использовано в качестве толщины на входе клети. Для моделирования обоих импульсных датчиков угловой скорости используется клавиатура SDK: кнопка «1» прибавляет один импульс, кнопка «2» - десять импульсов, кнопка «3» - сто импульсов. Превышение заданного значения относительной деформации сигнализируется зажиганием нечетных светодиодов (через один), звуковым сигналом и выводом необходимой информации на дисплей, при этом выводится значение разности между текущим и заданным значением деформации. Передача данных между SDK и ПК осуществляется по последовательному COM-порту (RS-232). В качестве ПК используется IBM-совместимый компьютер с наличием последовательного СОМ-порта и операционной системой семейства Windows.

микропроцессорный контроллер деформация дрессировочный

4. Описание программных средств для реализации проекта

Для создания программы на языке C++ использовалась среда разработки Borland C++Builder 6, а также набор программ для компиляции и загрузки кода на языке Ассемблера в стенд SDK.

Для взаимодействия стенда с ПК и работы с интерфейсом RS-232 использована библиотека ComPortLirary 3.10.

Для создания, компиляции и загрузки в стенд кода используется следующий набор программ:

**Ассемблер А51** преобразовывает исходный ассемблерный код в перемещаемый объектный модуль. Ассемблер А51 полностью поддерживает исходный код на языке Intel ASM-51. Ассемблер А51 поддерживает все микроконтроллеры семейства 8051. Набор регистров специального назначения (SFR) является стандартным. Вызов:

**А51 sourcefile [directives] А51 @ commandfile**

**sourcefile** Имя исходного файла на ассемблере.

**commandfile** Имя файла, содержащего командную строку ассемблера, включающую sourcefile и directives. Вы можете использовать командный файл для более простой компоновки исходного файла или в том случае, когда все директивы не помещаются в командной строке.

**directives** Параметры.

**Загрузчик/компоновщик BL51** объединяет один или несколько объектных модулей в один исполняемый файл. Компоновщик также разрешает внешние и глобальные ссылки и назначает абсолютные адреса перемещаемым сегментам программ. Компоновщик автоматически выбирает подходящие библиотеки рабочих программ и связывает только нужные модули библиотек. Вызов:

**BL51 inputlist [TO outputfile] [directives]**

**L51 inputlist [TO outputfile] [directives] BL51 @commandfile**

**L51 @commandfiIe**

**sourcefile** Имя исходного объектного файла, созданного компилятором А51 или С.

**commandfile** Имя файла, содержащего командную строку компилятора, включая sourcefile и directives. Можно использовать командный файл для более простой компоновки исходного файла или в том случае, когда все директивы не помещаются в командной строке.

**directives** Параметры элементов управления.

Шестнадцатеричный **конвертер ОН51** конвертирует объектные модули в абсолютных адресах в шестнадцатеричные файлы в формате Intel. Модули перед этим создаются с помощью компоновщика BL51 или конвертера ОС51.

**ОН51 absfile |HEXFILE (hexfile)**

**absfile**Имя объектного модуля, построенного в абсолютных адресах.

**hexfile**Имя шестнадцатеричного файла в формате Intel, который должен быть создан.

Для загрузки программы в МК используется загрузчик **T2.**

5. Программа для работы микропроцессорного контроллера на языке ассемблер

DPPDATA84h; адрес указателя страницы данных (data; page ponter)

ADCCON1 DATA0EFH; регистр управления АЦП

ADCCON2 DATA 0D8H; регистр управления АЦП

ADCCON3 DATA 0F5H; регистр управления АЦП

ADCDATALDATA 0xD9; регистр младшего байта оцифрованных; данных АЦП

ADCDATAH DATA0xDA; регистр старшего байта оцифрованных данных; АЦП

DACCON DATA0xFD; регистр управления ЦАП

DAC0L DATA0xF9; младший регистр данных ЦАП содержат

DAC0H DATA0xFA; старший регистр данных ЦАП содержат

PLLCONDATA0D7H; регистр управления частотой контроллера

TIMECONDATA0A1H; адрес счетчика временных интервалов

T3FD DATA0x9D; регистры настройки Таймера 3

T3CON DATA0x9E; регистр управления таймером 3

ORG0000h; начало области, в которой будет располагаться код

JMPSTART; переход на инициализацию системы

ORG0023H; вектор прерывания от последовательного порта (UART)

JMPPOSL\_PORT; переход на подпрограмму обработки прерывания

START:

MOVPLLCON,#3; настраиваем частоту ядра

MOVTMOD,#00H; T/C0 -таймер, режим0

MOVT3CON,#83H; устанавливаем скорость

MOVT3FD,#2DH; передачи – 9600 бод

MOVSCON,#50H; настройка последовательного порта; 0101 0000 (01 - 8-битный режим, 01 - режим 0, прием разрешен, 00 - биты используемые в режимах 1 и 2, 0 - флаг передачи; последов. порта, устанавливается; аппаратно после передачи 8-го бита, д.б. сброшен программно, 0 - флаг приема; последов порта, установливается аппаратно после приема 8-го бита, д.б. сброшен; программно)

MOVIE,#00010000B; разрешение прерывания от UART

SETBEA; разрешение прерываний

MAIN:; основная программа

CLRF0;

MOVR3,#0;

SCAN:; сканирование и проверка регистра R3;

K0:

CJNER3, #2,K1; если R3=2, то

CALLDAC\_ADC; перейти на подпрограмму ЦАП-АЦП,

MOVR3,#0; затем записать в R3 0; иначе перейти на метку К1

K1:

CJNER3,#1, SCAN; если R3=1, то дальше сканировать клавиши 1,2 и 3, в портивном случае перейти на метку SCAN;

MOV R5,#0FEH;

MOV R7,#00H; нам нужен 0-й байт 8-й страницы 08"00"00"Н

MOV R6,#00H; 08"00"00

CALLWRITE;

CALLCHECK123;

CJNE A,#00H,K2; если клавиша 1 не нажата, сканировать клавишу 2

CALLWAIT\_CHECK1; проверка на "залипание" клавиши 1

MOVSBUF,#01111000b; записываем в буфер значение символа;"x", бит TI устанавливается в 1,; инициируя прерывание по посл. порту

CALL DELAY; задержка

K2:; проверка нажатия клавиши 2

MOVR5,#0FDH;

MOV R7,#00H; нам нужен 0-й байт 8-й страницы 08"00"00"Н

MOV R6,#00H; 08"00"00

CALLWRITE;

CALLCHECK123;

CJNE A,#00H,K3; если клавиша 2 не нажата, сканировать клавишу 3

CALLWAIT\_CHECK2; проверка на "залипание" клавиши 2

MOVSBUF,#01111001b; записываем в буфер значение символа; "y", бит TI устанавливается в 1,; инициируя прерывание по посл. порту

CALL DELAY; задержка

K3:; проверка нажатия клавиши 3

MOVR5,#0FBH;

MOV R7,#00H; нам нужен 0-й байт 8-й страницы 08"00"00"Н

MOV R6,#00H; 08"00"00

CALLWRITE;

CALLCHECK123;

CJNE A,#00H,SCAN; если клавиша 3 не нажата, перейти на; метку SCAN

CALLWAIT\_CHECK3; проверка на "залипание" клавиши 2

MOVSBUF,#01111010b; записываем в буфер значение символа; "z", бит TI устанавливается в 1,; инициируя прерывание по посл. порту

CALLDELAY; задержка

JMP SCAN; переход на метку SCAN

CHECK123:; функция определения, нажата ли клавиша (1,2 или 3)

CALL READ; читаем байт по адресу 080000Н

ANLA,#10H

RET

WAIT\_CHECK1:

CALL READ;

CJNE A,#0FEH,WAIT\_CHECK1; ждать, если клавиша 1 зажата

RET

WAIT\_CHECK2:

CALL READ;

CJNE A,#0FDH,WAIT\_CHECK2; ждать, если клавиша 2 зажата

RET

WAIT\_CHECK3:

CALL READ;

CJNE A,#0FBH,WAIT\_CHECK3; ждать, если клавиша 3 зажата

RET

BUZZ:; Для управления динамиком в процедуре использованы таймер и регистр ПЛИС ENA (адрес;080004h).;2-4 биты регистр ENA управляют величиной напряжения на динамике, т.е. позволяют; задавать громкость звука. ;Для формирования звука генерируются прямоугольные импульсы с заданным периодом.

MOVPLLCON,#5;

MOVR1,#255; задаем продолжительность сигнала

MOVR2,#5;

AGAIN:

CLRTR0; выключаем таймер

MOVTH0,#00H;инициализация таймера

MOVTL0,#00H;

SETBTR0; включаем таймер

MOVR5,#010H; записываемое число в регистр DPTR,

;соответствует высокому уровню напряжения

MOVR7,#04H; нам нужен 4-й байт 8-й страницы 0800"04"Н

MOVR6,#00H; 08"00"04

REP1:

CALLWRITE; включаем сигнал

JNBTF0,REP1; удерживаем значение в течение полупериода

CLRTR0; выключаем таймер

MOVTH0,#00H; инициализация таймера

MOVTL0,#00H;

SETBTR0; включаем таймер

MOVR5,#08H; записываемое число в регистр DPTR,; соответствует низкому уровню напряжения

MOVR7,#04H; нам нужен 4-й байт 8-й страницы 08"00"04"Н

MOVR6,#00H; 08"00"04

REP2:

CALLWRITE; Выключаем сигнал, 0-е напряжение

JNBTF0,REP2; удерживаем значение в течение полупериода

CLRTR0; выключаем таймер

DJNZR1,AGAIN;

DJNZR2,AGAIN;

MOVPLLCON,#3;

RET;

LCD\_Putch:; вывод символа на дисплей, код которого

;передается через регистр R5

MOVR7,#01H; DATA\_IND ( 080001H)

MOVR6,#00H;

CALLWRITE; устанавливаем данные на шине данных ЖКИ

MOVR5,#05H; 0-й бит C\_IND отвечает за строб, 1-й бит –

;запись/чтение , 2-й бит за данные/команды, 05H=0000 0101B - запись данных и;установка строба

CALLSTROBE; строб

RET;

STROBE:; подача сигнала Е интерфеса ПЛИС <-> ЖКИ на; время не менее 500 нс, одновременно подается; сигнал R/W=0 запись, RS=1 - данные

MOVR7,#06H; C\_IND ( 080006H)

MOVR6,#00H;

CALLWRITE; установка строба

MOVR5,#0FEH

MOVR7,#06H; C\_IND ( 080006H)

MOVR6,#00H;

CALLWRITE; сброс строба

CALLDELAY; задержка на время исполнения команды

RET;

DELAY:; процедура задержки

MOVR0,#100;

L1:

MOVR1,#100;

L2:

DJNZR1,L2;

DJNZR0,L1;

RET;

CLEAR:; очистка дисплея

MOVR5,#01H;

MOVR7,#01H; DATA\_IND ( 080001H)

MOVR6,#00H;

CALLWRITE; устанавливаем данные на шине данных ЖКИ

MOVR5,#01H; команда очистки дисплея

CALLSTROBE; строб

MOVR7,#07H; нам нужен 7-й байт 8-й страницы 0800"07"Н

MOVR6,#00H; 08"00"07

MOVR5,#00000000B; инициализация светодиодов через один

CALLWRITE;

CLRF0;

RET;

POSL\_PORT:; наступило прерывание, бит TI или RI равен 1

JBRI,WAIT\_IN; если бит RI - принимаем данные

JBTI,WAIT\_OUT; если бит TI - пересылаем данные

WAIT\_IN:

CLRRI; сбрасываем флаг

MOVR5,SBUF; записываем в буфер значение R5

CJNER5,#61H,E1; если пришел символ 'a', то

MOVR3,#1; записать в R3 1

JMPE; переход на метку Е

E1:

CJNER5,#70H,E2; если пришел символ 'р', то

MOVR3,#2; записать в R3 2

JMPE; переход на метку Е

E2:

CJNER5,#73H,E3; если пришел символ 's', то

CALLCLEAR; очистить дисплей и погасить светодиоды

JMPE; переход на метку Е

E3:; если не пришел ни символ 'a', ни 'p', ни 's', то

MOVA,R5; это означает, что последовательно начали передаваться

MOVR4,A; значения, составляющие разность между текущим и заданным значениями деформаций

JBF0,E3a; если бит F0=0, тогда однократный

CALLSTOP; вывод информации: "Деформация Превышение: %"

E3a:

SETBF0;

CALLLCD\_NA; вывод символов, определяющих отклонение от; заданной деформации

E:

RETI;

WAIT\_OUT:; подпрограмма пересылки значения в послед. порт

CLR TI;

RETI;

DAC\_ADC:

CALLDELAY;

MOVDACCON,#01101101B; настройка регистра ЦАП; 0-12 битный режим,; 11-бит выбора диапазона, 0-Vdd(в); 01-выход ЦАП1=0 в, выход ЦАП0-; соответствует коду; 1-выходы ЦАПов изменяются сразу, как; только данные попадают в регистры; 01-ЦАП1 выключен, ЦАП0 включен

MOVADCCON1,#10101100B;10-дежурный режим, если не выполняется; преобразование; 10-коэф. деления тактовой частоты на 4; 11-число тактов задержки=4; 0-Бит запуска преобразования от Таймера 2; 0-бит разрешения внешнего запуска АЦП

MOVDAC0H,#00000011B;переслать значение в старший байт ЦАП

MOVDAC0L,#10000100B;переслать значение в мл. байт; (т.е. 1.1/5\*4095 = 909, т.е. 00000011 10000100)

MOVADCCON2,#00010000B;SCONV=1-Бит однократного преобразования; После того, как цикл завершился, бит; автоматически сбрасывается в 0

ADC\_WAIT:

MOVA,ADCCON3;

JBACC.7,ADC\_WAIT;Флаг занятости ЦАП (только для чтения).; Устанавливается аппаратно на; время цикла преобразования или; калибровки. Автоматически сбрасывается; ядром в конце преобразования или; калибровки.

MOV A, ADCDATAH; записать данные из старшего регистра АЦП в А

ANL A,#00001111B;стереть код канала АЦП, хранящегося; в старшем байте

MOV SBUF, A; переслать значение

CALL DELAY; задержка

MOV A,ADCDATAL; записать данные из младшего регистра АЦП в А

MOV SBUF, A; переслать значение

CALL DELAY; задержка

RET;

STOP:; подпрограмма окончания процесса

MOV R5,#11100000B; Д

CALLLCD\_Putch;

MOV R5,#01100101B; е

CALLLCD\_Putch;

MOV R5,#11100100B; ф

CALLLCD\_Putch;

MOV R5,#01101111B; о

CALLLCD\_Putch;

MOV R5,#01110000B; р

CALL LCD\_Putch;

MOV R5,#10111100B; м

CALLLCD\_Putch;

MOV R5,#01100001B; а

CALLLCD\_Putch;

MOVR5,#11100101B; ц

CALLLCD\_Putch;

MOVR5,#10111000B; и

CALLLCD\_Putch;

MOVR5,#11000111B; я

CALLLCD\_Putch;

MOVR5,#0C0H; сдвиг курсора на следующую строку

MOVR7,#01H; DATA\_IND ( 080001H)

MOVR6,#00H;

CALLWRITE; устанавливаем данные на шине данных ЖКИ

MOVR5,#01H; команда очистки дисплея

CALLSTROBE; строб

MOVR5,#10101000B; П

CALLLCD\_Putch;

MOVR5,#01110000B; р

CALLLCD\_Putch;

MOVR5,#01100101B; е

CALLLCD\_Putch;

MOVR5,#10110011B; в

CALLLCD\_Putch;

MOVR5,#11000011B; ы

CALLLCD\_Putch;

MOVR5,#11000001B; ш

CALLLCD\_Putch;

MOVR5,#01100101B; е

CALLLCD\_Putch;

MOVR5,#10111101B; н

CALLLCD\_Putch;

MOVR5,#10111000B; и

CALLLCD\_Putch;

MOVR5,#01100101B; е

CALLLCD\_Putch;

MOVR5,#00111010B;:

CALLLCD\_Putch;

MOVR5,#0CFH;; установка крайнего правого положения во второй строке ЖКИ

MOVR7,#01H; DATA\_IND ( 080001H)

MOVR6,#00H;

CALLWRITE; устанавливаем данные на шине данных ЖКИ

MOVR5,#01H; команда очистки дисплея

CALLSTROBE; строб

MOVR5,#00100101B; %

CALLLCD\_Putch;

MOVR5,#0CBH;установка положения после ':' во второй строке ЖКИ

MOVR7,#01H; DATA\_IND ( 080001H)

MOVR6,#00H;

CALLWRITE; устанавливаем данные на шине данных ЖКИ

MOVR5,#01H; команда очистки дисплея

CALLSTROBE; строб

MOVR7,#07H; нам нужен 7-й байт 8-й страницы 0800"07"Н

MOVR6,#00H; 08"00"07

MOVR5,#10101010B; инициализация светодиодов через один

CALLWRITE;

CALL BUZZ; звуковой сигнал

RET;

LCD\_NA:; вывод символов на дисплей, составляющий отклонение текущей деформации от заданного значения

MOVR7,#01H; DATA\_IND ( 080001H)

MOVR6,#00H;

MOVA,R4;

MOVR5,A;

CALLWRITE; устанавливаем данные на шине данных ЖКИ

MOVR5,#05H; 0-й бит C\_IND отвечает за строб, 1-й бит –; запись/чтение , 2-й бит за данные/команды,; 05H=0000 0101B - запись данных и;; установка строба

CALLSTROBE; строб

RET;

WRITE:; запись значения по адресу передаваемому в;R6 и R7, записываемое значение находится в R5

MOV DPL,R7; загружаем адрес в младший байт DPTR

MOV DPH,R6; загружаем адрес в старший байт DPTR

MOV R7,DPP; временно сохраним содержимое dpp в R7

MOV DPP,#08H; переключаемся на 8-ю страницу - 080007Н

MOV A,R5;

MOVX@DPTR,A; записываем значение

MOV DPP,R7; возвращаем страницу

RET;

READ:; чтение значения по адресу передаваемому в; регистрах R6 и R7, прочитанное значение записываем в А

MOV DPL,R7; загружаем адрес в младший байт DPTR

MOV DPH,R6; загружаем адрес в старший байт DPTR

MOV R7,DPP; временно сохраним содержимое dpp в R7

MOV DPP,#08H; переключаемся на 8-ю страницу - 080007Н

MOVX A,@DPTR; пересылаем в аккумулятор значение ячейки

MOV DPP,R7; возвращаем страницу

RET;

END.

6. Программа для персонального компьютера на языке с++ в среде Borland C++Builder 6

//--------------------------------------------------------------

#include <vcl.h>

#pragma hdrstop

#include <iomanip>

#include "Kursovik.h"

//--------------------------------------------------------------

#pragma package(smart\_init)

#pragma link "CPort"

#pragma link "CPortCtl"

#pragma resource "\*.dfm"

TForm1 \*Form1;

int connect=0, N\_vhod=0, N\_vyhod=0, vhod=0, vyhod=0, process=0, ADC\_H=0, ADC\_L=0, v\_vhod=15, P=50;

float ADC, e, h2, delta\_e, zad\_e=3, v\_vyhod=0;

AnsiString Str,delta\_es;

//--------------------------------------------------------------

\_\_fastcall TForm1::TForm1(TComponent\* Owner)

: TForm(Owner)

{

}

//--------------------------------------------------------------

void \_\_fastcall TForm1::Button1Click(TObject \*Sender)

{

if (connect==1)

{

zad\_e=StrToFloat(Edit6->Text); //считывание заданного значения деформации

process=1; //переменная, отвечающая за начало процесса подсчтеа и анализа параметров

vhod=0; //переменная, отвечающая за формирование входных импульсов

vyhod=0; //переменная, отвечающая за формирование выходных импульсов

ComPort1->WriteStr('p'); //отправляем символ 'p'

ComPort1->ClearBuffer(true,true); //очистка буфера

}

}

//--------------------------------------------------------------

void \_\_fastcall TForm1::ComPort1RxChar(TObject \*Sender, int Count)

{

ComPort1->ReadStr(Str,1); //читаем полученные данные

ComPort1->ClearBuffer(true,true); // записываем в Str

if (vhod==1)

{

if (Str == 'x') {N\_vhod=N\_vhod+1; Edit1->Text=N\_vhod;}

else if (Str == 'y') {N\_vhod=N\_vhod+10; Edit1->Text=N\_vhod;}

else if (Str == 'z') {N\_vhod=N\_vhod+100; Edit1->Text=N\_vhod;}

}

else if (vyhod==1)

{

if (Str == 'x') {N\_vyhod=N\_vyhod+1; Edit2->Text=N\_vyhod;}

else if (Str == 'y') {N\_vyhod=N\_vyhod+10; Edit2->Text=N\_vyhod;}

else if (Str == 'z') {N\_vyhod=N\_vyhod+100; Edit2->Text=N\_vyhod;}

}

if (process == 1)

{

if (ADC\_H == 0) ADC\_H = Str[1];

else

{

ADC\_L = Str[1];

if (ADC\_L<0) ADC\_L=255+ADC\_L;

ADC=(float)(ADC\_H<<8|ADC\_L)\*5/4095; //Данные с АЦП используются как толщина на входе

if (N\_vhod!=0 && N\_vyhod>=N\_vhod) //Проверка корректности данных по импульсам (количество импульсов на входе < кол-ва импульсов на выходе)

{

Edit3->Text=ADC; //Выводим толщину на входе

e=(float)(N\_vyhod-N\_vhod)/N\_vhod\*100; //Расчет текущего значения относительной деформации

Edit5->Text=e; //Выводим текущее значение относительной деформации

h2=ADC-e\*ADC/100; //Расчет толщины на выходе

Edit4->Text=h2; //Выводим толщину на выходе

Edit7->Text=v\_vhod; //Выводим скорость на входе

v\_vyhod=(float)v\_vhod\*ADC/h2; //Расчет скорости на выходе

Edit8->Text=v\_vyhod; //Выводим скорость на выходе

Edit9->Text=P; //Выводим усилие обжатия

ADC\_H = 0;

ADC\_L = 0;

process=0;

}

else Label5->Visible=true; //Если данные некорректны, вывод необходимой информацию

}

}

if (e>zad\_e) //Если текущее значение деформации больше заданного,то

{

delta\_e=e-zad\_e; //вычисляем разность между текущим значением и заданным

delta\_es=(AnsiString)delta\_e; //Преобразуем данную разность в строку

ComPort1->WriteStr(delta\_es[1]);ComPort1->ClearBuffer(true,true); //и отправляем первый байт разности

Sleep(1000);

ComPort1->WriteStr(delta\_es[2]);ComPort1->ClearBuffer(true,true); //отправляем второй байт разности

Sleep(500);

ComPort1->WriteStr(delta\_es[3]);ComPort1->ClearBuffer(true,true); //отправляем третий байт разности

Sleep(500);

ComPort1->WriteStr(delta\_es[4]);ComPort1->ClearBuffer(true,true); //отправляем четвертый байт разности

e=0;

}

}

//--------------------------------------------------------------

void \_\_fastcall TForm1::Button2Click(TObject \*Sender)

{

if (connect==1)

{

ComPort1->WriteStr('s');

ComPort1->ClearBuffer(true,true); //Очистка буфера

ComPort1->Connected=false; //Завершение соединения

ComPort1->Close(); //Закрытие порта

connect=0;

Edit1->Text=0;

Edit2->Text=0;

Edit3->Text=0;

Edit4->Text=0;

Edit5->Text=0;

Edit6->Text=0;

Edit7->Text=0;

Edit8->Text=0;

Edit9->Text=0;

N\_vhod=0;

N\_vyhod=0;

vhod=0;

vyhod=0;

process=0;

v\_vyhod=0;

ADC\_H=0;

ADC\_L=0;

Edit6->Text=zad\_e;

}

}

//--------------------------------------------------------------

void \_\_fastcall TForm1::Button4Click(TObject \*Sender)

{

Edit6->Text=zad\_e; //Вывод заданного значения деформации

ComPort1->Open(); //Открытие порта

ComPort1->Connected=true; //Установление соединения

connect=1;

}

//--------------------------------------------------------------

void \_\_fastcall TForm1::Button5Click(TObject \*Sender)

{

if (connect==1)

{

process=0;

vhod=1;

vyhod=0;

Label5->Visible=false;

ComPort1->WriteStr('a');

ComPort1->ClearBuffer(true,true);

}

}

//--------------------------------------------------------------

void \_\_fastcall TForm1::Button7Click(TObject \*Sender)

{

if (connect==1)

{

process=0;

vyhod=1;

vhod=0;

Label5->Visible=false;

ComPort1->WriteStr('a');

ComPort1->ClearBuffer(true,true);

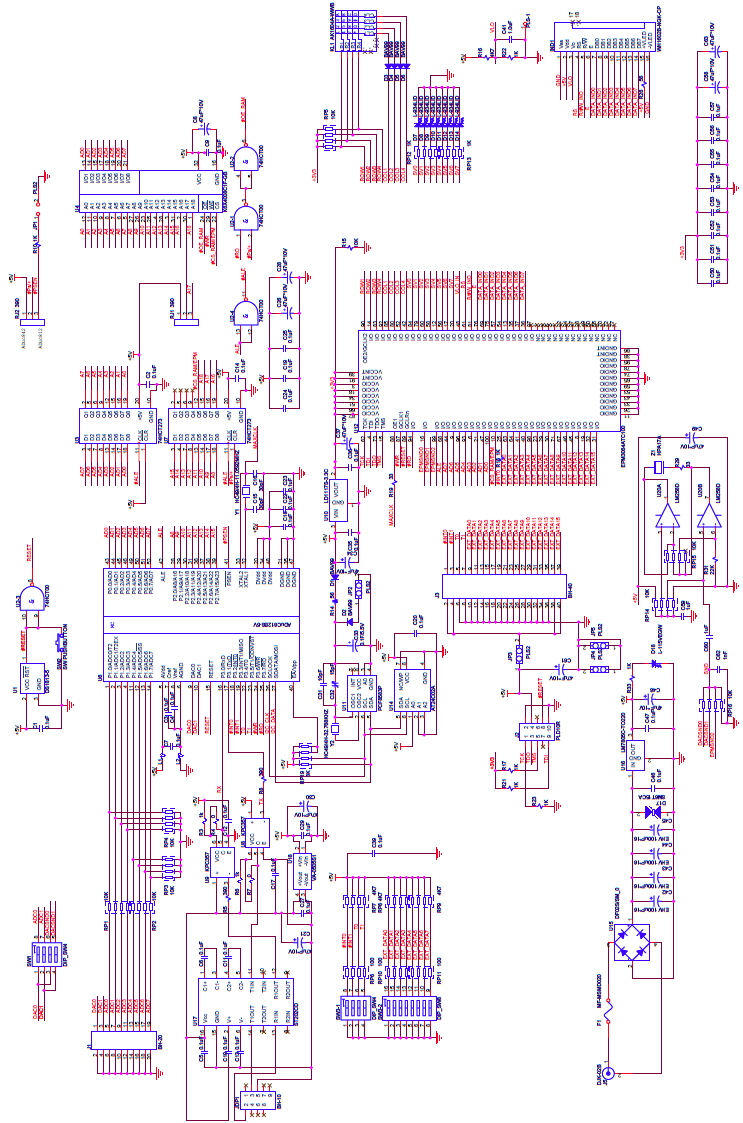
}

}

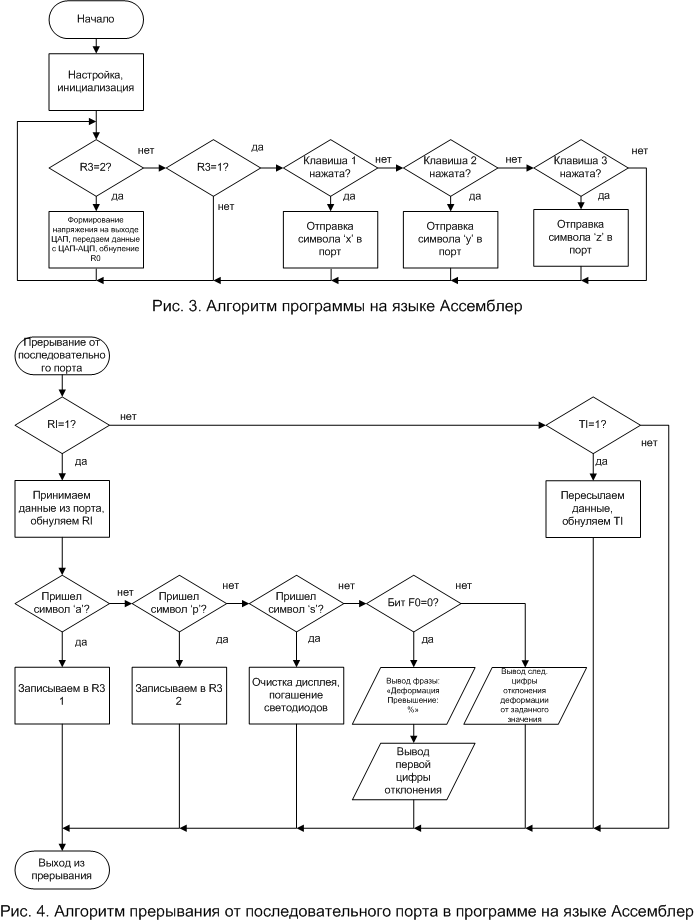
//--------------------------------------------------------------

7. Графическая часть

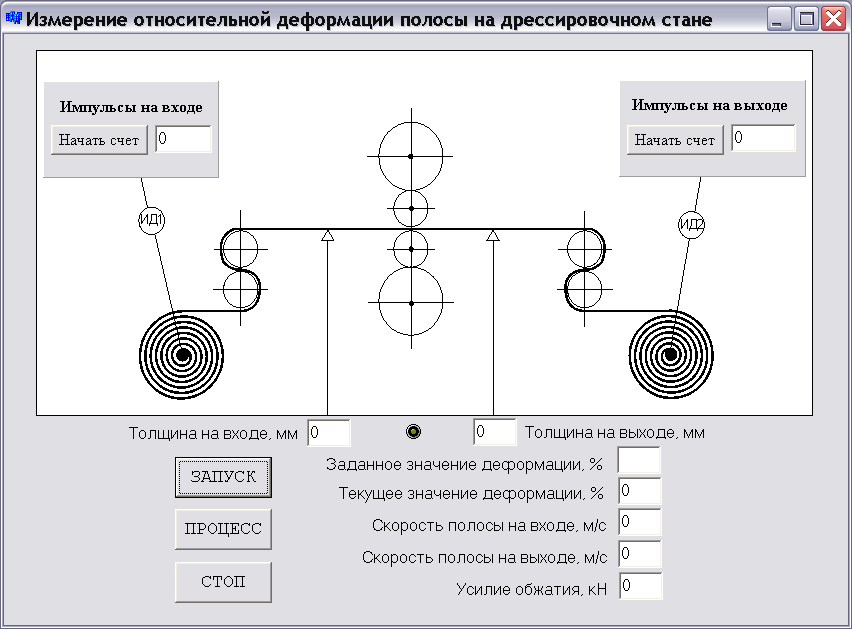
А. Электрическая принципиальная схема микропроцессорной системы



Б. Алгоритмы работы системы



В. Окно программы для персонального компьютера



В окне программы имеются поля для вывода рассчитанных и необходимых для расчетов параметров, имеется поле для ввода необходимого значения относительной деформации. Кнопка “ЗАПУСК” открывает COM-порт, моделируя, таким образом, начало технологического процесса. После нажатия на кнопку “Начать счет” какой-либо панели (“Импульсы на входе” или “Импульсы на выходе”) можно моделировать кнопками стенда SDK количество импульсов, пришедших с импульсных датчиков (кнопка 1 – 1 импульс, кнопка 2 – 10 импульсов, кнопка 3 – 100 импульсов). При нажатии на “ПРОЦЕСС” приходят данные со стенда после преобразования ЦАП-АЦП, рассчитываются все имеющиеся параметры и выводятся в соответствующие поля. Кнопка “СТОП” моделирует окончание процесса (закрывается последовательный порт), при ее нажатии обнуляются переменные и поля для вывода.

Заключение

Задача на курсовой проект состояла в проектировании микропроцессорной системы, обеспечивающей измерение относительной деформации полосы на дрессировочном стане. Для моделирования условий технологического процесса был использован стенд SDK, для которого была написана и загружена в него программа на языке Ассемблера. Взаимодействие с ПК организовано посредством COM-порта, через который передаются необходимые данные для анализа параметров процесса, а также управляющие сигналы. Расчет необходимых параметров, а толщина, скорость полосы на выходе, относительная деформация полосы проводится при использовании программы, написанной на языке С++.

Таким образом, по полученным результатам работы можно сделать вывод об успешном выполнении задания курсового проекта.

СПИСОК литературЫ

1. Джонсон Б., Скибо К., Янг М. Основы Visual studio.NET 2003 / пер. с англ. – Издательско-торговый дом "Русская редакция", 2003 – 464 с: ил.
2. Павловская Т.А. С/С++. Программирование на языке высокого уровня – СПб.: Питер, 2003 – 461 с: ил.
3. Павловская Т.А., Щупак Ю.А. C++. Объектно-ориентированное программирование: Практикум. – СПб.: Питер, 2006. – 265 с: ил.
4. Харахнин К.А. Основы проектирования микропроцессорных устройств автоматики на однокристальных микроконтроллерах: Учебное пособие. – Череповец: ЧГУ, 2007. – 234 с.
5. Техническая документация на SDK1.