

***RTE-V831-PC***

**USER S MANUAL (Rev. 2.00)**

**REVISION HISTORY**

Date	REV.	Chapter	Explanation of revision
Apr. 14 1997	1.02		First edition
Apr. 28 1998	2.00		A description of the use of the PARTNER monitor has been added.
		5.19	Corrections have been made to the following descriptions: JCPU-A, pin 26 (+3..3 V → +3.3 V), pin 98 (+5 V → GND)
		8.3	Logical address in the table (BC00-0000H → 4500-2000H)
		8.9	Request levels listed in the interrupt source table

**CONTENTS**

<b>1. INTRODUCTION</b> .....	<b>1</b>
1.1. NUMERIC NOTATION .....	1
<b>2. FUNCTIONS</b> .....	<b>1</b>
<b>3. MAJOR FEATURES</b> .....	<b>2</b>
<b>4. BASIC SPECIFICATIONS</b> .....	<b>2</b>
<b>5. BOARD CONFIGURATION</b> .....	<b>3</b>
5.1. RESET SWITCH (RESET).....	3
5.2. POWER JACK (JPOWER).....	3
5.3. SWITCH 1 (SW1) .....	3
5.4. SWITCH 2 (SW2) .....	4
5.5. LED .....	4
5.6. TEST PINS FOR ROM EMULATOR (TP) .....	4
5.7. CLOCK SOCKET (OSC1) .....	5
5.8. 72-PIN-SIMM SOCKETS (SIMM#1, SIMM#2).....	5
5.9. ROM SOCKETS .....	5
5.10. ROM SIZE SWITCHING JUMPER (JP1).....	5
5.11. ROM DIVISION SWITCHING JUMPER (JP2).....	5
5.12. JP3, JP4, JP5 .....	6
5.13. JUMPER FOR SWITCHING BETWEEN BASIC CLOCKS FOR USER TIMERS (JP6).....	6
5.14. SERIAL CONNECTOR (JSIO1, JSIO2).....	6
5.15. PARALLEL CONNECTOR (JPRT).....	7
5.16. AUDIO MINI-JACKS (JMIC-R, JMIC-L, JOUT) .....	7
5.17. DEBUGGING CONNECTOR (JDBG).....	8
5.18. EXTENSION BUS CONNECTOR (JEXT) .....	8
5.19. CPU CONNECTOR (JCPU-A, JCPU-B).....	9
<b>6. CONNECTION WITH THE HOST PC</b> .....	<b>11</b>
6.1. INSTALLATION ON THE ISA BUS .....	11
6.2. STANDALONE USE OF THE BOARD .....	11
<b>7. HARDWARE REFERENCES</b> .....	<b>12</b>
7.1. MEMORY AND I/O MAP.....	12
7.2. DETAILS OF MAPPING.....	13
<b>8. SYSTEM-I/O</b> .....	<b>14</b>
8.1. SYSTEM-I/O LIST .....	14
8.2. DIPSW1 READ PORT (4500-1000H [READ ONLY]) .....	14
8.3. 7-SEGMENT LED DISPLAY DATA OUTPUT PORT (4500-2000H [WRITE ONLY]) .....	15
8.4. COMMAND REGISTER #0 PORT (4500-3000H [READ/WRITE]).....	15
8.5. COMMAND REGISTER #1 PORT (4500-4000H [READ/WRITE]).....	15
8.6. COMMAND REGISTER #2 PORT (4500-5000H [READ/WRITE]).....	16
8.7. UART/PRINTER (TL16C552A) (4500-8000H TO 4500-A00CH) .....	16
8.8. TIC ( $\mu$ PD71054) (4500-B000H TO 4500-B00CH).....	17
8.9. INTERRUPT CONTROLLER (PIC) (4500-D000H TO 4500-D018H).....	18
8.10. AUDIO CONTROLLER (AUDCNT) (4580-0000H TO 4580-0010H, 4580-2000H).....	19
8.11. $\mu$ PD63310 REGISTER: AUDIO COD. (4580-1000H TO 4580-100FH).....	21
<b>9. INTERRUPTS AND DMA</b> .....	<b>22</b>
9.1. INTERRUPT .....	22
9.2. USING NMI .....	22
9.3. DMA CHANNEL .....	23

<b>10. EXT BUS SPECIFICATIONS.....</b>	<b>24</b>
10.1. PIN ARRANGEMENT AND SIGNALS.....	24
10.2. TIMING.....	25
10.3. NOTES ON USING THE EXT BUS.....	26
<b>11. SOFTWARE.....</b>	<b>27</b>
11.1. INITIALIZATION.....	27
11.2. LIBRARIES.....	27
11.3. USING TIMERS.....	28
11.4. FLASH ROM PROGRAMMING.....	29
11.5. AUDIO I/O.....	31
<b>12. DEVELOPMENT OF APPLICATIONS USING MASKABLE INTERRUPTS.....</b>	<b>33</b>
12.1. INTERRUPT VECTOR.....	33
12.2. INTERNAL INSTRUCTION RAM.....	34
12.3. GENERAL RESTRICTIONS/NOTES.....	34
12.4. RESTRICTIONS ON BREAKS IN THE INTERRUPT HANDLING ROUTINE.....	35
<b>13. APPENDIX A MULTI MONITOR.....</b>	<b>36</b>
13.1. BOARD SETTING.....	36
13.1.1. RTE for Win 32 installation.....	36
13.2. SWITCH SETTING.....	36
13.2.1. SW1 Setting.....	36
13.2.2. SW2 Setting.....	37
13.3. MULTI MONITOR.....	38
13.3.1. Monitor Work RAM.....	38
13.3.2. Interrupt.....	38
13.3.3. Interrupt for Forced Break.....	38
13.3.4. _INIT_SP Setting.....	38
13.3.5. Remote Connection.....	38
13.3.6. Monitor Execution Area.....	38
13.4. RTE COMMANDS.....	39
13.4.1. HELP(?).....	39
13.4.2. INIT.....	39
13.4.3. VER.....	39
13.4.4. INB, INH, INW.....	39
13.4.5. OUTB, OUTH, OUTW.....	40
13.4.6. DCTR Command.....	40
13.4.7. ITCR Command.....	40
13.4.8. CMCR Command.....	40
13.4.9. SFR Command.....	40
<b>14. APPENDIX B PARTNER MONITOR.....</b>	<b>41</b>
14.1. SWITCH SETTING.....	41
14.1.1. SW1 Setting.....	41
14.1.2. SW2 Setting.....	42
14.2. PARTNER MONITOR.....	43
14.2.1. Monitor Work RAM.....	43
14.2.2. Interrupt.....	43
14.2.3. Interrupt for Forced Break.....	43
14.2.4. SP Setting.....	43
14.2.5. Remote Connection.....	43
14.2.6. Monitor Execution Area.....	43

**1. INTRODUCTION**

The **RTE-V831-PC** is an evaluation board, conforming to the IBM-PC/AT ISA bus specification, that is designed to evaluate the NEC V831 RISC processor. The board features a V831 capable of operating at a maximum speed of 100 MHz, memory, serial and parallel interfaces, and inputs/outputs such as audio inputs/outputs. For interfacing with memory, the memory controller built into the V831 is used.

These functions enable the RTE-V831-PC to be used for a wide variety of applications including processor performance evaluation and application program development at the initial stage, and to also be used as an engine for demonstration and simulation.

The GHS Multi or NEC PARTNER source-level debugger can be used as a development software tool with the RTE-V831-PC. The type of monitor to be stored in ROM depends on the debugger type.

In ROM, the monitor specified at the time of purchase is stored. Even when neither of the debuggers is purchased together with the RTE-V831-PC, they can be purchased at anytime subsequently.

**1.1. NUMERIC NOTATION**

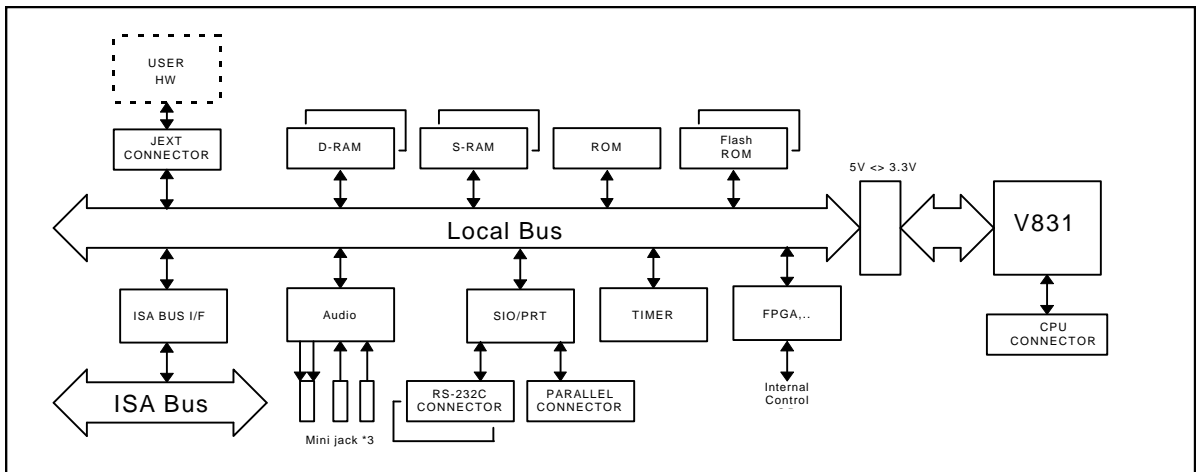
This manual represents numbers according to the notation described in the following table. Hexadecimal and binary numbers are hyphenated at every four digits, if they are difficult to read because of many digits being in each number.

Number	Notation rule	Example
Decimal number	Only numerals are indicated.	"10" represents number 10 in decimal.
Hexa-decimal number	A number is suffixed with letter H.	"10H" represents number 16 in decimal.
Binary number	A number is suffixed with letter B.	"10B" represents number 2 in decimal.

Number Notation Rules

**2. FUNCTIONS**

The overview of each function block of the RTE-V831-PC is shown below.



RTE-V831-PC Block Diagram

MULTI is a trademark of Green Hills Software, Inc. in the US.

### 3. MAJOR FEATURES

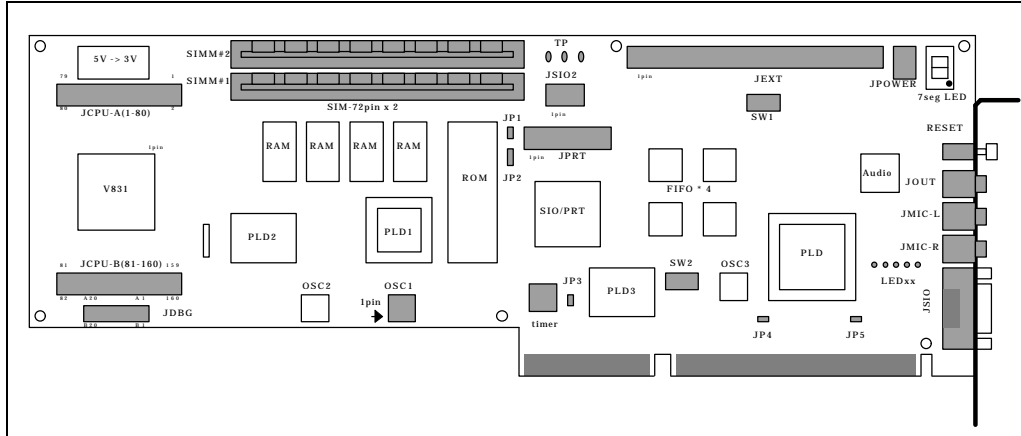
- Two types of monitor ROM are provided: one is used for the Green Hills Multi and the other for the NEC PARTNER.
- Real-time execution and evaluation at a high-level language level using Multi or PARTNER.
- A ROM emulator can be connected.
- A high-speed 512K bytes SRAM is installed as standard. Up to 16M bytes of DRAM can be installed using SIMMs.
- Two serial interfaces and one printer interface are provided.
- Two timer channels are provided. (One channel is used for the Multi monitor.)
- Two audio input channels and two audio output channels are provided.

### 4. BASIC SPECIFICATIONS

Processor	V831
CPU clock	100 MHz (max.)
Bus clock	33 MHz (max.)
Power consumption	+5 V (2.0 A)
Memory	
EPROM	128 KB 64 K × 16 bits (40-pin DIP) × 1 (512K bytes max.)
Flash ROM	8 MB 2 M × 8 bits × 4 (MBM29F016-120)
SRAM	512 KB 128 K × 8 bits × 4
DRAM	8 MB EDO-SIMM-72pin (16M bytes max. with two SIMMs)
I/O	
Serial (2 ch)	Equivalent to NS16550, 10-pin header, DB9 connector
Printer	PS2-compatible, 26-pin header
Audio input/output (2 ch)	■PD66310, Mini-jack (MIC × 2, LINEOUT × 1)
Timer	Equivalent to i8254, 500-ns resolution
I/O port	LED (7-segment) display/switch input
Others	
CPU connector	Connector with all function pins of the V831 connected
Standard external extension connector	RTE-PC standard 16-bit interface (1M byte, 16-bit bus)
Reset switch	Push type

**5. BOARD CONFIGURATION**

The physical layout of the major components on the RTE-V831-PC board is shown below. This chapter explains each component.



RTE-V831-PC Components Layout

**5.1. RESET SWITCH (RESET)**

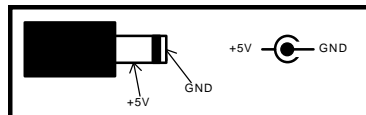
RESET is a reset switch for the entire board. Pressing this switch causes all the circuits including the CPU to be reset.

**5.2. POWER JACK (JPOWER)**

When this board is to be used as a standalone, that is, without being inserted in an ISA bus slot, the board should be supplied with power from an external power supply by connecting it to the JPOWER connector.

The external power should be one rated as listed below.

- Voltage: 5 V
- Current: Maximum of 2 A (excluding the current supplied to the JEXT connector)
- Mating connector: Type A (5.5 mm in diameter)
- Polarity:



**[Caution]** When attaching an external power supply to the board, be careful about its connector polarity. When inserting the board into the ISA bus slot, do not attach the JPOWER connector to an external power supply. It may result in a malfunction.

**5.3. SWITCH 1 (SW1)**

SW1 is a general-purpose input port switch. When the monitor is used, all SW1 switches except SW1-7 are already set. See Sections 13.2.1. and 14.1.1, for the switch settings for the Multi and PARTNER monitors, respectively. When the port is read, a switch being set to OFF represents 1, while its being set to ON represents 0. For details, see Section 8.2.

#### 5.4. SWITCH 2 (SW2)

SW2 is a switch for selecting the I/O address of the ISA bus. Switches 1 to 8 correspond to ISA addresses A4 to A11, respectively (A12 to A15 are fixed at 0). This means that the I/O address that can be selected ranges between 00xH and 03F<sub>x</sub>H. When a switch is open, it corresponds to 1. When it is closed, it corresponds to 0.

SW2	1	2	3	4	5	6	7	8
ISA address	A4	A5	A6	A7	A8	A9	A10	A11

SW2-to-ISA Address Correspondence

#### 5.5. LED

The LEDs are used to indicate statuses, as listed below.

LED	Description
POWER	Lights when power is supplied to the RTE-V831-PC board.
PLY	Lights in green when voice is output. Lights in red if an error occurs during voice output.
REC	Lights in green when voice is recorded. Lights in red if an error occurs during voice recording.
TOVER	Lights when a time-out occurs.
FLBUSY	Lights while the flash ROM is busy (during a write operation, erasure, etc.)

LED Indication

#### 5.6. TEST PINS FOR ROM EMULATOR (TP)

Test pins are used to connect a ROM emulator. They accept control signals from the ROM emulator. The following table lists the signal names and functions related to each test pin.

Signal	Input/output	Function
RESET-	Input	When a low level is supplied to this test pin, the CPU is reset. A reset request signal from the ROM emulator is connected to the test pin. The test pin is pulled up with 1k $\Omega$ .
NMI-	Input	When a low level is supplied to this test pin, an NMI signal is given to the CPU. This signal can be masked by software, so it is necessary to reset the mask. (See Section 8.9.) An NMI request signal from the ROM emulator is connected to the test pin. The test pin is pulled up with 1k $\Omega$ .
GND	---	This test pin is at a ground level. The ground level of the ROM emulator is connected to the test pin.

Test Pin Functions

### 5.7. CLOCK SOCKET (OSC1)

An oscillator for generating the clock signal to be supplied to the CPU is mounted in the OSC1 socket. With the V831, a PLL is used to generate a system clock. The frequency of the oscillator must be one-third the internal operating frequency of the V831. (A 33.33-MHz oscillator is mounted at the factory.)

The OSC1 socket accepts DIP 8-pin (half-type) oscillators.

**[Caution]** When you have to cut an oscillator pin for convenience, be careful not to cut it too short, or otherwise the frame (housing) of the oscillator may touch a tine in the socket, resulting in a short-circuit occurring.

### 5.8. 72-PIN-SIMM SOCKETS (SIMM#1, SIMM#2)

Each of the SIMM#1 and SIMM#2 sockets accepts an 8M-byte EDO type DRAM-SIMM; a total of 16M bytes can be mounted. When mounting only one SIMM, use the SIMM#1 socket.

The evaluation board is factory-fitted with an EDO-type DRAM-SIMM (8M bytes), mounted in the SIMM#1 socket. To install an additional SIMM in the SIMM#2 socket, use a SIMM of the same specifications.

**[Caution]** Both the SIMM#1 and SIMM#2 sockets accept only 72-pin, 8M-byte EDO-type DRAM SIMMs (for DOS/V). Never mount a SIMM of more than 8M bytes.

### 5.9. ROM SOCKETS

The RTE-V831-PC has ROM sockets to hold 40-pin ROM chips to provide standard 128K bytes (64K × 16 bits). When the ROM chips used here are to be replaced, their type should be 27C1024, 27C2048, or 27C4096, and the access time should be 150 ns or less. (JP1 and JP2 may need to be set according to the type and purpose of the ROM chips to be used.)

### 5.10. ROM SIZE SWITCHING JUMPER (JP1)

Jumper JP1 must be set according to the size of the installed ROM. Leave the jumper open when a ROM of 128K bytes (64K × 16 bits) or 256K bytes (128K × 16 bits) is mounted. Close the jumper when a ROM of 512K bytes (256K × 16 bits) is mounted.

**[Remark]** Jumper JP1 is left open at the factory.

### 5.11. ROM DIVISION SWITCHING JUMPER (JP2)

Jumper JP2 is used to select between the two modes of a mounted 128K-byte ROM. In one mode, the ROM is used as a contiguous 128K-byte area. In the second mode, the ROM is divided into two 64K-byte areas.

The 128K-byte ROM contains two codes of the monitor as standard. The lower half of the 128K-byte area contains one code (to be executed in the cacheable area), while the higher half contains the other code (to be executed in the uncacheable area).

1-2-3: At boot time, the ROM is viewed as a contiguous 128K-byte area.

The monitor is executed in the uncacheable area.

1-2-3: At boot time, only the lower 64K-byte area is visible.

The monitor is executed in the cacheable area.

**[Remark]** The factory-configured jumper setting depends on the monitor type. See Section 13.3.6. or 14.2.6.

**5.12. JP3, JP4, JP5**

Use jumpers JP3, JP4, and JP5 as set at the factory.

JP3: Open

JP4: Closed

JP5: Closed

**5.13. JUMPER FOR SWITCHING BETWEEN BASIC CLOCKS FOR USER TIMERS (JP6)**

JP6 is used to select which of two clocks is to be supplied to the timers (CH#1, CH#2) that can be used by applications.

1-2: 2 MHz (factory-set)

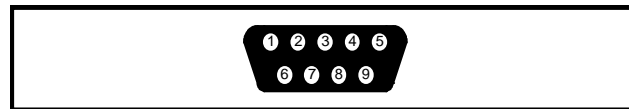
3-4: 4 MHz

**5.14. SERIAL CONNECTOR (JSIO1, JSIO2)**

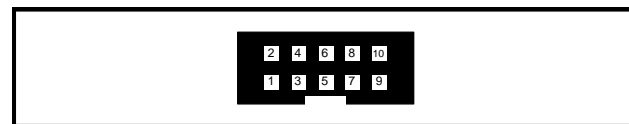
The JSIO1 and JSIO2 connectors are used for the RS-232C interface that is controlled by the serial controller (TL16C552A). JSIO1 is a 9-pin D-SUB RS-232C connector like that commonly used on the PC/AT, while JSIO2 is a pin plug type connector with a pitch of 2.54 mm. All signals on both of these connectors are converted to the RS-232C level. The figures and table below indicate the pin and signal arrangements of these connectors.

For the signals to be connected to the host, the table indicates two modes of wiring on the host: one for a 9-pin D-SUB connector, and the other for a 25-pin D-SUB connector. (Regular cross-cable wiring is used for these connections.)

The pin arrangement of JSIO2 will be identical to that of JSIO1 when a push-fit connector is used with a ribbon cable.



JSIO1 Pin Arrangement (Male)



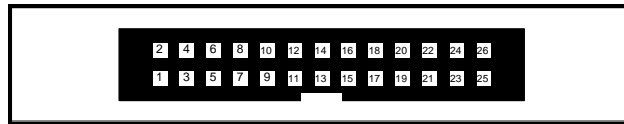
JSIO2 Pin Arrangement

JSIO1 pin No.	JSIO2 pin No.	Signal name	Input/ output	Connector pin number on the host side	
				D-SUB9	D-SUB25
1	1	DCD	Input		
2	3	RxD(RD)	Input	3	2
3	5	TxD(SD)	Output	2	3
4	7	DTR(DR)	Output	1, 6	6, 8
5	9	GND		5	7
6	2	DSR(ER)	Input	4	20
7	4	RTS(RS)	Output	8	5
8	6	CTS(CS)	Input	7	4
9	8	RI	Input		
--	10	NC			

JSIO1 and JSIO2 Connector Signals

**5.15. PARALLEL CONNECTOR (JPRT)**

The JPRT connector is used for parallel communication controlled by the parallel (printer) controller (TL16C552A). JPRT is a pin plug type connector with a 2.54 mm pitch. All signals on the connector are 5-V level signals. The figure and table below indicate the pin and signal arrangements of the connector. The pin arrangement of JPRT will be identical to that of the 25-pin D-SUB connector, like that commonly used on the PC/AT, when a push-fit connector is used with a ribbon cable.



JPRT Pin Arrangement

JPRT pin No.	Signal name	JPRT pin No.	Signal name
1	STB-	2	AUTO_FD-
3	D0	4	ERROR-
5	D1	6	INIT-
7	D2	8	SELECT_IN-
9	D3	10	GND
11	D4	12	GND
13	D5	14	GND
15	D6	16	GND
17	D7	18	GND
19	ACK-	20	GND
21	BUSY	22	GND
23	PE	24	GND
25	SELECT	26	NC

JPRT Connector Signals

**5.16. AUDIO MINI-JACKS (JMIC-R, JMIC-L, JOUT)**

Audio jacks are provided for two monaural microphone input channels and one stereo output channel. The input/output conditions of these jacks are indicated below.

**JMIC-R, JMIC-L**

Electrical input condition

140 mVp-p (Internal amplification: About 20 dB)

Physical shape of mating plug

Monaural mini-plug (3.5 DIA.) × 2 channels

**JOUT**

Electrical output condition

1.4 Vp-p

Physical shape of mating plug

Stereo mini-plug (3.5 DIA.) × 1 channel

**5.17. DEBUGGING CONNECTOR (JDBG)**

The JDBG connector is used to connect a debug tool based on the debug function built into the V831.

On-board connector: 8930E-040-178MS manufactured by KEL

Pin No.	Signal name	Pin No.	Signal name
A1	GND	B1	GND
A2	CLKOUT	B2	+3.3V
A3	GND	B3	GND
A4	TRCDATA0	B4	GND
A5	GND	B5	GND
A6	TRCDATA1	B6	GND
A7	GND	B7	GND
A8	TRCDATA2	B8	GND
A9	GND	B9	GND
A10	TRCDATA3	B10	GND
A11	GND	B11	GND
A12	DDI	B12	GND
A13	GND	B13	GND
A14	DCK	B14	GND
A15	GND	B15	GND
A16	DMS	B16	GND
A17	GND	B17	GND
A18	DDO	B18	GND
A19	GND	B19	GND
A20	DRST-	B20	GND

JDBG Connector Signals

**5.18. EXTENSION BUS CONNECTOR (JEXT)**

The JEXT connector is provided to enable memory or I/O extension. This connector is internally connected to the local bus of the board. For detailed information about the bus specifications, see Chapter 10.

**5.19. CPU CONNECTOR (JCPU-A, JCPU-B)**

The CPU connector signals are connected directly to the V831. Many signals are used on the board. So, be careful when extracting signals from the JCPU. The 3.3-V signal level is used.

Pin No.	Signal name	Pin No.	Signal name
1	GND	2	D2
3	D3	4	D4
5	D5	6	D6
7	D7	8	D8
9	+3.3V	10	GND
11	D9	12	D10
13	D11	14	+3.3V
15	GND	16	D12
17	D13	18	D14
19	D15	20	D16
21	D17	22	D18
23	D19	24	D20
25	D21	26	+3.3V
27	GND	28	D22
29	D23	30	D24
31	+3.3V	32	GND
33	D25	34	D26
35	D27	36	D28
37	D29	38	D30
39	D31	40	+3.3V
41	GND	42	LLMWR-
43	LUMWR-	44	ULMWR-
45	UUMWR-	46	MRD-
47	TXD	48	RXD
49	GND	50	+3.3V
51	SI/PORT2	52	SO/PORT1
53	SCLK-/PORT0	54	+3.3V
55	NC.	56	JCX2(*1)
57	GND	58	Reserve
59	+3.3V	60	GND
61	+3.3V	62	RESET-
63	DRST-	64	NMI-
65	BT16B	66	GND
67	+3.3V	68	GND
69	DMACK0	70	DMAAK1
71	DMAAK2	72	DMAAK3
73	DMARQ0	74	DMARQ1
75	DMARQ2	76	DMARQ3
77	REFRQ-	78	INTP03
79	INTP02	80	+3.3V

JCPU-A Connector Signals

\*1 JCX2 is a buffered signal.

Pin No.	Signal name	Pin No.	Signal name
81	GND	82	INTP01
83	INP00	84	TCLR
85	TI	86	INTP13
87	INTP11	88	INTP12/TO11
89	INTP10/TO10	90	CS7-
91	CS6-	92	+5V
93	GND	94	CS5-
95	CS4-	96	CS3-
97	+5V	98	GND
99	CS2-	100	CS1-
101	HLDK-	102	HLDRQ-
103	READY-	104	BCYST-
105	IORD-	106	IOWR-
107	+5V	108	GND
109	A23	110	A22
111	DDO	112	DMS
113	DCK	114	DDI
115	TRCDATA3	116	TRCDATA2
117	TRCDATA1	118	TRCDATA0
119	CLKOUT	120	+5V
121	GND	122	A21
123	A20	124	A19
125	A18	126	A17
127	A16	128	A15
129	A14	130	A13
131	A12	132	+5V
133	GND	134	A11
135	A10	136	+5V
137	GND	138	A9
139	A8	140	A7
141	A6	142	A5
143	A4	144	A3
145	A2	146	+5V
147	GND	148	+5V
149	GND	150	A1
151	WE-	152	OE-
153	RAS-	154	UUCAS-
155	ULCAS-	156	LUCAS-
157	LLCAS-	158	D0
159	D1	160	+5V

## JCPU-A Connector Signals

The connector used is the FX2-80P-1.27SV, manufactured by Hirose Electric Co., Ltd.

## 6. CONNECTION WITH THE HOST PC

### 6.1. INSTALLATION ON THE ISA BUS

When the RTE-V831-PC is installed in the ISA bus slot of the PC, power (+5 V) is supplied from the ISA bus to the board. In addition, the ISA bus can be used for communication with the debugger, so programs are down-loaded at high speed.

The RTE-V831-PC can be installed in the ISA bus slot according to the following procedure.

- <1> Set the I/O address of the PC using a DIP switch on the board. Be careful not to specify the same I/O address as used for any other I/O unit. See Section 5.4. for switch setting.
- <2> Turn off the power to the PC, open its housing, and check the ISA bus slot to be used. If the slot is equipped with a rear panel, remove the rear panel.
- <3> Insert the board into the ISA bus slot. Make sure that the board does not touch any adjacent board. Fasten the rear panel of the board to the housing of the PC with screws.
- <4> Turn on the power to the PC, and check that the POWER-LED on the board lights. **If the LED does not light, turn off the PC power immediately, and check the connection.** If the system does not start normally (for example, if an error occurs during installation of a device driver), it is likely that the set I/O address is the same as one already in use. Check the I/O address of the board by referring to the applicable manual of the PC or the board.
- <5> When the system turns out to be normal, turn off the PC power again, and put back its housing.

### 6.2. STANDALONE USE OF THE BOARD

When the RTE-V831-PC is used as a standalone rather than being installed in the PC, it requires an external power supply. In addition, communication with the debugger is supported by the RS-232C interface.

The RTE-V831-PC can be used as a standalone according to the following procedure.

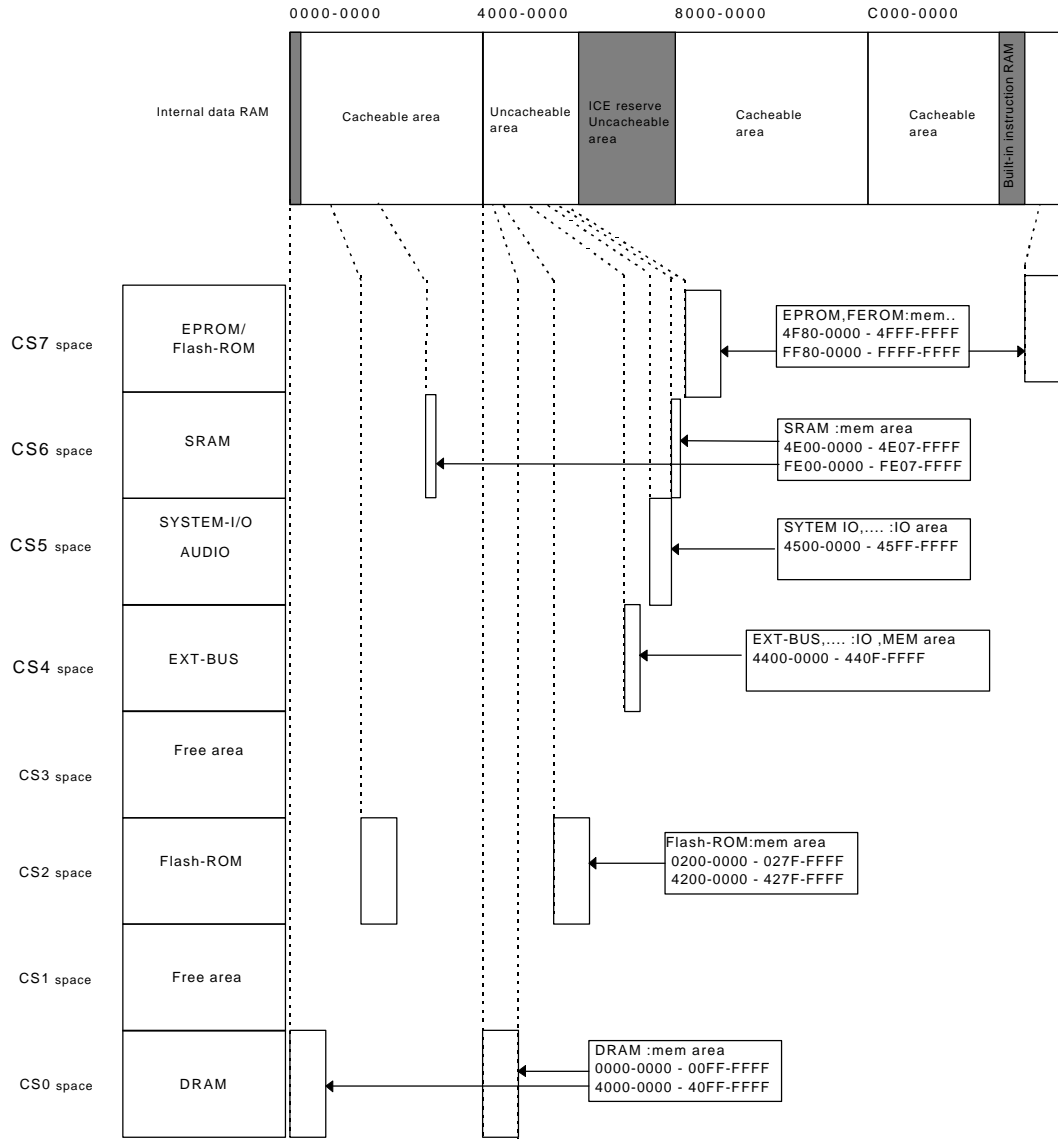
- <1> Get an RS-232C cable for connection with the host and an external power supply (+5 V, 2 A) on hand. Especially for the power supply, watch for its voltage and **connector polarity**. In addition, attach spacers to the four corners of the board, so it will not pose any problem wherever it is installed. See Sections 5.14. and 5.2. for RS-232C cable connection and the power supply connector, respectively.
- <2> Set the RS-232C baud rate using a DIP switch on the board. See Section 13.2.1. and 14.1.1. for the switch settings for the Multi and PARTNER monitors, respectively.
- <3> Connect the board to the host via an RS-232C cable. Also connect an external power supply to the JPOWER jack, then check that the POWER-LED on the board lights. **If the LED does not light, turn off the power immediately, and check the connection.**

7. HARDWARE REFERENCES

This chapter describes the hardware of the RTE-V831-PC.

7.1. MEMORY AND I/O MAP

The figure below shows the memory and I/O mapping on the board.



Memory and I/O Mapping

**[Caution]** The flash ROMs existing in the CS2 and CS7 spaces are physically identical. Before a flash ROM can be mapped into the CS7 space, a boot program starting with a reset vector must be written.

## 7.2. DETAILS OF MAPPING

Detailed information on mapping is provided below.

### **CS0 space (x000-000 to x0FF-FFFF, x800-000 to x8FF-FFFF)**

CS0 is the space for EDO-DRAM mounted in the SIMM#1 and SIMM#2 sockets. The size of the CS0 space is 16M bytes while it has a data bus width of 32 bits. The lower 8M bytes are mapped onto SIMM#1, while the higher 8M bytes are mapped onto SIMM#2. In the other space, an image appears at intervals of 16M bytes.

For interfacing with DRAM, the DRAM controller built into the V831 is used.

### **CS1 space (x100-000 to x1FF-FFFF, x900-000 to x9FF-FFFF)**

CS1 is not used.

### **CS2 space (x200-000 to x27F-FFFF, xA00-000 to xA7F-FFFF)**

CS2 is the space for flash ROM (MBM29F016PFTN-120 manufactured by Fujitsu: 2M × 8 bits, 120 ns). The size of the CS2 space is 8M bytes while it has a data bus width of 32 bits. A flash ROM image appears at intervals of 8M bytes.

Wait control is exercised by external hardware. Four wait clock cycles are always inserted.

### **CS3 space (x300-000 to x3FF-FFFF, xB00-000 to xBFF-FFFF)**

CS3 is not used.

### **CS4 space (x400-000 to x40F-FFFF, xC00-000 to xC0F-FFFF)**

CS4 is the 1M-byte extension bus space. This space is assigned as memory or I/O space by the V831's internal bus controller. An extension bus space image appears at 1M-byte intervals.

Wait control is exercised by external hardware. After two wait clock cycles have been inserted, wait control (ready) is exercised from the extension bus.

Before attempting to use this space, see Section 10.3.

### **CS5 space (x500-000 to x5FF-FFFF, xD00-000 to xDFF-FFFF)**

CS5 is the space for I/O devices including the timer, audio channels, and serial and parallel channels mounted on the board. The bus controller must be set to enable the assigning of this space as a 16-bit I/O space. Because full decoding is not performed, image spaces appear at various locations. So, never attempt to access other than the specified I/O addresses.

Wait control is exercised by external hardware. One wait clock cycle is always inserted for high-speed I/O, while seven wait clock cycles are always inserted for low-speed I/O.

For details of each I/O device, see Chapter 8.

### **CS6 space (x600-000 to x6FF-FFFF, xE00-000 to xEFF-FFFF)**

CS6 is the space for high-speed SRAM (μPD431008LE-15 manufactured by NEC: 128K × 8 bits, 15 ns). The size of the CS6 space is 512K bytes while its data bus width is 32 bits. An SRAM image appears at 512K-byte intervals.

Wait control is exercised by the bus controller built into the CPU. Access is possible with no wait clock cycles when a 33-MHz external clock is used.

### **CS7 space (x700-000 to x7FF-FFFF, xF00-000 to xFFF-FFFF)**

CS7 is the space for boot ROM. When the 16-bit bus (BT16B = 1) is selected, the EPROM is selected as boot ROM. When the 32-bit bus is selected, the flash ROM is selected as the boot ROM. (The setting of pin BT16B can be switched using SW1-8.)

As an EPROM, a 27C1024, or 27C2048 (150 ns or less) (40-pin DIP type) can be used. The evaluation board is factory-fitted with a 27C1024 incorporating the monitor.

Wait control is exercised by external hardware. Five wait clock cycles are always inserted for the EPROM, while four wait clock cycles are always inserted for the flash ROM.

## 8. SYSTEM-I/O

SYSTEM-I/O represents I/O devices mapped into the CS5 space. The I/O devices include the Audio, UART/PRINTER, TIC, PIO, and ISA bus interface. This chapter explains these devices. (A description of the ISA bus interface is omitted.)

### 8.1. SYSTEM-I/O LIST

The table below lists the SYSTEM-I/O areas and functions.

Address	Function	Remarks
4500-1000H	See the description of DIPSW1.	High-speed I/O
4500-2000H	7-segment LED display data setting	High-speed I/O
4500-3000H	Status	<b>Not made available</b>
4500-4000H	Command #1 setting/reference (PBHEn, PA0)	High-speed I/O
4500-5000H	Command #2 setting/reference (Time-Over-Clear, Flash-Reset)	High-speed I/O
4500-8000H to 4500-801FH	UART-CH#1 (TL16C552A) setting/reference	Low-speed I/O
4500-9000H to 4500-901FH	UART-CH#2 (TL16C552A) setting/reference	Low-speed I/O
4500-A000H to 4500-A01FH	Printer (TL16C552A) setting/reference	Low-speed I/O
4500-B000H to 4500-B00FH	Timer controller ( $\mu$ PD71054) setting/reference	Low-speed I/O
4500-C000H to 4500-C007H	ISA communication	<b>Not made available</b>
4500-D000H to 4500-D01FH	Interrupt controller setting/reference	Low-speed I/O
4580-0000H to 4580-001FH	Audio-Control register setting/reference	High-speed I/O
4580-1000H to 4580-100FH	$\mu$ PD63310 register setting/reference	High-speed I/O
4580-2000H	Audio-FIFO data register setting/reference	High-speed I/O

### 8.2. DIPSW1 READ PORT (4500-1000H [READ ONLY])

This port is used to read the status of DIPSW1 on the board. The table below indicates the data format.

Logical address	Data bus								Setting
	D7	D6	D5	D4	D3	D2	D1	D0	
4500-1000H input	SW4 -8	SW4 -7	SW4 -6	SW4 -5	SW4 -4	SW4 -3	SW4 -2	SW4 -1	<b>0 = ON</b> <b>1 = OFF</b>
Hardware assignment	BT16B	-		-	-	-	-	-	

The status of SW1, installed on the board, can be read. SW1-1 corresponds to switch 1 of SW1, while SW1-8 corresponds to switch 8 of SW1. When a bit is ON, 0 is read. When a bit is OFF, 1 is read. Bits 1 to 6 of SW1 are used by the Multi monitor to set operation. (See Section 13.2.).

SW1-8 is used to set the hardware.

#### SW1-8: Boot-time bus size (BT16B) and ROM specification

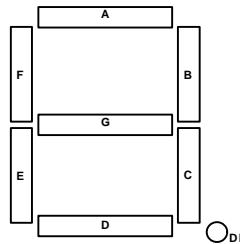
BT16B	Boot-time bus size and ROM
0	Boot on 32-bit bus from flash ROM
1	Boot on 16-bit bus from EPROM

**8.3. 7-SEGMENT LED DISPLAY DATA OUTPUT PORT (4500-2000H [WRITE ONLY])**

This port sets the data to be displayed on the 7-segment LED. The table below indicates the data format. When a bit is set to 0, the corresponding segment is turned on.

Logical address	Data bus								Setting
	D7	D6	D5	D4	D3	D2	D1	D0	
4500-2000H output	7SEG -DP	7SEG -G	7SEG -F	7SEG -E	7SEG -D	7SEG -C	7SEG -B	7SEG -A	0 = Turned on 1 = Turned off

The figure below illustrates the correspondence between the bits and the segments of the 7-segment LED.



**8.4. COMMAND REGISTER #0 PORT (4500-3000H [READ/WRITE])**

This port is an area reserved by the system. Never change the following settings:

Logical address	Register	Data bus							
		D7	D6	D5	D4	D3	D2	D1	D0
4500-4000H	<b>CMD#0</b> (initial value)	X	X	X	X	0 (0)	0 (0)	0 (0)	0 (0)

**8.5. COMMAND REGISTER #1 PORT (4500-4000H [READ/WRITE])**

This port is used for exercising byte enable (BHEn, A0) control when accessing the external extension bus (EXT bus).

Logical address	Register	Data bus							
		D7	D6	D5	D4	D3	D2	D1	D0
4500-4000H	<b>CMD#1</b> (initial value)	X	X	X	X	X	X	PBHE (0)	PA0 (0)

To access the external extension bus (EXT bus), set the following:

When the CS4 space (EXT bus) is mapped into the I/O space

Access type	PBHE-	PA0
Byte access to an even-numbered address	1	0
Byte access to an odd-numbered address	0	1
Half-word access to an even-numbered address	0	0

When the CS4 space (EXT bus) is mapped into the memory space

Keep PBHE- and PA0 set to 0 at all times, regardless of the access type. In a read cycle, half-word access is always performed. In a write cycle, BHEn and A0 on the extension bus are generated according to the access size.

**8.6. COMMAND REGISTER #2 PORT (4500-5000H [READ/WRITE])**

This port has the following functions:

Logical address	Register	Data bus							
		D7	D6	D5	D4	D3	D2	D1	D0
4500-5000H	<b>CMD#2</b> (initial value)	X	X	X	X	X	0	FRES (0)	TOVEN (0)

**TOVEN:** Controls the use of the time-over function. When the length of a bus cycle reaches 512 bus clocks, the time-over function returns READY-, and forcibly terminates the bus cycle.

TOVEN	Time-over function
0	Used (Reset value)
1	Not used

**FRES:** Resets the flash ROM.

FRES	Flash ROM
0	Reset clear (Reset value)
1	Reset

\* Access to the flash ROM is enabled only in the reset clear state.

**8.7. UART/PRINTER (TL16C552A) (4500-8000H TO 4500-A00CH)**

The Texas Instruments TL16C552A (dual asynchronous communications element with FIFO) LSI is used as UART/PRINTER controller. The TL16C552A provides two UART channels and one channel of the bidirectional printer port (PS2-compatible). It incorporates a 16-character FIFO buffer in the UART receive circuitry.

Each register of the TL16C552A is assigned as listed below. For an explanation of the function of each register, refer to the manual provided with the TL16C552A.

Address	Function	Read	Write
4500-8000H	UART-CH#1	RBR/DLL	THR/DLL
4500-8004H		IER/DLM	IER/DLM
4500-8008H		IIR	FCR
4500-800CH		LCR	LCR
4500-8010H		MCR	MCR
4500-8014H		LSR	LSR
4500-8018H		MSR	MSR
4500-801CH		SCR	SCR
4500-9000H	UART-CH#2	RBR/DLL	THR/DLL
4500-9004H		IER/DLM	IER/DLM
4500-9008H		IIR	FCR
4500-900CH		LCR	LCR
4500-9010H		MCR	MCR
4500-9014H		LSR	LSR
4500-9018H		MSR	MSR
4500-901CH		SCR	SCR
4500-A000H	PRINTER	Read-data	Write-data
4500-A004H		Read-status	-----
4500-A008H		Read-control	Write-control
4500-A00CH		-----	-----

TL16C552A Register Arrangement

The CLK input of the TL16C552A is connected to the 16-MHz clock.

### 8.8. TIC (*m*PD71054) (4500-B000H TO 4500-B00CH)

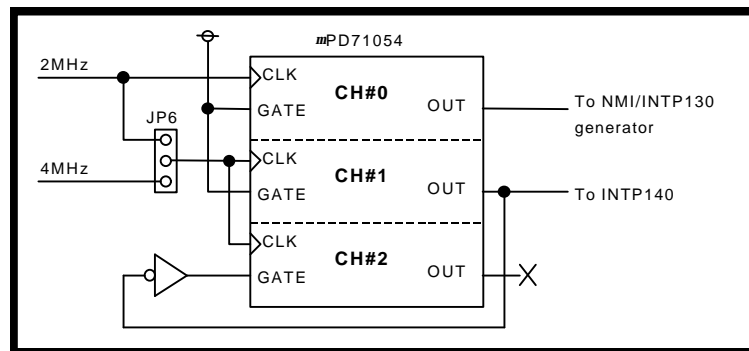
The NEC *m*PD71054 is installed as a TIC. The *m*PD71054 is compatible with the Intel i8254. It has three timers/counters. These timers/counters are used to generate monitor timer interrupts. Each register of the TIC is assigned as listed below.

Address	Read	Write
4500-B000H	COUNTER#0	COUNTER#0
4500-B004H	COUNTER#1	COUNTER#1
4500-B008H	COUNTER#2	COUNTER#2
4500-B00CH	-----	Control Word

TIC Register Arrangement

The channels of the TIC are connected as shown in the figure below. Channel 0 is connected to the PIC and used as the interval timer for the monitor. A 2-MHz clock input is connected to channel 0. Channel 1 can be used by a user program as necessary. It serves as the pre-scale counter for channel 2. Channel 2 can also be used by a user program as necessary.

A 2- or 4-MHz clock can be selected for connection to each of channels 1 and 2 using JP6 on the board.



The *m*PD71054 requires a command recovery time of 165 ns. To reserve the recovery time, dummy read from the ROM area is recommended. The TIC is reset upon a system reset.

#### Examples of modes

- CH#0: Mode 2 (rate generator)
- CH#1: Mode 2 (rate generator)
- CH#2: Mode 0 (down counter)

### 8.9. INTERRUPT CONTROLLER (PIC) (4500-D000H TO 4500-D018H)

The PIC mainly exercises interrupt-related control. The table below indicates the assignment of registers.

With the RTE-V831-PC, INT0 of the PIC is connected to NMI or INTP03 of the V831 according to the specification of NMI/INT3-. INT1 is connected to INTP02.

Logical address	Register	Data bus							
		D7	D6	D5	D4	D3	D2	D1	D0
4500-D000H	PIC INT0M	IM07	IM06	IM05	IM04	IM03	IM02	IM01	IM00
4500-D008H	PIC INT1M	IM17	IM16	IM15	IM14	IM13	IM12	IM11	IM10
4500-D010H	PIC INTR	IR7	IR6	IR5	IR4	IR3	IR2	IR1	IR0
4500-D018H	PIC INTEN	0	0	0	NMI/ INT3-	0	0	1	NMIEN

The INT0M and INT1M registers mask interrupts applied to INT0 and INT1, respectively. When the IM0x or IM1x bit is set to 1, the interrupt is enabled. When multiple bits are selected, each OR value activates an interrupt.

The INTR register is an interrupt status register, for which 1 is read whenever there is an interrupt request. This does not depend on the state of masking. To clear an edge interrupt request, the corresponding bit of this register must be set to 1.

The table below indicates the interrupt source assigned to each bit of IM0[0..7], IM1[0..7], and IR[0..7].

IM0, IM1, IR	Interrupt source	Request level
0	Timer 0 (mode 2)	Edge (rising)
1	Serial 0	Level (high)
2	Host (ISA communication)	Level (low)
3	Time-over	Level (low)
4	Timer 1 (mode 2)	Edge (rising)
5	Serial 1	Level (high)
6	Parallel (printer)	Level (high)
7	ISA-IRQ	Level

The INTEN register enables or disables all interrupts.

**NMIEN:** Disables a non-maskable interrupt (NMI) by hardware. At this time, the NMI pin is high.

NMIEN	NMI
0	Sets a mask. (Reset value)
1	Does not set a mask.

**NMI/INTP3-:** Specifies whether an INT0 interrupt is to be applied to NMI or INTP03.

NMI/INTP3-	INT0
0	INTP03 (Reset value)
1	NMI

**[Caution]** INT0 (NMI/INTP03) is used with the monitor. So, never modify the related registers. INT1 is released, and can be used freely.

**8.10. AUDIO CONTROLLER (AUDCNT) (4580-0000H TO 4580-0010H, 4580-2000H)**

AUDCNT controls digital data input to and output from the audio chip (μPD63310).

Logical address	Register	Data bus							
4580-0000H	<b>CONTROL</b>	D15	D14	D13	D12	D11	D10	D9	D8
		RST	0	0	0	0	RM1	RM0	REC
		D7	D6	D5	D4	D3	D2	D1	D0
		0	0	0	0	0	PM1	PM0	PLY
4580-0008H	<b>STATUS</b>	D15	D14	D13	D12	D11	D10	D9	D8
		0	FF1	HF1	EF1	0	ROV	RUD	REX
		D7	D6	D5	D4	D3	D2	D1	D0
		0	FF0	HF0	EF0	0	POV	PUD	PEX
4580-0010H	<b>MCLKDIV</b>	D15	D14	D13	D12	D11	D10	D9	D8
		0	0	0	0	0	0	0	0
		D7	D6	D5	D4	D3	D2	D1	D0
		0	0	0	DIV4	DIV3	DIV2	DIV1	DIV0
4580-2000H	<b>AUDIO DATA</b>	D15	...						D0
		MSB	L/R channel voice data						LSB

The CONTROL register controls voice recording/replay. (Read/write)

RST	Audio reset
0	Reset clear (Reset value)
1	Reset

PLY	Replay operation
0	Stop (Reset value)
1	Start

PM1	Overflow interrupt upon replay
0	Disable (Reset value)
1	Enable

PM0	Underflow interrupt upon replay
0	Disable (Reset value)
1	Enable

REC	Recording operation
0	Stop (Reset value)
1	Start

RM1	Overflow interrupt upon recording
0	Disable (Reset value)
1	Enable

RM0	Underflow interrupt upon recording
0	Disable (Reset value)
1	Enable

The STATUS register is a read-only register for indicating various statuses.

PEX	Replay status
0	Stopped
1	Being conducted

POV	Overflow upon replay
0	No overflow detected
1	Overflow detected

PUD	Underflow upon replay
0	No underflow detected
1	Underflow detected

FF0	HF0	EF0	FIFO for replay
0	0	1	Empty
0	1	0	Half-full
1	1	0	Full (FIFO write disabled)

REX	Recording status
0	Stopped
1	Being conducted

ROV	Overflow upon recording
0	No overflow detected
1	Overflow detected

RUD	Underflow upon recording
0	No underflow detected
1	Underflow detected

FF1	HF1	EF1	FIFO for recording
0	0	1	Empty (FIFO read disabled)
0	1	0	Half-full
1	1	0	Full

The MCLKDIV register is used to determine the MCLK frequency.

DIV4	DIV3	DIV2	DIV1	DIV0	MCLK 49.152/(DIV + 2)	Sampling frequency (MCLK/256)	Bytes/sec fs * 4
0	0	0	0	0	24.576 MHz		
0	0	0	0	1	16.384 MHz		
0	0	0	1	0	12.288 MHz	48.0 KHz	192.0 KB
0	0	0	1	1	9.830 MHz	38.4 KHz	153.6 KB
0	0	1	0	0	8.192 MHz	32.0 KHz	128.0 KB
0	0	1	0	1	7.022 MHz	27.5 KHz	109.7 KB
0	0	1	1	0	6.144 MHz	24.0 KHz	96.0 KB
0	0	1	1	1	5.461 MHz	21.3 KHz	85.3 KB
0	1	0	0	0	4.915 MHz	19.2 KHz	76.8 KB
0	1	0	0	1	4.468 MHz	17.5 KHz	69.8 KB
0	1	0	1	0	4.096 MHz	16.0 KHz	64.0 KB
0	1	0	1	1	3.780 MHz	14.8 KHz	59.1 KB
0	1	1	0	0	3.511 MHz	13.7 KHz	54.9 KB
0	1	1	0	1	3.277 MHz	12.8 KHz	51.2 KB
0	1	1	1	0	3.072 MHz	12.0 KHz	48.0 KB
0	1	1	1	1	2.891 MHz	11.3 KHz	45.2 KB
1	0	0	0	0	2.731 MHz	10.7 KHz	42.7 KB
1	0	0	0	1	2.587 MHz	10.1 KHz	40.4 KB
1	0	0	1	0	2.458 MHz	9.6 KHz	38.4 KB
1	0	0	1	1	2.341 MHz	9.1 KHz	36.6 KB
1	0	1	0	0	2.234 MHz	8.7 KHz	34.9 KB
1	0	1	0	1	2.137 MHz	8.3 KHz	33.4 KB
1	0	1	1	0	2.048 MHz	8.0 KHz	32.0 KB
1	0	1	1	1	1.966 MHz	7.7 KHz	30.7 KB
1	1	0	0	0	1.890 MHz	7.4 KHz	29.5 KB
1	1	0	0	1	1.820 MHz	7.1 KHz	28.4 KB
1	1	0	1	0	1.755 MHz	6.9 KHz	27.4 KB
1	1	0	1	1	1.695 MHz	6.6 KHz	26.5 KB
1	1	1	0	0	1.638 MHz	6.4 KHz	25.6 KB
1	1	1	0	1	1.586 MHz	6.2 KHz	24.8 KB
1	1	1	1	0	1.536 MHz	6.0 KHz	24.0 KB
1	1	1	1	1	1.489 MHz	5.8 KHz	23.3 KB

AUDIO DATA is a data port for audio data FIFO input/output; 16-bit data is input or output, in order, from Lch to Rch.

**8.11. mPD63310 REGISTER: AUDIO COD. (4580-1000H TO 4580-100FH)**

The mPD63310 register is assigned as indicated below. For details, refer to the data sheet provided with the mPD63310.

Address	Function	D5	D4	D3	D2	D1	D0
4580-1000H	Address register	Register number					
4500-1008H	Data register	Gain control data					

## 9. INTERRUPTS AND DMA

This chapter describes the interrupts and DMA for the RTE-V831-PC.

### 9.1. INTERRUPT

External interrupts are used as indicated below.

Interrupt	Source
NMI	Interrupt from the ROM emulator, and INT0 interrupt by PIC setting (for the monitor)
INTP03	INT0 interrupt by PIC setting (for the monitor)
INTP02	INT1 interrupt by PIC setting (for the user)
INTP01	Interrupt from the EXT (extension) bus
INTP00	Interrupt from the audio interface (when an error occurs)
INTP13	Not used
INTP12	Not used
INTP11	Not used
INTP10	Not used

Two interrupts (INT0, INT1) can be generated by selecting interrupt requests from the interrupt sources listed below with the interrupt controller (see Section 8.9.) on the board. INT0 is used for the system (used with the Multi monitor), while INT1 is used for a user application.

Interrupt source
Timer 0 (mode 2)
Serial 0
Host (ISA communication)
Time-over
Timer 1 (mode 2)
Serial 1
Parallel (printer)
ISA-IRQ

### 9.2. USING NMI

This section describes the method of using NMI for transporting the monitor, for example, to the board. NMI is edge-detected. NMI can be masked by hardware because the interrupt source is a level output. For an explanation of masking, see the description of the INTEN register in Section 8.9.

The following procedure applies when an NMI occurs.

- <1> Set the NMIEN of the PIC to 0 to mask the NMI by hardware.
- <2> Check the INTR of the PIC.
- <3> Perform NMI processing for the interrupt source, and clear the request.
- <4> Reset the NMIEN of the PIC to 1 to reset the mask.
- <5> Return from NMI processing.

**[Caution]** When the monitor is used, the operation of an INT0-related register of PIC, NMI, or INTP03 may cause the monitor to hang.

### 9.3. DMA CHANNEL

DMARQ-/AK-	DMARQ signal	Remark
CH0	Replay request	A DMA request for data to be written to the audio data buffer during replay. A timeout results in an underrun error.
CH1	Recording request	A DMA request for data to be read from the audio data buffer during recording. A timeout results in an overflow error.

**Remark** Set DMARQ-/AK-[1..0] to negative logic.

## 10. EXT BUS SPECIFICATIONS

The EXT bus, provided with JEXT connectors, is used to expand the memory and I/O units. The local bus of this board is connected to the JEXT connector.

### 10.1. PIN ARRANGEMENT AND SIGNALS

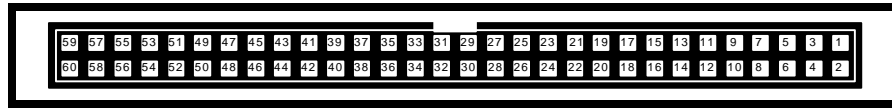
The following table shows the pin arrangement of the JEXT connector. The signals on these pins are also explained below.

Number	Signal name	Number	Signal name	Number	Signal name	Number	Signal name
1	+5V	2	+5V	31	GND	32	GND
3	D0	4	D1	33	A8	34	A9
5	D2	6	D3	35	A10	36	A11
7	D4	8	D5	37	A12	38	A13
9	D6	10	D7	39	A14	40	A15
11	GND	12	GND	41	+5V	42	+5V
13	D8	14	D9	43	A16	44	A17
15	D10	16	D11	45	A18	46	A19
17	D12	18	D13	47	BHE-	48	GND
19	D14	20	D15	49	GND	50	RD-
21	+5V	22	+5V	51	WR-	52	RESET-
23	A0	24	A1	53	GND	54	GND
25	A2	26	A3	55	READY	56	INT-
27	A4	28	A5	57	GND	58	GND
29	A6	30	A7	59	CPUCLK	60	GND

JEXT Connector Pin Arrangement

Signal name	Input/output	Function
A[1..19]	Output	Address bus signal, connected to the CPU address signal via a buffer
A0, BHE-	Output	Byte low/high enable signal
D[0..15]	Input/output	Data bus signal, connected to the CPU data bus signal via a buffer It is pulled up with a 10-k $\Omega$ resistor on the board.
RD-	Output	Read cycle timing signal, which becomes active only when the JEXT space is accessed.
WR-	Output	Write cycle timing signal, which becomes active only when the JEXT space is accessed.
READY	Input	Signal for notifying the V831 of the end of a cycle. It is valid for the JEXT space. To have the V831 reliably recognize READY, it is necessary to keep READY active until RD- or WR- becomes inactive. It is pulled up with a 10-k $\Omega$ resistor on the board.
INT-	Input	Active-low interrupt request signal. It is logically inverted and connected to the INTP01 pin of the V831 via a buffer.
RESET-	Output	Active-low system reset signal
CLK	Output	Clock signal, connected to the CLKOUT pin of the V831 via a buffer.

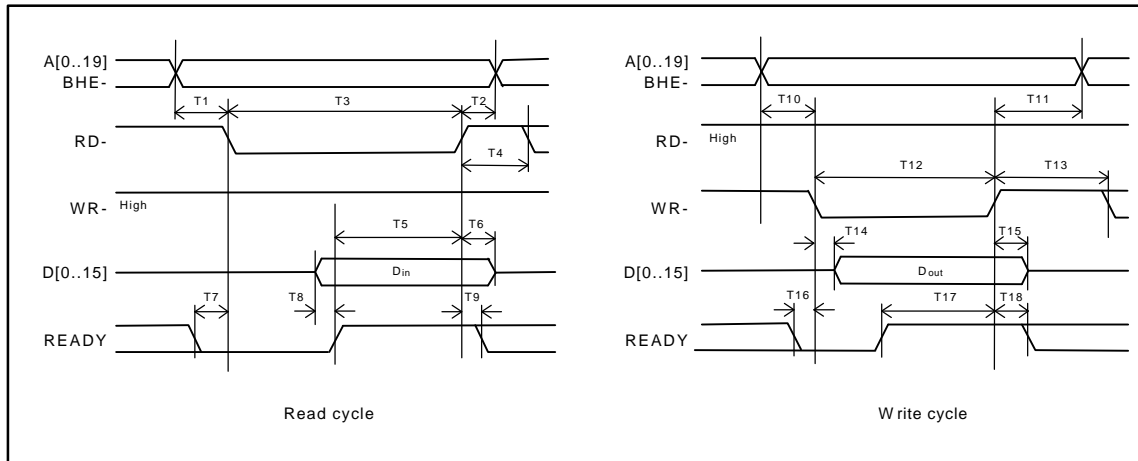
JEXT Connector Signals



JEXT Pin Arrangement

10.2. TIMING

The timing of the EXT bus is shown below.



JEXT Bus Cycle

Symbol	Description	Min. (ns)	Max. (ns)
T1	RD address setup time	0	
T2	RD address hold time	0	
T3	RD cycle time	50	
T4	RD cycle interval	20	
T5	RD data setup time	15	
T6	RD data hold time	0	
T7	RD READY WAIT setup time	0	
T8	RD READY setup time	0	
T9	RD READY hold time	0	
T10	WR address setup time	0	
T11	WR address hold time	20	
T12	WR cycle time	50	
T13	WR cycle interval	20	
T14	WR data delay time		20
T15	WR data hold time	20	
T16	WR READY WAIT setup time	0	
T17	WR READY setup time	0	
T18	WR READY hold time	0	

JEXT Bus AC Specifications

**10.3. NOTES ON USING THE EXT BUS**

When extending hardware on the EXT bus, note the following points.

(1) Memory and I/O space mapping

The EXT bus space is mapped onto the CS4 space of the V831. For access, select memory or I/O space with the bus controller (BCTC) built into the CPU.

(2) A0 and BHE- signal handling

The A0 and BHE- signals are used on the EXT bus. They are determined by access and port output from the V831. Before access, perform setting as explained in Section 8.5.

## 11. SOFTWARE

This chapter describes the initialization of the hardware of the RTE-V831-PC board, and explains how to use peripheral devices.

### 11.1. INITIALIZATION

The boot routine initializes the bus controller built into the V831 for external memory or I/O access. The wait control and DRAM timing values indicated below assume a bus clock of 33 MHz.

Register	Internal I/O address	Set value	Size	Remarks
BCTC	C000-0010H	30H	Byte	
DBC	C000-0012H	30H	Byte	
PWC0	C000-0014H	7470H	Half-word	
PWC1	C000-0016H	0000H	Half-word	SRAM: 0 wait state (CS6)
PIC	C000-0018H	0000H	Half-word	
DRC	C000-0020H	02H(82H)	Byte	EDO 60ns (hyper page mode)
RFC	C000-0022H	900EH	Half-word	Approx. 15 $\mu$ s

For detailed information about the registers, refer to the manual provided with the V831 CPU.

### 11.2. LIBRARIES

Libraries are required for programming using the C compiler for I/O accesses and other purposes. However, the methods of writing these libraries and passing their parameters described below are specific to the GHS-C. So, modifications may be required, for example, when another compiler is used.

```

/* I/O library */

/* GHS V800 compiler parameter passing */
/* arg0 : r6, arg1 : r7, arg2 : r8, return : r10 */

inb(int addr) /* Byte (8 bits) input */
{
    __asm(" in.b 0[r6], r10");
}

inh(int addr) /* Half-word (16 bits) input */
{
    __asm(" in.h 0[r6], r10");
}

inw(int addr) /* Word (32 bits) input */
{
    __asm(" in.w 0[r6], r10");
}

outb(int addr, int data) /* Byte (8 bits) output */
{
    __asm(" out.b r7, 0[r6]");
}

outh(int addr, int data) /* Half-word (16 bits) output */
{
    __asm(" out.h r7, 0[r6]");
}

outw(int addr, int data) /* Word (32 bits) output */
{
    __asm(" out.w r7, 0[r6]");
}

```

### 11.3. USING TIMERS

A sample time measurement is indicated below which uses timer 1 and timer 2 cascaded with each other by an external timer (8254) on the board. Timer 1 is initialized as an interval counter (mode 2), and timer 2 is initialized as a down counter (mode 0). By determining the counter values before and after a routine whose execution time is to be measured, the execution time can be calculated. Note that both timers function as down counters. Note also that command recovery (dummy read from the ROM area) is required for successive accesses to the external timer.

```

/* Sample execution time measurement using timers */

#define TIMERCLK 2000000 /* 2MHz */
#define INTERVAL(TIMERCLK * 10 / 1000) /* 10 ms (1/100) */
#define IOWAIT()(*(char *)0x4FFF0000) /* For I/O command recovery */

InitTimer() /* Timer initialization */
{
    outb(0x4500B00C, 0x74); IOWAIT(); /* Timer 1 set to mode 2 */
    outb(0x4500B004, INTERVAL); IOWAIT(); /* Lower digit count of timer 1 */
    outb(0x4500B004, INTERVAL / 256); IOWAIT(); /* Higher digit count of timer 1 */
    outb(0x4500B00C, 0xB0); IOWAIT(); /* Timer 2 set to mode 0 */
    outb(0x4500B008, 0xFF); IOWAIT(); /* Lower digit count of timer 2 */
    outb(0x4500B008, 0xFF); IOWAIT(); /* Higher digit count of timer 2 */
    return 0;
}

LatchTimer() /* Count Latch */
{
    int count1, count2, counts;

    outb(0x4500B00C, 0xDC); IOWAIT(); /* Timer 1/2 multiple latch */
    count1 = inb(0x4500B004); IOWAIT();
    count1 += inb(0x4500B004) * 256; IOWAIT(); /* Count of timer 1 */
    count2 = inb(0x4500B008); IOWAIT();
    count2 += inb(0x4500B008) * 256; IOWAIT(); /* Count of timer 2 */
    counts = INTERVAL * (0xFFFF - count2)
        + (INTERVAL - count1);
    return counts;
}

double total_time;

main()
{
    int start_count, stop_count;

    InitTimer();
    start_count = LatchTimer(); /* Start count value */
    func(); /* Stop count value */
    stop_count = LatchTimer();
    total_time = (double)(stop_count - start_count)
        / (double)TIMERCLK; /* Seconds */

    return 0;
}

#include <time.h>
func() /* Time measurement routine */
{
    ...
}

```

#### 11.4. FLASH ROM PROGRAMMING

A sample program for writing data into a flash ROM mounted on the board is indicated below. For writing to a flash ROM, data is written on a byte-by-byte basis in the uncacheable area. For detailed information about the programming algorithm, refer to the data sheet for the flash ROM.

```

/* Flash ROM write sample */

#define FLCMD_ERASE      0x80          /* Erase */
#define FLCMD_WRITE     0xA0          /* Write */
#define FLCMD_READ      0xF0          /* Read */
#define BIT_DQ(n)       (1<<(n))     /* Bit macro */

/* Flash-ROM Library */

static FlashCommand(int addr, int cmd) /* Command */
{
    int addr2;

    addr2 = addr & ~(0x7FF<<2);      /* Address 11 bit 0 clear */
    *(char *)(addr2 + (0x555<<2)) = 0xAA;
    *(char *)(addr2 + (0x2AA<<2)) = 0x55;
    *(char *)(addr2 + (0x555<<2)) = cmd;
    return 0;
}

static FlashDataPoll(int addr, int data) /* Busy check */
{
    int rdata;

    data &= 0xff;                    /* Byte */
    do {
        rdata= *(char *)addr & 0xff;
        if (((data ^ rdata) & BIT_DQ(7)) == 0) {
            break;
        }
    } while ((rdata & BIT_DQ(5)) == 0);
    rdata = *(char *)addr & 0xff;
    return (((data ^ rdata) & BIT_DQ(7)) == 0)? 0: -1;
}

FlashErase(int addr, int all)         /* Erase (whole/sector) */
{
    FlashCommand(addr, FLCMD_ERASE);
    FlashCommand(addr, (all)? 0x10: 0x30); /* Whole/sector */
    return FlashDataPoll(addr, 0xff);
}

FlashWrite(int addr, int data)        /* Write */
{
    FlashCommand(addr, FLCMD_WRITE);
    data &= 0xFF;
    *(char *)addr = data;
    return FlashDataPoll(addr, data);
}

FlashRead(int addr)                  /* Read (reset) */
{
    int data;

    FlashCommand(addr, FLCMD_READ);
    data = *(char *)addr & 0xFF;
    return data;
}

/* Copy monitor to flash ROM */

#define SRC_ADDR        0xFFFF0000   /* EPROM */
#define DST_ADDR        0x427F0000   /* FLASH ROM */
#define DATA_SIZE      0x10000      /* 64 KB */

main()
{
    int daddr, saddr, data;
    int i, err;

```

```

if (inb(0x45001000) & 0x80) == 0) {          /* Confirm EPROM boot */
    return -1;
}
daddr = DST_ADDR;
saddr = SRC_ADDR;

for (i = 0; i < 4; i++) {
    FlashRead(daddr + i);                  /* Initialization */
}
for (i = 0; i < DATA_SIZE; i++) {
    if ((i & (0xFFFF<<2)) == 0) {
        err = FlashErase(daddr, 0);        /* Erase sector */
        if (err) {
            break;                          /* Erase error */
        }
    }
    data = *(char *)saddr & 0xFF;
    err = FlashWrite(daddr, data);         /* Write */
    if (err) {
        break;                              /* Write error */
    }
    saddr++;
    daddr++;                                /* Next address */
}
daddr = DST_ADDR;
for (i = 0; i < 4; i++) {
    FlashRead(daddr + i);                  /* Initialization (End) */
}
return err;
}

```

After running this program with the debugger, terminate the program normally. Next, terminate the debugger, set SW1-8 to ON, then reset the board. Then, the monitor starts from the flash ROM.

**[Caution]** During writing to flash ROM, do not attempt to cause a break and subsequently resume operation, or to gain data access to the flash ROM. Otherwise, flash ROM command execution may be interrupted, possibly causing serious damage to the flash ROM.

## 11.5 AUDIO I/O

A sample program using the audio input/output interface mounted on the board is indicated below. For data input/output, the DMA built into the V831 is used.

```

/* Audio input/output sample */

#define DMA0          0xC0000030      /* Built-in DMA ch0 (replay) */
#define DMA1          0xC0000040      /* Built-in DMA ch1 (recording) */
#define AUDIO_DATA    0x45802000      /* Audio data (FIFO) */

static Set63310Reg(int reg, int data) /* Set mPD63310 register */
{
    outb(0x45801000, reg);           /* Write address register */
    outb(0x45801008, data);          /* Write data register
    return 0;
}

static Get63310Reg(int reg)          /* Acquire mPD63310 register */
{
    outb(0x45801000, reg);           /* Write address register */
    return inb(0x45801008) & 0x3F;   /* Read data register (6 bits) */
}

AudioInit()                          /* Audio initialization */
{
    outh(0x45800000, 0x8000);         /* Reset audio */
    outh(0x45800010, 8);              /* fs = 12 kHz */
    inh(0x45800010);                 /* Dummy read */
    outh(0x45800000, 0);              /* Reset clear */
    Set63310Reg( 0, 0);               /* IN1L 0db */
    Set63310Reg( 1, 0);               /* IN1R 0db */
    Set63310Reg(17, 0);               /* OUTDACL 0db*/
    Set63310Reg(18, 0);               /* OUTDACR 0db */
    return 0;
}

AudioTerm();                          /* Audio termination processing */
{
    Set63310Reg( 0, 0x20);            /* IN1L mute */
    Set63310Reg( 1, 0x20);            /* IN1R mute */
    Set63310Reg(17, 0x20);            /* INDACL mute */
    Set63310Reg(18, 0x20);            /* INDACR mute */
    outh(0x45800000, 0);              /* Stop command */
    return 0;
}

AudioPlay(int addr, int size)         /* Use replay processing DMA0 */
{
    outh(DMA0 + 0, addr >> 16);      /* DMA-DSA0H */
    outh(DMA0 + 2, addr);              /* DMA-DSA0L */
    outh(DMA0 + 4, AUDIO_DATA >> 16); /* DMA-DDA0H */
    outh(DMA0 + 6, AUDIO_DATA);        /* DMA-DDA0L */
    size = (size / 2 - 1) * 2;         /* DMA transfer count */
    outh(DMA0 + 8, size >> 16);        /* DMA-DBC0H */
    outh(DMA0 + 10, size);              /* DMA-DBC0L */
    outh(DMA0 + 12, (0<<12)           /* DMA-DCHC0
                    | (1<<10)         /* TYP DMARQ */
                    | (0<<8)          /* TBT Mem->I/O */
                    | (2<<6)          /* SAD Inc */
                    | (0<<5)          /* DAD Fix */
                    | (0<<4)          /* DAL Low */
                    | (1<<3)          /* DRL Low */
                    | (1<<1)          /* TM Demand */
                    | 1);              /* DS Half-word */
    outh(0x45800000, 0x0001);         /* Start replay */
    while ((inh(DMA0 + 12) & 1) != 0) /* Wait for DMA termination */
        ;
    while ((inh(0x45800008) & 0x10) == 0) /* Wait for FIFO empty */
        ;
    outh(0x45800000, 0);              /* Replay termination */
}

```

```

    return 0;
}

AudioRecord(int addr, int size)                /* Use record processing DMA1 */
{
    outh(DMA1 + 0, AUDIO_DATA >> 16);        /* DMA-DSA1H */
    outh(DMA1 + 2, AUDIO_DATA);               /* DMA-DSA1L */
    outh(DMA1 + 4, addr >> 16);              /* DMA-DDA1H */
    outh(DMA1 + 6, addr);                     /* DMA-DDA1L */
    size = (size / 2 - 1) * 2;                /* DMA transfer count */
    outh(DMA1 + 8, size >> 16);               /* DMA-DBC1H */
    outh(DMA1 + 10, size);                     /* DMA-DBC1L */
    outh(DMA1 + 12, (0<<12)                   /* DMA-DCHC1      TTYP (DMARQ) */
        | (2<<10)                               /*                TBT I/O->MEM */
        | (2<<8)                                 /*                SAD fix */
        | (0<<6)                                 /*                DAD inc */
        | (0<<5)                                 /*                DAL Low */
        | (0<<4)                                 /*                DRL Low */
        | (1<<3)                                 /*                TM demand */
        | (1<<1)                                 /*                DS half-word */
        | 1);                                    /*                Enable */

    outh(0x45800000, 0x100);                  /* Start recording */
    while ((inh(DMA1 + 12) & 1) != 0)
        ;                                       /* Wait for DMA termination */
    outh(0x45800000, 0);                       /* Recording termination */
    return 0;
}

#define COUNT    0x10000                      /* L/R data sample count */

int buffer[COUNT];                           /* L/R data buffer */

main()
{
    inb(0xC000006E);
    outb(0xC000006E, 1);                      /* DMA-DC MEM=1 */
    AudioInit();                               /* Initialization */
    AudioRecord((int)buffer, sizeof(buffer)); /* Recording */
    AudioPlay ((int)buffer, sizeof(buffer)); /* Replay */
    AudioTerm();                               /* Termination */
    return 0;
}

```

Audio data consists of 16-bit data for each of Lch (left) and Rch (right). Data is to be input and output, in order, from L1 to R1 to L2 to R2 and so forth for both recording and replay.

## 12. DEVELOPMENT OF APPLICATIONS USING MASKABLE INTERRUPTS

This chapter describes the methods of developing an application on the RTE-V831-PC by using a maskable interrupt, and related restrictions.

### 12.1. INTERRUPT VECTOR

The V831 interrupt vector area of addresses FFFF-FE00H to FFFF-FFFFH is fixed in the ROM, and cannot be rewritten. So, for the NEC monitor, an alternate vector area is allocated in the SRAM; in a vector at addresses FFFF-FE00H to FFFF-FFFFH, an instruction for causing a branch to the alternate vector area is placed. If, for example, an interrupt with exception code FE00H is generated, the CPU interrupt function causes a branch to address FFFF-FE00H, where an instruction for causing a branch to the corresponding alternate vector area is placed. This means that, by rewriting the alternate vector area in the same way as with the original vector area, a branch to the user program interrupt handling routine can be caused when an interrupt is generated.

The difference from an ordinary V831 program is that a vector area is fixed in ROM, and no rewriting by a program is required. However, a program running on the monitor must rewrite the alternate vector area to enable an interrupt.

With the monitor of the RTE-V831-PC, an alternate vector area is allocated at FE07-0000H to FE07-01FFH in SRAM. So, for an interrupt with exception code FE00H, an instruction for causing a branch to the interrupt handling routine is to be written at address FE07-0000H; for an interrupt with exception code FE10H, an instruction for causing a branch to the interrupt handling routine is to be written at address FE07-0010H, and so forth. Moreover, the V831 CPU contains cache memory, so that the cache must be cleared after the vector is rewritten. Otherwise, an instruction may be executed before rewriting.

A sample program for alternate vector rewriting is given below (when the relative address from the interrupt handling routine to an alternate vector area is within 26 bits).

```

#define VECT_CPU 0xfffffe00                /* Start of CPU interrupt vector */
#define VECT_RAM 0xfe070000              /* Start of alternate interrupt vector */
#define VECT(n) ((VECT_CPU - n) + VECT_RAM) /* Find interrupt vector address */

main()
{
    extern void __interrupt IntEntry();    /* Interrupt handling routine */
    int addr, offs, inst;

    /* Allocation of alternate vector address for 0xfffffe30 of CPU vector,
       and creation of JR dest26 instruction for branching to interrupt handling routine */
    addr = VECT(0xfffffe30);
    offs = (int)IntEntry - addr;
    inst = 0xa8000000 | (offs & 0x3fffffe); /* 32-bit instruction JR dest26 */

    /* Vector replacement */
    di(); /* Interrupt disable __asm("di"); */
    *((unsigned short *)(addr + 0)) = (inst >> 16) & 0xffff; /* Higher 16-bit code */
    *((unsigned short *)(addr + 2)) = (inst & 0xffff); /* Lower 16-bit code */
    outw(0xFFFFFFFF4, 3); /* Clear cache */

    /* Interrupt device initialization, etc. */
    .....
    ei(); /* Interrupt enable __asm("ei"); */
    .....
}

```

## 12.2. INTERNAL INSTRUCTION RAM

With the V831, a maskable interrupt vector can be placed in the internal RAM by a CPU function (IHA bit of the HCCW system register). When this function is used, the vector setting requirement does not differ between an ordinary V831 program and a program using the monitor on the RTE-V831-PC.

For an explanation of the method of vector usage with the internal instruction RAM and that of modifying the contents of the internal instruction RAM, refer to the manual provided with the CPU.

When a program is placed in the internal instruction RAM for purposes including interrupt handling, the user is required to pay careful attention to the compiled object codes. For the switch-case statement of C, in particular, a jump table is created in the instruction code, and a code for causing a branch by referencing the table may be generated. Such a reference to the table is made with the LD instruction. However, the internal instruction RAM cannot be referenced by the LD instruction, so that the program may perform an unpredictable operation.

## 12.3. GENERAL RESTRICTIONS/NOTES

This section describes restrictions and notes relating to the debugging of an application using a maskable interrupt.

- 1) If an interrupt is generated before alternate vector setting, or if an interrupt is generated with other than a valid alternate vector set, a break occurs at the point where the interrupt is generated. This is because the initial value of the alternate vector is an instruction for causing a branch to the break handling routine of the monitor ROM.
- 2) If the relative address from an alternate vector area to the interrupt handling routine exceeds 26 bits, the contents of at least one register must be destroyed, or a branch relay point must be created, to cause a branch to the interrupt handling routine.
- 3) An alternate vector area is protected as a ROM monitor management area, so that the area cannot be rewritten by downloading a program. The user may consider the use of the following method; that is, a vector area is defined as an independent section in the source program, and such an area is assigned as an alternate vector area by a link-time parameter. This method cannot be used, however, because downloading will invariably fail.
- 4) Immediately after rewriting an alternate vector area, always flush the cache memory in the CPU. Otherwise, a vector existing before alternate vector rewriting may be used.
- 5) All peripherals, including interrupt-related peripherals, can be initialized only with the reset switch on the board. This means that if, after a program is executed, another program is loaded, the peripherals will still be in the statuses set by the previous program. So, use the procedure below when, after executing a program that uses a peripheral, another program is to be loaded and executed.
  - (1) For resetting, press the reset switch of the RTE-V831-PC.
  - (2) Load and execute another program.
- 6) Before setting the EI (interrupt enable) state, set the DI (interrupt disable) state at the start of program execution, then set the peripherals and vectors.

**12.4. RESTRICTIONS ON BREAKS IN THE INTERRUPT HANDLING ROUTINE**

The following restrictions are imposed on breaks in the interrupt handling routine:

- 1) During a break, all maskable interrupts are rejected.
- 2) The single step function sets a temporary breakpoint in the next instruction. So, when a user program placed in the EI (interrupt enable) state is subject to single stepping, an interrupt is accepted even during single stepping.
- 3) After a break in the interrupt handling routine, exiting from the interrupt handling routine by single stepping is impossible. (Specifically, single stepping based on the last "}" of the interrupt handling routine is disabled.) Similarly, IRET instruction single stepping is impossible.
- 4) The Return function of the debugger does not support return from an interrupt handling routine to the original routine.

### 13. APPENDIX A MULTI MONITOR

This chapter describes how to make the settings required to establish a connection between the Multi monitor stored in ROM and the Multi debugger on the host. It also provides notes on the use of the Multi monitor.

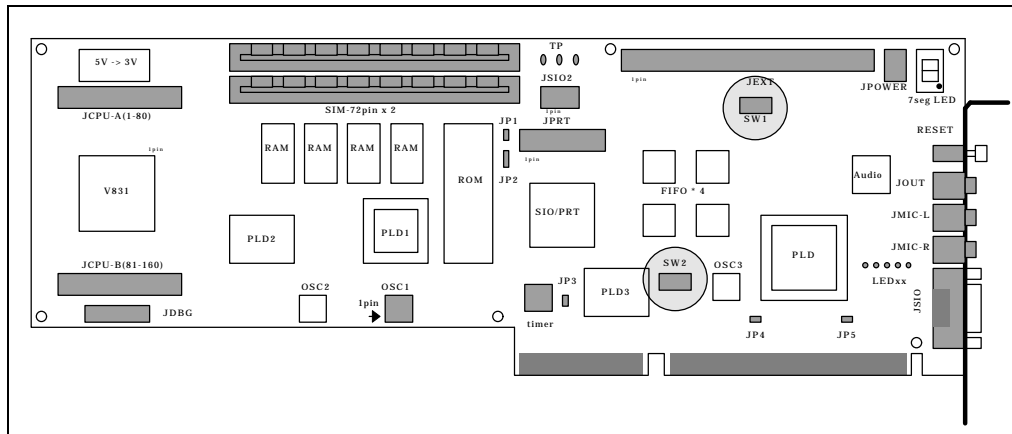
#### 13.1. BOARD SETTING

##### 13.1.1. RTE for Win 32 installation

When the board is used with the Multi debugger, communication software called RTE for Win32 must be installed in the PC. Refer to the RTE for Win32 Installation Manual (supplied with this product) for installation and test methods.

#### 13.2. SWITCH SETTING

The RTE-V831-PC board has two DIP switches. The DIP switches can be used to set up the evaluation board. The switch layout is shown below.



Location of the Switches on the RTE-V831-PC Board (SW1, SW2)

##### 13.2.1. SW1 Setting

SW1 is a switch for general-purpose input ports. For the Multi monitor in the factory-installed ROM, SW1 is used as shown below.

SW1	1	2	Baud rate
Setting	ON	ON	Not used
	OFF	ON	38400 baud
	ON	OFF	19200 baud
	OFF	OFF	9600 baud

(Factory-set)

Baud Rate Setting

SW1	3	4	Profiler period
Setting	ON	ON	Timer is not used.
	OFF	ON	200 Hz    5 ms
	ON	OFF	100 Hz    10 ms
	OFF	OFF	60 Hz    16.67 ms

(Factory-set)

Profiler Period Setting

SW1	5	Debug mode
Setting	ON	7-segment LED used by the monitor
	OFF	Normal use state (Factory-set)

Debug Mode Setting

SW1	6	Break interrupt
Setting	ON	INTP3 used (Factory-set)
	OFF	NMI used

Break Interrupt Setting

SW1-7 is not used with the Multi monitor.

SW1	8	Boot-time bus size (BT16B) and ROM
Setting	ON	Boot on 32-bit bus (BT16B = 0) from flash ROM
	OFF	Boot on 16-bit bus (BT16B = 1) from EPROM (Factory-set)

Boot-Time Bus Size and ROM Setting

### 13.2.2. SW2 Setting

SW2 is a switch for selecting the I/O address of the ISA bus. Switches 1 to 8 correspond to ISA addresses A4 to A11, respectively (A12 to A15 are fixed at 0). This means that the I/O address that can be selected ranges between 000xH and 03FxFH. When a switch is set to OFF, it corresponds to 1. When a switch is set to ON, it corresponds to 0. Generally, SW2 is set to any value between 20xH and 3FxFH.

SW2	1	2	3	4	5	6	7	8	
Address	A4	A5	A6	A7	A8	A9	A10	A11	I/O address
ON/ OFF	0	0	0	0	0	1	0	0	020xH (Factory-set)

SW2-to-ISA I/O Address Correspondence

### 13.3. MULTI MONITOR

The ROM chip on the board is incorporated with the Multi monitor. The following cautions should be observed when the board is connected to the Multi debugger as the host.

#### 13.3.1. Monitor Work RAM

The monitor uses the first 64K bytes area in the SRAM as work RAM. In other words, user programs are not allowed to use logical addresses FE07-0000H to FE07-FFFFH.

#### 13.3.2. Interrupt

When using an interrupt with a user program, see Chapter 12.

#### 13.3.3. Interrupt for Forced Break

NMI or INTP03 can be selected as an interrupt to be used by the monitor for a forced break (the Halt button for the Multi debugger) or a timer (such as profiler). INTP03 is to be selected when a program using the DMA is used (for audio input/output, for example). In other cases, the use of NMI is recommended. The differences between the two interrupts are listed below.

##### When NMI is used

A break can occur at any time during user program execution. If, however, a break occurs during DMA operation, the DMA is also terminated. (The DMA does not restart even upon reexecution.)

##### When INTP03 is used

No break will occur if the user program uses an interrupt with a higher priority than INTP03, or if interrupts are disabled. DMA operation is continued even during a break.

#### 13.3.4. \_INIT\_SP Setting

\_INIT\_SP (stack pointer initial value) is set to FE06-FFFCH (highest SRAM address) by the monitor. (\_INIT\_SP can be changed in the Multi debugger environment.)

#### 13.3.5. Remote Connection

Either serial or ISA bus connection can be selected to connect the monitor with the Multi debugger. To switch from serial connection to ISA bus connection or vice versa, it is necessary to reset the monitor (by pressing the reset switch on the rear panel) and run the Check RTE32.exe utility of RTE for Win32.

#### 13.3.6. Monitor Execution Area

The 128K-byte ROM contains two codes; the lower 64K-byte area contains the code to be executed in the cacheable area, while the higher 64K-byte area contains the code to be executed in the uncacheable area.

To enable a program to be downloaded at higher speed, select the cacheable area.

Note that monitor intervention (including profile timer operation) temporarily flushes the cache used during user program execution, so that data obtained from profile measurements may have a larger error. To prevent this from occurring, select the uncacheable area.

The use of the uncacheable area is factory-set (with 1-2 of JP2 closed).

### 13.4. RTE COMMANDS

When the monitor and server are connected, the TARGET window is opened. The RTE commands can be issued in this window. The following table lists the RTE commands.

Command	Description
HELP, ?	Displays help messages.
INIT	Initializes.
VER	Displays the version number.
INB, INH, or INW	I/O read
OUTB, OUTH, OUTW	I/O write
DCTR, INTR, PLLCR, CMCR	Changes or displays the internal registers.
SFR	Displays or sets the internal I/O.

RTE Commands

Some commands require parameters. All numeric parameters such as addresses and data are assumed to be hexadecimal numbers. The following numeric representations are invalid:

0x1234 1234H \$1234

#### 13.4.1. HELP(?)

<Format> HELP [command-name]

Displays a list of RTE commands and their formats. A question mark (?) can also be used in place of the character string HELP. If no command name is specified in the parameter part, the HELP command lists all usable commands.

<Example> HELP SFR

Displays help messages for the SFR command.

#### 13.4.2. INIT

<Format> INIT

Initializes the RTE environment. Usually, this command should not be used.

#### 13.4.3. VER

<Format> VER

Displays the version number of the current RTE environment.

#### 13.4.4. INB, INH, INW

<Format> INB [address]  
INH [address]  
INW [address]

Reads from an I/O register. The INB, INH, and INW commands access in byte, halfword, and word units, respectively. If an address is omitted, the previous address is assumed.

<Example> INB 1000

Reads a byte from an I/O register at 1000H.

**13.4.5. OUTB, OUTH, OUTW**

<Format>   OUTB [[address] data]  
              OUTH [[address] data]  
              OUTW [[address] data]

Writes to an I/O register. The OUTB, OUTH, and OUTW commands access in byte, halfword, and word units, respectively. If an address or data is omitted, the previous address or data is assumed.

<Example>   OUTH 2000 55AA

Writes the halfword 55AAH to the I/O register at 2000H.

**13.4.6. DCTR Command**

<Format>   DCTR [ALL]

Displays the contents of DCTR registers. There are 256 DCTR registers. Among these 256 registers, the contents of the registers whose valid bit is on are displayed except when ALL is specified. If ALL is specified, the contents of all DCTR registers are displayed. The DCTR registers are mapped on the I/O space f2000000h-f2000fffh.

**13.4.7. ICTR Command**

<Format>   ICTR [ALL]

Displays the contents of ICTR registers. There are 128 ICTR registers. Among these 128 registers, the contents of the registers whose valid bit is on are displayed except when ALL is specified. If ALL is specified, the contents of all ICTR registers are displayed. The ICTR registers are mapped on the I/O space fa000000h-fa000fffh.

**13.4.8. CMCR Command**

<Format>   CMCR [=] VALUE

Specifies a value in the cache memory control register (CMCR).

**13.4.9. SFR Command**

<Format>   SFR [register-name [=data]]

When a register name is specified with data omitted, the data read from the register is displayed. When a register name is specified, and data is specified after =, the data is written to the register. The size of data is automatically determined according to the valid size of the specified register. For details of the internal I/O registers, refer to the manual provided with the V831 CPU.

<Example 1>  SFR

A list of registers is displayed.

<Example 2>  SFR IMR

The contents of the IMR register are displayed.

<Example 3>  SFR IMR=55AA

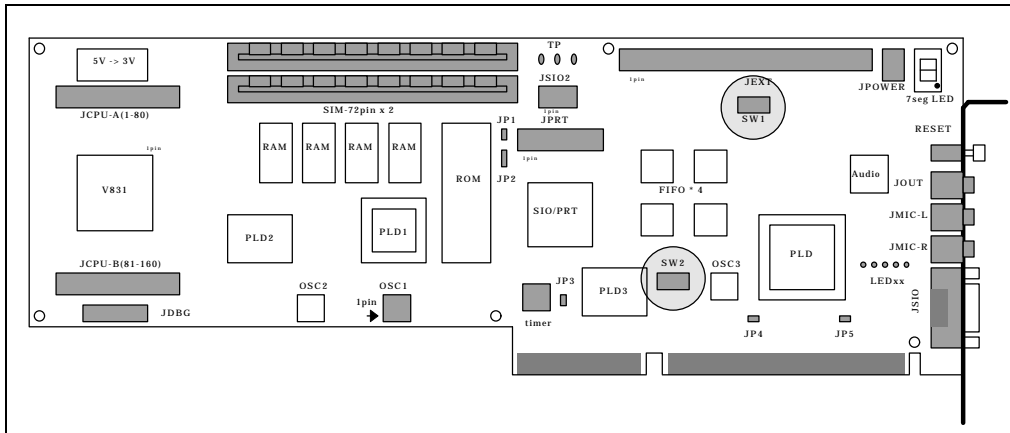
Data 55AAH is written into the IMR register.

**14. APPENDIX B PARTNER MONITOR**

This chapter describes how to make the settings required to establish a connection between the PARTNER monitor stored in ROM and the debugger on the host. It also provides notes on the use of the PARTNER monitor.

**14.1. SWITCH SETTING**

The RTE-V831-PC board has two DIP switches. The DIP switches can be used to set up the evaluation board. The switch layout is shown below.



Location of the Switches on the RTE-V831-PC Board (SW1, SW2)

**14.1.1. SW1 Setting**

SW1 is a switch for general-purpose input ports. For the PARTNER monitor in the factory-installed ROM, SW1 is used as shown below.

SW1	1	2	Baud rate
Setting	ON	ON	115200 baud
	OFF	ON	38400 baud
	ON	OFF	19200 baud
	OFF	OFF	9600 baud (Factory-set)

Baud Rate Setting

SW1	3	4	Profiler period
Setting	ON	ON	Timer is not used. (Factory-set)

Profiler Period Setting

SW1	5	Debug mode
Setting	ON	7-segment LED used by the monitor
	OFF	Normal use state (Factory-set)

Debug Mode Setting

SW1	6	Break interrupt	
Setting	ON	INTP3 used	(Factory-set)
	OFF	NMI used	

Break Interrupt Setting

SW1-7 is not used with the monitor.

SW1	8	Boot-time bus size (BT16B) and ROM	
Setting	ON	Boot on 32-bit bus (BT16B = 0) from flash ROM	
	OFF	Boot on 16-bit bus (BT16B = 1) from EPROM	(Factory-set)

Boot-Time Bus Size and ROM Setting

#### 14.1.2. SW2 Setting

SW2 is a switch for selecting the I/O address of the ISA bus. Switches 1 to 8 correspond to ISA addresses A4 to A11, respectively (A12 to A15 are fixed at 0). This means that the I/O address that can be selected ranges between 000xH and 03FxFH. When a switch is set to OFF, it corresponds to 1. When a switch is set to ON, it corresponds to 0. Generally, SW2 is set to any value between 20xH and 3FxFH.

SW2	1	2	3	4	5	6	7	8	
Address	A4	A5	A6	A7	A8	A9	A10	A11	I/O address
ON/ OFF	0	0	0	0	0	1	0	0	020xH (Factory-set)

SW2-to-ISA I/O Address Correspondence

## 14.2. PARTNER MONITOR

### 14.2.1. Monitor Work RAM

The monitor uses the first 64K bytes area in the SRAM as work RAM. In other words, user programs are not allowed to use logical addresses FE07-0000H to FE07-FFFFH.

### 14.2.2. Interrupt

When using an interrupt with a user program, see Chapter 12.

### 14.2.3. Interrupt for Forced Break

NMI or INTP03 can be selected as an interrupt to be used by the monitor for communications and a forced break (the ESC button). INTP03 is to be selected when a program using the DMA is used (for audio input/output, for example). In other cases, the use of NMI is recommended. The differences between the two interrupts are listed below.

#### When NMI is used

A break can occur at any time during user program execution. If, however, a break occurs during DMA operation, the DMA is also terminated. (The DMA does not restart even upon reexecution.)

#### When INTP03 is used

No break will occur if the user program uses an interrupt with a higher priority than INTP03, or if interrupts are disabled. DMA operation is continued even during a break.

### 14.2.4. SP Setting

The stack pointer initial value is set to FE06-FFFCH (highest SRAM address) by the monitor.

### 14.2.5. Remote Connection

Either serial or ISA bus connection can be selected to connect the monitor with the PARTNER debugger. Use RTSETUP.exe to make the necessary settings. For details, refer to the PARTNER manual.

### 14.2.6. Monitor Execution Area

The 128K-byte ROM contains two codes; the lower 64K-byte area contains the code to be executed in the cacheable area, while the higher 64K-byte area contains the code to be executed in the uncacheable area.

If the monitor is executed in the cacheable area, the monitor execution speed is faster. So, the program is downloaded at a higher speed. Normally, use the cacheable area (the area is factory-set).

The use of the cacheable area is factory-set (with 2-3 of JP2 closed).

- Memo -

RTE-V831-PC User' s Manual

M6A1MNL01

*Midas lab*