R31JP Reference Manual

This manual describes the R31JP. The actual hardware board that we will share with you is an ñMIT versionò that has some custom upgrades. However, this manual accurately describes the ñcore hardwareò on the board. Ask if you have questions.

Some parts of this manual are ñmost relevantò.
These sections include:

Section 1.1 (Introduction and Hardware Overview)
Chapter 3 (Operating Notes)
Chapter 5 (Memory Configuration ï but NOTE ï your RAM is NOT Battery-backed.)
Chapter 7 (CPLD Equations, if you are curious about the XC9536)
Chapter 9 (8051 Microcontroller Overview)
Chapter 10 (8051 ON-CHIP Facilities)
Chapter 11 (Interfacing the Microcontroller System)
SCHEMATICS (Last pages of the document)

We will add additional information about the R31JP you are using in lecture and lab, this document is a ñstarter.ò

TABLE OF CONTENTS

1 INTRODUCTION

The R-31JP with Reads51 (Rigel's Embedded Applications Development System) constitutes a complete system for developing embedded control applications. Reads51 is an Integrated Development Environment (IDE) that currently supports Rigel's 8051 family embedded controller boards. Reads51 contains an assembler, SmallC-compatible C compiler, editor, monitor program, debugger and chip simulator for the 8051 family of microcontrollers. The R-31JP's prototyping area allows custom circuits to be added to the board. Efficient software development and rapid hardware prototyping are combined in a single integrated development environment. Starting November 1999, each board will be populated with an 87C52/89C52 with a Basic interpreter loaded into internal ROM.

The R-31JP board is programmable in Assembly, Basic, C, and Forth. The assembly, SmallC compatible compiler, and Basic are free with each board. We also offer a fuzzy logic software package, FLASH, for programming the 8051.

The R-31JP Version 1.1 board has the following improvements over version 1.0.

- Industrial grade reset chip to guard against brownout or blackout conditions.
- A socketed clock oscillator to generate a very stable CPU clock. The board speed may effortlessly be changed by replacing the oscillator and CPU.
- Optional 128K RAM capability.

1.1 Hardware Overview

R-31JP is designed to work with a variety of 8031 microcontrollers in the 40-pin DIP package. These include but are not limited to: the Dallas Semiconductor 80C320/87C520, the SIEMENS C500 series of processors, the 83C51/52, 87C51/52, 89C51/52 and the Intel 8052 Basic chip. The instruction set for all of these microcontrollers is the MCS-51 instruction set. The R-31JP uses external RAM during the development cycle. Once an application program is developed, it may be permanently placed in EPROM. With an application-specific program installed, the R-31JP may be used as an embedded controller.

The R-31JP board:

- 8031 / 8051 family in the 40-pin DIP format
- 32K of SRAM / optional 128K SRAM / optional battery backed RAM
- 32K of monitor EPROM / optional 32K EEPROM
- 12 general purpose digital input/output bits
- All system signals are available on a 40-pin header
- Address and Data lines are de-multiplexed on two 23-pin headers
- Two-way reset feature gives access to all interrupt vectors
- Fully duplexed RS232 serial port with a RS 232 driver
- Serial port terminates at a DB9 connector
- Power on LED
- Power supplied to the board by way of a 2 position terminal block
- Board operates on +5 volts
- Power consumption is less than 60mA fully populated
- Operating temperature 0 to 70C
- Machine screw sockets under all IC's
- 2 layer 4" x 7" board with a 4" X 3.25" prototyping area for wire-wrapping
- Mounting holes in corners

1.2 Software Overview

1.2.1 Reads51

Rigel Corporation offers 2 versions of our Reads51 software. Please select the version that will work best in your system. We recommend new user's select Reads51 Toolchain 4.

1. Reads51 4.x (IDE, SmallC-compatible 8051 compiler, assembler, linker, editor, chip simulator, assembly language debugger, monitor, 95/98/NT)
2. Reads51 version 2.0 (IDE, assembler, editor, debugger, monitor, DOS -- runs in Win 3.1 box)

Reads51, version 4.x, is Rigel Corporation's Integrated Development Environment for the 8051 family of processors. Reads51 constitutes a complete system for developing embedded control applications when used with Rigel Corporation's 8051 boards. Efficient software development and rapid hardware prototyping are combined in a single integrated development environment. Reads51 v4.x includes an IDE, SmallC-compatible 8051 compiler, assembler, linker, editor, chip simulator, assembly language debugger, and monitor. Reads51 v4.x is written in native 32-bit code to run on Windows95/98 and WindowsNT. Reads51 includes a sophisticated project management system to simplify code reusability and version control. Reads51 supports a full debugger in assembly language. The debugger allows you to step through your code with breakpoints and variable watches as the compiled code runs on the target board, similar to the operation of an in-circuit emulator.

V4 Compiler
The compiler is written to accompany Rigel's educational packages. It is SmallC compatible (integer and char only, one-dimensional arrays, one level of indirection, i.e. pointers). Please refer to books on SmallC for more information.

Running Compiler-Generated Code
The compiler is written for Rigel's 8051 family of boards. Currently the memory map is fixed. It assumes overlapped code and data memory. After building your project, download it to the board and swap the memory (R31JP or R515JC slide switch or R-535J RUN button) so that RAM occupies the lower block. Release the RESET button to run the code.

The Reads51 software has the following distinctive features:

- Project management for organized software development
- Multiple project management with drag and drop module transfers
- Enhanced graphical user interface for easy monitoring
- Stand alone compiler and editor applications connected to Reads51 in a client/server fashion

The 8051 boards are designed to communicate with a PC (IBM PC or compatible) acting as a host. The host-to-board communications are carried out through a serial port (COM1 - COM4).

The monitor program (RROS) includes a monitor system and user-accessible system calls for control and communication support. The RROS monitor may be used to communicate with an ASCII terminal when the PC host is unavailable. The source code of the user-accessible systems calls is provided. These routines as well as all examples in the User's Guide and on the distribution disk may be used or incorporated into applications by the registered buyer without any royalties, fees, or limitations. Rigel Corporation is not responsible for the suitability or correctness of the example software. Refer to the warranty for additional information.

1.2.2 BASIC Interpreter

The BASIC interpreter programmed in the 87C52/89C52 chips on the R-31JP board is the standard Intel interpreter released to public domain by Intel. For more information, you may contact Rigel. We have an Intel Basic Manual in PDF format you may download from our web site. You will need to contact us for access to this file.

1.2.3 Rb52 Host

The RbHost combines a terminal emulator and an editor with BASIC language syntax highlighting. It facilitates writing BASIC programs and downloading them to Rigel's embedded control boards. The BASIC interpreter may be in internal ROM, external ROM, or in RAM.
Most BASIC interpreters, such as the Intel 8052-BASIC, are complete development systems placed in internal ROM of the microcontroller. However, most interpreters do not have powerful edit functions. Thus, it is much

more convenient to write source programs in RbHost and then download them to the interpreter. Similarly, RbHost provides a convenient way to store the source programs on the host PC.

1.2.4 Example Software

Tutorial source code is provided to experiment with the capabilities of the R-31JP board and Reads51. Examples are designed to illustrate the features of the 8051 family of microcontrollers, specifically digital and serial input/output, timers and counters, and interrupt logic. The example software may be found in the Work directory of the Reads51 software. Please refer to the comments embedded in the programs for further information.

Users are encouraged to modify the circuit diagrams and example software in developing their own specific applications. The source code of the user-accessible systems calls, as well as all examples on the distribution disk may be used or incorporated into applications by the registered buyer without any royalties, fees, or limitations. Rigel Corporation is not responsible for the suitability or correctness of the example software. Refer to warranty for additional information.

1.3 Package List

Your R-31JP / READS package includes the following:

1. R-31JP with an 87C52 with Basic (Nov 1999)
2. R-31JP User's Guide
3. Assembly instructions and parts package (for unassembled kits only).
4. Reads51 software and User's manual which is downloadable from the web. May be included with the package.

A serial modem cable with a male DB9 connector and a regulated 5 volt 500mA power source are to be supplied by the user.

2 SOFTWARE SETUP

2.1 System Requirements

Reads51 Version 4.x is designed to work with an IBM PC or compatible, 486 or better, running Windows 95, 98, or Windows NT. The newest version of the software is always available to download off our web site, www.rigelcorp.com. We encourage you to check our web site often to keep up-to-date.

2.2 Software Installation, Reads51

If you receive a CD from Rigel, follow these steps:

1. Place the CD-ROM in your drive.
2. Go to the Rigel Products | 8051 Software | Reads51 | Win95-nt | and click on the SetupReads400.exe file. The program will then install in your system.
3. Follow the standard install directions answering the questions with the appropriate answers

If you download the software from the web, (www.rigelcorp.com)

1. Click on the SetupReads400.exe file. The program will then install in your system.
2. Follow the standard install directions answering the questions with the appropriate answers

2.3 Quick Start

2.3.1 R-31JP Start-Up When Using Reads51

1. Check to make sure a jumper is in the EA# header.
2. Check to make sure both jumpers in J4 and J5 are in the ROM position.
3. Check to make sure the slide switch is in the MON position.
4. Connect the R-31JP to the PC host via a modem cable.
5. Connect the R-31JP to a well-regulated 5-Volt supply. The red LED should light up when power is connected.
6. Run the Reads51 IDE by selecting **Start | Programs | Reads51**. You may also start Reads51 by double clicking on the Reads51 short cut icon if installed.
7. Specify the serial port (COMM Port) that is connected to the board by opening the **Options | TTY Options** window.
8. Select the Toolchain and Target platform by selecting **Options | Toolchain/Target** and selecting Reads51 Toolchain v4 and the target RROS.
9. Open the TTY window using the menu command **View | TTY Window**.
10. Press RESET on the embedded controller board and observe the prompt in the TTY window.

2.3.2 Verifying that the Monitor is Loaded

Make sure the TTY window is active, clicking the mouse inside the TTY window to activate it if necessary. Then type the letter '*H*' (case insensitive) to verify that the monitor program is responding. The 'H' command displays the available single-letter commands the monitor will recognize.

The READS monitors use single-letter commands to execute basic functions. Port configurations and data, as well as memory inspection and modifications may be accomplished by the monitor. Most of the single-letter commands are followed by 4 hexadecimal digit addresses or 2 hexadecimal digit data bytes.

The list of monitor commands is displayed with the H command while the monitor program is in effect. The H command displays the following table.

B xxxx	sets Break point at address xxxx
C xxxx-xxxx	displays Code memory
D xx-xx	displays internal Data ram
D xx=nn	modifies internal Data ram
D xx-xx=nn	fills a block of internal Data ram
G xxxx	Go - starts executing at address xxxx
H	Help - displays monitor commands
K	Kills (removes) break point

L	down Loads Intel hex file into memory
P x	displays data on Port x
P x=nn	modifies data on Port x to nn
R	displays the contents of the Registers
S	displays Special function register addresses
S xx-xx	displays Special function registers
S xx=nn	modifies Special function registers
S xx-xx=nn	fills Special function registers
X xxxx-xxxx	displays eXternal memory
X xxxx=nn	modifies eXternal memory
X xxxx-xxxx=nn	fills eXternal memory

A single-letter command may be followed by up to 3 parameters. The parameters must be entered as hexadecimal numbers. Each 'x' above represents a hexadecimal digit (characters 0..9, A..F). Intermediate spaces are ignored. Alphabetic characters are converted to upper case. The length of the command string must be 16 characters or less. The command syntax is:

Letter [address][-address][=data]<CR>.

2.3.2 Downloading and Running an Assembly Program

1. Use the use the **Project I Open Project** command to open the project AbsoluteAssembly01.
2. Assemble the program using the **Compile I Build** command.
3. Use the **Compile I Toggle Mode** command to switch to the Run/Debug Mode.
4. Click on the **Compile I Run** command and specify the starting address 8000 (hex).
5. Again the **Compile I Toggle Mode** command to revert back to the Build Mode.

2.3.3 Downloading and Running a C Program

1. Use the **Project I Open Project** command to open the project Hello.
2. Compile the program and download it to the board using the **Compile I Build and Download** command. The project will be compiled and the resultant HEX code will be downloaded to the target board.
3. Press and hold the RESET button on the board. While the RESET button is pressed, flip the MON / RUN switch to the RUN position. This swaps the memory map on the board so that RAM occupies low memory. The HEX code downloaded to RAM executes when you release the RESET button.

2.4 Basic Set-Up

2.4.1 R-31JP Start-Up When Using Basic

1. Remove the jumper from the EA# header.
2. Check to make sure both jumpers in J4 and J5 are in the ROM position.
3. Insert the BASIC jumper.
4. Check to make sure the slide switch is in the RUN position.
5. Connect the R-31JP to the PC host via a modem cable.
6. Connect the R-31JP to a well-regulated 5-Volt supply. Both LED's should light up when power is connected.
7. Open the TTY window and press the space bar on the keyboard several times. The Basic prompt should appear in the TTY window.
8. You may now begin to write your BASIC program.

Please note that when the BASIC jumper is inserted both LEDs light up, and that the MON/RUN slide switch has no effect.

2.4.2 System Requirements for the Rb52 Host

Rb52 Host is designed to work with an IBM PC or compatible, 486 or better, running Windows 95, Windows 98, or Windows NT.

2.4.3 Software Installation, Rb52 Host

From the CD. Go to the **Rigel Products I 8051 Software I Rb52 Host I** and click on the .exe file. The program will begin to load in your system. Follow the standard install directions answering the questions with the appropriate answers.

From the file downloaded from the web. Click on the .exe file. The program will begin to load in your system. Follow the standard install directions answering the questions with the appropriate answers.

3 OPERATING NOTES

3.1 Overview

R-31JP uses the 8031 family of microcontrollers in the 40-pin DIP package. These microcontrollers use the MCS-51 instruction set. The R-31JP uses external RAM during the development cycle. Once an application program is developed, it may be permanently placed in EPROM. With an application-specific program installed, the R-31JP may be used as an embedded controller. Starting November 1999, we will be populating the R-31JP with the 87C52 processor with the Basic Interpreter loaded into ROM. The following are general guides to operating the board.

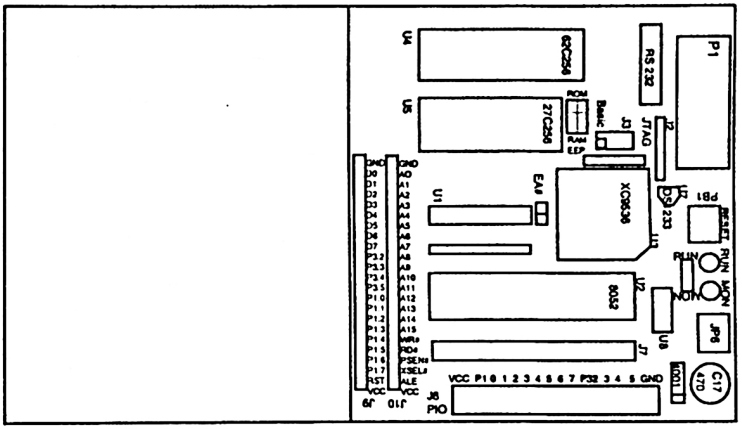


Figure 3.1 Top Board Overlay

The R-31JP has 14 terminal blocks connected to Port 1, 4 bits of Port 3 and Ground and VCC. Each port may be used as either an input or an output port. When used as output ports, it is recommended that the input ports, first write ports sink current. Similarly, when used as drive the port low. 1's to the ports and have the external signal bits may be connected The figures in Section 7 shows how the port devices. to switches, LEDs, or higher power output

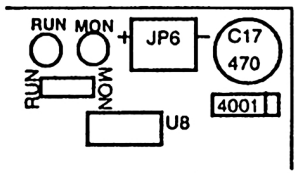


Figure 3.2 Power Connector

3.2 Power

Power is brought to the R-31JP board by a two-position screw- type terminal block, J6. A well-regulated (+/- 5%) 5V DC source is required. The (+) and (-) terminals are marked on the board. Note that a diode is placed across the input in reverse. If the power is applied to the R-31JP board in reverse polarity, the diode will short the power supply attempting to prevent damage to the board.

3.3 Serial Port

The serial port is accessed through an RS-232 level converter. The microcontroller supports the transmit and receive signals. A minimal serial port is constructed with just 3 lines: transmit, receive, and ground, disregarding all hardware handshake signals. P1 of the R-31JP is a DB-9 female connector used to connect the board to an IBM compatible PC. A straight-through modem cable may be used. That is a cable connecting pin 2 of the R-31JP to pin 2 of the host, and similarly pin 3 to pin 3, and pin 5 to pin 5.

3.4 Switches

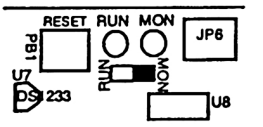


Figure 3.3 Default Settings

The R-31JP has one reset button and one slide switch. The push button resets the board. The slide switch changes the memory configuration. The slide switch in the MON position, causes the EPROM in socket U5 to occupy the lower 32K of memory, and the static RAM in socket U4 to occupy the higher 32K of memory. With this configuration, the microcontroller executes the monitor program in the EPROM. The slide switch in the RUN position causes the EPROM to occupy the higher 32K of memory, and the static RAM to occupy the lower 32K of memory.

3.5 LEDs

There are two LEDs on the R-31JP. The LEDs indicate the current operating mode of the board. They also provide visual confirmation that power is applied.

The LEDs light up depending on the operating mode which in turn is determined by the position of the slide switch. The MON LED (red) is lit when the R-31JP (with an 8031) is run with the monitor program in the lower 32K of memory. The RUN LED (green) is lit when the board is operated with the RAM in the lower 32K of memory.

3.6 Prototyping Area

There are two traces that follow the edge of the prototyping area. These traces are visible from the bottom of the board. Each trace may be connected to VCC (5 volts) or GND (ground). **Do not connect GND and VCC to the same trace.** Two pads are placed at the either end of the traces to facilitate the distribution of power to the prototyping area.

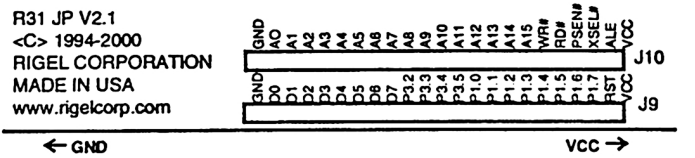


Figure 3.4 GND and VCC Pads

4 JUMPER SELECTION

4.1 EA#

EA# (J1) is the 2 post header in the center of the board. EA# selects between external memory on the board and the on-board ROM of a processor, when present. The jumper, when installed, causes all instructions to be fetched from external memory. For most processors, the jumper will need to be installed. When using the Basic on the 87C52 the EA# jumper will need to be removed to access the on-board ROM where the Basic is located.

4.2 EPROM / RAM Select

J4 and J5 are 3 post headers marked ROM/RAM on the top overlay of the board. The U5 socket on the board may be populated with EPROM, RAM or EEPROM. The position of the jumpers in J4 and J5 determine which memory device is activated in the U5 socket. Both jumpers need to be populated and they need to be moved as a pair for the proper operation of the board. With both jumpers connecting the center posts to the left posts (marked ROM), the EPROM is selected. With both jumpers connecting the center posts to the right posts (marked RAM), the RAM/EEPROM mode is selected. In this configuration either a RAM or EEPROM may be placed in the U5 socket. (U4 can be populated with a RAM or EEPROM and no jumper selection is needed for this socket.) The board is populated from the factory with an EPROM in U5 and a RAM in U4.

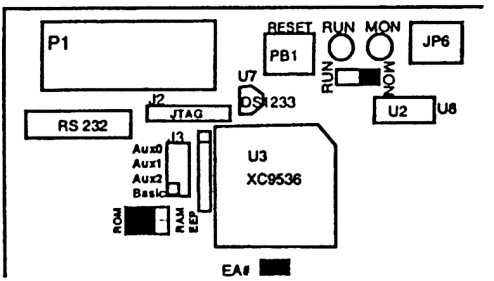


Figure 4.1 Default Jumper Selection For 80C52

4.3 J3

J3 is a two by four header. These headers should be left unpopulated unless you are using BASIC on the board. If you are using BASIC you should insert a jumper in this header in the position marked BASIC.

4.4 J2, JTAG

The header marked J2 on the board is the JTAG header used for programming the PLD on the board.

4.5 Default Settings When Using Reads51

The following are the default settings for the R-31JP as sent from the factory.

1. A jumper in the EA# header.
2. Both jumpers in J4 and J5 in the ROM position.
3. The slide switch in the MON position.

4.6 Default Settings When Using BASIC

1. The jumper in the EA# header removed.
2. Both jumpers in J4 and J5 in the ROM position.
3. A jumper inserted in J3, in the BASIC position.
4. The slide switch in the RUN position.

5 MEMORY CONFIGURATION

The R-31JP has 64K of memory, of which 32K is EPROM and 32K is static RAM. Program memory and external data memory are decoded to overlap. In this configuration, programs may be downloaded from the PC host and placed in RAM as data, and subsequently executed as program instructions.

5.1 Memory Addressing

The 8031 family of microcontrollers address 64K of program. The microcontroller may read from both external data memory and external code memory using the `movx` and `movc` instructions. The microcontroller may write to external data memory but may not write to external code memory.

The microcontroller pin Program Segment Enable (PSEN#) is activated (made logic 0) when a byte is to be read from external program memory, and pin Read (RD#), when a byte is to be read from external data memory. By combining these signals by an AND gate (PSEN# AND RD#), the same physical 64K memory block is made to appear as both external code and data memory. The default R-31JP configuration overlaps external data and code memory blocks by combining PSEN# and RD# in this manner. This allows downloading and running programs on the R-31JP. The pound sign (#), when used as a suffix signifies that the signal is active when low. For example, external memory is enabled when the EA# line is held at logic low (close to ground voltage).

5.2 Memory Options

R-31JP has a 28-pin socket U5 and a 32-pin socket U4. The socket U5 holds 32K of memory. Socket U5 accepts a 27C256 EPROM, a 28C256 EEPROM or 62C256 static RAM. See section 4.2 for jumper selection. U4 accepts a 62C256 static RAM, a 28C256 EEPROM, or a 128K RAM.

U5 is mapped as the lower half of memory, [0..7FFFh] and U4 holds the higher half of memory [8000h..FFFFh] when the slide switch S1 is in the MON position. In its RUN position, the high and low 32K memory blocks are swapped. That is, U5 is decoded to be the high memory block and U4, the low memory block. The slide switch is used when downloaded programs need direct access to the interrupt vectors located in low memory. Such a program, with its origin at 0, is first downloaded into a RAM device placed in U4. While holding the RESET button down, S1 is moved to its RUN position. Now the program in U4 is in the low memory block, starting at address 0. Releasing the RESET button executes the user program in U4. The user program may then have direct access to all interrupt vectors.

5.3 Stand-Alone Mode

When you finalize your code and want to run it on the board in a stand-alone mode you have two options, you can put your code in an EPROM or run it from a battery-backed RAM.

5.3.1 Running Code from EPROM

Once your code is debugged, you may burn it in an EPROM and put it on the board in place of the standard RROS EPROM found in U5. The starting address for your program must be set to zero. The memory map is the same as RROS: 0-7FFFh is EPROM and 8000h-FFFFh is RAM so the jumpers and switch (MON) stays in the same position as the RROS mode. In this configuration when you press reset on the board (or at power up) program execution starts at address 0 -- the beginning of your program.

Please remember that your program should use the address range 8000h-FFFFh for external data memory (external RAM).

Also remember that if your application uses any of the system utility functions (such as the serial communication functions), they must be included in your code. You may simply include the source (given in "syscalls.inc") of the utility functions into your application.

5.3.2 Running Code from Battery-Backed RAM

5.3.2.1 U4

Once your code is debugged, you may download it into a battery-backed RAM placed in U4. The starting address for your program must be set to zero. The jumper settings remain the same, the slide switch must be

changed to the RUN position. In this configuration when you press reset on the board (or at power up) program execution starts at address 0 -- the beginning of your program.

Please remember that if your application uses any of the system utility functions (such as the serial communication functions), they must be included in your code. You may simply include the source (given in "syscalls.inc") of the utility functions into your application.

5.3.2.2 U5

Once your code is debugged, you may download it into a battery-backed RAM placed in U4. The starting address for your program must be set to zero. Once you have downloaded your program to the battery-backed RAM you may place the battery-backed RAM into U5 and place a standard RAM in U4. The jumper settings J4 and J5 determine which memory device is activated in the U5 socket. With a battery-backed RAM in U5 both jumpers need to be shifted to the right position, connecting the center posts to the right posts (marked RAM). The slide switch must be MON position. The memory map is the same as RROS: 0-7FFFh is now battery-backed RAM and 8000h-FFFFh is RAM. In this configuration when you press reset on the board (or at power up) program execution starts at address 0 -- the beginning of your program.

Please remember that your program should use the address range 8000h-FFFFh for external data memory (external RAM).

Also remember that if your application uses any of the system utility functions (such as the serial communication functions), they must be included in your code. You may simply include the source (given in "syscalls.inc") of the utility functions into your application.

6 HEADERS

6.1 System Headers

All system signals are available on the 40-pin header marked J7. The pin assignments are given below. These correspond one-to-one with the processor pins. Pin 35 and 37 are not connected on this header, as they are the crystal pins of the processor. Refer to the circuit diagram for additional information.

Pin	Signal	Pin	Signal
1	P1.0	2	VCC
3	P1.1	4	P0.0
5	P1.2	6	P0.1
7	P1.3	8	P0.2
9	P1.4	10	P0.3
11	P1.5	12	P0.4
13	P1.6	14	P0.5
15	P1.7	16	P0.6
17	RESET	18	P0.7
19	P3.0	20	EA#
21	P3.1	22	ALE
23	P3.2	24	PSEN#
25	P3.3	26	P2.7
27	P3.4	28	P2.6
29	P3.5	30	P2.5
31	P3.6	32	P2.4
33	P3.7	34	P2.3
35	-	36	P2.2
37	-	38	P2.1
39	GND	40	P2.0

6.2 System Bus

The system bus is available on two 23-pin headers marked J9, J10. The pin assignments are given below. The Address and Data lines of the processor are de-multiplexed on this header. The system bus facilitates interfacing the R-31JP to external memory-mapped input/output devices. The pin XIOSEL# (eXternal I/O SELect) is low for the address range FE00h-FEFFh. No memory access is performed in this address range. The user may decode the address, along with the XIOSEL#, RD#, and WR# signals to select memory-mapped peripherals.

J9 (BUS 1)		J10 (BUS 2)	
Pin	Signal	Pin	Signal
1	VCC	1	VCC
2	RESET	2	ALE
3	P1.7	3	XIOSEL#
4	P1.6	4	PSEN#
5	P1.5	5	RD# (P3.6)
6	P1,4	6	WR# (P3.7)
7	P1.3	7	A15 (P2.7)
8	P1.2	8	A14 (P2.6)
9	P1.1	9	A13 (P2.5)
10	P1.0	10	A12 (P2.4)
11	P3.5	11	A11 (P2.3)
12	P3.4	12	A10 (P2.2)
13	P3.3	13	A9 (P2.1)
14	P3.2	14	A8 (P2.0)
15	D7 (P0.7)	15	A7
16	D6 (P0.6)	16	A6
17	D5 (P0.5)	17	A5
18	D4 (P0.4)	18	A4
19	D3 (P0.3)	19	A3
20	D2 (P0.2)	20	A2
21	D1 (P0.1)	21	A1
22	D0 (P0.0)	22	A0
23	GND	23	GND

7 CPLD EQUATIONS

7.1 November 1999

```
module r31jp
Title 'R31JP -  single (top) ABEL file CPLD implementation'

r31jp device;

Declarations
// inputs
A8       pin  7;
A9       pin  8;
A10      pin  9;
A11      pin 11;
A12      pin 12;
A13      pin 13;
A14      pin 14;
A15      pin 18;
ALEIN    pin 40;
ALEDISX  pin  2;
PSENX    pin 42;
PPX      pin  3;
PEX      pin  4;
WRX      pin  5;
RDX      pin  6;
AUX0     pin 19;
AUX1     pin 20;
AUX2     pin 22;
BASICX   pin 24;
MONRUN   pin 27;
RSTINX   pin 39;

// outputs
MA15     pin 35 istype 'com';
MA16     pin 34 istype 'com';
ALEOUT   pin 33 istype 'com, neg';
ROMSELX  pin  1 istype 'com, pos';
ROMRDX   pin 44 istype 'com, pos';
ROMWRX   pin 43 istype 'com, pos';
RAMSELX  pin 38 istype 'com, pos';
RAMRDX   pin 37 istype 'com, pos';
RAMWRX   pin 36 istype 'com, pos';
LEDMONX  pin 28 istype 'com, pos';
LEDRUNX  pin 29 istype 'com, pos';
RSTOUT   pin 25 istype 'com, neg';
XIOSELX  pin 26 istype 'com';

// --- CUTOFF ---
// the BASIC interpreter generates the signal PPX (ProgramPulse#) to burn ROMs.
// EEPROM write signals need to be shorter, similar to the regular WR# in
length.
// CUTOFF uses ALE to disables the ROMWRX, effectively shortening PPX.
CUTOFF   node istype 'reg';  // terminates PPX by the BASIC interpreter to burn
EEPROMs
```

```
Equations

XIOSELX     = ! ( !A8 & A9 & A10 & A11 & A12 & A13 & A14 & A15 );  //
[0xFE00..0xFEFF]
RSTOUT      = ! RSTINX;

CUTOFF     := 1;
CUTOFF.CLR  = PPX;

* --- BASIC mode ---
* the AUX0 jumper determines ROMWR#:

* AUX0 jumper     EEPROM write mode
* -----------     -----------------
*  removed         BASIC PROGx
*  inserted        WR# (erase EEPROM)
*
* insert AUX0 to program or erase EEPROMs (see documentation)

 when(!BASICX & AUX2) then
  {
   MA15       = A15;
   MA16       = 0;
   ALEOUT     = ALEIN & ALEDISX;

   RAMSELX    = A15 # !XIOSELX;
   ROMSELX    = !A15 # !XIOSELX;

   RAMRDX     = RDX;
   RAMWRX     = WRX;

   ROMRDX     = PSENX & RDX;
   when(AUX0) then       // BASIC PROGx
    {
     CUTOFF.CLK = ALEIN;
     ROMWRX     = PPX # CUTOFF;
     }
    else // WR#
    {
     CUTOFF.CLK = 0;
     ROMWRX     = WRX;
     }

   LEDMONX    = 0;
   LEDRUNX    = 0;
   }
* --- NVRAM mode ---
* 128K RAM (LOW RAM : 0..FFFF, HIGH RAM : 10000..1FFFF)

*        CODE        XDATA
*       ------      -------
* MON : EPROM       LOW RAM
* RUN : LOW RAM     HIGH RAM
 else when(BASICX & !AUX0 & AUX1 & AUX2) then
  {
   ALEOUT     = ALEIN;
   CUTOFF.CLK = 0;  // CUTOFF not used
```

```
    ROMWRX      = 1;

    when(MONRUN) then
     {
      MA16      = 0;
      ROMWRX    = 1;
      LEDMONX   = 0;
      LEDRUNX   = 1;

      when(!PSENX) then
       {
        MA15     = A15;
        ROMSELX  = !XIOSELX;
        ROMRDX   = 0;
        RAMSELX  = 1;
        RAMRDX   = 1;
        RAMWRX   = 1;
        }
       else
        {
         MA15     = !A15;       // RROS downloads invert A15
         RAMSELX  = (RDX & WRX) # !XIOSELX;
         RAMRDX   = RDX;
         RAMWRX   = WRX;
         ROMSELX  = 1;
         ROMRDX   = 1;
         }
      }
     else // RUN mode
      {
       ROMSELX =  1;          // both CODE and XDATA are in RAM
       ROMRDX  =  1;
       ROMWRX  =  1;
       RAMSELX = !XIOSELX;

       MA15        = A15;
       LEDMONX     = 1;
       LEDRUNX     = 0;

       when(!PSENX) then
        {
         MA16     = 0;
         RAMRDX   = 0;
         RAMWRX   = 1;
         }
        else
         {
          MA16     = 1;
          RAMRDX   = RDX;
          RAMWRX   = WRX;
          }
        }
    }
  --- RROS mode ---
   else
    {
     MA16        = 0;
```

```
    ALEOUT      = ALEIN;
    CUTOFF.CLK = 0;  // CUTOFF not used

    RAMRDX      = PSENX & RDX;
    RAMWRX      = WRX;
    ROMRDX      = PSENX & RDX;
    ROMWRX      = 1;

    when(MONRUN) then
     (
      MA15 =  A15;
      ROMSELX =  A15 # !XIOSELX;
      RAMSELX = !A15 # !XIOSELX;
      LEDMONX = 0;
      LEDRUNX = 1;
      )
     else
      (
       MA15    = !A15;
       ROMSELX = !A15 # !XIOSELX;
       RAMSELX =  A15 # !XIOSELX;
       LEDMONX = 1;
       LEDRUNX = 0;
       )
    )

end r31jp
```

Boots to mimon

if xiosels = 0 (we're trying to go to I/O)

all Selx lines go high.

all mem chips deactivated

7.2 January 1998-November1999

```
    module CtrlBus
    Title 'CtrlBus'

    Declarations
    A15      PIN;
    ALE      PIN;
    ALEDISX  PIN;
    PSENX    PIN;
    PPX      PIN;
    PEX      PIN;
    WRX      PIN;
    RDX      PIN;
    BASICX   PIN;
    MON      PIN;
    AUX0     PIN;
    AUX1     PIN;
    AUX2     PIN;

    ROMSELX PIN istype 'com, pos';
    ROMRDX  PIN istype 'com, pos';
    ROMWRX  PIN istype 'com, pos';
    RAMSELX PIN istype 'com, pos';
    RAMRDX  PIN istype 'com, pos';
    RAMWRX  PIN istype 'com, pos';
    A15M    PIN istype 'com';
    A16M    PIN istype 'com';
    ALEOUT  PIN istype 'com, neg';
```

```
LEDMONX PIN istype 'com, pos';
LEDRUNX PIN istype 'com, pos';

Equations
 A16M     = 0;
 ALEOUT   = ALE;
 RAMRDX   = PSENX & RDX;
 RAMWRX   = WRX;
 ROMRDX   = PSENX & RDX;
 ROMWRX   = 1;

 when(MON) then
 {
 A15M     =  A15;
 ROMSELX  =  A15;
 RAMSELX  = !A15;
 LEDMONX  = 0;
 LEDRUNX  = 1;
 }
 else
       {
 A15M     = !A15;
 ROMSELX  = !A15;
 RAMSELX  =  A15;
 LEDMONX  = 1;
 LEDRUNX  = 0;
 }
end CtrlBus
```

8 R-31JP WITH OPTIONAL PROCESSORS

The R-31JP may be used with a variety of 8051 processors in the 40-pin dip package. The following sections give instructions for some of the more popular 8051 variations. For more information on these microcontrollers please contact the manufacturer. The addresses of major manufacturers are listed at the end of this User's Guide.

8.1 A 83C51 / 87C51

The 83C51 and 87C51 processors are 8051 compatible with user programmable ROM/EPROM memory added. These features make the chips very popular for embedded applications. The 83C51 is One Time Programmable (OTP), which means the program cannot be changed once it is burned into the chip. The 87C51 is an EPROM version. It can be erased and reprogrammed multiple times. When using these microcontrollers on the R-31JP, remove jumper J1 (EA#). This will allow you to use the on-board ROM in the processor. The jumpers in J4 and J5 need to be configured according to the memory devices you are using.

8.2 The DS80C320

The Dallas Semiconductor's DS80C320 is a fast 80C31/80C32-compatible microcontroller. The increase in speed is obtained in two ways. First by using 4 clocks per machine cycle instead of the normal 12. Second, the processor will run at clock speeds of up to 25MHz. These two changes result in an apparent execution speed of 62.5 MHz. Typical applications will see a speed improvement of 2.5 times using the same code and same clock. Enhanced features include:

- 4 clocks/machine cycle
- Wasted cycles removed
- Runs DC to 25MHz clock rates
- Uses less power for equivalent work
- Dual data pointers
- Power-fail reset
- Programmable Watchdog timer
- Early-warning power-fail interrupt
- Two full-duplex hardware serial ports
- 13 total interrupt sources with 6 external interrupts

The R-31JP comes populated with an 11.0592MHz clock. In order to run the DS80C320 at 25MHz the clock needs to be replaced. This can easily be done by the end user. Rigel will also populate the board with different clocks before shipment if requested.

To use the DS80C320 on the R-31JP, jumper EA# needs to be installed. The jumpers in J4 and J5 need to be configured according to the memory devices you are using.

8.3 The SAB-C500 Series

The SAB-C500 processors are Siemens' high-speed 8051 compatible microcontrollers. They come in three versions, the 501, 502, and 503. The SAB-C501 and the SAB-C502 will run on the R-31JP. At this time the SAB-C503 is only available in the 44 PLCC package. These processors have the following enhanced features:

SAB-C501

- Versions for up to 40MHz clock frequency
- Three 16-bit timers / counters
- USART
- Six interrupt sources, two priority levels
- Power saving modes

SAB-C502

- Versions for up to 20MHz clock frequency
- Eight data pointers for indirect addressing of program and external data memory

- Three 16-bit timers / counters
- USART with programmable 10-bit Baud rate-Generator
- Six interrupt sources, two priority levels
- Programmable 15-bit Watchdog timer
- Oscillator Watchdog
- Fast power on reset
- Power saving modes

The R-31JP comes populated with an 11.0592MHz clock. In order to run the C500 processors at faster speeds, the clock needs to be replaced. This can easily be done by the end user. Rigel will also populate the board with different clocks before shipment if requested.

To use the SAB-C501 / SAB-C502 on the R-31JP, jumper EA# needs to be installed. The jumpers in J4 and J5 need to be configured according to the memory devices you are using.

8.4 The DS5000 / DS5000T

The Dallas Semiconductor's DS5000 is compatible with the 8051, but has addition features. These features include:

- A high speed, nonvolatile static CMOS RAM for program and/or data memory storage
- Capable of modifying its own program and/or data memory in end use
- 128 internal nonvolatile registers for variable retention
- Initial downloading of software in end system via on-chip serial port
- Maintains all nonvolatile resources for 10 years in the absence of VCC
- Orchestrates orderly shutdown and automatic restart on power up/down
- Automatic restart on detection of errant software execution
- Executes encrypted software to prevent unauthorized disclosure

The Dallas Semiconductor's DS5000T is an enhanced DS5000, compatible with the 8051, but with addition features which include:

- Embedded clock/calendar
- Internal lithium cell preserves clock function in the absence of VCC
- Permits logging of events with time and date stamp
- Clock accuracy of better than 2min/month @25C

To use the DS5000 / DS5000T on the R-31JP, jumper EA# needs to be installed. The jumpers in J4 and J5 need to be configured according to the memory devices you are using.

9 8051 MICROCONTROLLER OVERVIEW

The 8051 family of microcontrollers are 8-bit controllers capable of addressing 64K of program memory and a separate 64K of data memory. The 8031 has 128 bytes of internal Random Access Memory (RAM). The 8031 has 2 timer/counters, a serial port, 4 general purpose parallel input/output ports, and interrupt control logic with 5 sources of interrupts. Besides internal RAM, the 8031 has various Special Function Registers (SFR) which are the control and data registers for on-chip facilities. The SFRs also include the accumulator, the B register, and the Program Status Word (PSW) which contains the CPU flags. Programming the various internal hardware facilities of the 8031 is achieved by placing the appropriate control words into the corresponding SFRs.

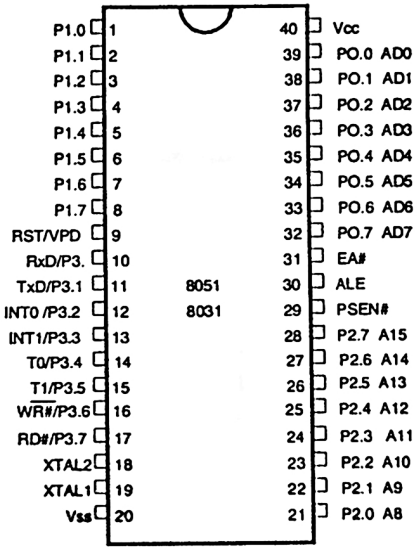


Figure 9.1 Pinout of the 8051 Microcontroller

9.1 External Data and Program Memory

The 8031 can address 64K of external data memory and 64K of external program memory. These may be separate blocks of memory, so that up to 128K of memory can be attached to the microcontroller. The 8031 has two separate read signals, RD# and PSEN#. The first is activated when a byte is to be read from external data memory, the other, from external program memory. Both of these signals are so-called active low signals. That is, they are cleared to logic level 0 when activated. All external code is fetched from external program memory. In addition, bytes from external program memory may be read by special read instructions such as the MOVC instruction. There are separate instructions to read from external data memory, such as the MOVX instruction. That is, the instructions determine which block of memory is addressed, and the corresponding control signal, either RD# or PSEN# is activated during the memory read cycle. A single block of memory may be mapped to act as both data and program memory. In order to read from the same block using either the RD# signal or the PSEN# signal, the two signals are combined with a logic AND operation. This way, the output of the AND gate is low when either input is low. Separating program and data increases the reliability of the microcontroller, since there are no instructions to write to the program memory. A ROM device is ideally suited to serve as program memory.

If the program fits into the on-chip ROM and if the internal RAM is sufficient, the MCS-51 family of microcontrollers require no additional logic to implement a complete controller system. The following discusses the 8031 in detail.

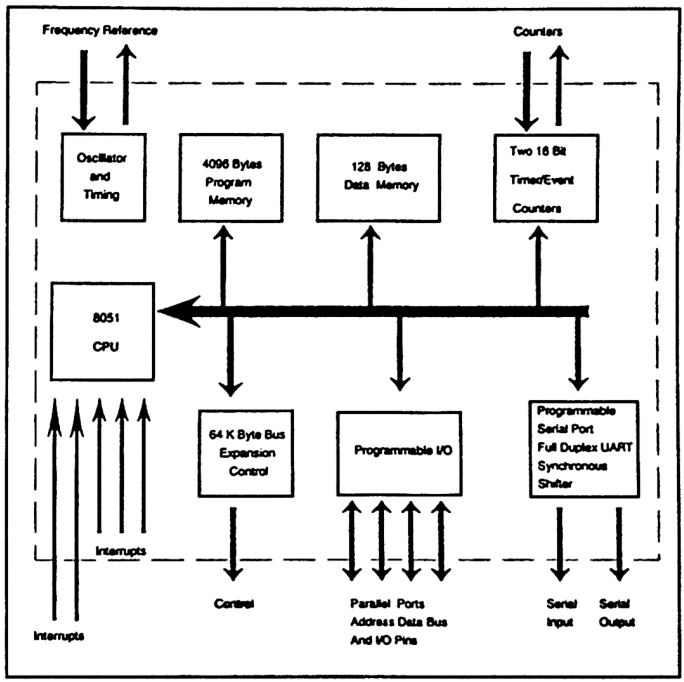


Figure 9.1 Block Diagram of the 8031 Microcontroller

9.2 On-Chip Memory

9.2.1 Internal Random Access Memory

The internal RAM of the 8031 contains the registers and the bit addressable registers as well as general purpose RAM, as illustrated in Figure 9.2. Random access memory is used as a general purpose scratch pad. Memory location addresses range from 0 to 7Fh. Although all internal RAM locations are accessible two blocks of internal RAM can be accessed in different modes.

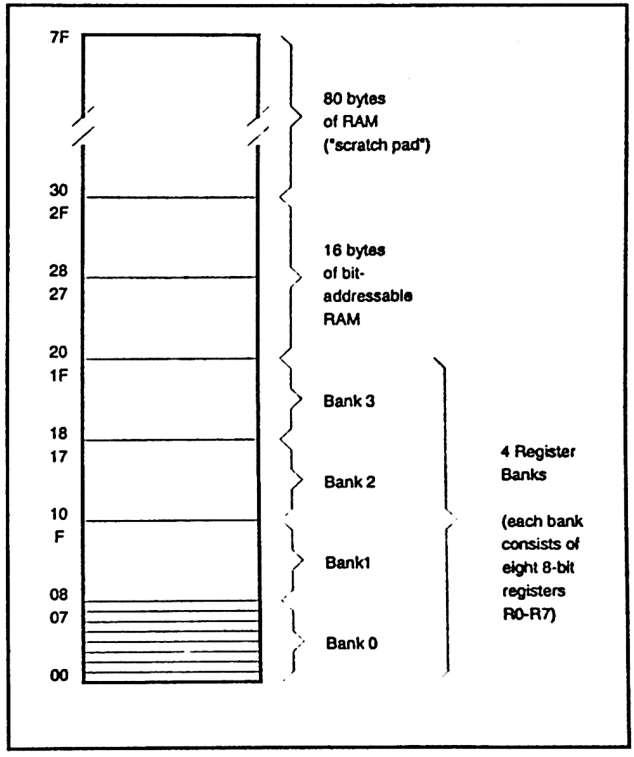


Figure 9.2 The Memory Map of the 8031 Internal RAM

9.2.2 Registers

The block of internal RAM [0..1FH] is used for the registers. Specifically, there are four banks, with 8 registers in each bank. The registers of a bank are denoted referred to as R0 to R7. Register bank 0, 1, 2, and 3 contain internal RAM locations [0..7], [8..FH], [10H..17H], and [18H..1FH], respectively. In order to address a register, first the register bank it belongs to must be selected as the active register bank. Such selection is done by flags in the PSW. It is often a good idea to dedicate register banks to key subroutines or interrupt service routines.

9.2.3 Bit-Mapped Memory

A block of memory, in the range [20H..2FH], is bit-addressable. Notice that there are 128 bits in this block. Each bit has its bit address from 0 to 7FH. Bit 0 of byte 20H has bit address 0 and bit 7 of byte 2FH has bit address 7FH.

9.2.4 Special Function Registers

SFRs contain data and control registers. Each on-chip facility such as the timers, the serial port, and the interrupt system, has one or more dedicated SFRs. The serial port, for example, is controlled by the SFR SCON, while its associated data is read from or written to the SFR SBUF. Individual bits of SCON set the different modes of the serial port. SCON is referred to as a control register, while SBUF is called a data register.

SFRs have byte addresses in the range [80h..FFh]. The 8031 has 128 bytes of internal RAM which occupy the addresses [0..7Fh]. The SFRs are mapped into the next block of 128 bytes, in the range of [80h..FFh]. Table A is a list of the 8031 SFRs.

Register	Mnemonic	Internal Address
Port 0 Latch	P0	80H
Stack Pointer	SP	81H
Data Pointer (as a word)	DPTR	82H-83H
Data Pointer Low Byte	DPL	82H
Data Pointer High Byte	DPH	83H
Power Control	PCON	87H
Timer/Counter Control	TCON	88H
Timer/Counter Mode Control	TMOD	89H
Timer/Counter 0 Low Byte	TL0	8AH
Timer/Counter 1 Low Byte	TL1	8BH
Timer/Counter 0 High Byte	TH0	8CH
Timer/Counter 1 High Byte	TH1	8DH
Port 1 Latch	P1	90H
Serial Port Control	SCON	98H
Serial Data Port	SBUF	99H
Port 2 Latch	P2	A0H
Interrupt Enable	IE	A8H
Port 3 Latch	P3	B0H
Interrupt Priority Control	IP	B8H
Program Status Word	PSW	D0H
Accumulator	ACC or A	E0H
B Register	B	F0H

Table A. 8031 Special Function Registers

Of the SFRs, SP, DPTR, DPL, DPH, PCON, SBUF, TMOD, TL0, TL1, TH0, and TH1 are only byte-addressable. That is, these SFRs can only be read, written to, operated on, and compared as full bytes. The remaining SFRs, ACC, B, PSW, P0, P1, P2, P3, IE, IP, SCON, and TCON are also bit-addressable, meaning that their individual bits can also be read, written to, operated on, and compared.

9.2.5 Read Only Memory

Some members of the 8031 family have on-chip ROM. The 8051 and 8052 have 4K (kilobytes) and 8K of factory masked ROM. The 8751 and 8752 are EPROM versions of the 8051 and 8052. Internal ROM occupies the lowest block of program memory. The EA# (External Enable) pin of the microcontroller is tied to +5 volts to allow the program to be fetched directly from internal ROM. Of course, any program which resides above the 4K or 8K block is fetched from external program memory. If EA# is connected to ground (0 volts), then all of the program is fetched from external memory. The pound sign (#), when used as a suffix signifies that the signal is active when low. For example, external memory is enabled when the EA# line is held at logic low (close to ground voltage).

9.2.6 Program Counter

The Program Counter is internally used to point to the next instruction byte to be fetched. It is not directly accessible. However, it is modified by the branching instructions such as jump (JMP) and call (CALL). It can also be used as a base address for indexed addressing when reading from the program memory.

10 8051 ON-CHIP FACILITIES

This section gives a general overview of the generic 8051 features which are standard. There are many enhanced varieties of the 8051 microcontroller but all start with these basic features.

10.1 Parallel Input/Output Ports

The 8031 has 4 parallel input/output ports. When a port is to be used as an output port, the data is put in the corresponding SFR. The value of an output port is changed when a new value is latched. When a port is to be used as an input port, the value FFH must first be written to the port. Then any input which pulls the pin voltage low will be recognized as a zero. The port can then be read from the corresponding SFR. The latched output drives the port pin to logic level 1 if there is no external circuit sinking the current on the pin.

Ports 0, 2, and 3 have alternative functions. That is, the individual pins of these ports may be used as general digital input/output lines, or alternatively, be used for their secondary functions. The secondary function of Ports 0 and 2 is to interface with external memory. When external program or data memory is accessed, Port 2 outputs the high byte of the 16-bit address. Port 0 first outputs the low byte of the 16-bit address, and then sends or receives the data byte.

Alternative functions of Port 3 pins include interrupt and timer inputs, serial port input and output, and control signals for interfacing with external memory. Table B summarizes the alternative functions of Port 3.

Bit	Alternate Function	Mnemonic/Designation
0	Serial Input Port	RXD
1	Serial Output Port	TXD
2	External Interrupt 0	INT0#
3	External Interrupt 1	INT1#
4	Timer/Counter 0 External Input	T0
5	Timer/Counter 1 External Input	T1
6	External Memory Write Strobe	WR#
7	External Memory Read Strobe	RD#

Table B. Alternative Functions Of Port 3.

In order to implement the alternative function, the corresponding SFR bit must be set (made equal to 1).

10.2 System Clock Generator

The 8031 contains hardware to generate a system clock (oscillator) using an external crystal and two external capacitors. The oscillator frequency is the same as the crystal frequency. The oscillator frequency divided by 12 (prescaled by 12) may be used as an input by the on-chip timers. Using a 12MHz crystal, the timer inputs then are at 1MHz.

The 8031 uses 12 oscillator cycles per machine cycle. The 8031 has 255 operation codes grouped as 111 instructions. Since many of the frequently used instructions take only one machine cycle, the 8031 is considered to be capable of roughly 1 Million Instructions Per Second (MIPS). Many 8031-based systems use a crystal frequency of 11.0592MHz. This choice is due to the fact that many high Baud rates can be generated by this clock frequency. With a 12MHz clock, a maximum of 4800 Baud is the practical limit. Baud rates up to 19200 can be generated using an 11.0592MHz crystal. The R-51JX uses the 11.0592 MHz crystal when populated with the 8031 processor.

10.3 Serial Port

The serial port is controlled by the SFR SCON. Data to and from the serial port is channeled through the SFR SBUF. Once the serial port is configured, simply writing the byte to SBUF initiates the serial transmission. Similarly, a received byte is read from SBUF. Note that although SBUF appears as a single SFR, its hardware implementation contains two separate buffers, one for transmission and one for receiving the bytes.

A serial transmission and a serial receive may take place simultaneously, which in serial communications is referred to as a full duplex operation.

10.4 Timer/Counters

The 8031 has 2 internal timer/counters. Each timer/counter has two SFRs dedicated to it. Associated with timer/counter 0 are TH0 and TL0, and with timer/counter 1, TH1 and TL1. These registers make the timer/counter 16 bits wide. In operation, at each input pulse, the count stored by one or two SFRs is incremented. A timer uses the system clock as the source of its input pulses. Thus timer increments are periodical while a counter uses external pulses to increment its count. The external pulses are received through two of the bits of Port 3 which, as alternative functions, are assigned to the two timer/counters. The counter/timers also generate interrupts, provided that the associated interrupts are not masked by the Interrupt Enable (IE) SFR. When the count overflows (and restarts from 0) an interrupt is generated.

10.5 Interrupt Control

The 8031 has are 5 sources of interrupts: TF0, TF1, INT0#, INT1#, and the serial port events. TF0 and TF1 of the TCON register constitute the two timer/counter interrupts. TF0 and TF1 are generated when the associated counter overflows. INT0# and INT1# constitute the two external interrupts. These interrupts are caused by external signals received through bits 2 and 3 of Port 3 as alternative functions. The interrupt control hardware may be programmed to respond to either the falling edge of the external signals or to the low level of the external signals. The final source of interrupts are from the serial port. The interrupt flags RI and TI of SCON SFR are combined by an OR gate so that either flag may generate an interrupt. It is the responsibility of software to check both the TI and RI flag to determine which caused the interrupt.

There are two SFRs associated with interrupt control. The Interrupt Enable Register (IE) is used to mask individual interrupts. Interrupt Priority Register (IP) assigns either a high priority or a low priority to each of the 5 interrupt sources. When two interrupt sources are programmed to the same priority level, priorities are determined by the following order. IE0 has the highest priority, followed by, TF0, IE1, TF1, and finally by RI-or-TI.

Provided that it is not masked (disabled), the processor acknowledges an interrupt and branches to pre-specified locations in program memory by a LCALL instruction. Notice that the PSW is not pushed onto the stack. Below are the fixed memory locations where interrupt service routines must start.

Interrupt	Service Routine Address	Default Priority
IE0	3H	1 (Highest)
TF0	BH	2
IE1	13H	3
TF1	1BH	4
RI-or-TI	23H	5 (Lowest)

11 INTERFACING THE MICROCONTROLLER SYSTEM

11.1 Memory Mapped I / O

Header J8 provides all of the necessary signals for interfacing additional circuitry to the R-51JX. Figure 11.1 shows how an 8255 Programmable Peripheral Interface is connected to the microcontroller signals. The 8255 ports A, B, and C are decoded at external data addresses FE00h, FE01h, and FE02h, respectively. The 8255 control register occupies address FE03h. If only one device is interfaced to the controller, no other address lines need be decoded. With this arrangement, the ports and control register also respond to the image addresses. For example, FE04h, FE08h, FE0Ch, ... FEFCh all address Port A.

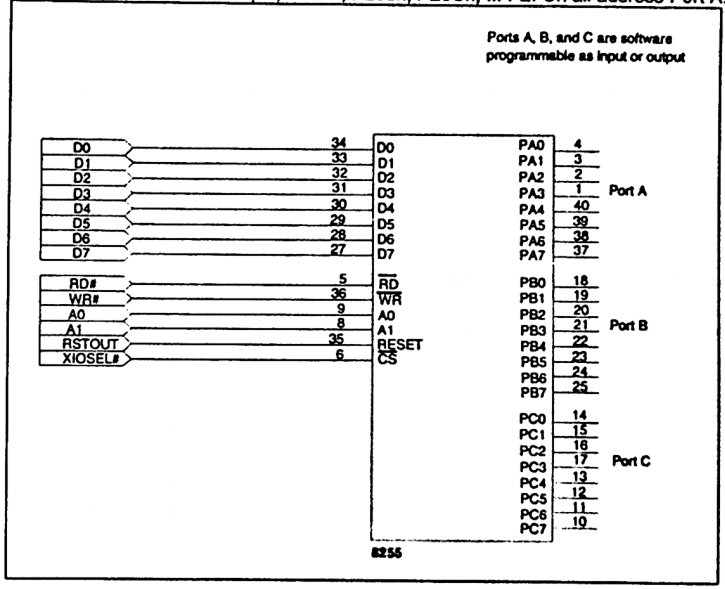


Figure 11.1 Adding Memory-Mapped I/O ports to the R-31JP

The 8255 ports A, B, and C are decoded at external data addresses FE00h, FE01h, and FE02h, respectively.

If more than one device (Chip Select) is to be interfaced with the processor, the address lines must further be decoded. Figure 11.2 shows how a 74138 decoder can be used to decode the address lines A3 to A5 to obtain 8 separate Chip Select signals.

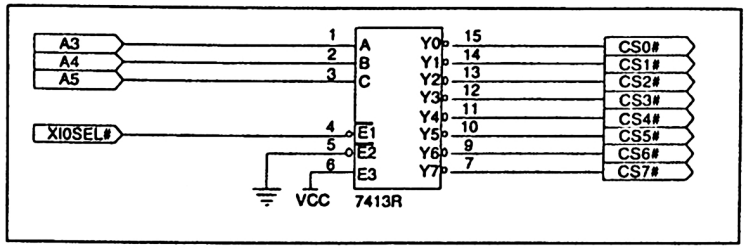


Figure 11.2 Decoding the Memory-Mapped I/O Space for Eight Devices.

The 74138 activates one of the eight CS# (Chip Select) signals (CS0# to CS7#) depending on the address line A3-A5. The 74138 decoder itself is activated by the XIOSEL# signal from the processor. Since the address lines A0, A1, and A2 are not decoded, each CS# line actually selects a block of 8 addresses. The individual devices that are activated by the CS# lines can use A0-A2 to further decode the address space. For example, eight 8255s may be connected to the processor, as shown in Figure 11.2 each with its own CS# line. Each 8255 would then make use of A0 and A1 to further decode the addresses of the individual ports.

11.2 Simple Input / Output Circuits

The following figure give simple input and output circuits that may be built on a breadboard, and interfaced with the microcontroller.

Analog inputs are typically obtained by a potentiometer. In some applications it is desirable to have diode clamps to prevent the analog input voltage to exceed VCC or GND.

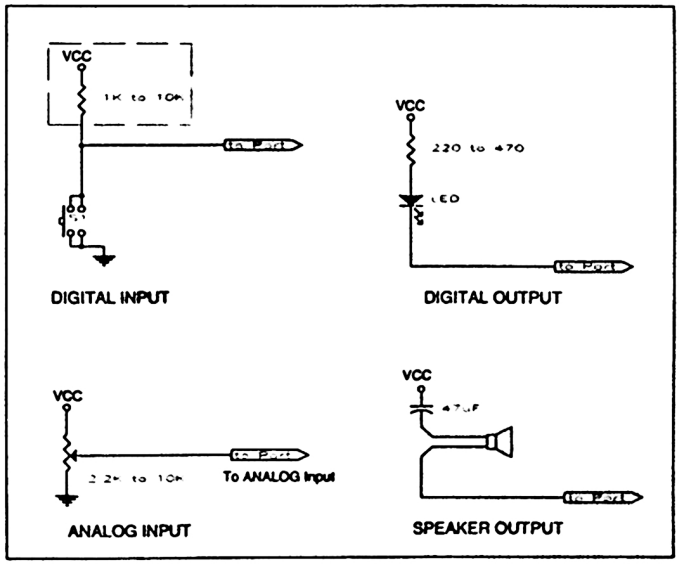


Figure 11.3 Simple User Input/Output Devices.

11.3 Driving High Current Loads

Driving high-current loads requires external power sources and switching devices, such as the darlington switching transistor shown in the below figure. The diode protects the transistor from reverse transients when driving inductive loads such as DC motors or solenoids. The capacitors also smooth the voltage across the load. The above circuit turns the load on when the microcontroller port is at logic level 1. Since the microcontroller ports are at level 1 upon reset, the load will be on immediately after reset, for example, after a power failure that resets the microcontroller. If the external power supply is not connected to the same source, the load will remain energized even if the microcontroller is not powered. The microcontroller has the capability of turning the power off by making the output port 0. The below circuit was used to control a battery-powered cooling fan. The design assured that the cooling fan would remain on in case the microcontroller lost power.

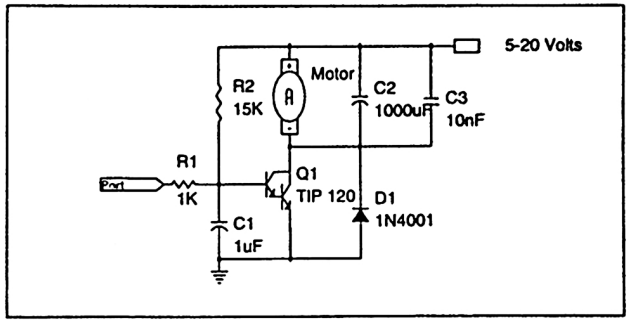


Figure11.4 DC motor control.

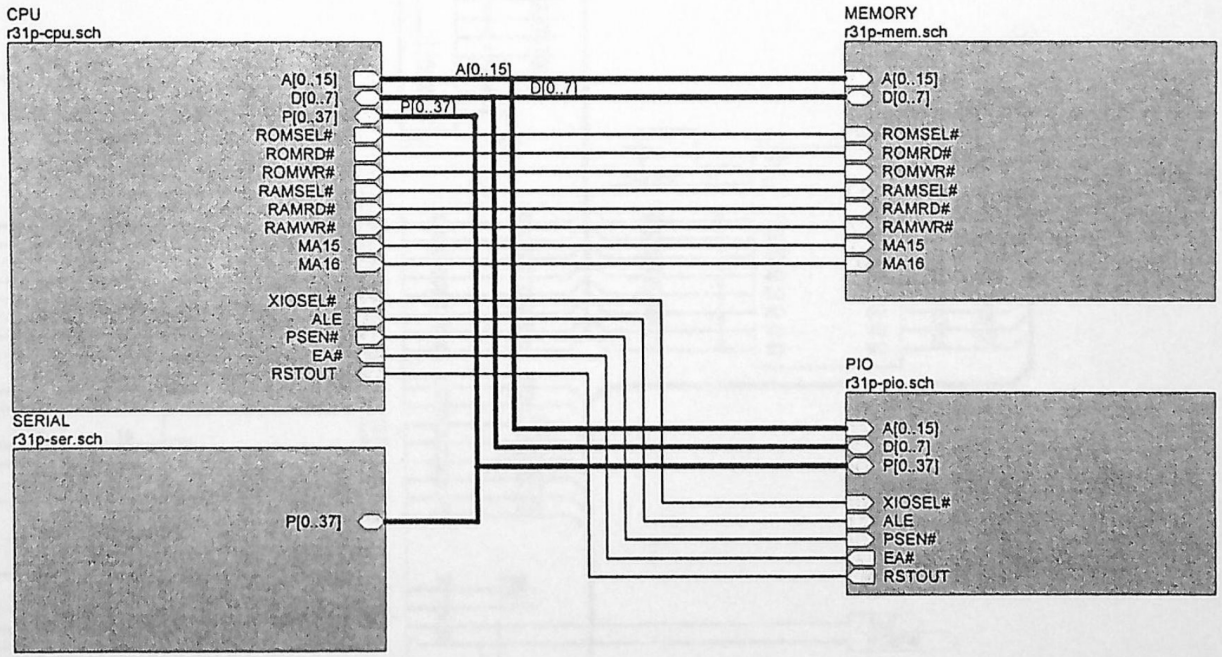
The Port may be pulsed with a varying duty cycle to change the speed.

12 PARTS LISTS

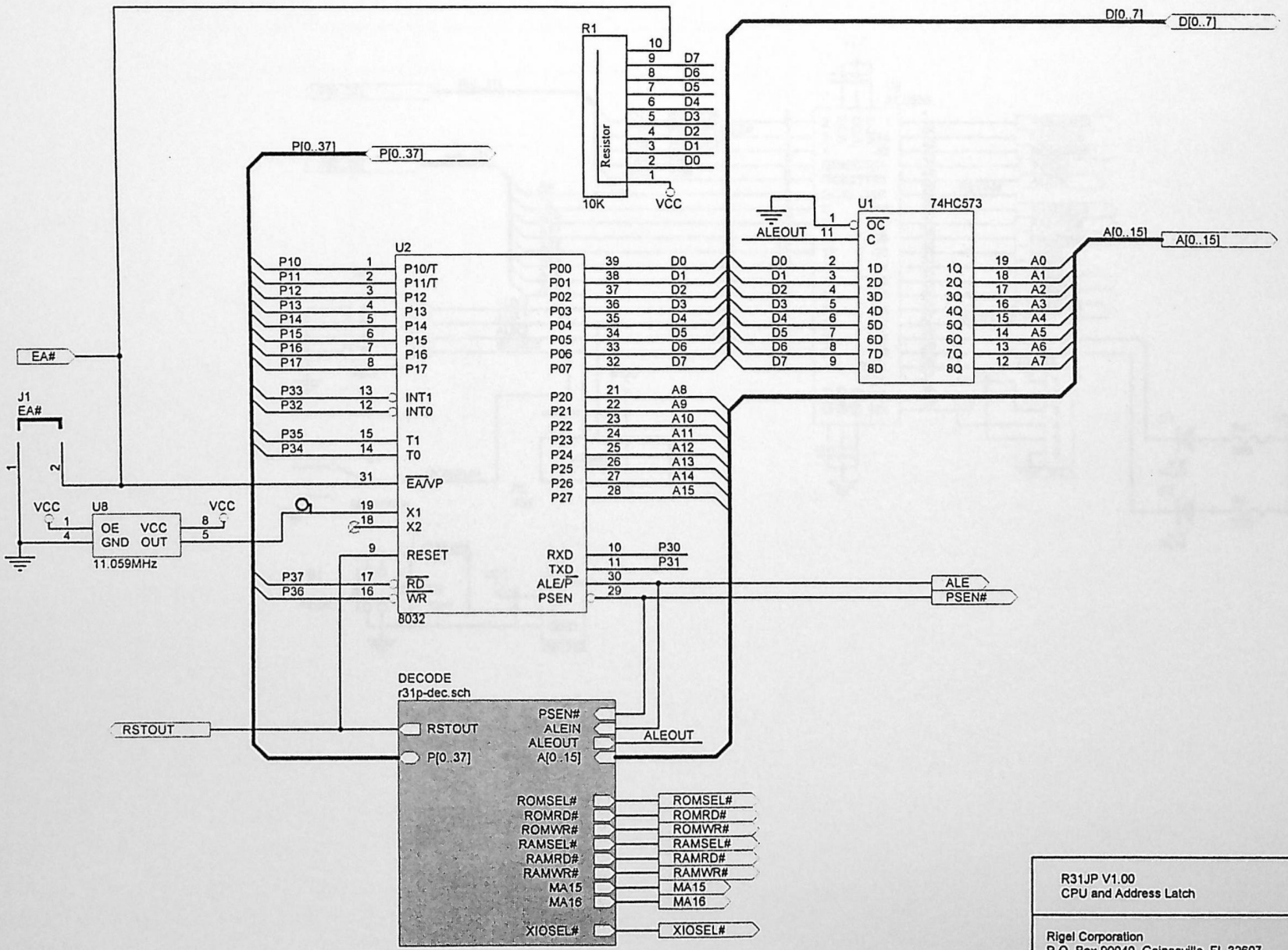
R-31JP Bill of Materials

The bill of materials given below lists all components by their reference as they appear on the board top overlay. Note that this list does not include the sockets.
Revised: June 23, 2000
List Of Materials

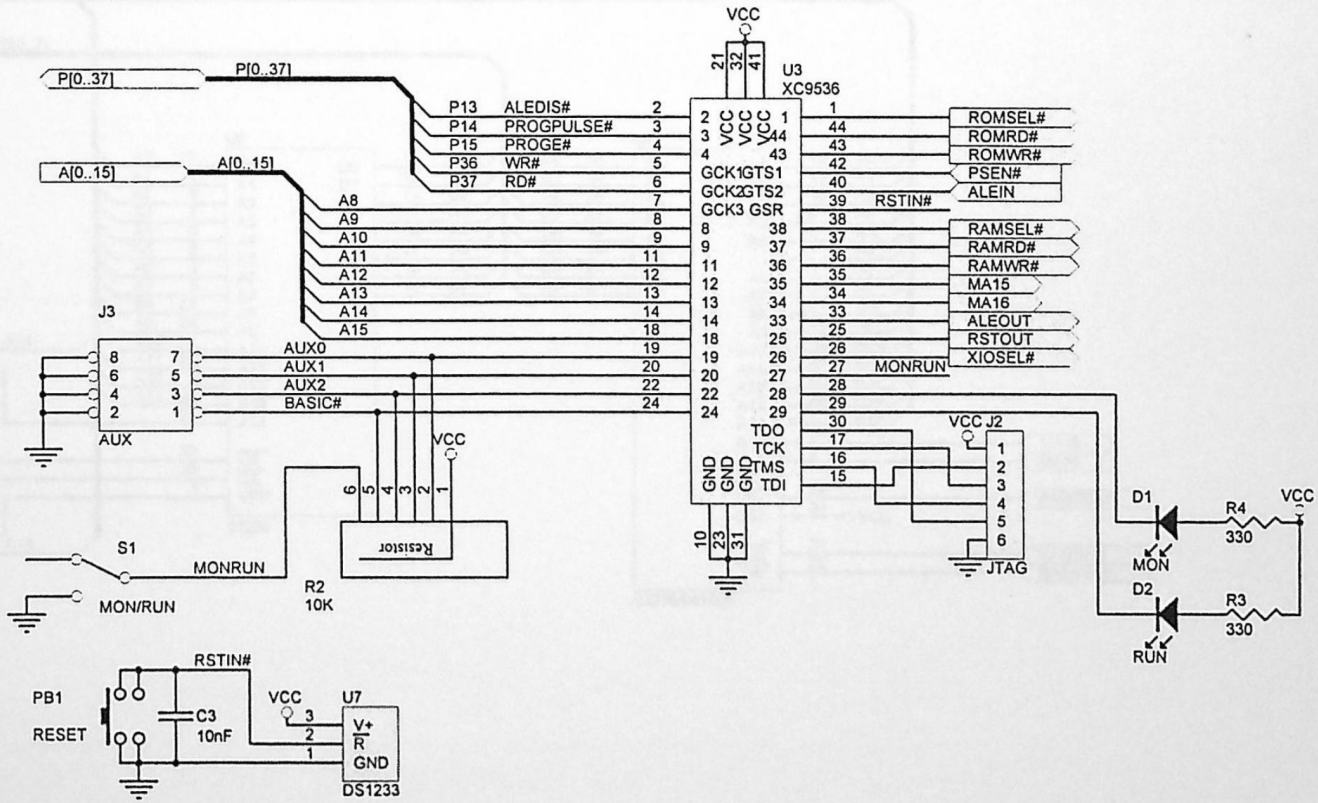
Item	Quantity	Part	Reference	Description
1	12	10nF	C1,2, 3,8-16	Axial capacitor
2	4	1.0uF CAP	C4-7	Electrolytic capacitor
3	1	470uF16V (220uF, 330uF)	C17	Electrolytic capacitor
4	2	47uF16V (22uF)	C19, C18	Electrolytic capacitor
5	1	10K 10GANG	R1	Gang Resistor
6	1	10K 6GANG	R2	Gang Resistor
7	2	330 OHM	R3, R4	1/2 Watt Resistor
8	1	RED LED	MON, D1	LED (red)
9	1	GREEN LED	RUN, D2	LED (green)
10	1	1N4001	D3	Diode
11	1	MON/RUN	S1	Slide Switch
12	1	RESET	PB1	Push-button
13	1	RS232	P1	DB9
14	1	EA#	J1	1X2 Header
15	1		J3	2X4 Header
16	1	ROM/RAM		2X3 Header
17	8	PIO	J8	Terminal blocks (14)
18	1	PWR	JP6	Terminal block (2)
19	1	SIGNALS	J7	40-pin header
20	2	BUS	J9, J10	23-pin Header
21	1	J-TAG	J2	1X6 Header
22	1	DS1233	U7	Reset chip
23	1	74HCT573	U1	Octal latch
24	1	89C52	U2	Microcontroller
25	1	XC9536-15PC44C	U3	CPLD
26	1	62C256	U4	Static RAM
26	1	27C256	U5	32K EPROM
27	1	MAX232	U6	Serial
28	1	11.0592 clock	Y1	Clock



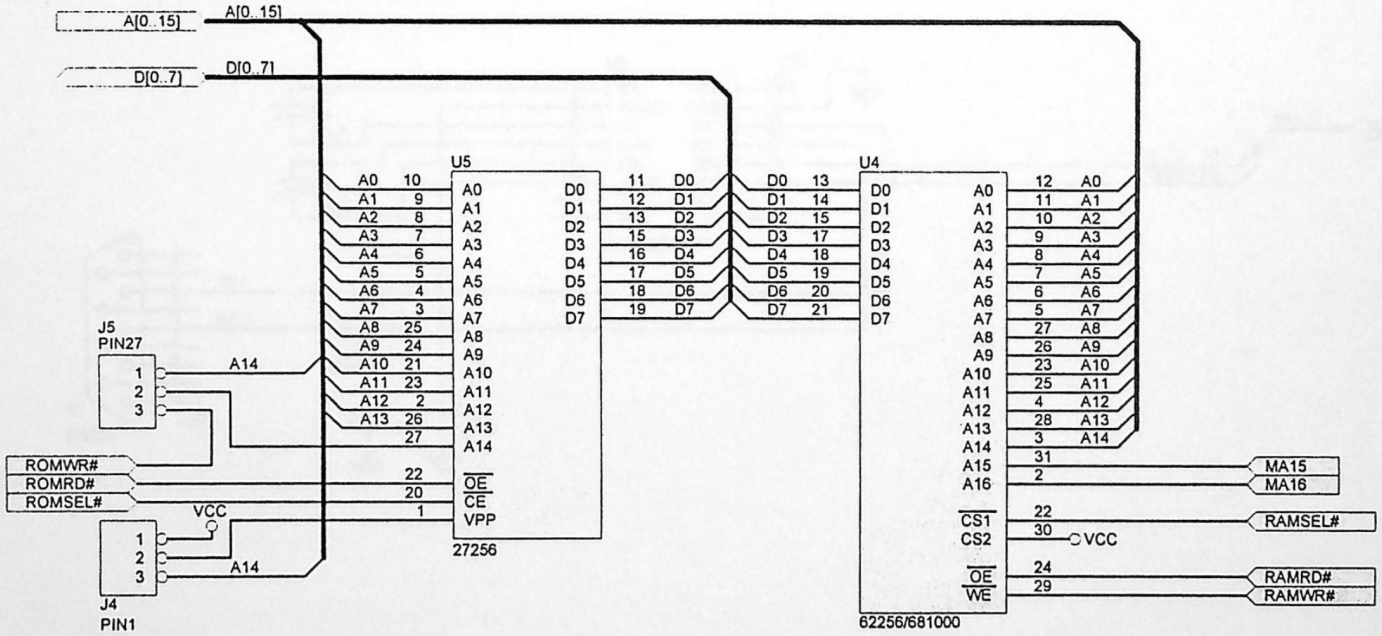
R31JP V1.00 System Block Diagram	August 20, 1994 Revision 1.00
Rigel Corporation P.O. Box 90040, Gainesville, FL 32607 Phone: (904) 373-4629	Sheet 1 of 6



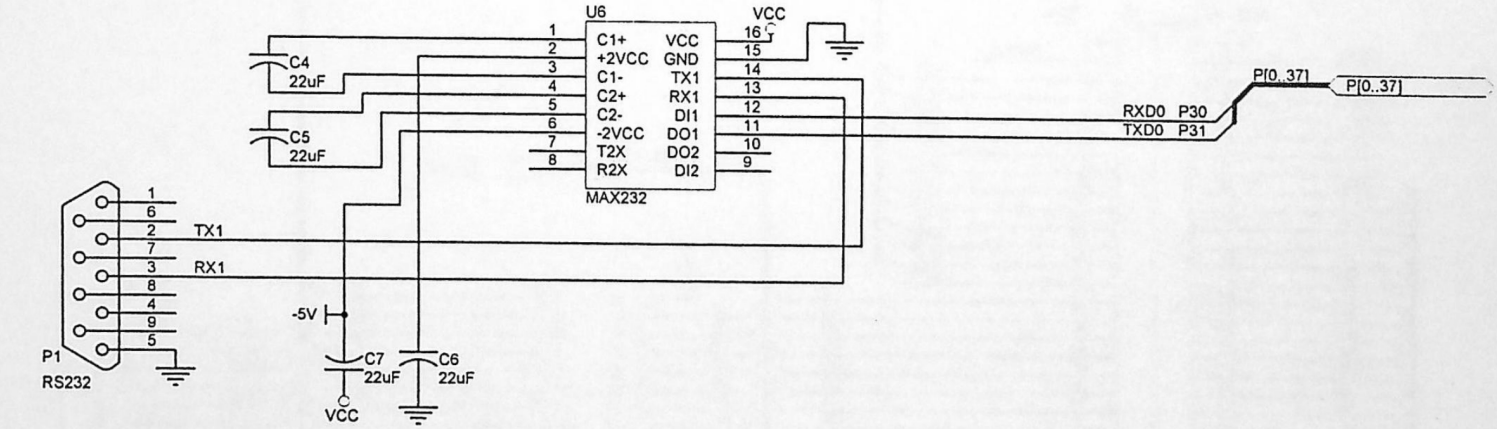
R31JP V1.00 CPU and Address Latch	August 20, 1994 Revision 1.00
Rigel Corporation P.O. Box 90040, Gainesville, FL 32607 Phone: (904) 373-4629	Sheet 2 of 6

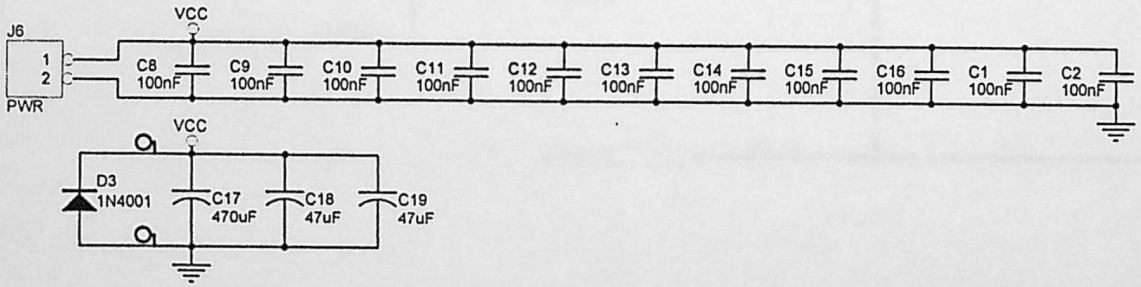


R31JP V1.00 Decode Logic	August 20, 1994 Revision 1.00
Rigel Corporation P.O. Box 90040, Gainesville, FL 32607 Phone: (904) 373-4629	Sheet 3 of 6

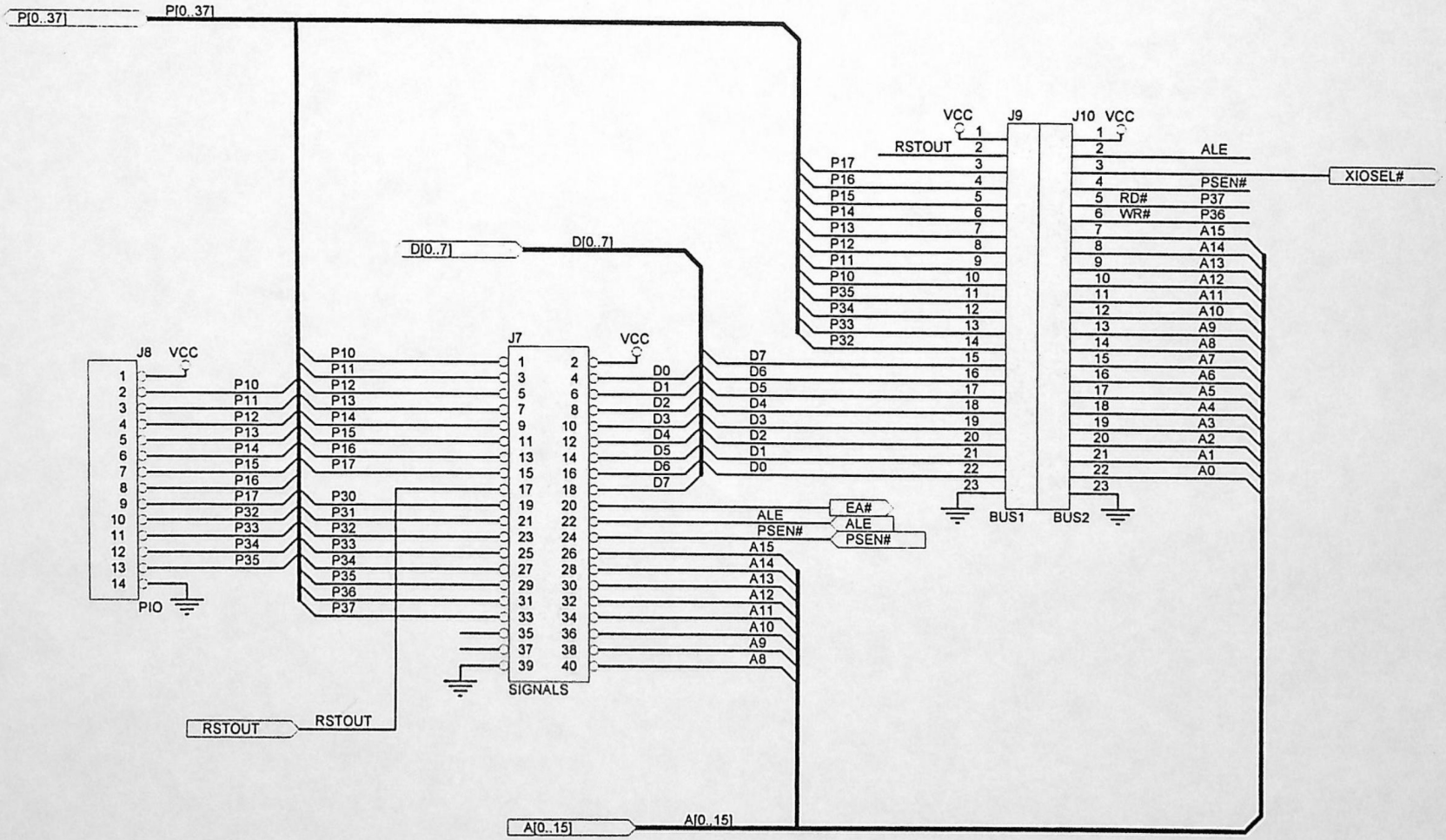


R31JP V1.00 Memory	August 20, 1994 Revision 1.00
Rigel Corporation P.O. Box 90040, Gainesville, FL 32607 Phone: (904) 373-4629	Sheet 4 of 8





R31JP V1.00 Serial Port and Power	August 20, 1994 Revision 1.00
Rigel Corporation P.O. Box 90040, Gainesville, FL 32607 Phone: (904) 373-4629	Sheet 5 of 6



R31JP V1.00 Parallel Input/Output	August 20, 1994 Revision 1.00
Rigel Corporation P.O. Box 90040, Gainesville, FL 32607 Phone: (904) 373-4629	Sheet 6 of 6