**Лабораторная работа №4**

**Тема**: Программная реализация последовательного порта ПРИНЯТИЯ ИНФОРМАЦИИ

**Цель**: Определить архитектуру программного обеспечения для МК.

теоретические сведения

Для передачи данных в микроконтроллерной и компьютерной технике используют различные протоколы и аппаратно-программные средства. С целью изучения возможностей передачи информации последовательным одно проволочным интерфейсом, рассмотрим гипотетическую задачу:

ЗАДАЧА:

Входная информация: Принять байт (значение байта находится в пределах 0..9), который передается последовательно от старшего к младшему биту по протоколу указанным на рисунке 1.

Исходная информация: Изобразить принятое значение на светодиодном индикаторе, рис. 2.



  Рис. 1 - Протокол последовательной передачи байта



Рис. 2 - подключение светодиодов для изображения цифр.

**Решение**

1) Для создания программного обеспечения нужно определить его архитектуру (структуру). Структура программного обеспечения определяется ее составными блоками. Степень детализации структуры может быть разной, и повышаться с развитием разработки. Примером разработки структуры программного обеспечения может быть следующая структурная схема программного обеспечения (рис. 3).

При создании структурной схемы программного обеспечения рассматривают основные блоки, которые нужно реализовать для полной функциональности системы. Фактически, на этом этапе проходит процесс разбиения основной задачи на ряд более простых задач, которые можно решать отдельно.

Например, отдельной задачей является система, которая будет измерять длину импульса в 1 или 2 миллисекунды. Такая система будет состоять из аппаратной частью отсчета времени, системой синхронизации начала и конца отсчета.

Отдельно, как вариант решения задачи, можно предложить системе измерять напряжение через 1,5 миллисекунды после перехода напряжения от низкого до высокого положения. В случае получения длинного единичного импульса, результатом измерения будет состояние "1", иначе "0". Это упрощает общий алгоритм, потому что вместо измерения времени между двумя переходами, нам нужно запустить измерение однотипных паузы, значительно проще задачей, чем измерения времени между разнородными событиями.



Рис. 3 - Структурная схема программного обеспечения

Разработав структурную схему программного обеспечения сделав анализ работы ее структурных единиц, результаты записывают в виде функциональной схемы программного обеспечения.

2) Разработка функциональной схемы решает задачу определения порядка работы программу и функциональное назначение ее структурных единиц. Поэтому функциональная схема является продолжением структурной схемы, только она не отвечает на вопрос "из чего состоит программа", а дает ответ на вопрос "в каком порядке и что делает программа". Примером функциональной схемы для решения поставленной задачи может быть схема, которая показана на рис. 4.

Рис. 4 - Функциональная схема программного обеспечения

Фактически, такая схема содержит алгоритм работы программного обеспечения, и как для предыдущей схемы, функциональная схема может содержать, обобщая блоки и объединения блоков. Такое использование позволит также использовать разбиение выделенных подзадач на еще более мелкие и простые задачи. Логика взаимодействия таких составляющих элементов программного обеспечения как раз и изображена на показанной схеме.

3) После разработки структурной и функциональных схем приступают к разработке алгоритма работы основной программы и вспомогательных программ-функций в виде блок-схем, решающих отдельные подзадачи. Для сложных информационных систем разработке алгоритмов может предшествовать построение диаграммы движения информации по уже созданным схемам. Такая диаграмма позволит выявить ошибки на предыдущих этапах, выявить временные конфликты, дублирование потоков и их сечения по уровням обработки. При обнаружении таких ошибок нужно вернуться к предыдущим этапам для изменения разработанных схем и диаграмм.

Блок схема основной программы должна содержать только общую логику работы. Фрагменты кода на языках программирования использовать в таких блок-схемах неуместно, что может маскировать логическое назначение кода. В идеальном случае, блок-схема объясняет назначение и семантический смысл действий и их порядок, а фактическая реализация является уже языково не зависит от маскировкой смысла использованных выражений. Идею понятности блок-схемы можно проиллюстрировать фрагментами блок-схем, показанном на рис. 5:



Рис. 5 - Пример семантической наполненности записи алгоритма

Программное обеспечение для работы на микроконтроллере получает управление сразу после подачи напряжения, и выполняется, пока устройство не будет отсоединен от питания. Такой принцип работы показано на следующей блок-схеме, которая является общим алгоритмом поставленной задачи изображения принятой цифры на светодиодном индикаторе (рис. 6):



Рис. 6 - Блок-схемы алгоритмов программного обеспечения

Также определим действия для принятия байта блок-схемы алгоритмов принятия байта и перевода значение цифры в код для ее изображения:



Рис. 7 - блок-схемы алгоритмов программного обеспечения

После определения работы программы, можно записывать алгоритмы любом языке программирования. Некоторые моменты зависят от командной архитектуры среды выполнения программного кода, их учитывают именно на этом этапе.

С блок-схем видно, что нужно решить некоторые аппаратно зависимые проблемы. Например, для счета времени нам нужно настроить таймер, для которого изменение счетчика происходила 10000 раз за 1 секунду. Для настройки частоты счета таймера используют регистр TCCR1B, в котором биты 2,1,0 - CS12, CS11, CS10 задают выбор тактирования. Эти биты определяют источник импульсов для счета для таймера-счетчика 1.

Таблица 1.

Выбор источника импульсов для счета

|  |  |  |  |
| --- | --- | --- | --- |
| **CS12** | **CS11** | **CS10** | **Опис** |
| 0 | 0 | 0 | Таймер/лічильник зупинено |
| 0 | 0 | 1 | CK |
| 0 | 1 | 0 | CK/8 |
| 0 | 1 | 1 | CK/64 |
| 1 | 0 | 0 | CK/256 |
| 1 | 0 | 1 | CK/1024 |
| 1 | 1 | 0 | Зовнішній T1, 0 → 1 |
| 1 | 1 | 1 | Зовнішній T1, 1 → 0 |

Конечно, для нашей цели полное соответствие и точность не нужны, ведь есть потребность отличить импульсы в 1 и 2 миллисекунды, поэтому по факту замер времени в 1,2-1,8 миллисекунды позволит решить основную задачу.

Пусть имеем внешний кварцевый резонатор на 4МГц. Тогда при установке делителя импульсов в 256, получим для 16-ти разрядного счетчика частоту 15625 Гц с периодом переполнения 4,13 секунды. Тогда для 1,5 миллисекунды счетчик изменит показатели на 23,4375 единицы. Счетчик таймера является целочисленным, поэтому мы можем реагировать на показатель 23 или 24 что дает интервал 1,5 миллисекунды с более чем 5% точностью. Для решения поставленной задачи эти показатели являются достаточными и не нуждаются в уточнении.

Тогда для настройки таймера 1, нужно к регистру **TCCR1B** занести константу 4. Значение отсчетов времени можно прочитать с парного регистра счетчика **TCNT1L**, **TCNT1H**. Чтобы чтение и запись двух байтов проходило одновременно, для работы с ними используется дополнительный аппаратный регистр (**TEMP**):

- Запись в таймер-счетчика 1: При записи старшего байта в **TCNT1H**, значение передается в регистр **TEMP**. Затем, при записи младшего байта, значение одновременно с **TEMP** передается таймеру-счетчика 1. То есть, при записи 16-разрядного значения, первым должен записываться байт в **TCNT1H**.

- Чтение таймера-счетчика 1: При считывании младшего байта с **TCNT1L**, он попадает к процессору, но значение с **TCNT1H** сохраняется в **TEMP**, то есть одновременно считываются все 16-разрядов. При следующем чтении регистра **TCNT1H**, значение берется из регистра TEMP.

Теперь можно перейти к написанию собственно кода на выбранном языке программирования.

// 3D09h

#include <avr / io.h>

void initPINS (void); // инициализация порта

void initTIMER1 (void); // инициализация таймера

void wait0 (void); // задержка

void wait1 (void); // задержка

void wait15 (void); // задержка

char toKod (char b);

char getByte (void);

int main ()

{

initPINS (); // инициализация порта

initTIMER1 (); // настройка таймера

PORTB = 0; // отключение порта B

char b, k; // переменный

wait1 (); // проверка порта D 0 бита

while (1)

{

wait0 (); //

b = getByte (); //

k = toKod (b); // функция возвращает значение которое нужно

PORTB = k; // вывести на дисплей

wait1 (); //

}

return 0;

}

// инициализация порта

void initPINS (void) {

DDRD = 0b00000000; // порт D на вход

DDRB = 0b11111111; // порт B на выход

PORTB = 0xFF; // открытие порта B

}

// инициализация таймера

void initTIMER1 (void) {

TCCR1B = 4; // задаем параметр предделителя

} // установка на 0b00000100 = 256

// задержка

void wait0(void) {

char b = 1; // переменная для проверки состояния

while (b == 1) { // порта D пока b = 1 выполняется цикл

b = PIND & 0b00000001;} // как только значение изменилось выход из функции

}

// задержка

void wait1 (void) {

char b = 0; // переменная для проверки состояния

while (b == 0) { // порта D пока b = 0 выполняется цикл

b = PIND & 0b00000001;} // как только значение изменилось выход из функции

}

// задержка

void wait15 (void) {

TCNT1H = 0 // обнуление значения старшего байта счетчика

TCNT1L = 0 // обнуление значения младшего байта счетчика

char b = 0; // переменная для диагностики младшего байта счетчика

char t = 0; // переменная для диагностики старшего байта счетчика

while (b < 24) {

b = TCNT1L; //

t = TCNT1H;

}

}

// возврат бинарного кода для вывода на дисплей

char toKod (char b) {

if (b == 0) // если 0

return 0b01110111;// вернуть 0

if (b == 1) ...

...

if (b == 9) return 0b01111101;

return 0;

}

char getByte (void) {

char n = 8; // количество битов

char b = 0; // обнуление значения

while (n > 0) {

n--; инкремент цикла

wait1 (); // пока PIND=0

wait15 ();

if ((PIND & 1) == 1) b | = (1 << n)

wait0 ();

}

return b;

}

Задание:

№ Задача (вход / выход выбрать произвольно)

0 Подавать / снимать напряжение до светодиода, если ко входу подано единичный импульс длиннее 1 сек.

1 Зажечь светодиод на 1 сек. при одновременном поступлении единичного состояния на два входа.

2 Зажечь светодиод PORTD.0, если была правильная последовательность одиночных импульсов на PORTB.0..9

3 Посчитать количество импульсов на входе за 4 сек.

4 Определить время между двумя импульсами с точностью 1 мс.

5 Сделать мигалку со светодиодом на 3 вспышки за 1 с.

6 Определить время содержание единичного сигнала на входе с точностью 1 мс.

7 Сделать поочередное возгорание светодиодов с частотой 10Гц.

8 Определить факт наличия импульса долгом за 0,5 с.

9 Сделать делитель входной частоты на 5. (одно переключение на 5 входных)

1. Разработать структурную схему программного обеспечения.

2. Разработать функциональную схему программного обеспечения.

3. Разработать блок-схемы алгоритмов работы программного обеспечения.

4. Запишите с объяснениями настройки портов ввода:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

5. Запишите с объяснениями настройки таймера:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

6. Ответьте на контрольные вопросы.

Контрольные вопросы:

1. Какие параметры нужно задать, чтобы таймер считал с частотой 100000Гц:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

2. Какой регистр используется для сравнения со счетчиком таймера?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

3. Зачем мерить время подачи импульсов напряжения (придумать пример)?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

4. Как можно подключать светодиод к МК?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

5. Как можно подключать кнопку в МК?

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_