

Блок схема ЭВМ

4.2 Структурная схема ЭВМ

Структурная схема mcs51 выбирается для определения необходимых ресурсов памяти и проектирования функциональной микропрограммы для команд теста

Вследствие общности необходимо в микропрограммировании видеть всю структурную схему первоначально заданной – детали конкретизируются для конкретных команд теста. Показано также разделение структурной схемы на блоки для дальнейшей их детализации на уровне функциональной схемы в MAXPLUS



1. Задание ресурсов памяти прогрпммной модели в Микро51

//ulong codDCM; 32-бит код микрокоманды в блоке

Uchar CODE[2048]; // программная память

// тест в кодах команд + таблица векторов и начало

РС=0х22;

CODE[PC++]=0x22; //ret

CODE[PC++]=0x24; CODE[PC++]=0x80; //add a,#10 CODE[PC++]=0x28; //adda,r0

CODE(PC++]=0; //Nop для контроля прерываний CODE[PC++]=0x82; CODE[PC++]=Oxe7; //anl ACC.7 CODE[PC++]=0x11; CODE[PC++]=Ox22; //acall 0x22 CODE[PC++]=0; //конец теста

}

Instr->Clear();

1E=0; //сброс запросов прерываний

ЕХ1=ЕХ0=ЕА=0; //запрет прерываний

//сброс масок прерываний на пульте CheckBoxl->State=cbUnchecked; //кодирование CheckBox2->State=cbUnchecked; //EX 1=0 CheckBox3->State=cbUnchecked; //EX0=0 CheckBox4->State=cbUnchecked; //EA=0

ComboBox2->Clear(); //сброс окна регистров ComboBox6->Clear(); //сброс стека

for(char i=0; i<8; i++)

Reg(i); //исходные значения в регистрах !=0 StateMCU(); //начальное состояние регистров il=l; //начало кодов в декодерах DCMi il=0 - пустая МК CheckBox8->State=cbUnchecked; CheckBox9->State=cbUnchecked;

}

4) Функциональная Микропрограмма в Микро51

switch(ADC[IR]); //декодирование команды {

//микропрограмма входа в прерывание

//0 mk

GoToInt(); RAMK++; //схема регистрации запросов прерываний INT0,1 - запросы в ОКНЕ

//1 mk вход в прерывание

if (!intra2) goto decoder; //переход к исполнению команды основной программы

//2 mk

SP++;RAMK++;

//3 mk

Ram(SP++]=PC;RAMK++;

//4 mk

Ram|SPJ=PO>>8; PC=vect; Clearlnt(); RAMK++;

//5 mk Завершение входа в прерывание PC=vect

 RamjSp|=SP; RAMK=0;

//---

decoder: //исполнение команд

{lR=CODE[PC++];RAMK=(ADC[IR]<<3)+1;} **//преобразование кода в адрес**

Switch(IR){ //декодирование кода команды

case 0: //1.0 mk Nop-команда пропуск цикла

{lnstr->Text= “Nop” ; RAMK=0; //переход в начало микропрограммной памяти

goto finish;}//завершение микропрограммы

case 1: //ljmp adr

//1 - чтение старшего байта РСН адреса в Wrk

{Wrk=CODE[PC++]; RAMK++;

//2- чтение младшего байта PCL и запись PCH.PCL в PC

 PC=CODE[PC] |(Wrk<<8); RAMK=0;}

Goto finish; //завершение микропрограммы- переход к следующей

case 3: //ret

{ Wrk=Ram[SP--]; RAMK++;

//2 запись адреса из Стека в PC

PC=(Wrk<<8)|Ram[SP--]; RAMK++; //PCL

// 3 сохранить SP в Ram

Ram[Sp]=SP; RAMK=0;

} goto finish;

case 4: // add a,ri

//1 чтение регистра

{ PB= Ram[(PSW&0x18)|(IR&0x3)];RAMK++;

//2 операция в АЛУ и формирование признаков и сохранение

АСС =АСС+РВ, Ram[Acc]=ACC; Ram[Acc] = Acc; RAMK++; Pswc(“add”);

//3 сqохранение PsW в SFR

Ram[Psw]=PSW; RAMK=0;} goto finish;

case 2: //add a,#d

//1 -чтение операнда из памяти Code

 { PB=CODE [PC++]; RAMK++;

//1 - - операция в АЛУ и формирование признаков результата

АСС=АСС+РВ, RAMK++; Ram[Acc]=ACC; Pswc(“add”);

//3

Ram[Psw]=PSW; RAMK=0; } goto finish;

case 5: // acall met

//11 - чтение второго байта команды и пре-инкремент указателя

{Wrk=CODE[PC++]; SP++; RAMK++;

//2 запись в Стек РС(7-0) и постинкремент указателя

Ram[SP++]=PC;RAMK++;

ИЗ -старший байт PC в стек и формирование PC Ram[SP]=PC»8; RAMK++; PC=((PC&0xf800)|Wrk)|((IR&0xE0)>>5)<<8;}

//4- сохранение продвинутого указателя в SFR

Ram[Sp]=SP; RAMK=0;} goto finish; case 6: // anl c,bit

//1- чтение адреса бита из второго байта команды

{Wrk=CODE[PC++]; RAMK++;

//2 - чтение бит адресуемого байта из RAM

if(W7) PB=Ram [Wrk&0xf8];

else { PB=Ram[0x20|((Wrk&0x78)>>3)]; RAMK++;}

// 3 - выполнение операции с битами

If ((l«(Wrk&0x7)&PB)>;

else PSW=PSW&0x7f;

RAMK++;

//I4 - сохранение PSW в SFR по адресу Psw

Ram[Psw]=PSW; RAMK=0;} goto finish;

 case 7: //reti

//1

{ Wrk=Ram[SP--]<<8; RAMK++;

//2

PC=(Wrk<<8)|RamlSP--]; RAMK++;

//3

Ram[Sp]=SP; RAMK=0; intra2=0;}

//Вывод состояния регистров в НЕХ-коде

finish: //возврат к контролю прерывания и чтению команды- StateMCU();

}

5. Принципы кодирования микропрограммы

Для кодирования микрокоманд используются символы, обозначающие микрооперации в разных блоках структурной схемы.

Например, микрокоманда выборки первого байта команды

{IR=CODE[PC++];RAMK=(ADClIR<<«3)+1;}

выполняется в течение нескольких микротактов в проекте схемы ЭВМ в MaxPlus

clk4: “SelbusB=Com, Unicon = Wrir, Unicontr =Ramkl, Unicod 6 = Incpc “

clkl: CODE[PC] ->BusB

В блоке Bus8: “SelbusB=Code”

clk2: BusB -> IR, PC++, BusB -> ADC, (ADC[BnsB]«3).001-> RAMК

 В блоке Control “Unicontr = Wrir, Unicontr =Ramkl “

16mem “Unicodl6 = Incpc”

clk3: ROMM[RAMK]-> DC, DC[] ->

MicroCodMem(“SelbusB=Code, Unicontr = Wrir, Unicontr =Ramkl, Unicod 16 = Incpc “

Таблицы кодирования

//1 блок 16mem ==== (DCM 1)=

struct mikl6{// структура микрокоманды управления блоком 16mem

uchar selpc; // 2 бит

uchar unicod 16; 5 бит

uchar selaxa ; 1 бит }

mkl6 ; 7 bit ширина мк

char \*Selpc ="AcalI,Pc+a,Vect,Call “j

II 1 2 3 бит 2

char \*Unicodl6 = "Wrpc,Wrxda,Incpc,Incdptr,Wrdptr **//код микрооперации 1 2 3 4 5 бит 5**

\*Selaxa=”P2.ri,Dptr”;

H 0// 1 1 бит

//3 блок BITA===(DCM 3)=======

struct ЬиЦ ефуктура микрокоманды

uchar selpsw; //1

uchar unibit; //6

} bita

void Nevvbita(void)

vbita.unibit=0; bita.selpsw=0;}

char \*Unbitbit=”Bitor,Bitand,Nebit,Newbit,Setb,Clrb, Movcb ";

char \*Selpsw="BusB,Bitsw;

//1

//4 блок RALU===(DCM4)==- =

struct ralu { //структура микрокоманды управления блоком ralu 19 бит

uchar шор; //3

uchar selacc; //3

uchar selb; //2

uchar uniralu; //9

uchar selpa; //1

uchar selpb; //2

} alu; //19 бит

char \*Mop="Suba,Subb,Add,Or,Xor,And,H

1 2 34567 3 бит

char \*Uniralu =" Wrbloc,Ci., Mula, Diva, Wrwrk,Incwrk,Wlocacc

 1 2 3 4 5 6 7 7 бит

char \* Selacc ="BusB, Mulow, Quot, Laic,Rale, F,Xdata

0 1 2 3 4 5 6 3 бит

char \*SelB ="Wrk,Mulhigh,Remain

0 1 2 2 бит

char \*Selpa ="Pa,Acc

0 1

char \* Selpb =" 0, Pci, Pb”

0 1 2 2 бит

/5 блок Ports— (DCM 2 )=

uint mkporta;

struct ralu

{uchar adpi; uchar adpin;

uchar uniport;

} ports;

\*Adpin=”P0,Pl,P2,P3”;

0 1 2 3

\* Adpin=”Pin0.Pin 1 ,Pin2,Pin3”;

 0 1 2 3

//6 блок Interrupt==

//управление в Contr

//блок7 bus8 (dcm7)

//сокращенные адреса рабочих регистров при обращении к теневым в SFR struct BUSS8

{ //структура микрокоманды управления блоком ralu

uchar selbusb; //4

uchar selbusa; //3

uchar adsfr; //7

uchar selsp; //3

uchar unibus8; //1

} mkbus8; //21 бит

char \*SelbusB="Code,Wrk,Pch,Pcl,PinPi,Pi.Tcon, Acc,Ram,F, Psw,Sp,Bita “;

 // 0 1 2 3 4 5 6 7 8 9 10 11 12 4 бит

char \*SelbusA="Ar,Wrk,Asfr,Sp,Aram

 // 0 1 2 3 4 3 бит

char \*Unibus8="Ari,Wram, 'Wrsp,Wrpsw";

 // 2 3 4 4 бит

char \*Selsp= “Dcrsp, 0x07,Incsp, Busb";

 // 1 2 3 4 2 бит

char \*Asfr=” “

 6 бит

//блок8 **Control** (dcm8.mif)

uchar unicontr;

char \*Unicontr — **"Clrarnk,Ramkl,Wrir,Eintra,Clreci,Clrinta “;**

***//*** 0 123 45 6**бит**

//блок ROMM (DCM9.mif)

uint ifromm; //Selif,Neiff,dcrom)

char **\*Selif** ="Cc,Zacc,Bitwrk,Wrk7,PswC,lntra,Neiff “;

 1 2 3 4 5 6 7 3+1 бит

