SH-7045 マイコンボードキット

**256KBフラッシュROM内蔵、28MHz高速動作、
AD内蔵、4kB RAM内蔵、5V書き込み、
高速・大容量8Mbit SRAM搭載**

HITACHI SH/7045F 使用

SH-7045F マイコンボードキット

256kB フラッシュ ROM 内蔵、28MHz 高速動作、AD 内蔵
4kB RAM 内蔵、5V 書き込み、高速・大容量 8Mbit SRAM 搭載

◆日立製 32 ビット CPU SH/7045F を使用したマイコンボードです。1 チップに ROM/RAM ・周辺回路をすべて内蔵しており、ボードはシンプルかつ高性能です。

◆32 ビット RISC アーキテクチャー搭載で 28MHz の高速動作を実現しています。

◆命令処理においてパイプライン機構を採用していますので、原則として 1 命令 1 クロックで動作し、同じ周波数で H8 とくらべて 4 倍程度の処理速度向上を実現しています。

◆256k バイト大容量フラッシュメモリを CPU チップに内蔵しています。プログラムを 100 回以上書き換え可能です。従来の CPU に不可欠な EP-ROM を取り付ける必要がなくなりました。メモリ空間は最大 4G バイトでさらに ROM ・ RAM を拡張することができます。

◆高速・高分解能 AD/DA コンバータを内蔵しています。

◆最大 106 本(付属 SRAM 使用時は 41 本)の I/O ポートを装備しています。

◆高速 RS232C ドライバーレシーバーを内蔵しており、パソコンや他のマイコンとの通信も容易に行えます。

◆8M ビット高速 SRAM 装備で 3.5 インチ FD サイズです。ピンヘッダ付きで機器組み込みに最適です。

- 書き込みは、5V 単一電源のため、容易にブートモード設定ができます。
- 開発ソフトはフリーのＣコンパイラの gcc が附属しています。Ｈ８／ＯＳのＳＦ－７０４５版が附属していますのでソフト開発が容易に行えます。

■SH-7045F ボードの主な仕様■

メモリ	ROM	256k バイト	外部拡張可能
	RAM	4k バイト	外部拡張可能
	SRAM	1M バイト	外部拡張
周辺回路	DMAC	最大 4 チャンネル	
	MTU	5 チャンネル	
	CMT	2 チャンネル	
	DTC	33 + ソフトウェア起動	
	WDT	ウオッチドッグタイマー	インターバルタイマーとして使用可能
	SCI	独立 2 チャンネル	
	A/D	10 ビット分解能×8 チャンネル	
	I/O ポート	最大 106 本	付属 SRAM 使用時は 41 本

■開発用ソフトウェア(Ｃコンパイラ、ライターソフト)について■

開発用ソフトウェア(CD-R)は、SH/7045F ボードのみのセットには、付属していません。SH/7045F 開発セットに付属しています。

1)ライター(書き込み)ソフトは、キット附属の h8write でブートモードにして書き込みます。h8write の使用方法はＣＤのＷＩＮフォルダの h8write.doc をごらんください。

2)Ｃコンパイラは、フリーの GCC を使います。Cygwin(gcc)インストール方法はＣＤの cygwin フォルダの conpile.html をごらんください。

3)Ｈ８／ＯＳでユーザーソフトをパソコンからｓｈマイコンのＲＡＭに転送するソフト（put.exe）が附属しています。put.exe の使用方法はＣＤのＷＩＮフォルダのｐｕｔ.doc をごらんください。

■部品表■　※印の部品は実装半田付け済みです。

番　　号	数	品　　名	備　　考
U1	1	HD64F7045F28	※SH－2マイコン
U2	1	S80945AN	※リセットIC
U3，4	2	74VHC08	※TTL
U5	1	74VHC21	※TTL
U6	1	SP202	※RS232C
U7，8	2	6216255	※4Mbit高速SRAM
U9	1	27C4096 実装用ICソケット 40 ピン	27C4096 は附属していません
C1	1	電解コンデンサ　22μF16V	
C2～C19	18	チップコンデンサ　0．1μF	※
C20	1	チップコンデンサ　470pF	※
C21，22	2	チップコンデンサ　18pF	※
R1～3	3	チップ抵抗　220Ω	※
R4	1	チップ抵抗　3KΩ	※
R5，6	2	チップ抵抗　10KΩ	※
RA1～8	8	チップ抵抗モジュール　47K×8	※
SW1	1	タクトスイッチ	※リセットスイッチ
XT1	1	クリスタル　14.32MHz	2 倍モードで 28.64MHz 動作
J1	1	Dサブ9ピンメスコネクタ	基板取り付け用
CN1	1		電源コネクタ
ピンヘッダ		CN2，3，4，JP1、JP2	150ピン分(2×40 を折って使用)
ピンソケット	2	CN2，3用	2×30
ショートピン	10	JP1、JP2用	
ソフト	1	SH－2開発用CD	基板のみの場合は附属しません。

■基板製作■

動作部品は、ほとんど実装半田付け済みです。取り付ける部品は、クリスタルX1、コンデンサC1、コネクタ類のみです。

1、コンデンサC1は極性があります。足の長い方が＋です。基板印刷の＋に合わせて取り付けてください。

2、クリスタルX1は極性はありません。基板に密着させますと、パターンがショートする場合がありますので0．5mmほど隙間をあけて取り付けてください。

3、CN2，3，4，JP1、JP2は2×40ピンのピンヘッダから、それぞれに折って使用します。
JP1、JP2はショートピンを挿入しますので、部品面（SHマイコンが半田付けしてある面）に半田付けしてください。CN2，3，4は用途に合わせてどちら側に取り付けてもかまいません。
通常は、CN2，3は半田面、CN4は部品面に取り付けてください。

4、J1（Dサブ9ピン）、CN1（電源コネクタ）は部品面に取り付けてください。

5、U6はICソケットのみ附属しています。必要な場合のみ取り付けてください。
（SH－7045Fは書き換え可能なフラッシュROMを内蔵していますのでU6無しで動作します。）

■動作モード設定■

このボードは、8M ビット SRAM が装着されていますが、この他にメモリ、周辺ペリフェラル等を外部に拡張することが可能です。

ジャンパースイッチで動作モードを設定できます。　通常は、モード２で使用します。

ただし、ボードに装着されている SRAM を使用しない場合は、シングルチップモードにも設定できますが、回路図をよく理解してから I/O ポートを使用してください。

ジャンパを開放すると「1」になり、短絡すると「0」になります。ただし、ＦＷＰ、ＭＤ１はＢＯＯＴを短絡すると常に「0」になります。

動作モード	BOOT	FWP	MD3	MD2	MD1	MD0	内蔵 ROM	CS0 空間
モード 0	開放	開放	短絡	開放	短絡	短絡	無効	16 ビット幅
モード 1	開放	開放	短絡	開放	短絡	開放	無効	32 ビット幅
モード 2	開放	開放	短絡	開放	開放	短絡	有効	8/16/32 ビット幅
モード 3	開放	開放	短絡	開放	開放	開放	有効	16M バイト
ユーザプログラムモード	開放	短絡	短絡	開放	開放	短絡	有効	8/16/32 ビット幅
ブートモード	短絡	X	短絡	開放	X	X	有効	

■動作周波数■

メーカーの動作保証範囲は、28.7MHz 以下です。

このキットは 28.64MHz 用のクリスタル 14.32MHz が付属し、28.64MHz で動作します。(×2モード)

書き込みプログラムは 28.64MHz 用になっています。

(書き込まれたプログラムは別の周波数でも動作可能です。)

MD3	MD2	クロックモード	本キットでの動作周波数
短絡	短絡	×１	１４．３２MHｚ
短絡	開放	×２	２８．６４MHｚ（ライタープログラム、Ｈ８／ＯＳ対応）
開放	短絡	×４	
開放	開放	リザーブ	

■ブートモード(書き込みモード)と書き込み用パソコン接続■

内蔵フラッシュ ROM に書き込むにはブートモードで書き込みます。書き込み電圧は 5V です。

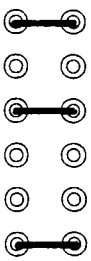
パソコンとの接続は、SH/7045F ボードの J1 シリアルポートをパソコンのシリアルポート(COM ポート)に接続します。また、Ｊ１（ＲＳ２３２Ｃ）を有効にする為に JP2 の 3-4、６－８を短絡します。

接続は、市販の RS-232C ストレートシリアルケーブルを用います。

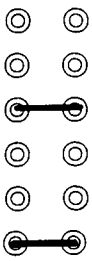
ブートモードにするには、JP1 の「BOOT」「MD3」をジャンパで短絡します。「BOOT」ジャンパを短絡すると CPU チップのＭＤ１、ＦＷＰが「０」になりブートモードになります

ライター(書き込み)ソフトは、キット附属の h8write で書き込みます。h8write の使用方法はCDのWINフォルダの h8write.doc をごらんください。

書き込み動作「ブートモード」



通常動作「モード２」



■電源■

電源は書き込み動作時、通常動作時共、DC5Vです。電源レギュレータは内蔵していません。

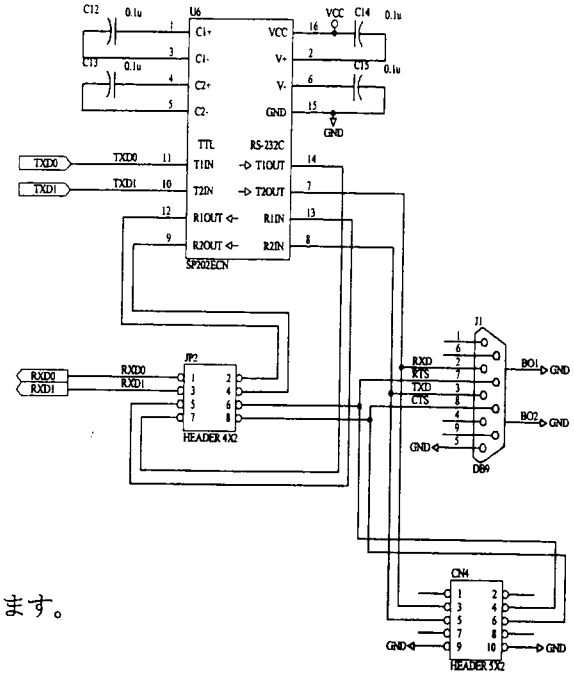
電流は200mA以上の電源をご使用ください。

■JP2 ジャンパー設定■

この基板のシリアルポート回路は右図の様になっています。

Ｊ１、ＣＮ４にＲＳ２３２Ｃ入出力が接続されます。

JP2 は、シリアルポート回路設定です。



ＪＰ２	
１－２を短絡 ５－６を短絡	ＲＸＤ０がＣＮ４－４、Ｊ１－７に Ｕ６を通して接続される
３－４を短絡	ＲＸＤ１がＣＮ４－５、Ｊ１－３に Ｕ６を通して接続される
７－８を短絡	ＴＸＤ０がＣＮ４－６、Ｊ１－８に Ｕ６を通して接続される
６－８を短絡	Ｊ１－７とＪ１－８が接続されパソコン のＲＴＳ－ＣＴＳ信号が折り返す

ＴＸＤ１は常にＵ６を通してＣＮ４－３、Ｊ１－２に接続されています。

ブートモード書き込み時には JP2 の３-４、６－８を短絡します。

■付属 SRAM の使い方■

付属 SRAM は、モード２で使用します。

付属 SRAM のアドレスは、400000H～4FFFFFH までとなります。

ＲＯＭはＣＳ０でモード１で使用します（ＲＯＭはキットに附属していません）

■コネクタ→ピン配置表■

ＣＮ２				ＣＮ３			
ピン番号	名称	ピン番号	名称	ピン番号	名称	ピン番号	名称
1	D0	31	D30	1	+5V	31	PE8
2	D1	32	D31	2	GND	32	PE7
3	D2	33	A21	3	+5V	33	PE6
4	D3	34	A20	4	GND	34	PE5
5	D4	35	A19	5	-WDTOVF	35	PA5
6	D5	36	A18	6	GND	36	TxD1
7	D6	37	A17	7	-WRL	37	RxD1
8	D7	38	A16	8	-WRH	38	PA2
9	D8	39	A15	9	-WRHL	39	TxD0
10	D9	40	A14	10	-WRHH	40	RxD0
11	D10	41	A13	11	PA20	41	AVREF
12	D11	42	A12	12	-RD	42	GND
13	D12	43	A11	13	PA18	43	PF6
14	D13	44	A10	14	PA21	44	PF7
15	D14	45	A9	15	PB2	45	PF4
16	D15	46	A8	16	PA19	46	PF5
17	D16	47	A7	17	PB4	47	PF2
18	D17	48	A6	18	PB3	48	PF3
19	D18	49	A5	19	PA9	49	PF0
20	D19	50	A4	20	PB5	50	PF1
21	D20	51	A3	21	PA7	51	PE4
22	D21	52	A2	22	PA8	52	PE3
23	D22	53	A1	23	PE15	53	PE2
24	D23	54	A0	24	PA6	54	PE1
25	D24	55	+5V	25	PE13	55	PE0
26	D25	56	GND	26	PE14	56	PA15
27	D26	57	+5V	27	PE12	57	PA16
28	D27	58	GND	28	PE11	58	PA17
29	D28	59	+5V	29	PE10	59	-NMI
30	D29	60	GND	30	PE9	60	-RESET

H8/OSを使ったソフトウェアの作りかた

(以下の説明がＣＤの h8_os フォルダの docs フォルダに sh_usage.doc で入っています。コンパイル時などにカット&ペーストでご利用されると便利です。)

1. ヘッダーファイル

C 言語から H8/OS のシステムコールを使う場合は、コンパイルに必ず gcc を使い、C ソースファイルのなかで<h8/syscall.h>のヘッダ定義を必ずしてください。
また、各 SH/H8 チップごとにヘッダーファイルが用意されていますので、使用すると便利でしょう。

<h8/reg3067.h>　H8/3067F、H8/3068F、H8/3069F 用(I/O ポート定義)
<h8/reg3048.h>　H8/3048F、H8/3052F 用(I/O ポート定義)
<h8/reg3667.h>　H8/3664F 用(I/O ポート定義)
<h8/reg704x.h>　SH/7045F 用(I/O ポート定義)

ヘッダーファイルは、h8_os¥include¥ 以下にあり、それを cygwin 上のプロンプトで以下の操作でフォルダを作成して、そのフォルダにコピーをします。

（1）フォルダ作成
```
mkdir /usr/local/sh-coff/include/sh
mkdir /usr/local/sh-coff/include/h8
```
（2）コピー
```
cp *.h /usr/local/sh-coff/include/sh
cp *.h /usr/local/sh-coff/include/h8
```

2. プログラム作成例(SH-7045F)

H8/OS の機能を使ったプログラムの作成例として、SH-7045F で I/O ポート PE0 を 1 秒ごとに"H"と"L"を繰り返すプログラム"blink.c"を作成してみます。

```
/* ソースファイル名は、blink.c */
#include <sh/reg704x.h>    /* SH-7045F で使用する I/O ポートを定義 */
#include <h8/syscall.h>    /* H8/OS を使う場合に必ず指定をする */

int main() {
        PECR1 = 0;
        PECR2 = 0;
        PEIOR = 0xffff;          /* PE0-PE15 のポートを出力に設定する */
        while(1) {
            PEDR = 0x00; /* PB0 を L にする */
            sleep(10);   /* 1 秒の時間待ち */
            PEDR = 0x01; /* PB0 を H にする */
            sleep(10);   /* 1 秒の時間待ち */
    }
}
```

sleep は、H8/OS のシステムコールです。

3. ROM 化用の実行ファイルの作成

まず、以下のようにして中間ファイルを作成します。

```
sh-coff-gcc -O -m2 -T rom7045.x -o blink.coff -nostartfiles 7045crt0.s blink.c -lc
```

gcc のオプションで同じソースから ROM、RAM ターゲットの実行ファイルが作成できます。
gcc のオプションの意味の概略は以下のとおりです。

(1) -O : 最適化する

(2) -m2: SH2 用でコンパイル
(3) -T rom7045.x : メモリ定義ファイルの指定。SH-7045F の ROM 化用には"rom7045.x"を指定。
(5) -o blink.coff : 中間ファイルのファイル名を指定する。
(6) -nostartfiles : 組み込みマイコンの場合は必ず指定する。
(7) 7045crt0.s : SH-7045 用の ROM 化プログラム用のスタートアップルーチン
(8) blink.c : C ソースファイル名。
(9) C ライブラリを使う場合に指定。

次に中間ファイルからモトローラ S 形式(mot ファイル)に変換すると ROM 化用の実行ファイルができあがります。kern7045.mot と作成した mot ファイル（blink.mot など）を合体させて書き込み用の mot ファイルが完成します

sh-coff-objcopy -O srec blink.coff blink.mot

使いかたは以下のとおりです。

sh-coff-objcopy -O srec [中間ファイル名] [mot ファイル名]
copy kern7045.mot+[mot ファイル名] [ROM 化ファイル名]
（copy はＭＳ－ＤＯＳのコマンドです。windows のメモ帖やワードパットで kern7045.mot の後に
作成した mot ファイル（blink.mot など）を貼り付けて合体させても良いです。）

ボードをブートモードに設定して内蔵 ROM に転送します。
転送が終了したら、通常モードで起動すれば、ROM 化したプログラムが起動します。

4. デバッグ用 RAM 実行ファイルの作成

デバッグ用 RAM 実行ファイルを実行するには、あらかじめ、コマンドインタープリタ付きの H8/OS を内蔵 ROM に書き込んでおきます。
例えば、SH-7045F では、plus7045.mot がそれに該当します。
ROM 化用の実行ファイルとデバッグ用 RAM 実行ファイルにする場合のソースプログラムは共通です。
まず、以下のようにして中間ファイルを作成します。

sh-coff-gcc -O -m2 -T ram7045.x -o blink.coff -nostartfiles ramcrt0.s blink.c -lc

gcc のオプションの意味の概略は以下のとおりです。

(1) -T ram7045.x : メモリ定義ファイルの指定。SH-7045F の RAM 用には"ram7045.x"を指定。
(2) shramcrt0.s : RAM 用プログラムのスタートアップルーチン

次に中間ファイルからモトローラ S 形式(mot ファイル)に変換するとデバッグ用 RAM 実行ファイルができあがります。

sh-coff-objcopy -O srec blink.coff blink.mot

デバッグ用 RAM 実行ファイルを H8 に転送するには、put コマンドを使います。
put コマンドは、パソコン上の実行ファイルを plus7045.mot が書き込まれた SH マイコンへシリアルケーブルを介してデータ転送をするツールです。転送するには、シリアルポートを使用します。

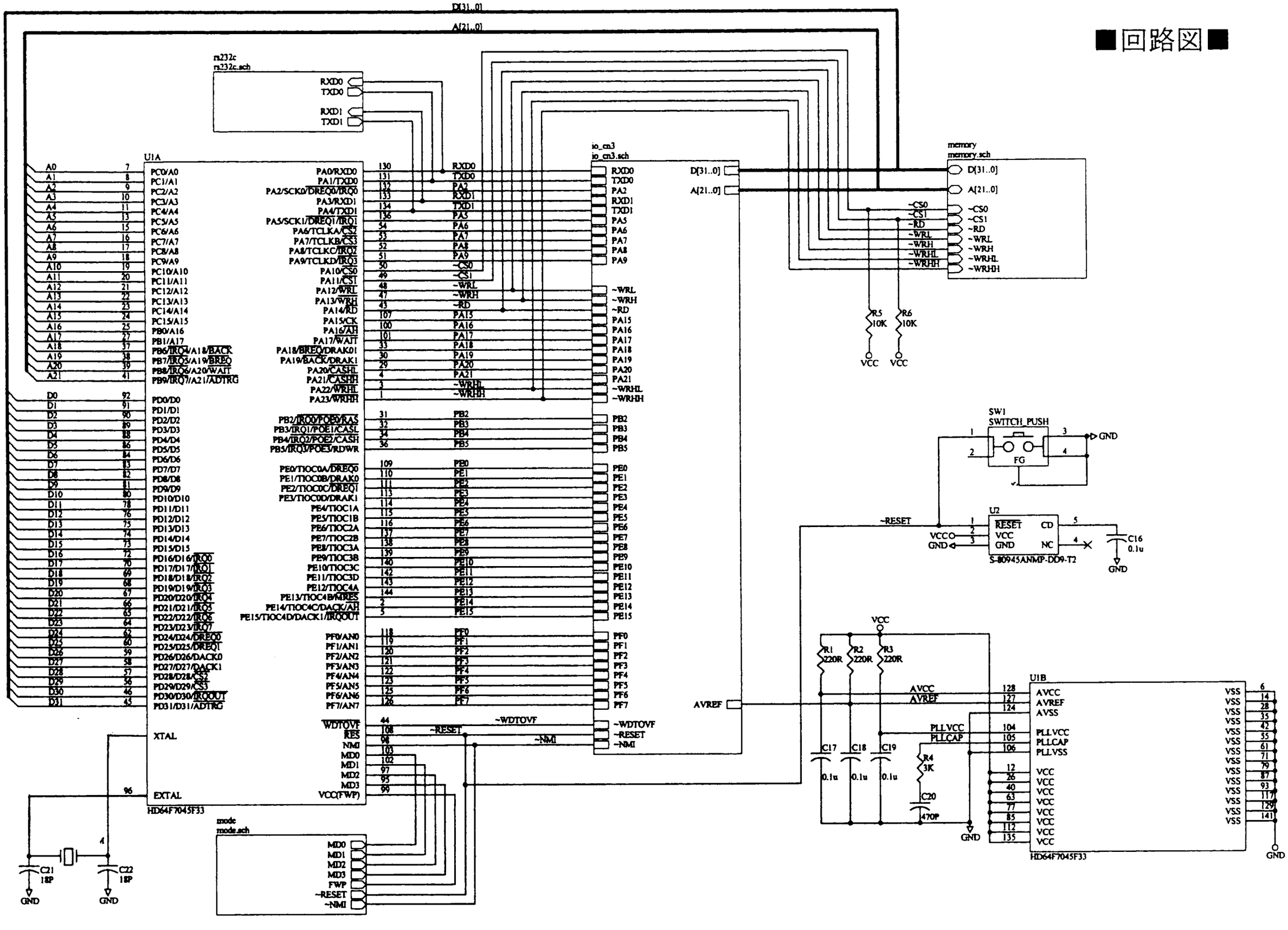
put blink.mot

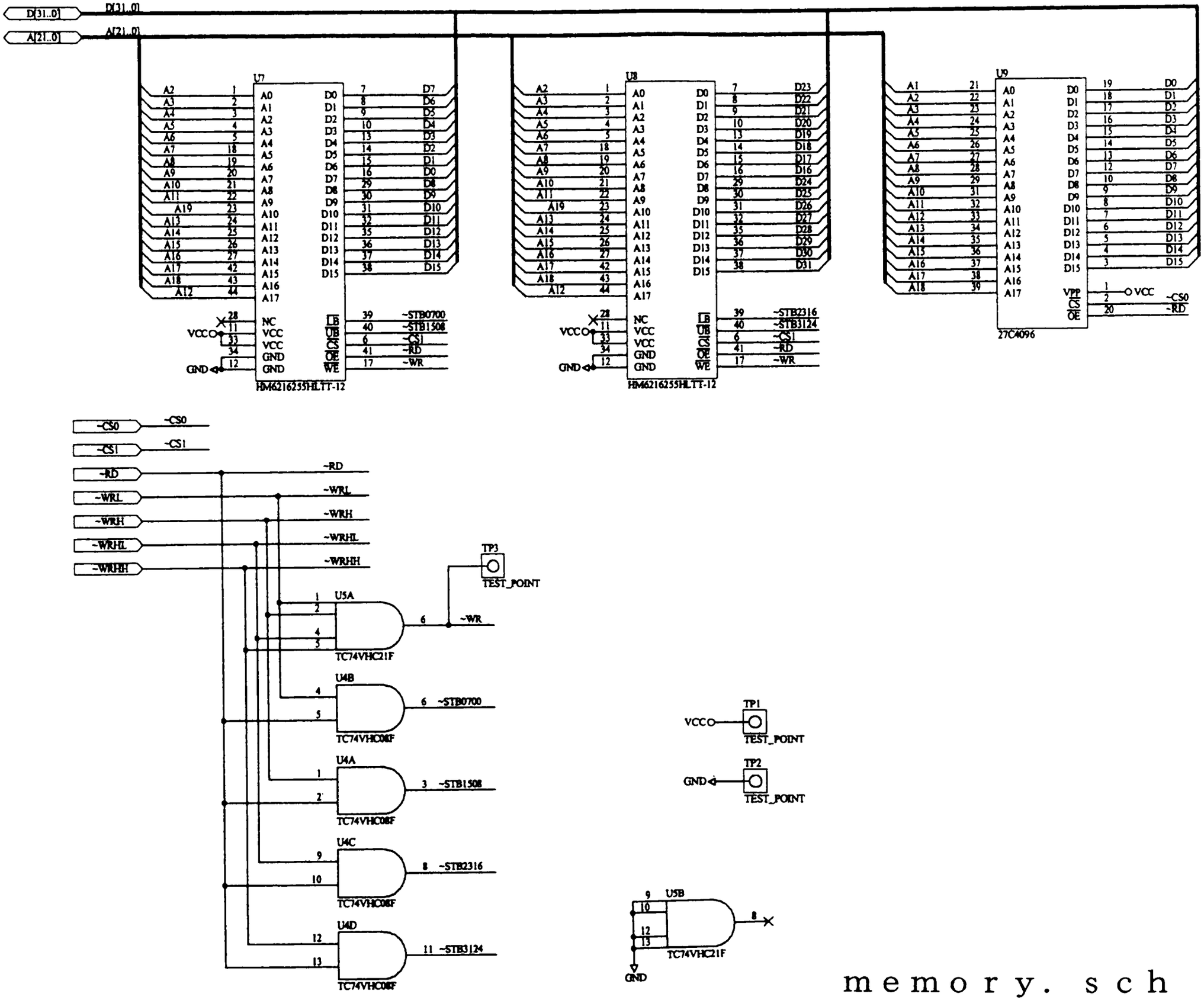
転送が終了したら、

put -g ffff7a0

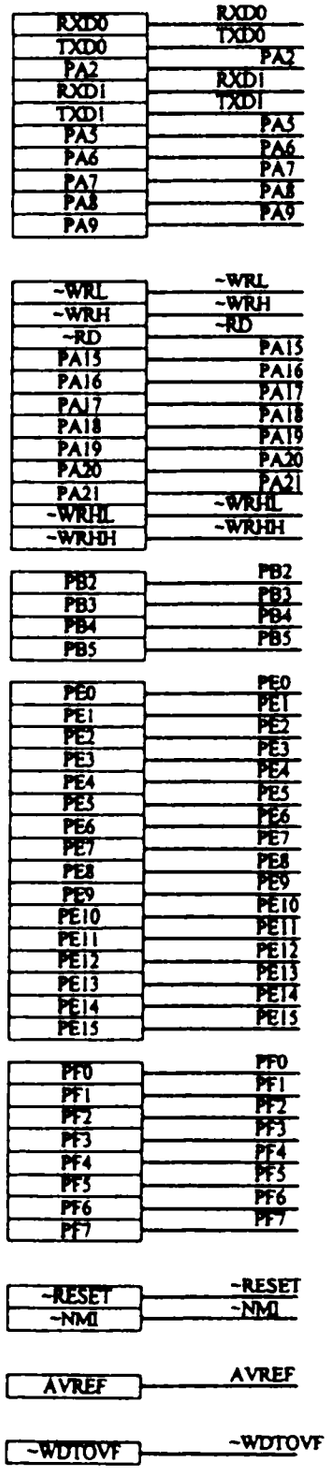
で blink.mot が起動します。

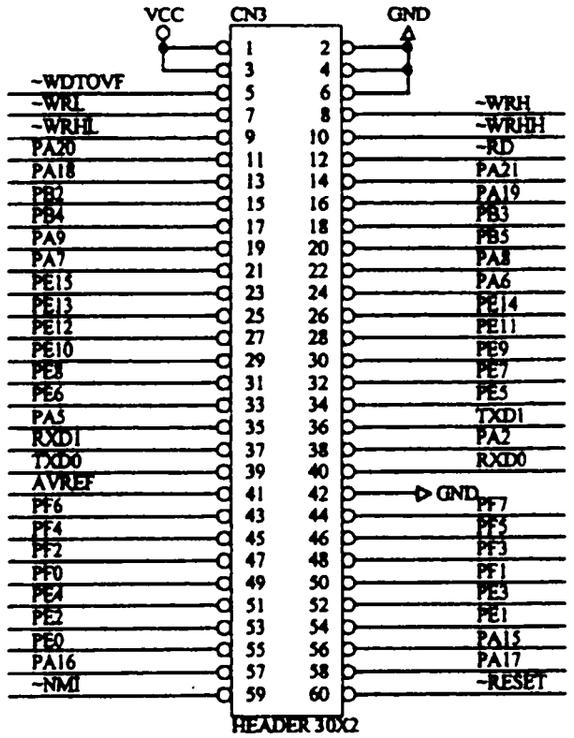
■回路図■

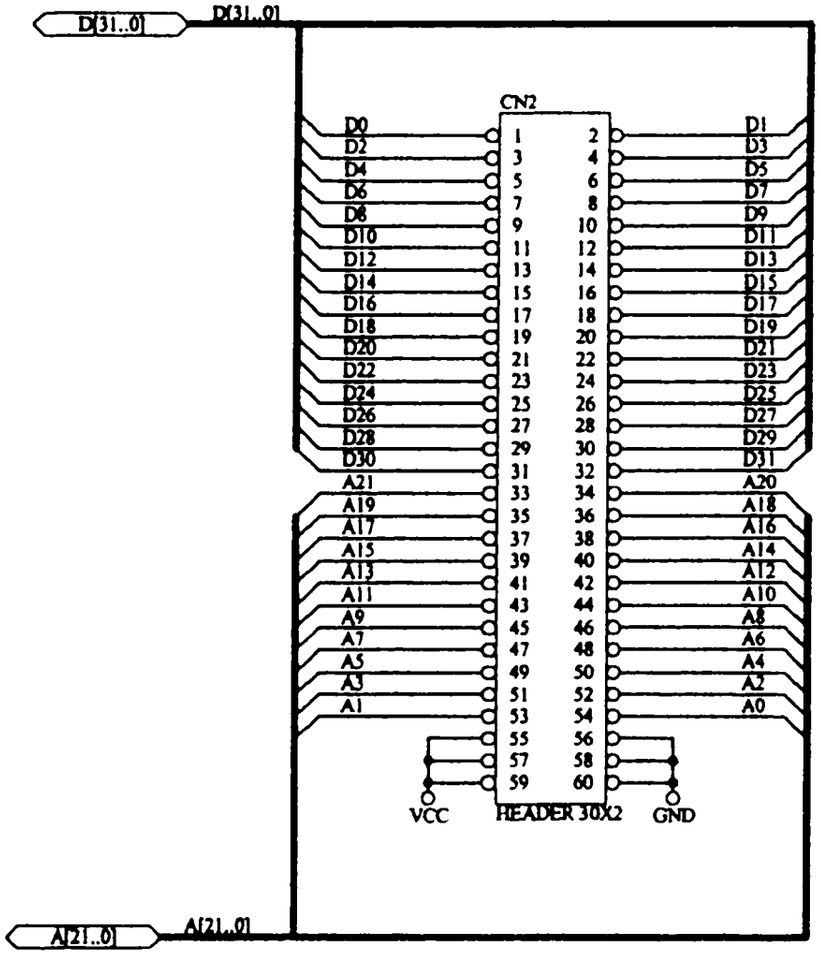

D[31..0]
A[21..0]
rs232c
rs232c.sch
RXD0
TXD0
RXD1
TXD1
U1A
HD64F7045F33
XTAL
EXTAL
VCC(FWP)
io_cn3
io_cn3.sch
memory
memory.sch
~CS0
~CS1
~RD
~WRL
~WRH
~WRHL
~WRHH
R5 10K
R6 10K
VCC
SW1
SWITCH_PUSH
GND
FG
U2
S-80945ANMP-DD9-T2
~RESET
C16 0.1u
R1 220R
R2 220R
R3 220R
C17 0.1u
C18 0.1u
C19 0.1u
R4 3K
C20 470P
U1B
AVCC
AVREF
AVSS
PLLVCC
PLLCAP
PLLVSS
C21 18P
C22 18P
mode
mode.sch
~WDTOVF
~NMI
AVREF

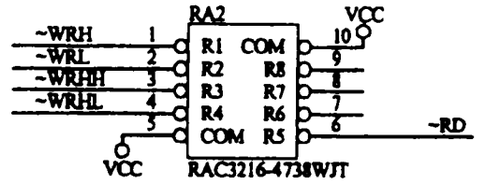

D[31..0]
A[21..0]
U7
HM6216255HLTT-12
U8
HM6216255HLTT-12
U9
27C4096
~STB0700
~STB1508
~STB2316
~STB3124
~CS0
~CS1
~RD
~WR
VCC
GND
~WRL
~WRH
~WRHL
~WRHH
TP3
TEST_POINT
U5A
TC74VHC21F
U4B
TC74VHC08F
U4A
TC74VHC08F
U4C
TC74VHC08F
U4D
TC74VHC08F
TP1
TP2
U5B
TC74VHC21F

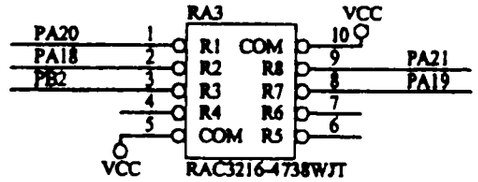
memory.sch

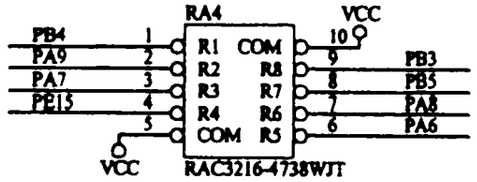

RXD0
TXD0
PA2
RXD1
TXD1
PA3
PA6
PA7
PA8
PA9
~WRL
~WRH
~RD
PA15
PA16
PA17
PA18
PA19
PA20
PA21
~WRHL
~WRHH
PB2
PB3
PB4
PB5
PE0
PE1
PE2
PE3
PE4
PE5
PE6
PE7
PE8
PE9
PE10
PE11
PE12
PE13
PE14
PE15
PF0
PF1
PF2
PF3
PF4
PF5
PF6
PF7
~RESET
~NMI
AVREF
~WDTOVF

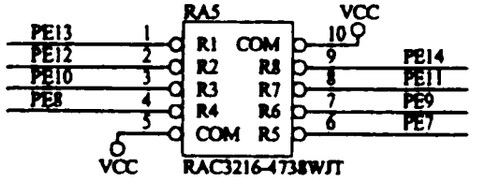

VCC
CN3
GND
~WDTOVF
~WRL
~WRHL
PA20
PA18
PB2
PB4
PA9
PA7
PE15
PE13
PE12
PE10
PE8
PE6
PA5
RXD1
TXD0
AVREF
PF6
PF4
PF2
PF0
PE4
PE2
PE0
PA16
~NMI
~WRH
~WRHH
~RD
PA21
PA19
PB3
PB5
PA8
PA6
PE14
PE11
PE9
PE7
PE5
TXD1
PA2
RXD0
GND
PF7
PF5
PF3
PF1
PE3
PE1
PA15
PA17
~RESET
HEADER 30X2

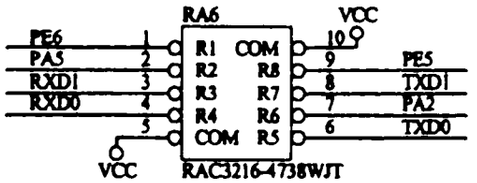

D[31..0]
D[31..0]
CN2
D0
D2
D4
D6
D8
D10
D12
D14
D16
D18
D20
D22
D24
D26
D28
D30
A21
A19
A17
A15
A13
A11
A9
A7
A5
A3
A1
D1
D3
D5
D7
D9
D11
D13
D15
D17
D19
D21
D23
D25
D27
D29
D31
A20
A18
A16
A14
A12
A10
A8
A6
A4
A2
A0
VCC
HEADER 30X2
GND
A[21..0]
A[21..0]

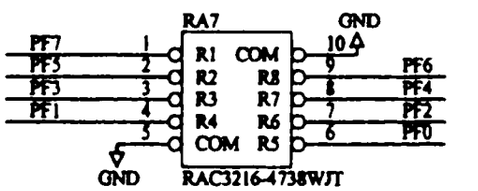

RA2
VCC
~WRH
~WRL
~WRHH
~WRHL
~RD
R1
R2
R3
R4
COM
R8
R7
R6
R5
RAC3216-4738WJT

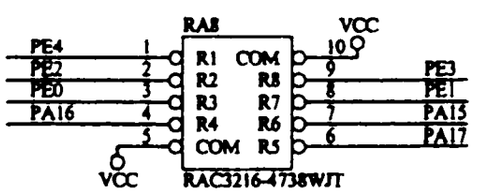

RA3
VCC
PA20
PA18
PB2
PA21
PA19
RAC3216-4738WJT


RA4
VCC
PB4
PA9
PA7
PE15
PB3
PB5
PA8
PA6
RAC3216-4738WJT

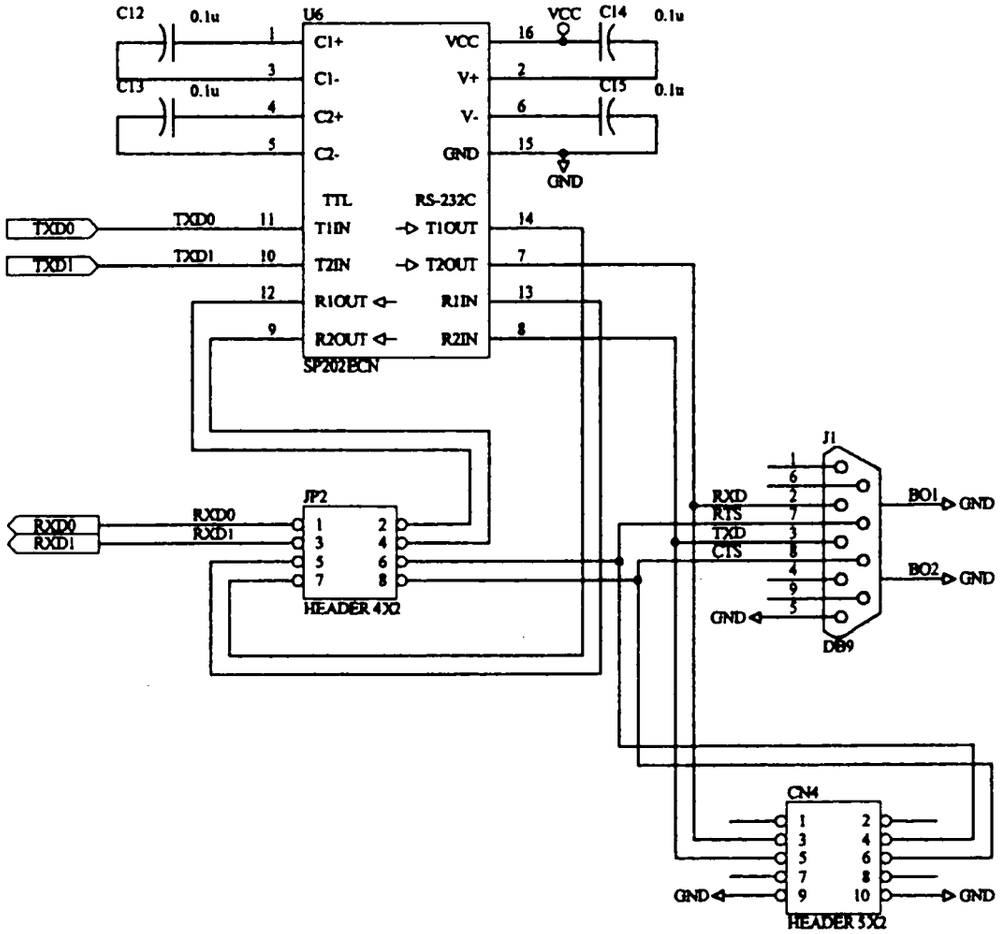

RA5
VCC
PE13
PE12
PE10
PE8
PE14
PE11
PE9
PE7
RAC3216-4738WJT


RA6
VCC
PE6
PA5
RXD1
RXD0
PE5
TXD1
PA2
TXD0
RAC3216-4738WJT

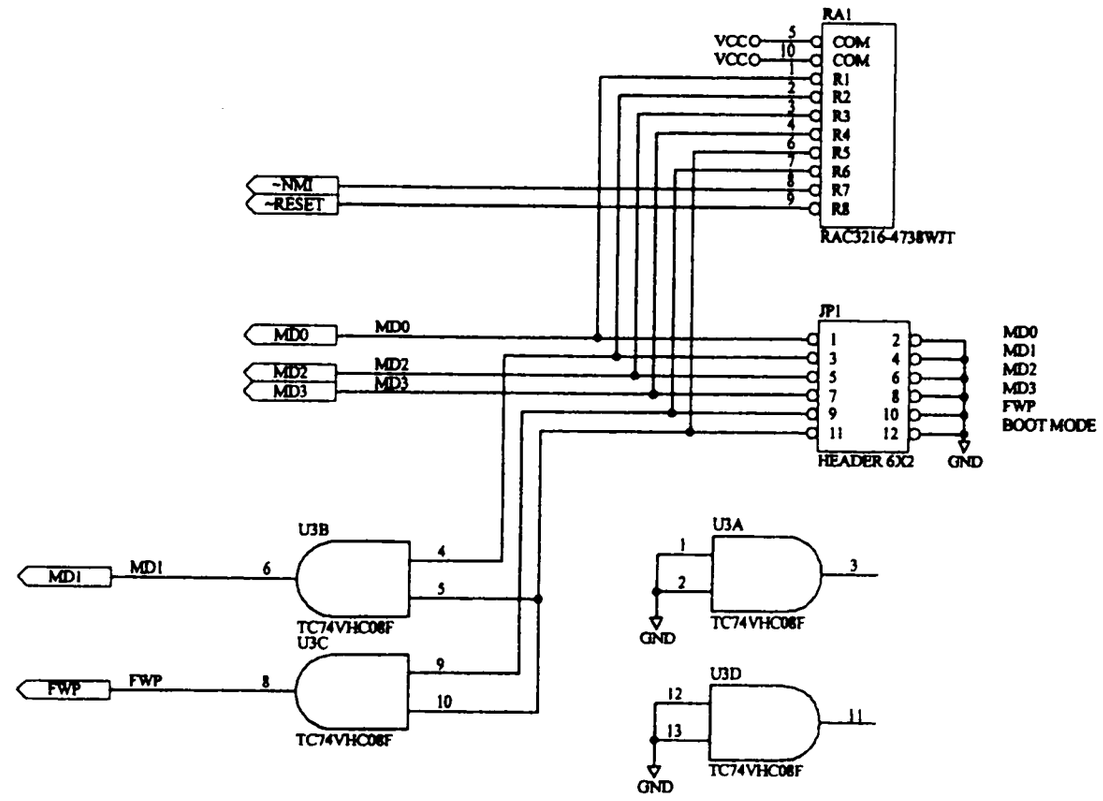

RA7
GND
PF7
PF5
PF3
PF1
PF6
PF4
PF2
PF0
RAC3216-4738WJT

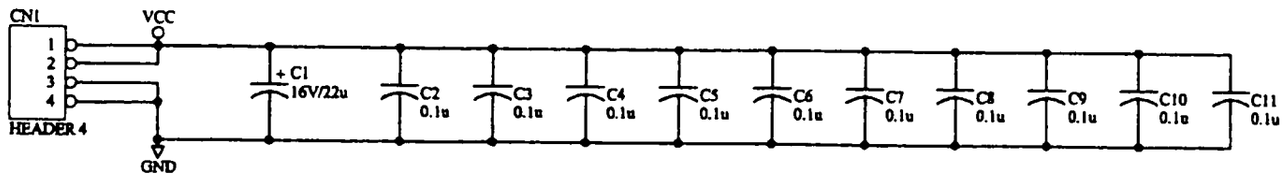

RA8
VCC
PE4
PE2
PE0
PA16
PE3
PE1
PA15
PA17
RAC3216-4738WJT

io_3.sch



r s 2 3 2 c . s c h





m o d e . s c h