Введение

В настоящее время встроенные компьютерные системы получают все большее распространение из-за их высокого качества и надежности, а так же простоты обработки информации.

Микроконтроллеры являются ядром многих современных устройств и приборов, в том числе и бытовых. Главной особенностью микроконтроллеров является то, что с их помощью легче и зачастую гораздо дешевле реализовать различные схемы.

Одними из самых популярных микроконтроллеров на сегодняшний день являются микроконтроллеры фирмы Atmel. Они основаны на новой RISC-архитектуре, которая была разработана с использованием достижений полупроводниковой микроэлектроники и возможностей программного обеспечения 1990-х годов. Созданные в результате микроконтроллеры имеют самое высокое соотношение производительность/потребление энергии, доступное на рынке 8-разрядных микроконтроллеров.

Языки программирования высокого уровня быстро становятся стандартным методом программирования для встроенных микроконтроллеров из-за уменьшения времени разработки и упрощенной поддержки сопровождения. Архитектура микроконтроллеров фирмы Atmel была разработана совместно с экспертами языка Си, чтобы гарантировать, что аппаратные средства и программное обеспечивают получение высокоэффективного кода.

В данном курсовом проекте разрабатывается устройство управления системой измерения веса, построенном на микроконтроллере ATmega16 фирмы Atmel.

1. Анализ поставленной задачи

Система измерения веса построена на весовом индикаторе WE2108, который является измерительным усилителем для подключения 4х датчиков веса. Сигнал датчиков веса усиливается и преобразуется прибором в цифровую форму и отображается на встроенном дисплее. Управление системой в целом осуществляется микроконтроллером ATmega16, который принимает от WE2108 сигнал превышения заданного уровня срабатывания и выдает соответствующие сигналы управления задвижками бункера и транспортером.

Структурная схема системы приведена на рисунке 1.1.

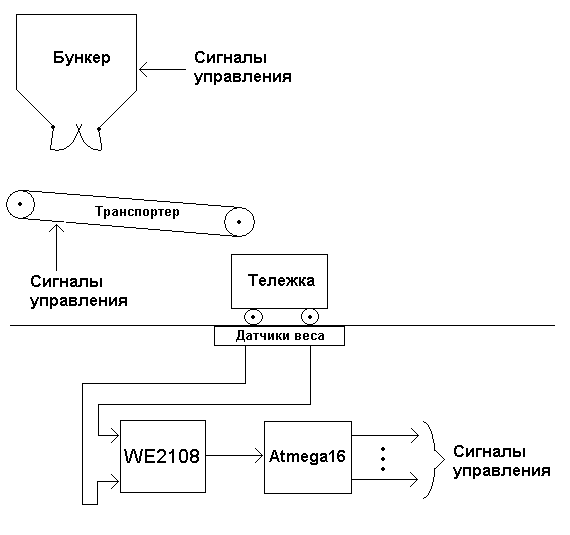


Рисунок 1.1 - Структурная схема системы

Работа системы заключается в следующем: в бункере находится груз (например, песок), который необходимо отсыпать дозированными порциями. Тележка заезжает на платформу, на которой установлены весовые датчики. Прибор WE2108 позволяет задать функцию тарирования, при выполнении которой запоминается текущее значение массы тележки и это значение вычитается из последующих значений веса. Т.е., если платформа с весовыми датчиками нагружена только пустой тележкой, то прибор показывает нулевое значение. При нажатии кнопки Пуск микроконтроллер ATmega16 выдает сигнал включения транспортера, а после выдержки времени в 5 секунд на открытие бункера. Тележка начинает наполняться грузом. Как только вес тележки с грузом достигнет заданного, WE2108 выдает сигнал о превышении установленного веса. ATmega16 получает этот сигнал, закрывает бункер и после выдержки времени в 5 секунд останавливает транспортер.

Предложенный для решения задачи микроконтроллер ATmega16 обладает следующими характеристиками:

1. напряжение питания+5 V
2. размер памяти программ 16 К
3. размер EEPROM512 В
4. размер внутренней SRAM 1 К
5. порты ввода/вывода4x8 bit
6. четыре таймера счетчика
7. программируемый последовательный УСАПП

Этих свойств микроконтроллера вполне достаточно для обеспечения управления задвижками бункера и временем работы транспортера.

2. Проектирование принципиальной схемы устройства

2.1 Схема включения микроконтроллера

Микроконтроллер AVR типа ATmega16 имеет напряжение питания +5 V, которое подводится к выводу VCC. Так как питание всего нашего устройства +10 V, то питание к микроконтроллеру должно подводится через понижающий импульсный стабилизатор. Кроме этого, микроконтроллер имеет еще два вывода для питания порта А (или АЦП), один из которых (AVCC) должен быть подключен к напряжению питания даже если порт А не используется. Если используется АЦП, то это питание должно подключатся через фильтр низких частот. Второй вывод (AREF) используется для подачи напряжения смещения на АЦП.

В нашем устройстве АЦП не используется, поэтому к выводам VCC и AVCC можно подключить предварительно стабилизированное питание +5 V от стабилизатора, а вывод AREF подключить к общей шине земли.



Рис.2.1 – Условное обозначение ATmega16

2.2 Формирование тактовых импульсов

Тактовые импульсы для работы микроконтроллера можно формировать с помощью либо встроенного генератора импульсов, либо подключая внешний генератор. Внешний генератор в данном случае не является приемлемым выбором, так как для получения импульсов заданной частоты для него нужна отдельная схема деления. Внутренний генератор, хотя и обладает меньшей стабильностью, по сравнению с внешним генератором (но стабильность можно повысить, подключив кварцевый резонатор), имеет встроенный делитель тактовой частоты. Предварительный делитель позволяет делить тактовую частоту на 8, 64, 256 и 1024. Выбор коэффициента деления тактовой частоты задается битами CS02, CS01, CS00 регистра управления т/сч0 TCCR0 или битами CS12, CS11, CS10 регистра управления т/сч1. Для данной схемы выбран коэффициент деления 1024, обеспечивающий максимальную задержку.

2.3 Схема сброса

Схема сброса должна формировать импульс логического нуля для подачи его на инверсный вывод сброса микроконтроллера RESET. Это импульс должен формироваться, например, при нажатии на кнопку сброса устройства или при переключении ключа.

Для такой схемы подходит включение транзистора, показанное на рисунке 2.2. Когда ключ открыт, транзистор находится в закрытом состоянии и на вход RESET AVR поступает сигнал логической единицы. Когда ключ замыкается, транзистор открывается и на вход сброса устройства поступает сигнал логического нуля.



Рис. 2.3 – Схема формирования импульса сброса

2.4 Схема стабилизатора напряжения

В качестве стабилизатора напряжения в нашем устройстве используется импульсный понижающий стабилизатор LM2574, который обладает следующими характеристиками:

1. входное напряжение - до 60 V (для HV версий)
2. выходное напряжение - 3.3 V, 5 V, 12 V, 15V
3. выходной ток - 0.5 А

Схема включения стабилизатора для преобразования +10 V - +5 V приведена на рисунке 2.4.



Рис. 2.4 - Схема включения стабилизатора напряжения LM2574

2.5 Входные и выходные устройства

Входными устройствами в данном проекте являются весовой индикатор WE2108, который имеет вывод превышения заданного уровня срабатывания и датчики положения заслонки бункера (открыт/закрыт). На выводе превышения заданного уровня срабатывания формируется уровень лог. 0, когда превышен заданный вес, предварительно занесенный в память прибора WE2108.

Весовой индикатор WE2108 является измерительным усилителем для подключения 4х датчиков веса. Сигнал датчиков веса усиливается и преобразуется прибором в цифровую форму и отображается на встроенном дисплее. WE2108 является полностью самостоятельным устройством со встроенным микропроцессором и позволяет выполнять широкий спектр задач связанных с измерением веса. Допустимое напряжение питания для прибора составляет от +10В до +35В постоянного тока.

Выходные устройства – сигналы управления задвижками бункера и транспортером выдаваемые через порты микроконтроллера ATmega16. Также предусмотрена индикация количества отгрузок на 4х разрядном 7-ми сегментном индикаторе.

3. Разработка программного обеспечения микроконтроллера

3.1 Разработка алгоритма программы

Программа должна выполнять следующие функции:

1. обработка нажатия кнопки Пуск;
2. управление заслонками бункера и работой транспортера;
3. обработку сигнала превышения заданного веса;
4. выдачу количества отгрузок на 4х разрядный 7-ми сегментный индикатор.

Выполнение этих операций происходит по следующему алгоритму:



Рис. 3.1 – Алгоритм программы.

3.2 Проектирование функций микроконтроллера

Процедура инициализации микроконтроллера должна состоять из процедур или операторов инициализации всех узлов самого микроконтроллера и всех периферийных устройств, и установить все начальные значения для их регистров..

Порт А работает на вывод данных, а порт С работает на прием сигналов. Поэтому при их настройке необходимо в соответствующие биты регистров DDRx, которые управляют направлением передачи данных, записать значение 0 или 1, тем самым, настроив их выводы на прием или передачу данных.

3.2.1 Инициализация таймера/счетчика0

8-разрядный таймер/счетчик может получать импульсы тактовой частоты – СК с предварительного делителя (СК/8, СК/64, СК/8256, СК/1024), импульсы с внешнего вывода или быть остановлен соответствующими установками регистра TCCR0. Флаг переполнения таймера находится в регистре TIFR. Разрешение и запрещение прерываний от таймера управляется регистром TIMSK.

SREG – регистр состояния

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| I | T | H | S | V | N | Z | C |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

Рис 3.2 - регистр состояния SREG

I – общее разрешение прерываний, 1 – разрешено.

Т – хранение копируемого бита.

Н – флаг половинного переноса.

S – бит знака.

V – флаг переполнения дополнения до двух.

N – флаг отрицательного результата.

Z – флаг нулевого результата.

C – флаг переноса.

TCCR0 - данный регистр служит для управления частотой и источником тактовых импульсов таймера (см. таблицу)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| - | - | - | - | - | CS02 | CS01 | CS00 |
| R | R | R | R | R | R/W | R/W | R/W |

Рис. 3.3 – Регистр управления таймером/счетчиком0 TCCR0

Биты 7…3 зарезервированы и всегда читаются как 0.

Биты 2, 1, 0 – выбор коэффициента деления предварительного делителя.

|  |  |  |  |
| --- | --- | --- | --- |
| CS02 | CS01 | CS00 | Значение |
| 0 | 0 | 0 | Стоп |
| 0 | 0 | 1 | СК |
| 0 | 1 | 0 | СК/8 |
| 0 | 1 | 1 | СК/64 |
| 1 | 0 | 0 | СК/256 |
| 1 | 0 | 1 | СК/1024 |
| 1 | 1 | 0 | Вн. Сигнал Т0, нар. фронт |
| 1 | 1 | 1 | Вн. Сигнал Т0, спад. фронт |

Рис. 3.4–Выбор коэффициента деления предварительного делителя.

TCNT0 – данный регистр служит для загрузки и считывания показаний счетчика/таймера0.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MSB |  |  |  |  |  |  | LSB |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

Рис. 3.5 – регистр текущего значения счетчика TCNT0

TIMSK – данный регистр служит для установки и сброса флагов разрешения работы прерываний таймеров.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| OCIE2 | TOIE2 | TICIE1 | OCIE1A | OCIE1B | TOIE1 | OCIE0 | TOIE0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

Рис. 3.6 – регистр масок прерываний таймеров

TOIE1 – разрешение прерывания по переполнению таймера/счетчика 0

OCIE1A - разрешение прерывания по совпадению таймера/счетчика 1

TIFR - данный регистр содержит флаги прерываний таймеров

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| OCF2 | TOV2 | ICF1 | OCF1A | OCF1B | TOV1 | OCF0 | TOV0 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

Рис. 3.7 – регистр флагов прерываний таймеров.

TOV1 – флаг переполнения таймера/счетчика 0. 1 – произошло переполнение

OCF1A – флаг выхода совпадения 1А. 1 – произошло совпадение значения таймера/счетчика 1 и данных в регистре OCR1A.

3.2.2 Инициализация таймера/счетчика1

16-разрядный таймер/счетчик может получать импульсы тактовой частоты – СК с предварительного делителя (СК/8, СК/64, СК/8256, СК/1024), импульсы с внешнего вывода или быть остановлен соответствующими установками регистра TCCR1B. Флаги состояния таймера (переполнения, совпадения и захвата) и управляющие сигналы находятся в регистре TIFR. Разрешение и запрещение прерываний от таймера управляется регистром TIMSK.

Таймер/счетчик1 поддерживает функцию совпадения, используя регистр совпадения OCR1A в качестве источника для сравнения с содержимым счетчика. Функция совпадения поддерживает очистку счетчика по совпадению.

TCNT1 - данный регистр служит для загрузки и считывания показаний счетчика/таймера.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| MSB |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  | LSB |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

Рис. 3.8 – регистр текущего значения счетчика TCNT1

TCCR1B – регистр управления таймером/счетчиком1

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| ICNC1 | ICES1 | - | WGM13 | WGM2 | CS12 | CS11 | CS10 |
| R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |

Рис. 3.9 – Регистр управления таймером/счетчиком1 TCCR1B

Биты 2, 1, 0 – выбор коэффициента деления предварительного делителя.

|  |  |  |  |
| --- | --- | --- | --- |
| CS02 | CS01 | CS00 | Значение |
| 0 | 0 | 0 | Стоп |
| 0 | 0 | 1 | СК |
| 0 | 1 | 0 | СК/8 |
| 0 | 1 | 1 | СК/64 |
| 1 | 0 | 0 | СК/256 |
| 1 | 0 | 1 | СК/1024 |
| 1 | 1 | 0 | Вн. Сигнал Т1, нар. фронт |
| 1 | 1 | 1 | Вн. Сигнал Т1, спад. фронт |

Рис. 3.10–Выбор коэффициента деления предварительного делителя.

Инициализация таймера/счетчика1 происходит только при вызове функции Time wate (), которая устанавливается делитель частоты на 1024, путем записи значения PrescalerTmr1= 5 в регистр TCCR1B.

OCR1A – регистр совпадения А таймера/счетчика1. В этом регистре хранятся данные, которые непрерывно сравниваются с текущим значением таймера/счетчика1. Действие по совпадению задается регистрами управления таймером/счетчиком1 и регистром состояния.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| MSB |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  | LSB |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

Рис. 3.11 - регистр совпадения А таймера/счетчика1 OCR1A

Рассчитаем значение, которое необходимо занести в OCR1A для сравнения с содержимым таймера/счетчика1.

Количество тактовых импульсов, которые приходят на таймер/счетчик1 за 1 секунду, рассчитывается по формуле:

, где



К – значение делителя частоты

f – значение частоты генератора

Выбираем значение делителя частоты 1024, т.к. это максимально возможный коэффициент деления/задержки. Частота генератора задается путем подключения кварцевого резонатора к соответствующим выводам контроллера. В данном случае был выбран кварц на 7, 3728\*106 Гц.



Т.к. необходима выдержка в 5 секунд то:

7200\*5=36000

Т.к. прерывание по совпадению таймера/счетчика1 происходит только на следующий такт после совпадения, необходимо от значения 36000 вычесть 1. Это значение присваивается переменной comp\_t1=35999, которая заносится в OCR1A.

void Time\_wate(void)

{#asm("cli")

TCCR1B = PrescalerTmr1;

OCR1A = comp\_t1;

#asm("sei")

while (Tmr1Flag == 0)

}

3.3 Процедура обработки нажатия кнопки Пуск

Процедура обработки нажатия кнопки Пуск, вызывается при переполнении таймера/счетчика0. Использование таймера для проверки состояния порта через заданные интервалы времени является некоторой защитой от импульсных помех. Еще более повысить устойчивость приема сигнала от датчика импульсов в условиях помех можно за счет использования мажоритарного элемента. Если известно текущее состояние входного сигнала и два его предшествующих состояния, то значение сигнала определяется по следующему принципу:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| OldOldPortSignal | OldPortSignal | NewPortSignal | RealSignal | Comment |
| 0 | 0 | 0 | 0 | НОЛЬ! |
| 0 | 0 | 1 | 0 | Кажется, ноль |
| 0 | 1 | 0 | 0 | Кажется, ноль |
| 0 | 1 | 1 | 1 | Кажется, единица |
| 1 | 0 | 0 | 0 | Кажется, ноль |
| 1 | 0 | 1 | 1 | Кажется, единица |
| 1 | 1 | 0 | 1 | Кажется, единица |
| 1 | 1 | 1 | 1 | ЕДИНИЦА! |

void CheckButton (void)

{

unsigned char b;

static char OldPortSignal;

static char OldOldPortSignal;

NewPortSignal = PINC&1; //select PC0 - START\_button

b = PINC&7; //select PC1,PC2 - bunker sensors

if((NewPortSignal != OldPortSignal) & (NewPortSignal != 0)) // Positive front found

{ RealSignal = (NewPortSignal ^ OldPortSignal) ^ OldOldPortSignal;

OldOldPortSignal = OldPortSignal;

OldPortSignal = RealSignal;

if(b) //bunker CLOSED!

{

PORTA=0x01; //PA0 - transporter ON!

Time\_wate(); //wate 5 sec!

Tmr1Flag = 0;

PORTA=0x03; //PA0&PA1 - bunker OPEN!

}

}

}

3.4 Процедура индикации

Выводит на на 4х разрядный 7-ми сегментный индикатор количество совершенных отгрузок. Индикатор подключен к порту В микроконтроллера. Управление разрядами индикатора осуществляется через порт D.

В процедуре используются 3 функции.

Первая Bin2BCD\_4Digit выполняет преобразование числа отгрузок, представленных в двоичном виде, в BCD число и поразрядно заносит его в массив BufBCD:

void Bin2BCD\_4Digit (unsigned int data)

{

unsigned char i;

for(i=0;i<4;i++)

{

BufBCD[i] = data % 10;

data /= 10;

}

Вторая функция Bin2Seg\_4Digit преобразовывает полученное BCD число в семисегментный эквивалент (путем вызова функции Bin2Seg) и заносит результаты в массив BufSeg. Семисегментные эквиваленты представлены в виде заранее определенных значений

#define Dig0 SegA + SegB + SegC + SegD + SegE + SegF + 0

#define Dig1 0 + SegB + SegC + 0 + 0 + 0 + 0

#define Dig2 SegA + SegB + 0 + SegD + SegE + 0 + SegG

#define Dig3 SegA + SegB + SegC + SegD + 0 + 0 + SegG

#define Dig4 0 + SegB + SegC + 0 + 0 + SegF + SegG

#define Dig5 SegA + 0 + SegC + SegD + 0 + SegF + SegG

#define Dig6 SegA + 0 + SegC + SegD + SegE + SegF + SegG

#define Dig7 SegA + SegB + SegC + 0 + 0 + 0 + 0

#define Dig8 SegA + SegB + SegC + SegD + SegE + SegF + SegG

#define Dig9 SegA + SegB + SegC + SegD + 0 + SegF + SegG

#define DigMinus 0 + 0 + 0 + 0 + SegЕ + 0 + 0

unsigned char Bin2Seg (unsigned char data)

{

switch(data)

{

case 0: return Dig0;

case 1: return Dig1;

case 2: return Dig2;

case 3: return Dig3;

case 4: return Dig4;

case 5: return Dig5;

case 6: return Dig6;

case 7: return Dig7;

case 8: return Dig8;

default: return Dig9;

}

}

//== Convert 4 digits from BinBuf[] into SegBuf[] ==========

void Bin2Seg\_4Digit (void)

{

unsigned char i;

for(i=0;i<4;i++)

{

BufSeg[3-i] = Bin2Seg(BufBCD[i]);

}

Третья функция выполняет собственно индикацию, выводя через порт В полученный семисегментный эквивалент числа отгрузок.

void Ind (void)

{

unsigned char i;

for(i=0;i<4;i++)

{

PositionPort = AllDigitsOFF;

SymbolPort = BufSeg(i);

PositionPort = DigitNmb(i)

}

4. Листинг программы

Файл ind.h

//== Include files =================================

#include <mega16.h>

#define PortBMask 0xFF

//== Common declarations ============================

#define SymbolPort PORTB

#define SegA 1 // aa

#define SegB 2 // f b

#define SegC 4 // f b

#define SegD 8 // gg

#define SegE 16 // e c

#define SegF 32 // e c

#define SegG 64 // dd

#define Dig0 SegA + SegB + SegC + SegD + SegE + SegF + 0

#define Dig1 0 + SegB + SegC + 0 + 0 + 0 + 0

#define Dig2 SegA + SegB + 0 + SegD + SegE + 0 + SegG

#define Dig3 SegA + SegB + SegC + SegD + 0 + 0 + SegG

#define Dig4 0 + SegB + SegC + 0 + 0 + SegF + SegG

#define Dig5 SegA + 0 + SegC + SegD + 0 + SegF + SegG

#define Dig6 SegA + 0 + SegC + SegD + SegE + SegF + SegG

#define Dig7 SegA + SegB + SegC + 0 + 0 + 0 + 0

#define Dig8 SegA + SegB + SegC + SegD + SegE + SegF + SegG

#define Dig9 SegA + SegB + SegC + SegD + 0 + SegF + SegG

#define DigMinus SegA + SegB + SegC + SegD + 0 + SegF + SegG

#define PositionPort PORTD

#define Position0 0

#define Position1 1

#define Position2 2

#define Position3 3

#define AllDigitsOFF 4

#define DigitNmb = [254, 253, 251, 247]

//== Global Variables =========================

unsigned char BufSeg[4];

unsigned char BufBCD[4];

void Bin2BCD\_4Digit (unsigned int data);

//== Convert binary char into 7 Segment Code =============

unsigned char Bin2Seg (unsigned char data)

{

switch(data)

{

case 0: return Dig0;

case 1: return Dig1;

case 2: return Dig2;

case 3: return Dig3;

case 4: return Dig4;

case 5: return Dig5;

case 6: return Dig6;

case 7: return Dig7;

case 8: return Dig8;

default: return Dig9;

}

}

//== Convert 4 digits from BinBuf[] into SegBuf[] ============

void Bin2Seg\_4Digit (void)

{

unsigned char i;

for(i=0;i<4;i++)

{

BufSeg[3-i] = Bin2Seg(BufBCD[i]);

}

}

//== Convert int value into 4 Digits of SegBuf[] ==============

void Bin2BCD\_4Digit (unsigned int data)

{

unsigned char i;

for(i=0;i<4;i++)

{

BufBCD[i] = data % 10;

data /= 10;

}

}

//== Show the next Digit ================

void Ind (void)

{

unsigned char i;

for(i=0;i<4;i++)

{

PositionPort = AllDigitsOFF;

SymbolPort = BufSeg(i);

PositionPort = DigitNmb(i)

}

}

Файл main.c

//== Include files =========================

#include <mega16.h>

#include <bitsm16.h>

#include <ind.h>

#include "ctype.h"

#include "stdlib.h"

//== Common declarations ==========================

#define Tmr0\_Reload 1;

#define PrescalerTmr0 4; // timer0 counts clk/256

#define PrescalerTmr1 5; // timer1 counts clk/1024

#define comp\_t1 36000-1; // coBnaDeHie 1ro taimepa

//== Global Variables ==========================

unsigned char Tmr0Flag;

unsigned char Tmr1Flag;

unsigned char Tmr0Cnt;

unsigned char NewPortSignal;

unsigned char RealSignal;

unsigned char kol\_otg;

//== Port Initialisation ===================================

void Init(void)

{

DDRA = 0xff; //Port A pins as output

PORTA = 0x00; //Send 0xFF to PortA output pins

DDRC = 0; //Port C pins as input

PORTC = 0xfe; //Turn ON PullUP for PortC pins

#asm("cli")

TIMSK = (1 << TOIE0)|(1 << OCIE1A); //Enable Timer0 & Timer1 Interrupt

TCNT0 = Tmr0\_Reload;

TCCR0 = PrescalerTmr0;

#asm("sei")

}

//== Time wate ====================================

void Time\_wate(void)

{

#asm("cli")

TCCR1B = PrescalerTmr1;

OCR1A = comp\_t1;

#asm("sei")

while (Tmr1Flag == 0)

{}

}

//== CheckButton ====================================

void CheckButton (void)

{

unsigned char b;

static char OldPortSignal;

static char OldOldPortSignal;

NewPortSignal = PINC&1; //select PC0 - START\_button

b = PINC&7; //select PC1,PC2 - bunker sensors

if((NewPortSignal != OldPortSignal) & (NewPortSignal != 0)) // Positive front found

{RealSignal = (NewPortSignal ^ OldPortSignal) ^ OldOldPortSignal;

OldOldPortSignal = OldPortSignal;

OldPortSignal = RealSignal;

if(b) //bunker CLOSED!

{

PORTA=0x01; //PA0 - transporter ON!

Time\_wate(); //wate 5 sec!

Tmr1Flag = 0;

PORTA=0x03; //PA0&PA1 - bunker OPEN!

}

}

}

//== Main Procedure =====================

void main(void)

{

Init();

while(1)

{

unsigned char c;

if (Tmr0Flag)

{

Tmr0Flag=0;

Tmr0Cnt++;

if(Tmr0Cnt>10)

{

Tmr0Cnt=0;

CheckButton();

}

}

c = PINC&4; //select PC3 - overflow signal from WE2108

if(c)

{

PORTA=0x07; //PA2 - bunker CLOSE!

Time\_wate(); //wate 5 sec!

Tmr1Flag = 0;

PORTA=0x00; //PA0 - tranporter OFF!

}

kol\_otg++;

Bin2BCD\_4Digit(kol\_otg);

Bin2Seg\_4Digit();

Ind ();

}

}

//== Interrup vectors ===============================

interrupt [TIM0\_OVF] void TIMER0\_OVF\_interrupt(void)

{

TCNT0=Tmr0\_Reload;

Tmr0Flag = 1;

}

interrupt [TIM1\_COMPA] void TIMER1\_COMP\_interrupt(void)

{

Tmr1Flag = 1;

#asm("cli")

TCCR1B = 0;

#asm("sei")

}

//==End =========================================

5. Проектирование печатной платы устройства

Для разводки печатной платы была использована программа LayoutPlus пакета OrCad. Данная программа позволяет производить разводку печатных плат, как в автоматическом режиме, так и в ручном.

Схему необходимо создавать таким образом, что бы на ней не оставалось ни одного свободного входа/выхода или висячего провода. Необходимо убрать все вспомогательные компоненты схемы (источники тока, напряжения, генераторы и другие), и прописать в свойствах компонентов раздела Footprint тип корпуса.

В установках LayoutPlus выставляется количество слоев, ширина дорожек, расстояние между дорожками, между дорожками и контактными площадками, угол поворота дорожек. Затем производится разводка в автоматическом режиме, создается координатная сетка, и проставляются размеры.

В результате была получена печатная плата, показанная в приложении Б.. Плата изготавливается из фольгированного стеклотекстолита толщиной 1 - 5 мм. Все отверстия на плате должны быть металлизированы

Заключение

В данном курсовом проекте была разработана устройство управления системой измерения веса, построенном на микроконтроллере ATmega16 фирмы Atmel.

Микроконтроллер принимает сигналы от датчиков веса и управляет ленточным транспортером и бункером, тем самым, обеспечивая автоматическое взвешивание груза дозированными порциями.

Для устройства управления в пакете OrCad была разработана печатная плата.

Разработанное устройство управления является полностью законченным прибором со всеми необходимыми функциями. Но при желании или необходимости его возможности могут быть расширены.

Приложение

