RTE-V831-PC

ユーザーズ・マニュアル (Rev. 2.00)

Midas lab

改訂履歴

実施日	Revision	章	内容
1997 年 3 月 18 日	1.00		初版
1997 年 4 月 4 日	1.01	5.19	誤記訂正 JCPU-A 74pin -> DMARQ1
1997 年 4 月 14 日	1.02	8.10 5.19	MCLKDIV レジスタ表の修正 誤記訂正 JCPU-A 76pin -> DMARQ3
1998 年 4 月 28 日	2.00	 5.19 8.3 8.9	PARTNER 用モニタ使用時の説明を追加 合わせて、以下の誤記訂正 JCPU-A 26pin(+3..3V -> +3.3V),98pin(+5V->GND) 表中の論理アドレス(BC00-0000 -> 4500-2000H) 割り込み要因表中の要求レベル

目次

1. はじめに

「RTE-V831-PC」は、NEC 製の RISC プロセッサ V831 の評価を目的とした PC/AT ISA バス・タイプの評価ボードです。ボードは、最高 100MHz で動作する V831 とメモリ、シリアル・パラレルインターフェースやオーディオ入出力などの I/O で構成されます。メモリとのインターフェースは、V831 が内蔵するメモリコントローラを使用して行います。

これらの機能を利用して、プロセッサの性能評価、デモ、シミュレータの実行エンジン、アプリケーション・プログラムの初期段階の開発など、幅広くご利用いただけます。

本製品は、開発用のソフトウェアツールとして、GHS 社の Multi と自社製の PARTNER のどちらかかをソースレベルデバッガとしてご使用になれます。ご使用になるデバッガによって、ROM に搭載するモニタは異なります。

ROM は、購入時にご指定頂いたモニタが搭載されています。デバッガを同時に購入されていない場合は、それぞれ別売りされていますので、別途お買い求め下さい。

1.1. マニュアル表記について

本書では、数字の表記については表の表記を用います。16 進数や 2 進数の表記では、桁数が多くて読みにくい場合は、4 桁ごとに"-"（ハイフン）を入れてあります。

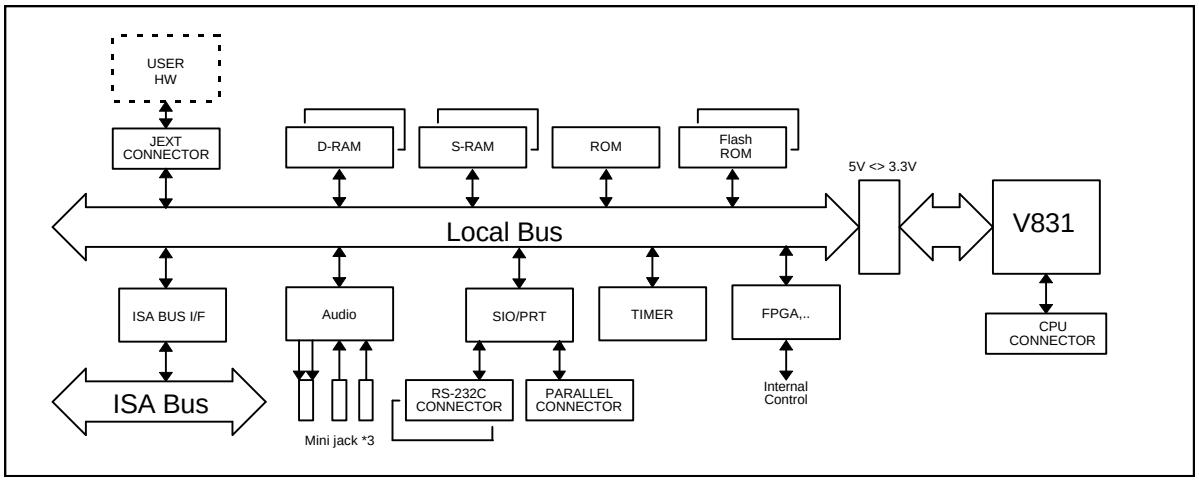
進数	表記規則	例
10 進数	数字のみを示します	"10"は 10 進数の"10"を示します
16 進数	数字の末尾に"H"を記します	"10H"は 10 進数の"16"を示します
2 進数	数字の末尾に"B"を記します	"10B"は 10 進数の"2"を示します

数字表記規則

MULTI は米国 Green Hills Software, Inc の商標です。

2. 機能

RTE-V831-PC の機能ブロックの概要を図に示します。



RTE-V831-PC ブロック図

3. 主な特徴

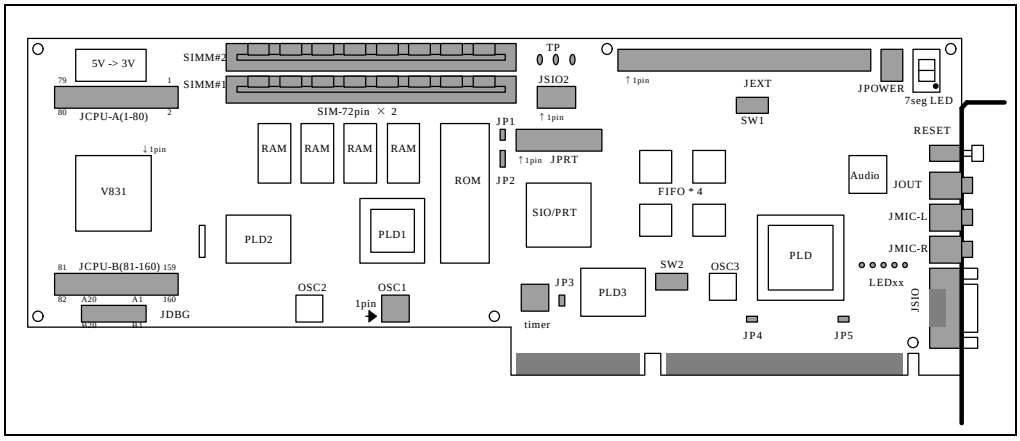
- GreenHills 社の MULTI と自社製 PARTNER 用のモニタ ROM を２種類用意しています。
- MULTI または PARTNER を使用した高級言語レベルでのリアルタイム実行・評価が可能です。
- ROM エミュレータが接続できます。
- 高速 SRAM を 512KB 標準搭載し、DRAM は SIMM により，16MB まで搭載できます。
- シリアル(2ch)&プリンタ(1ch)のインタフェースを用意しています。
- タイマー2ch を搭載しています。（MULTI モニタで 1ch 使用）。
- 音声の入出力チャンネルをそれぞれ 2ch 搭載しています。

4. 基本仕様

プロセッサ	V831	
CPU **クロック**	100MHz (max)	
バスクロック	33MHz (max)	
消費電力	+5V (2.0A)	
メモリ		
EPROM	128KB	64K×16bit (40pin-DIP)×1(max.512KB)
FlashROM	8MB	2M×8bit×4 (MBM29F016-120)
SRAM	512KB	128K×8bit×4
DRAM	8MB	EDO-SIMM-72pin (SIMM*2 **で** max16MB)
I/O		
シリアル(2ch)	NS16550 **相当**	10 **ピンヘッダ**，DB9 **コネクタ**
プリンタ	PS2 **互換**	26 **ピンヘッダ**
音声入出力(2ch)	uPD66310	**ﾐﾆｼﾞｬｯｸ**(MIC * 2,LINEOUT* 1)
タイマ	i8254 **相当**	**分解能** 500nS
IO **ポート**	LED(7seg)**表示／スイッチ入力**	
その他		
CPU **コネクタ**	V831 **の全機能ピンを接続したコネクタ**	
標準外部拡張コネクタ	RTE-PC **標準** 16bit I/F(1MB,16bit **バス**)	
リセット・スイッチ	Push **式**	

5. ボードの構成

下図は RTE-V831-PC **ボード上の主要な部品の物理的な配置です。ここでは、それぞれの部品について説明します。**



RTE-V831-PC **の部品配置図**

5.1. リセット・スイッチ（RESET）

RESET **は本ボード全体のリセット・スイッチです。このスイッチを押すと** CPU **を含む全ての回路がリセットされます。**

5.2. 電源ジャック（JPOWER）

本ボードを ISA **バス・スロットに挿さずに単体で使用する場合に、**JPOWER **コネクタに外部電源**

を接続して電源を供給します。

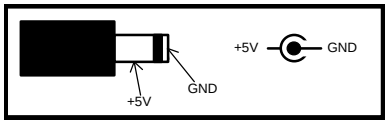
JPOWER ジャックに供給する電源は、下記の通りです。

電圧：５Ｖ

電流：最大２A（ただし、JEXT コネクタへの供給分を含まず）

適合コネクタ：Type A（Φ5.5mm）

極性：



【注意】外部から電源を給電する場合は、電源プラグの極性に注意してください。また、ISA バス・スロットに挿して使用する場合には、JPOWER に電源を接続しないでください。故障の原因になります。

5.3. スイッチ1（SW1）

SW1 は、汎用の入力ポートのスイッチです。モニタを使用する場合には、SW1-7 を除き、割り当て済みです。詳細は、それぞれのモニタの章（13.2.1 SW1 の設定 または、14.1.1 SW1 の設定）を参照してください。尚、ポートからの読み出し時、スイッチは、OFF で 1、ON で 0 の値になります。詳細は、「8.2 DIPSW1 読み出しポート(4500-1000H [Read Only])」を参照してください。

5.4. スイッチ2（SW2）

SW2 は、ISA バスの I/O アドレスを選択するスイッチです。スイッチの番号 1～8 が ISA バスのアドレス A4～A11 に対応しています（A12～A15 は０固定）。したがって、I/O アドレスとして 00xH～03FxH が選択できます。スイッチは、OFF で 1、ON で 0 の値となります。

SW2 番号	1	2	3	4	5	6	7	8
アドレス	A4	A5	A6	A7	A8	A9	A10	A11

SW2 のアドレス対応

5.5. LED

LED は、各種ステータスを示しています。表に内容を示します。

名称	内容
POWER	ボードに電源が供給されている時に点灯
PLY	音声を出力中緑色に点灯 音声を出力中にエラーが発生した時に赤色に点灯
REC	音声を録音中緑色に点灯 音声を録音中にエラーが発生した時に赤色に点灯
TOVER	タイムオーバー発生時に点灯
FLBUSY	フラッシュ ROM がビシー中（書き込み、イレーズ中等）点灯

LED ステータス

5.6. ROM エミュレータ用テストピン（TP）

TP は、ROM エミュレータを接続する際に使用するテストピンです。下記の制御信号が入力できます。表に信号名と機能を示します。

信号名	入出力	機能
RESET-	入力	Low レベル入力により、CPU がリセットされます。 ROM エミュレータからのリセット要求信号を接続します。 1KΩでプルアップされています。
NMI-	入力	Low レベル入力により、CPU に NMI が入ります。ただし、ソフトウェアによってマスクできますので、解除する必要があります。（「8.9 割り込みコントローラ（PIC）(4500-D000H～4500-D018H)」参照下さい。） ROM エミュレータからの NMI 要求信号を接続します。 1KΩでプルアップされています。
GND	ーーー	GND。ROM エミュレータの GND と接続します。

TP 端子の機能

5.7. クロック・ソケット（OSC1）

OSC1 ソケットには、CPU に供給するクロックのためにオシレータを実装します。V831 では、システム・クロックの生成に PLL を使用しています。OSC1 ソケットへは、V831 内部の動作周波数の３分の１の周波数のオシレータを実装してください。（出荷時は、33.33MHz を実装）

OSC1 ソケットには、DIP8 ピンタイプ（ハーフタイプ）のオシレータを実装してください。

【注意】オシレータの足を切って実装する場合、足が短かすぎるとフレーム（外装）部分が、ソケットの端子とショートしてしまいますのでご注意願います。

5.8. 72pin-SIMM ソケット（SIMM#1,SIMM2#）

SIMM#1、#2 ソケットには、それぞれ、8M バイトの EDO タイプの DRAM-SIMM が実装でき、合計で 16M バイトまで実装できます。１枚のみ実装する場合は SIMM#1 ソケットに実装してください。

出荷時、SIMM#1 ソケットには EDO タイプの DRAM-SIMM(8M バイト)が実装されています。SIMM#2 に追加する場合は、同じ仕様のものをご使用ください。

【注意】SIMM#1,#2 ソケットに実装できる DRAM は、EDO-8M バイトの 72 ピン SIMM（DOS/V 機用）のみです。8M バイトを超える容量の SIMM は実装しないでください。

5.9. ROM ソケット

ROM ソケットには、標準で 128K バイト（64K×16 ビット）の 40 ピン ROM が実装されています。変更する場合は、27C1024,27C2048,27C4096 タイプで、アクセス・タイムが 150ns 以下のものをご使用ください。(使用する ROM の種類や使い方によって、JP1,JP2 の設定が必要です)

5.10. ROM 容量切り替えジャンパ(JP1)

JP1 は実装する ROM の容量によって切り替えるジャンパです。128K バイト（64K×16 ビット）と 256K バイト（128K×16 ビット）の ROM を実装する場合はオープンにしておきます。512K バイト（256K×16 ビット）の ROM を実装する場合はショートします。

【備考】出荷時の設定は、オープンです。

5.11. ROM の分割切り替えジャンパ(JP2)

128K バイトの ROM を実装時に連続した 128K バイトとして使用するか、64K バイトに分割して使用するか切り替えるためのジャンパです。

搭載している ROM は、128K バイトの前半にキャッシュ領域で実行するコードが、後半にアンキャッシュ領域で実行するコードの二つが入ってます。

[1-2] -3 :ブート時 ROM は連続した 128k バイト領域として見えます。

.....モニタはアンキャッシュ領域で実行します。

1-[2-3] :ブート時 ROM の下位 64Kバイトのみが見えます。

.....モニタはキャッシュ領域で実行します。

【備考】出荷時の設定は、モニタによって異なります。それぞれの章（13.3.6 モニタの実行領域、または、14.2.6 モニタの実行領域）を参照下さい。

5.12. JP3,JP4,JP5

出荷時の状態でご使用ください。

JP3 :オープン
JP4 :ショート
JP5 :ショート

5.13. ユーザ用タイマ基本クロックの切り替えジャンパ(JP6)

アプリケーションで使用可能なタイマ(CH#1,CH#2)へ供給するクロックの切り替えに使用します。

[1 - 2] :2MHz（出荷時の設定）

[3 - 4] :4MHz

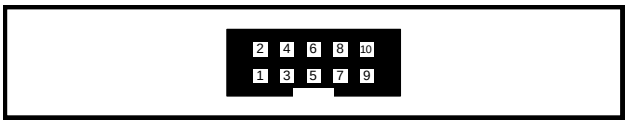
5.14. シリアル・コネクタ（JSIO1,JSIO2）

JSIO1,JSIO2 コネクタは、シリアル・コントローラ（TL16C552A）によって制御される RS-232C インターフェース用のコネクタです。コネクタの形状は、JSIO1 は PC/AT で用いられる一般的な D-SUB9 ピンの RS-232C コネクタ、JSIO2 は 2.54mm ピッチのピンプラグ型のコネクタです。何れも、全ての信号は RS-232C レベルに変換されています。コネクタのピン番号と内容は図と表の通りです。

表には、ホストと接続する場合の接続信号について、ホスト側が D-SUB9 ピンの場合と D-SUB25 ピンの場合の布線をそれぞれ示してあります（一般的なクロスケーブルの布線です）。

また JSIO2 のピン配置は、リボンケーブルに対して圧接型コネクタを使用した場合、JSIO1 のピン配置と同じになるようになっています。

JSIO1 ピン配置（オス）



JSIO2 ピン配置

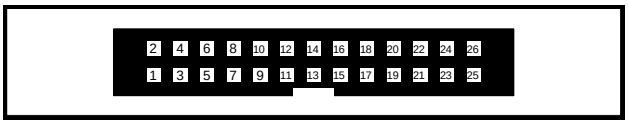
JSIO1 ピン番号	JSIO2 ピン番号	信号名	入出力	ホストの接続ピン番号	
				D-SUB9	D-SUB25
1	1	DCD	入力		
2	3	RxD(RD)	入力	3	2
3	5	TxD(SD)	出力	2	3
4	7	DTR(DR)	出力	1, 6	6, 8
5	9	GND		5	7
6	2	DSR(ER)	入力	4	20
7	4	RTS(RS)	出力	8	5
8	6	CTS(CS)	入力	7	4
9	8	RI	入力		
--	10	NC			

JSIO1,2 コネクタ信号

5.15.パラレル・コネクタ（JPRT）

JPRT コネクタは、パラレル（プリンタ）・コントローラ（TL16C552A）によって制御されるパラレル通信用のコネクタです。コネクタの形状は、2.54mm ピッチのピンプラグ型のコネクタです。何れも、全ての信号は 5V レベルです。コネクタのピン番号と内容は図と表の通りです。

また JPRT のピン配置は、リボンケーブルに対して圧接型コネクタを使用した場合、PC/AT で用いられている一般的な D-SUB25 ピンのピン配置と同じになっています。



JPRT ピン配置

JPRT ピン番号	信号名	JPRT ピン番号	信号名
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 コネクタ信号

5.16. Audio 用ミニジャック(JMIC-R,JMIC-L,JOUT)

Audio 用のジャックとして、モノラルのマイク入力が 2ch、ステレオの出力が 1ch 用意されています。それぞれの入出力条件は以下の通りです。

JMIC-R,JMIC-L

電気的な入力条件

140mVp-p （内部アンプ増幅 約 20db）

適合する物理的なプラグの形状

モノラル・ミニプラグ（φ3.5 DIA） × 2ch

JOUT

電気的な出力条件

1.4Vp-p

適合する物理的なプラグの形状

ステレオ・ミニプラグ（φ3.5 DIA） × 1ch

5.17. デバッグ用コネクタ(JDBG)

V831 に内蔵しているデバッグ機能を利用した、デバッグ・ツールを接続するためのコネクタです。

基板側のコネクタ　：KEL 社　8930E-040-178MS

ピン番号	信号名	ピン番号	信号名
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	TRCDAT3	B10	GND
A11	GND	B11	GND
A12	DDI	B12	GND
A13	GND	B13	GND
A14	DCK	B14	GND
A15	GND	B!5	GND
A16	DMS	B16	GND
A17	GND	B17	GND
A18	DDO	B18	GND
A19	GND	B19	GND
A20	DRST-	B20	GND

JDBG コネクタ信号

5.18. 拡張バス・コネクタ［(JEXT)

JEXT コネクタは、メモリや I/O などを拡張できるように用意されたコネクタです。このコネクタには、本ボードの内部のローカル・バスが接続されています。バスの詳細仕様は、「10 EXT バス仕様」を参照ください。

5.19. CPU コネクタ(JCPU-A,JCPU-B)

CPU コネクタの信号は、V831 と直結した信号です。多くの信号は、ボード内部で使用していますので、JCPU から信号を引き出す場合は、注意が必要です。尚、信号のレベルは、3.3V です。

ピン番号	信号名	ピン番号	信号名
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 コネクタ信号

*1:JCX2 は、バッファされた信号です。

ピン番号	信号名	ピン番号	信号名
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	HLDAK-	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-B **コネクタ信号**

使用しているコネクタは、ヒロセ電機製の FX2-80P-1.27SV **です。**

6. ホストPC との接続

6.1. ISA バスに実装する場合

ボードを PC の ISA バス・スロットに実装することにより、、ISA バスからボードへ電源（+5V）が供給されます。デバッガとの通信に ISA バス経由が使用できるため、プログラムの高速なダウンロードが実現できます。

ISA バス・スロットへの実装手順は以下の通りです。

① ボード上のディップ・スイッチにより、PC の I/O アドレスを設定します。I/O アドレスは他の I/O と重ならないように注意してください。スイッチの設定については「5.4 スイッチ 2（SW2）」を参照ください。

② PC の電源を切って筐体をあけ、ボードを実装する ISA バス・スロットを確認します。実装するスロットにリアパネルが付いている場合は、そのリアパネルを外します。

③ ボードを ISA バス・スロットに差し込み、ボードが隣接のボードなどと接触していないかを確認し、ボードに付いているリアパネルを PC の筐体にネジで固定します。

④ PC の電源を入れ、ボードの POWER-LED が点灯することを確認します。**LED が点灯しない場合は、すぐにPC の電源を切り接続を確認してください。**システムが正常に立ち上がらない（デバイス・ドライバの組み込みでエラーが発生するなど）場合は、設定した I/O アドレスが他の I/O と重なっている可能性があります。PC のマニュアルや他に実装されているボードのマニュアルなどを参照して、ボードの I/O アドレスを再確認してください。

⑤ システムが正常と判断できたら、再度 PC の電源を切ってから筐体を元に戻します。

6.2. ボード単体で使用する場合

PC に組み込まず、ボード単体で使用する場合は、外部からの電源供給が必要となります。デバッガとの通信は RS-232C 経由で行います。

ボードを単体で使用する場合の手順は以下の通りです。

① ホストと接続するための RS-232C ケーブルと、電源供給のための外部電源（+5V 2A）を用意してください。特に電源については、電圧とコネクタの極性に注意してください。また、ボードの４隅にスペーサを取り付けるなど、設置場所にも問題がないようにしてください。RS-232C ケーブルの結線は「5.14 シリアル・コネクタ（JSIO1,JSIO2）」、電源コネクタについては、「5.2 電源ジャック（JPOWER）」を参照してください。

② ボード上のディップ・スイッチで、RS-232C のボーレートを設定します。スイッチの設定については、使用するモニタの各章「13.2.1 SW1 の設定、または、14.1.1 SW1 の設定、」を参照ください。

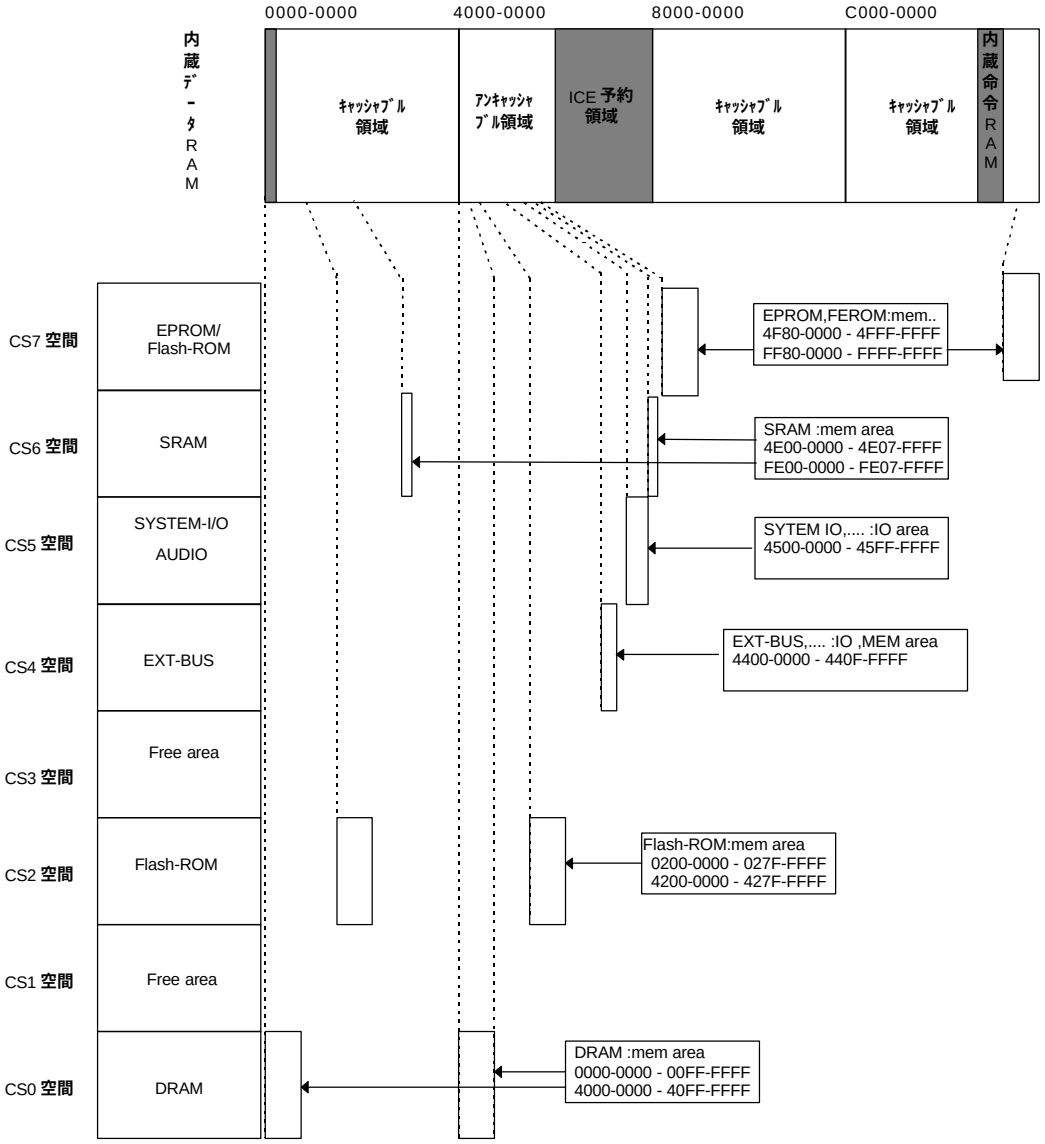
③ ホストと RS-232C ケーブルで接続して、JPOWER ジャックに電源を接続し、、ボードの POWER-LED が点灯することを確認します。**LED が点灯しない場合は、すぐに電源を切り接続を確認してください。**

7. ハードウェア・リファレンス

ここでは、RTE-V831-PC ボードのハードウェアについて記述します。

7.1. メモリ・I/O のマップ

ボードのメモリと I/O の割り付けは、以下の通りです。



メモリ・I/O マップ

【注意】CS2 と CS7 領域に存在する Flash ROM は、物理的に同じものです。Flash ROM を CS7 の領域にマップする場合には、予めリセットベクタから開始するブートプログラムを書き込んでおくことが必要です。

7.2. マップ詳細

マップの詳細を以下に示します。

CS0 空間 (x000-000～x0FF-FFFF,x800-000～x8FF-FFFF)

SIMM#1,#2 ソケットに実装される EDO-DRAM 用の空間で、16M バイトの容量と 32 ビットデータバス幅を持ちます。SIMM#1 に前半の 8M バイトが、SIMM#2 に後半の 8M バイトがマップされます。その他の領域には、16M バイトおきにイメージが現れます。

DRAM とのインターフェースは、V831 内蔵の DRAM コントローラを使用しています。

CS1 空間 (x100-000～x1FF-FFFF,x900-000～x9FF-FFFF)

未使用の領域です。

CS2 空間 (x200-000～x27F-FFFF,xA00-000～xA7F-FFFF)

Flash ROM（富士通製の MBM29F016PFTN-120：2M * 8bit,120nS）用の空間で、8M バイトの容量と 32 ビットデータ幅を持ちます。8M バイトおきに Flash ROM のイメージが現れます。

ウェイト制御は外部の H/W で行なっていて、4 クロック分のウェイトが常に入ります。

CS3 空間 (x300-000～x3FF-FFFF,xB00-000～xBFF-FFFF)

未使用の領域です。

CS4 空間 (x400-000～x40F-FFFF,xC00-000～xC0F-FFFF)

1M バイトの EXT バスの領域です。V831 内部のバスコントローラにより、メモリ空間か I/O 空間として割り付けます。1M バイトおきに EXT バス空間のイメージが現れます。

ウェイト制御は外部の H/W で行なっていて、２クロック分のウェイトが常に入った後、EXT バスからのウェイト（レディ）制御が働きます。

使用するにあたっては、「10.3 EXT バス使用時の注意事項」を参照して下さい。

CS5 空間 (x500-000～x5FF-FFFF,xD00-000～xDFF-FFFF)

ボードに搭載しているタイマ、Audio,シリアル、パラレルなどの I/O デバイス領域で、16bit の I/O 空間としてバスコントローラを設定する必要があります。フルデコードを行っていないため、各所にイメージ空間が現れますので、記載された I/O アドレス以外へのアクセスは行わないでください。

ウェイト制御は外部の H/W で行なっていて、高速 I/O では 1 クロック、低速 I/O では 7 クロックのウェイトが常に入ります。

各 I/O デバイスの詳細については、「8 SYSTEM-IO」を参照して下さい。

CS6 空間 (x600-000～x6FF-FFFF,xE00-000～xEFF-FFFF)

高速 SRAM（日本電気製の uPD431008LE-15：128K * 8bit,15nS）用の空間で、512K バイトの容量と 32 ビットデータ幅を持ちます。512K バイトおきに SRAM のイメージが現れます。

ウェイト制御は CPU 内部のバスコントローラで行ない、33MHz の外部クロック時に０ウェイトでアクセスできます。

CS7 空間(x700-000～x7FF-FFFF,xF00-000～xFFF-FFFF)

ブート ROM 用の空間で、16 ビットバス（BT16B=1）の時は EPROM、32 ビットバスの時は Flash ROM がブート ROM として選択されます（BT16B 端子は、SW1-8 で切り替わります)。

EPROM は 40pin DIP タイプの 27C1024,27C2048,27C2048（150nS 以下）が使用できます。出荷時にはモニタが格納された 27C1024 が実装されています。

ウェイト制御は外部の H/W で行なっていて、EPROM では５クロック、Flash ROM では４クロッ

クのウェイトが常に入ります。

8. SYSTEM-IO

SYSTEM-IO は、CS5 の空間にマップされた I/O デバイスで Audio,UART/PRINTER, TIC, PIO, ISA バス・インターフェースなどがあります。ここではこれらの I/O デバイスについて説明します（ISA バス・インターフェースについては、説明を省略します）。

8.1. SYSTEM-IO 一覧

SYSTEM-IO の一覧を下記の表に示します。

アドレス	機能	備考
4500-1000H	DIPSW1 参照	高速 I/O
4500-2000H	7 セグメント LED 表示データ設定	高速 I/O
4500-3000H	ステータス	非公開
4500-4000H	コマンド＃１設定／参照(PBHEn,PA0)	高速 I/O
4500-5000H	コマンド＃２設定／参照(Time-Over-Clear,Flash-Reset)	高速 I/O
4500-8000H～4500-801FH	UART-CH#1(TL16C552A)設定／参照	低速 I/O
4500-9000H～4500-901FH	UART-CH#2(TL16C552A)設定／参照	低速 I/O
4500-A000H～4500-A01FH	PRINTER(TL16C552A)設定／参照	低速 I/O
4500-B000H～4500-B00FH	タイマコントローラ(uPD71054)設定／参照	低速 I/O
4500-C000H～4500-C007H	ISA 通信	非公開
4500-D000H～4500-D01FH	割り込みコントローラ設定／参照	低速 I/O
4580-0000H～4580-001FH	Audio-Control レジスタ設定／参照	高速 I/O
4580-1000H～4580-100FH	uPD63310 レジスタ設定／参照	高速 I/O
4580-2000H	Audio-FIFO データレジスタ設定／参照	高速 I/O

8.2. DIPSW1 読み出しポート(4500-1000H [Read Only])

ベースボード上の DIPSW1 の状態を読み出すためのポートです。データ・フォーマットを下表に示します。

論理アドレス	データバス								内容
	D7	D6	D5	D4	D3	D2	D1	D0	
4500-1000H 入力	SW4 -8	SW4 -7	SW4 -6	SW4 -5	SW4 -4	SW4 -3	SW4 -2	SW4 -1	**0=ON** 1=OFF
ハード割り付け	BT16B	-		-	-	-	-	-	

ボード上に実装されている SW1 の状態を読み出せます。SW1-1 が SW1 の"1"のスイッチに、SW1-8 が SW1 の"8"のスイッチに対応しています。また、該当するビットのスイッチが ON で 0 が、OFF で 1 が読み出されます。SW1 の 1～6 ビットは MULTI 用モニタが動作設定用のスイッチとして使用しています（「13.2 スイッチの設定」を参照）。

SW1-8 はハードウェアの設定を行っています。

SW1-8 :ブート時のバスサイズ（BT16B）及び ROM を指定します。

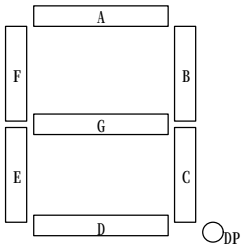
BT16B	ブート時のバスサイズ及び ROM
0	32bit,Flash ROM からブート
1	16bit,EPROM からブート

8.3. 7 セグメント LED 表示データ出力ポート(4500-2000H [Write Only])

7 セグメント LED に表示するデータを設定します。データ・フォーマットを下表に示します。該当するビットに 0 を設定すると対応するセグメントが点灯します。

論理アドレス	データバス								内容
	D7	D6	D5	D4	D3	D2	D1	D0	
4500-2000H 出力	7SEG -DP	7SEG -G	7SEG -F	7SEG -E	7SEG -D	7SEG -C	7SEG -B	7SEG -A	**0=点灯** 1=消灯

7 セグメント LED のビット対応は、下図の通りです。



8.4. コマンドレジスタ#0 ポート(4500-3000H [Read/Write])

コマンドレジスタ＃０は、システムの予約領域です。以下の設定から変更しないでください。

論理アドレス	レジスタ	データバス							
		D7	D6	D5	D4	D3	D2	D1	D0
4500-4000H	CMD#0 (初期値)	X	X	X	X	0 (0)	0 (0)	0 (0)	0 (0)

8.5. コマンドレジスタ＃１ポート(4500-4000H [Read/Write])

コマンドレジスタ＃１は、外部拡張バス（EXT バス）に対してアクセスする場合のバイトイネーブル（BHEn,A0 信号）を制御するためのポートです。

論理アドレス	レジスタ	データバス							
		D7	D6	D5	D4	D3	D2	D1	D0
4500-4000H	CMD#1 (初期値)	X	X	X	X	X	X	PBHE (0)	PA0 (0)

外部拡張バス(EXT バス)に対しアクセスする場合には、以下の通り、設定して行ってください。

CS4 空間（EXT バス）を I/O 空間にマップした場合

アクセスするタイプ	PBHE-	PA0
偶数アドレスへのバイトアクセス	1	0
奇数アドレスへのバイトアクセス	0	1
偶数アドレスへのハーフワードアクセス	0	0

CS4 空間（EXT バス）をメモリ空間にマップした場合

アクセスのタイプに関係なく、常に PBHE- ,PA0 共に'0'を設定してください。リードサイクルではハーフワードアクセス固定となり、ライトサイクルではアクセスサイズによって EXT バスの BHEn, A0 が生成されます。

8.6. コマンドレジスタ#2 ポート(4500-5000H [Read/Write])

コマンドレジスタ＃２は、以下の機能が割り付けられています。

論理アドレス	レジスタ	データバス							
		D7	D6	D5	D4	D3	D2	D1	D0
4500-5000H	CMD#1 (初期値)	X	X	X	X	X	0	FRES (0)	TOVEN (0)

TOVEN :タイムオーバー機能の使用を制御します。タイムオーバー機能は、バスサイクルが 512 バスクロック以上となった場合に、強制的に READY-を返してバスサイクルを終了させます。

TOVEN	タイムオーバー機能
0	使用しない （リセット値）
1	使用する

FRES :フラッシュ ROM のリセットを行います。

FRES	Flash ROM
0	リセット解除 （リセット値）
1	リセット

*リセット解除状態でないとフラッシュ ROM へのアクセスはできません。

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

UART/PRINTER コントローラとして TEXAS INSTRUMENTS 製の TL16C552A(DUAL ASYNCHRONOUS COMMUNICATIONS ELEMENT WITH FIFO)LSI を使用しています。TL16C552A は、UART を 2 チャンネル、双方向プリンタ・ポート（PS2 互換）を 1 チャンネル備えており、UART の受信部には 16 キャラクタ分の FIFO バッファを内蔵しています。

TL16C552A の各レジスタは、表のように割り付けられています。各レジスタの機能については、TL16C552A のマニュアルを参照してください。

アドレス	機能	読み出し	書き込み
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 レジスタ配置

TL16C552A の CLK 入力には 16MHz のクロックが接続されています。

8.8. TIC（uPD71054）(4500-B000H～4500-B00CH)

TIC は NEC 製の uPD71054 が実装されています。uPD71054 は Intel 製の i8254 と互換であり、3 つのタイマ／カウンタを持っています。これらのタイマ／カウンタにより、モニタのタイマ割り込みの生成を行っています。

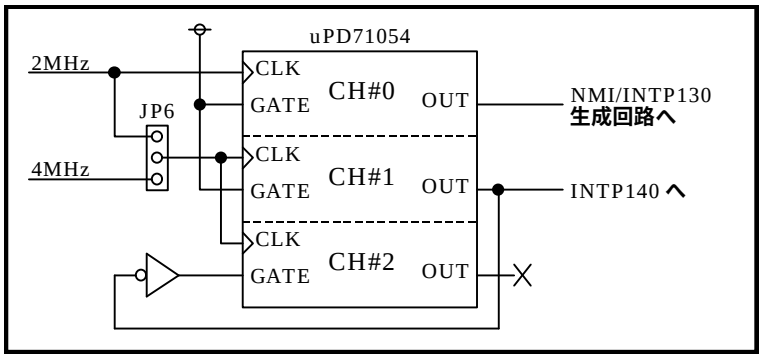
TIC の各レジスタは、表の通りに割り当てられています。

アドレス	読み出し	書き込み
4500-B000H	COUNTER#0	COUNTER#0
4500-B004H	COUNTER#1	COUNTER#1
4500-B008H	COUNTER#2	COUNTER#2
4500-B00CH	-----	Control Word

TIC のレジスタ配置

TIC の各チャンネルは下図のように接続されています。チャンネル 0 は、PIC に接続され、モニタ用のインターバル・タイマとして使用されます。クロック入力には 2MHz が接続されています。チャンネル 1 は、ユーザのプログラムで自由に使用することができ、チャンネル 1 はチャンネル 2 のプリスケール・カウンタとして機能します。チャンネル 2 は、ユーザのプログラムで自由に使用することができます。

チャンネル１およびチャンネル２に接続されているクロックは、ボード上の JP6 により 2MHz か 4MHz のいずれかを選択して接続できます。



uPD71054 はコマンド・リカバリ・タイムとして 165ns 必要とします。リカバリ・タイムの確保の方法として、ROM 領域へのダミーリードを推奨します。TIC は、システム・リセットによってリセットされます。

使用モード例

CH#0 ：モード２（レートジェネレータ）
CH#1 ：モード２（レートジェネレータ）
CH#2 ：モード０（ダウンカウンタ）

8.9. 割り込みコントローラ（PIC）(4500-D000H～4500-D018H)

PIC は、主に割り込み関係の制御を行ないます。レジスタ割り付けは下表の通りです。

RTE-V831-PC では、PIC の INT0 は、NMI/INT3-の指定により、V831 の NMI か INTP03 に接続され、INT1 は INTP02 に接続されます。

論理アドレス	レジスタ	データバス							
		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	NMI EN

INT0M,INT1M レジスタはそれぞれ INT0,INT1 に入力する割り込みをマスクします。IM0x, IM1x のビットが"1"の時にイネーブルとなり、複数ビットを選択した場合にはそれぞれの OR で割り込みがアクティブとなります。

INTR レジスタは割り込みステータスで、割り込み要求がある場合に"1"が読み出せます。これはマスク状態に関係ありません。またエッジ割り込み要求の解除（クリア）には、このレジスタの対応ビットに"1"を書き込みます。

IM0[0..7],IM1[0..7].IR[0..7]の各ビットに割り付けられている割り込み要因は以下の通りです。

IM0,IM1,IR	割り込み要因	要求レベル
0	タイマ０（モード２）	エッジ (立ち上がり)
1	シリアル０	レベル (high)
2	ホスト（ISA 通信）	レベル (Low)
3	タイムオーバー	レベル (Low)
4	タイマ１（モード２）	エッジ (立ち上がり)
5	シリアル１	レベル (High)
6	パラレル（プリンタ）	レベル (High)
7	ISA-IRQ	レベル

INTEN レジスタは、割り込み全体のイネーブル／ディセーブルなどを制御します。

NMIEN: ノンマスカブル割り込み（NMI） をハード的に禁止することができます。この時、NMI 端子の状態は High レベルとなります。

NMIEN	NMI
0	マスクする （リセット値）
1	マスクしない

NMI/INTP3-: INT0 の割り込みを NMI と INTP03 のどちらに入れるかの指定です。

NMI/INTP3-	INT0
0	INTP03 （リセット値）
1	NMI

【注意】INT0（NMI/INTP03）はモニタで使用していますので、関連するレジスタを変更しないでください。INT1 は解放されていますので、自由に使用可能です。

8.10. AUDIO コントローラ（AUDCNT）(4580-0000H～4580-0010H,4580-2000H)

AUDCNT は、AUDIO チップ(uPD63310)とのデジタルデータ入出力を制御します。

論理アドレス	レジスタ	データバス							
4580-0000H	**CONTROL**	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	**STATUS**	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	**MCLKDIV**	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	**AUDIO DATA**	D15	・・・						D0
		MSB	L/R チャネル音声データ						LSB

CONTROL レジスタは、音声の録音／再生を制御するレジスタです。(Read/Write)

RST	Audio リセット
0	リセット解除 （リセット値）
1	リセット

PLY	再生動作
0	停止 （リセット値）
1	起動

PM1	再生時オーバーフロー割り込み
0	ディセーブル （リセット値）
1	イネーブル

PM0	再生時アンダーフロー割り込み
0	ディセーブル （リセット値）
1	イネーブル

REC	録音動作
0	停止 （リセット値）
1	起動

RM1	録音時オーバーフロー割り込み
0	ディセーブル （リセット値）
1	イネーブル

RM0	録音時アンダーフロー割り込み
0	ディセーブル （リセット値）
1	イネーブル

STATUS レジスタは、リード専用レジスタで各種のステータスを示します。

PEX	再生状態
0	停止中
1	実行中

POV	再生時オーバーフロー
0	なし
1	発生

PUD	再生時アンダーフロー
0	なし
1	発生

FF0	HF0	EF0	再生用 FIFO
0	0	1	エンプティ
0	1	0	ハーフフル
1	1	0	フル（FIFO 書き込み禁止）

REX	録音状態
0	停止中
1	実行中

ROV	録音時オーバーフロー
0	なし
1	発生

RUD	録音時アンダーフロー
0	なし
1	発生

FF1	HF1	EF1	録音用 FIFO
0	0	1	エンプティ（FIFO 読み出し禁止）
0	1	0	ハーフフル
1	1	0	フル

MCLKDIV レジスタは、MCLK の周波数を決定します。

DIV4	DIV3	DIV2	DIV1	DIV0	MCLK 49.152 / (DIV+2)	サンプリング 周期 (MCLK / 256)	バイト／秒 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	45.2KB
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 は、音声データの FIFO 入出力用のデータポートで、Lch, Rch の順番で 16 ビットのデータを入出力します。

8.11. uPD63310 レジスタ:Audio cod.(4580-1000H～4580-100FH)

uPD63310 のﾚｼﾞｽﾀは、以下の通り割り付けられます。詳細には、uPD63310 のデータシートを参照ください。

アドレス	機能	D5	D4	D3	D2	D1	D0
4580-1000H	ｱﾄﾞﾚｽ･ﾚｼﾞｽﾀ	ﾚｼﾞｽﾀ番号					
4500-1008H	ﾃﾞｰﾀ･ﾚｼﾞｽﾀ	利得ｺﾝﾄﾛｰﾙﾃﾞｰﾀ					

9. 割込みとDMA

RTE-V831-PC の割込みと DMA について説明します。

9.1. 割り込み

外部割り込みは、以下の通り使用します。

割り込み	要因
NMI	ROM ｴﾐｭﾚｰﾀからの割り込みと PIC の設定による INT0 の割り込み（ﾓﾆﾀ用）
INTP03	PIC の設定による INT0 の割り込み（ﾓﾆﾀ用）
INTP02	PIC の設定による INT1 の割り込み（ﾕｰｻﾞ用）
INTP01	EXT（拡張）バスからの割り込み
INTP00	AUDIO 回路（エラー発生時）からの割り込み
INTP13	未使用
INTP12	未使用
INTP11	未使用
INTP10	未使用

ボード上の割り込みコントローラ（「8.9

割り込みコントローラ（PIC）(4500-D000H～4500-D018H)」参照）により、以下の割り込みソースから任意の割り込み要求を選択し、２種の割り込み(INT0,INT1)を生成できます。INT0 はシステム用（MULTI 用のモニタで使用）、INT1 はユーザアプリケーション用です。

割り込み要因
タイマ０（モード２）
シリアル０
ホスト（ISA 通信）
タイムオーバー
タイマ１（モード２）
シリアル１
パラレル（プリンタ）
ISA-IRQ

9.2. NMI の使用方法

本ボード上にモニタ等を移植する場合に NMI を使用する方法を説明します。NMI はエッジ検出されますが、割り込みソースはレベル出力のため NMI をハード的にマスクできるようにしています。マスク方法については、「8.9

割り込みコントローラ（PIC）(4500-D000H～4500-D018H)」の INTEN レジスタに関する記述を参照してください。

NMI が発生した場合は、下記の手順で処理します。

① PIC の NMIEN に"0"を設定して、NMI をハード的にマスクする。
② PIC の INTR を検査する。
③ 要求元のための NMI 処理を行ない、要求をクリアする。
④ PIC の NMIEN に"1"を設定して、マスクを解除する。
⑤ NMI 処理から復帰する。

【注意】モニタを使用する場合には、NMI, INTP03, PIC の INT0 に関連したレジスタの操作を行わないで下さい。モニタがハングアップする場合があります。

9.3. DMA チャンネル

DMARQ-/AK-	DMARQ 信号	備考
CH0	再生時の要求	再生時に Audio のデータバッファにデータを書き込むことを要求する DMA 要求です。間に合わなかった場合、アンダーラン・エラーが発生します。
CH1	録音時の要求	録音時に Audio のデータバッファからデータを読み出すことを要求する DMA 要求です。間に合わなかった場合、オーバフロー・エラーが発生します。

備考：

１．DMARQ-/AK-[1..0]は、共に負論理に設定します。

10. EXT バス仕様

EXT バスは、メモリや I/O などを拡張用のバスで、JEXT コネクタが用意されています。このコネクタには、ボード内部のローカル・バスが接続されています。

10.1. ピン配置と信号の説明

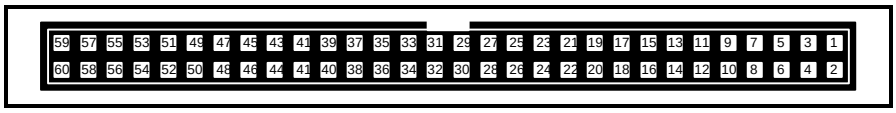
以下に JEXT コネクタのピン配置と信号の説明を示します。

番号	信号名	番号	信号名	番号	信号名	番号	信号名
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 コネクタピン配置

信号名	入出力	機能
A[1..19]	出力	アドレス・バス信号。CPU のアドレス信号をバッファして接続。
A0, BHE-	出力	バイトロー／ハイ・イネーブル信号。
D[0..15]	入出力	データ・バス信号。CPU のデータ・バス信号をバッファして接続。 また、ボード上で 10KΩプルアップ。
RD-	出力	リード・サイクルのタイミング信号。JEXT 空間のアクセス時のみ、アクティブになる。
WR-	出力	ライト・サイクルのタイミング信号。JEXT 空間のアクセス時のみ、アクティブになる。
READY	入力	サイクルの終了を V831 に通知する信号。JEXT 空間のみで有効。 確実に V831 に READY を認識させるためには、RD-もしくは WR-がインアクティブになるまで READY をアクティブに保つことが必要。 また、ボード上で 10KΩプルアップ。
INT-	入力	Low アクティブの割り込み要求信号。バッファ後に論理反転されて、V831 の INTP01 に接続。
RESET-	出力	Low アクティブのシステム・リセット信号。
CLK	出力	クロック信号。V831 の CLKOUT 信号をバッファして接続。

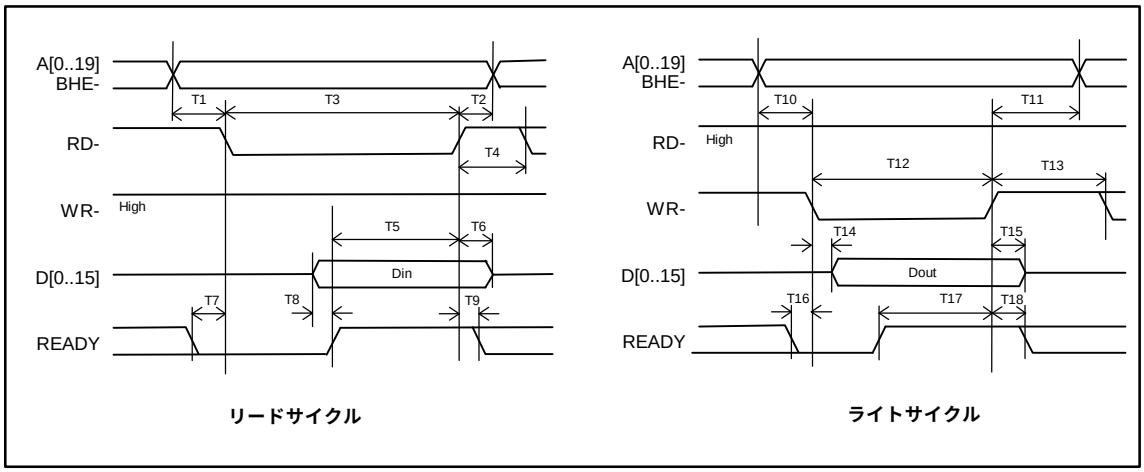
JEXT コネクタ信号



JEXT **のピン配置**

10.2. タイミング

EXT **バスのタイミングを示します。**



JEXT **バス・サイクル**

記号	内容	MIN(ns)	MAX(ns)
T1	RD **アドレス　セットアップ時間**	0	
T2	RD **アドレス　ホールド時間**	0	
T3	RD **サイクル時間**	50	
T4	RD **サイクル間隔**	20	
T5	RD **データ　セットアップ時間**	15	
T6	RD **データ　ホールド時間**	0	
T7	RD READY　WAIT **セットアップ時間**	0	
T8	RD READY **セットアップ時間**	0	
T9	RD READY **ホールド時間**	0	
T10	WR **アドレス　セットアップ時間**	0	
T11	WR **アドレス　ホールド時間**	20	
T12	WR **サイクル時間**	50	
T13	WR **サイクル間隔**	20	
T14	WR **データ　遅延時間**		20
T15	WR **データ　ホールド時間**	20	
T16	WR READY　WAIT **セットアップ時間**	0	
T17	WR READY **セットアップ時間**	0	
T18	WR READY **ホールド時間**	0	

JEXT **バス** AC **スペック**

10.3. EXT バス使用時の注意事項

EXT バスにハードウェアを拡張する場合には、以下の点に注意してください。

(1) メモリ-I/O 空間の割り付け

EXT バスは、V831 の CS4 空間に割り付けられています。CPU 内部のバスコントローラ BCTC によって、メモリまたは I/O 空間のどちらかを選択し、選択した空間でアクセスして下さい。

(2) A0, BHE-信号の扱い

EXT バス上の信号で A0,BHE-は、V831 からのアクセスとポート出力によって決定されます。「8.5 コマンドレジスタ＃１ポート(4500-4000H [Read/Write])」を参照し、正しく設定してからアクセスを行って下さい。

11. ソフトウェア

RTE-V831-PC ボードのハードウェアの初期化と周辺デバイスの使用方法について説明します。

11.1. 初期化

ブートルーチンでは外部メモリ／I/O をアクセスするために V831 の内部バスコントローラを初期化します。ウェイト制御や DRAM タイミングなどは、バスクロック 33MHz での値としています。

レジスタ	内部 I/O アドレス	設定値	サイズ	備考
BCTC	C000-0010H	30H	byte	
DBC	C000-0012H	30H	byte	
PWC0	C000-0014H	7470H	half-word	
PWC1	C000-0016H	0000H	half-word	SRAM 0 ウェイト（CS6）
PIC	C000-0018H	0000H	half-word	
DRC	C000-0020H	02H(82H)	byte	EDO 60ns (HyperPageMode)
RFC	C000-0022H	900EH	half-word	約 15μsec

レジスタの詳細については、V831 CPU マニュアルを参照ください。

11.2. ライブラリ

C コンパイラでプログラムする時に必要となる I/O アクセスなどのライブラリです。ただし、これらの記述やパラメータ受け渡し方法などは、GHS-C の場合のものです。他のコンパイラ等を使用する場合には、変更が必要となる場合があります。

```
/* I/O 入出力ライブラリ */

/* GHS V800 コンパイラ パラメータ受け渡し */
/* arg0：r6, arg1：r7, arg2：r8, return：r10 */

inb(int addr)                       /* バイト（8 ビット）入力 */
{
    __asm("  in.b 0[r6], r10");
}

inh(int addr)                       /* ハーフワード（16 ビット）入力 */
{
    __asm("  in.h 0[r6], r10");
}

inw(int addr)                       /* ワード（32 ビット）入力 */
{
    __asm("  in.w 0[r6], r10");
}

outb(int addr, int data)                        /* バイト（8 ビット）出力 */
{
    __asm("  out.b r7, 0[r6]");
}

outh(int addr, int data)                        /* バイト（8 ビット）出力 */
{
    __asm("  out.h r7, 0[r6]");
}

outw(int addr, int data)                        /* バイト（8 ビット）出力 */
{
    __asm("  out.w r7, 0[r6]");
}
```

11.3. タイマの使用法

ボード上の外部タイマ（8254）でカスケード接続されたタイマ１とタイマ２を使用した時間計測のサンプルを示します。タイマ１はインターバルカウンタ（モード２）、タイマ２は、ダウンカウンタ（モード０）として初期化して、時間計測するルーチンの前後でカウンタ値を求めておくことで実行時間が算出できます。ただし、タイマのカウント値はどちらもダウンカウンタとなることに注意してください。また、外部タイマの連続アクセスではコマンドリカバリ（ROM 領域にダミーリード）が必要となります。

```
/* タイマによる実行時間計測サンプル */

#define TIMERCLK        2000000                 /* 2MHz */
#define INTERVAL        (TIMERCLK * 10 / 1000)  /* 10ms (1/100) */
#define IOWAIT()        (*(char *)0x4FFF0000)   /* I/O コマンドリカバリ用 */

InitTimer() /* タイマ初期化 */
{
            outb(0x4500B00C, 0x74);             IOWAIT();   /* タイマ 1 モード 2 */
            outb(0x4500B004, INTERVAL);         IOWAIT();   /* タイマ 1 下位カウント */
            outb(0x4500B004, INTERVAL /256);    IOWAIT();   /* タイマ 1 上位カウント */
            outb(0x4500B00C, 0xB0);             IOWAIT();   /* タイマ 2 モード 0 */
            outb(0x4500B008, 0xFF);             IOWAIT();   /* タイマ 2 下位カウント */
            outb(0x4500B008, 0xFF);             IOWAIT();   /* タイマ 2 上位カウント */
            return 0;
}

LatchTimer()            /* カウントラッチ */
{
            int count1, count2, counts;

            outb(0x4500B00C, 0xDC);             IOWAIT();   /* タイマ 1/2 マルチプルラッチ */
            count1  = inb(0x4500B004);          IOWAIT();
            count1 += inb(0x4500B004) *256;     IOWAIT();   /* タイマ 1 カウント */
            count2  = inb(0x4500B008);          IOWAIT();
            count2 += inb(0x4500B008) *256;     IOWAIT();   /* タイマ 2 カウント */
            counts  =  INTERVAL * (0xFFFF - count2)
               + (INTERVAL - count1);
            return counts;
}

double total_time;

main()
{
            int start_count, stop_count;

            InitTimer();
            start_count = LatchTimer();         /* スタートカウント値 */
            func();
            stop_count = LatchTimer();          /* ストップカウント値 */
            total_time = (double)(stop_count - start_count)
                        / (double)TIMERCLK;     /* 秒数 */
            return 0;
}

#include <time.h>

func()                  /* 時間計測ルーチン */
{
            ....
}
```

11.4. Flash ROM プログラミング

ボードに搭載された Flash ROM にデータを書き込むサンプルプログラムです。Flash ROM の書き込みは、アンキャッシュ領域でバイト単位で行います。プログラミング・アルゴリズムについての詳細は、Flash ROM のデータシートを参照してください。

```
/* Flash ROM 書き込みサンプル */

#define FLCMD_ERASE     0x80                    /* 消去 */
#define FLCMD_WRITE     0xA0                    /* 書き込み */
#define FLCMD_READ      0xF0                    /* 読み込み */
#define BIT_DQ(n)       (1<<(n))                /* ビットマクロ */

/* Flash-ROM ライブラリ */

static FlashCommand(int addr, int cmd)          /* コマンド */
{
            int addr2;

            addr2 = addr & ~(0x7FF<<2);                     /* アドレス 11 ビット 0 クリア */
            *(char *)(addr2 + (0x555<<2)) = 0xAA;
            *(char *)(addr2 + (0x2AA<<2)) = 0x55;
            *(char *)(addr2 + (0x555<<2)) = cmd;
            return 0;
}

static FlashDataPoll(int addr, int data)        /* ビジーチェック */
{
            int rdata;

            data &= 0xff;                                   /* バイト */
            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)                   /* イレース (全体／セクタ) */
{
            FlashCommand(addr, FLCMD_ERASE);
            FlashCommand(addr, (all)? 0x10: 0x30);          /* 全体／セクタ */
            return FlashDataPoll(addr, 0xff);
}

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

FlashRead(int addr)                             /* 読み込み (リセット) */
{
            int data;

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

```
/* EPROM 上のモニタをフラッシュ ROM にコピーする */

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

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

            if ((inb(0x45001000) & 0x80) == 0) {                    /* EPROM ブートの確認 */
                        return -1;
            }
            daddr = DST_ADDR;
            saddr = SRC_ADDR;

            for (i = 0; i < 4; i++) {
                        FlashRead(daddr + i);                       /* 初期化 */
            }
            for (i = 0; i < DATA_SIZE; i++) {
                        if ((i & (0xFFFF<<2)) == 0) {
                                    err = FlashErase(daddr, 0);     /* セクタ消去 */
                                    if (err) {
                                                break;              /* 消去エラー */
                                    }
                        }
                        data = *(char *)saddr & 0xFF;
                        err = FlashWrite(daddr, data);              /* 書き込み */
                        if (err) {
                                    break;                          /* 書き込みエラー */
                        }
                        saddr++;                                    /* 次のアドレス */
                        daddr++;
            }
            daddr = DST_ADDR;
            for (i = 0; i < 4; i++) {
                        FlashRead(daddr + i);                       /* 初期化（終了） */
            }
            return err;
}
```

このプログラムをデバッガで走らせて正常終了した後にデバッガを終了して、SW1-8 を ON にしてボードをリセットすると Flash ROM からモニタが立ち上がります。

【注意】Flash-ROM への書き込み中に、ブレークやその後の再実行、Flash ROM 領域へのデータアクセスなどは行わないようにしてください。Flash ROM のコマンド実行がそれらのアクセスによって中断されると Flash ROM に致命的なダメージを与える可能性があります。

11.5. 音声入出力

ボードに搭載されている音声入出力インターフェースを使用したプログラムサンプルです。データの入出力には、V831 の内蔵 DMA を使用しています。

```
/* 音声入出力サンプル */

#define DMA0                        0xC0000030              /* 内蔵 DMA ch0（再生）*/
#define DMA1                        0xC0000040              /* 内蔵 DMA ch1（録音）*/
#define AUDIO_DATA      0x45802000              /* AUDIO データ（FIFO）*/

static Set63310Reg(int reg, int data)           /* uPD63310 レジスタ設定 */
{
            outb(0x45801000, reg);                          /* アドレスレジスタ ライト */
            outb(0x45801008, data);                         /* データレジスタ ライト */
            return 0;
}

static Get63310Reg(int reg)                     /* uPD63310 レジスタ獲得 */
{
            outb(0x45801000, reg);                          /* アドレスレジスタ ライト */
            return inb(0x45801008) & 0x3F;                  /* データレジスタ リード（6 ビット） */
}

AudioInit()                                     /* Audio 初期化 */
{
            outh(0x45800000, 0x8000);                       /* リセット Audio */
            outh(0x45800010, 8);                            /* fs = 12KHz */
            inh(0x45800010);                                /* ダミーリード */
            outh(0x45800000, 0);                            /* リセット解除 */
            Set63310Reg( 0, 0);                             /* IN1L    0db */
            Set63310Reg( 1, 0);                             /* IN1R    0db */
            Set63310Reg(17, 0);                             /* OUTDACL 0db*/
            Set63310Reg(18, 0);                             /* OUTDACR 0db */
            return 0;
}

AudioTerm()                                     /* Audio 終了処理 */
{
            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)                   /* 再生処理 DMA0 使用 */
{
            outh(DMA0 + 0, addr >> 16);                     /* DMA-DSAOH */
            outh(DMA0 + 2, addr);                           /* DMA-DSAOL */
            outh(DMA0 + 4, AUDIO_DATA >> 16);   /* DMA-DDAOH */
            outh(DMA0 + 6, AUDIO_DATA);                     /* DMA-DDAOL */
            size = (size / 2 - 1) * 2;                      /* DMA 転送カウント数 */
            outh(DMA0 + 8, size >> 16);                     /* DMA-DBCOH */
            outh(DMA0 + 10, size);                          /* DMA-DBCOL */
            outh(DMA0 + 12, (0<<12)                         /* DMA-DCHC0            TTYP DMARQ */
                      | (1<<10)                             /*                      TBT Mem->I/O */
                      | (0<<8)                              /*                      SAD Inc */
                      | (2<<6)                              /*                      DAD Fix */
                      | (0<<5)                              /*                      DAL Low */
                      | (0<<4)                              /*                      DRL Low */
                      | (1<<3)                              /*                      TM Demand */
                      | (1<<1)                              /*                      DS Half-word */
                      |  1     );                           /*                      Enable */
            outh(0x45800000, 0x0001);                       /* 再生スタート */
            while ((inh(DMA0 + 12) & 1) != 0)
                        ;                                   /* DMA 終了を待つ */
```

```
            while ((inh(0x45800008) & 0x10) == 0)
                        ;                                   /* FIFO エンプティまで待つ */
            outh(0x45800000, 0);                            /* 再生終了 */
            return 0;
}

AudioRecord(int addr, int size)                 /* 録音処理 DMA1 使用 */
{
            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 転送カウント数 */
            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);                                    /* 録音スタート */
            while ((inh(DMA1 + 12) & 1) != 0)
                        ;                                   /* DMA 終了を待つ */
            outh(0x45800000, 0);                            /* 録音終了 */
            return 0;
}

#define COUNT           0x10000                 /* L/R データサンプリング数 */

int buffer[COUNT];                              /* L/R データバッファ */

main()
{
            inb(0xC000006E);
            outb(0xC000006E, 1);                            /* DMA-DC MEM=1 */
            AudioInit();                                            /* 初期化 */
            AudioRecord((int)buffer, sizeof(buffer));       /* 録音 */
            AudioPlay  ((int)buffer, sizeof(buffer));       /* 再生 */
            AudioTerm();                                            /* 終了 */
            return 0;
}
```

音声データは、Lch（左）／Rch（右）**ごとに各** 16 **ビットデータで、録音／再生ともにスタートから必ず** L 1, R1, L2 , R2, ...**の順番でデータ入出力します。**

12. マスカブル割り込みを使用したアプリケーションの開発

RTE-V831-PC 上でマスカブル割り込みを使用したアプリケーションの開発を行う場合の方法と、制限事項について説明します。

12.1. 割り込みベクタ

V831 の割り込みベクタ領域である FFFF-FE00H～FFFF-FFFFH 番地は、ROM により固定されていて書き換えることができません。そこで弊社が提供するモニタでは、SRAM 上に代替えのベクタ領域を用意し、FFFF-FE00H～FFFF-FFFFH 番地のベクタには、その代替えベクタ領域への分岐命令が置かれています。例えば、例外コード FE00H の割り込みが発生すると、CPU の割り込み機能により FFFF-FE00H 番地に分岐し、そこには対応する代替えベクタ番地への分岐命令があります。したがって、ユーザ・プログラムでは、この代替えベクタ領域を本来のベクタ領域と同じように書き換えれば、割り込み発生時にユーザ・プログラムの割り込み処理ルーチンに分岐するようにできます。

通常の V831 のプログラムと異なるのは、通常はベクタ領域は ROM 化の時点で固定されており、プログラムで書き換える必要はありません。しかし、モニタ上で動作するプログラムの場合、自分で代替えベクタを書き換えてから割り込みを許可する必要があります。

RTE-V831-PC のモニタでは、代替えベクタ領域を SRAM 上の FE07-0000H～FE07-01FFH に確保しています。したがって、例外コード FE00H の割り込みは FE07-0000H 番地、例外コード FE10H の割り込みは FE07-0010H 番地...に目的の割り込み処理に分岐する命令を書き込みます。また、V831 CPU はキャッシュメモリを内蔵していますので、ベクタを書き換えた後にキャッシュをクリアすることが必要です。クリアしないと、書き換え前の命令が実行されてしまう場合があります。

代替えベクタを書き換えるためのプログラム例を下記に示します（割り込み処理ルーチンから代替えベクタ領域への相対アドレスが 26 ビット以内の場合）。

```
#define VECT_CPU     0xfffffe00                                  /* CPU 割り込みベクタ先頭 */
#define VECT_RAM     0xfe070000                                  /* 代替え割り込みベクタ先頭 */
#define VECT(n)      ((VECT_CPU - n) + VECT_RAM)                 /* 割り込みベクタ番地を求める */

main()
{
          extern void __interrupt IntEntry();                    /* 割り込み処理ルーチン */
          int addr, offs, inst;

          /* CPU ベクタの 0xfffffe30 に対応する代替えベクタ番地と
             割り込みルーチンへの “JR dest26” 命令の作成      */
          addr = VECT(0xfffffe30);
          offs = (int)IntEntry - addr;
          inst = 0xa8000000 | (offs & 0x3fffffe);                /* 32 ビット命令 JR dest26 */

          /*  ベクタの置き換え */
          di();                                                  /*  割り込み禁止 __asm(“ di”); */
          *((unsigned short *)(addr + 0)) = (inst >> 16) & 0xffff;      /* 上位 16 ビットコード */
           *((unsigned short *)(addr + 2)) = (inst     ) & 0xffff;      /* 下位 16 ビットコード */
           outw(0xFFFFFFF4, 3);                                  /* キャッシュクリア*/

          /* 割り込みデバイスの初期化など */
          ……
          ei();                                                  /* 割り込み許可 __asm(“ ei”); */

          ……

}
```

12.2. 内蔵命令RAM

V831 の場合 CPU の機能として、マスカブル割り込みのベクタを内蔵命令 RAM に配置することが可能です（システム・レジスタ HCCW の IHA ビット）。この機能を使用した場合は、V831 の通常のプログラムと、RTE-V831-PC 上でモニタを使用したプログラムの場合で、ベクタの設定に関して異なることはありません。

内蔵命令 RAM のベクタの使用方法や、内蔵命令 RAM の内容の変更方法については、CPU のマニュアルを参照してください。

また、割り込み処理に限らずプログラムを内蔵命令 RAM に配置する場合には、コンパイルされたオブジェクトコードに注意する必要があります。特にＣの switch-case 文は、命令コード中にジャンプテーブルが作成され、このテーブル参照により分岐を行うコードが生成されることがあります。このテーブル参照は LD 命令で行われますが、内蔵命令 RAM は LD 命令による参照ができないためプログラムは予期しない動作を行う結果となります。

12.3. 一般的な制限事項／注意事項

マスカブル割り込みを使用したアプリケーションをデバッグする上での制限事項と注意事項を下記に示します。

１）代替えベクタの設定前に割り込みが発生した場合や、代替えベクタを正しく設定しないで割り込みが発生した場合には、割り込み発生時点でのプログラム位置でブレークします。これは、代替えベクタの初期値がモニタ ROM のブレーク処理ルーチンへの分岐命令になっているためです。

２）代替えベクタ領域から割り込み処理ルーチンまでの相対アドレスが 26Bit を超える場合、割り込み処理ルーチンへの分岐のために、少なくとも１つ以上のレジスタの値を壊すか、分岐の中継点を作る必要があります。

３）代替えベクタ領域は、ROM モニタの管理領域として保護されているため、プログラムのダウンロードで書き換えることはできません。したがって、ソース・プログラム上ベクタ領域を独立したセクションとして定義し、リンク時のパラメータによりそのセクションを代替えベクタ領域に割り付けることはできません。

４）代替えベクタ領域を書き換えた直後に CPU 内蔵のキャッシュ・メモリをフラッシュしてください。この操作を忘れると、代替えベクタ書き換え前のベクタが使用されてしまうことがあります。

５）割り込み関係を含む全てのペリフェラルは、ボード上のリセット・スイッチによってのみ初期化されます。したがって、一度プログラムを実行した後にプログラムを再ロードして動作させる場合、前のプログラム実行による影響がペリフェラル上に残ってしまいます。したがって、ペリフェラルを使用するプログラムを動作させた後に、再度プログラムを始めから動作させる場合は、以下の手順にしたがってください。

(1) RTE-V831-PC のリセット・スイッチを押しリセットします。
(2) プログラムをリロードして再実行します。

６）プログラムの先頭で DI（割り込み禁止）状態にしてから、ペリフェラルやベクタの設定をした後、EI（割り込み許可）状態にするようにしてください。

12.4. 割り込み処理ルーチン内でのブレークに関する制限事項

割り込み処理ルーチン内でブレークした場合の制限事項を以下に記します。

１）ブレーク中、全てのマスカブル割り込みは受け付けません。

２）シングルステップ機能は、次の命令にテンポラリ・ブレーク・ポイントを設定する方式を取っています。したがって、EI（割り込み許可）状態のユーザ・プログラムをシングル・ステップする場合、シングルステップ中にも割り込みを受け付けます。

３）割り込みルーチン中でブレーク後に、シングルステップによって割り込み処理ルーチンから抜けることはできません（具体的には、割り込みルーチンの最後の"}"でのシングルステップができません）。同様に、IRET 命令のシングルステップもできません。

４）デバッガの"Return"機能で、割り込み処理ルーチンから元のルーチンへ戻ることはできません。

13. APPEDIX.A MULTI モニタ

MULTI 用のモニタ ROM を使用して、ホストの MULT デバッガと接続して使用する場合の設置方法と使用上の注意事項について説明します。

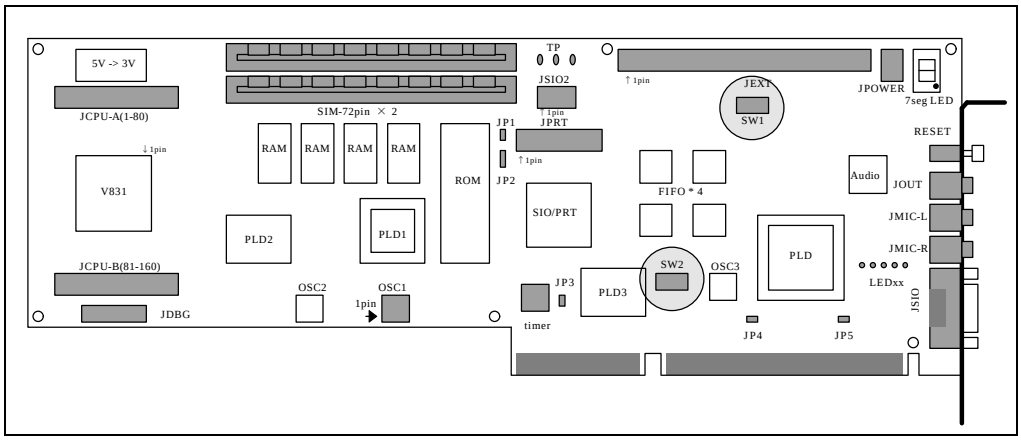
13.1. ボードの設置

13.1.1. RTE for Win32 のインストール

MULTI デバッガを使用する場合には、PC に通信用のソフトウェア（RTE for Win32）をインストールする必要があります。ソフトウェアのインストールとテストについては、添付の「RTE for Win32 インストール・マニュアル」を参照してください。

13.2. スイッチの設定

ボードには、２つの設定用ディップ・スイッチが設けられています。スイッチの箇所を図に示します。



ボードのスイッチの位置(SW1,SW2)

13.2.1. SW1 の設定

SW1 は、汎用の入力ポートのスイッチですが、実装されている MULTI 用のモニタでは、以下の通り使用しています。正しく設定してご使用ください。

SW1 番号	1	2	ボーレート
設定	ON	ON	未使用
	OFF	ON	38400 baud
	ON	OFF	19200 baud
	OFF	OFF	9600 baud （出荷時の設定）

ボーレートの設定

SW1 番号	3	4	プロファイラ周期
設定	ON	ON	タイマを使用しない
	OFF	ON	200 Hz 5 ms
	ON	OFF	100 Hz 10 ms
	OFF	OFF	60 Hz 16.67ms （出荷時の設定）

プロファイラ周期の設定

SW1 番号	5	デバッグモード
設定	ON	7segLED をモニタが使用
	OFF	通常の使用状態 （出荷時の設定）

デバッグモードの設定

SW1 番号	6	ブレーク割り込み
設定	ON	INTP3 を使用 （出荷時の設定）
	OFF	NMI を使用

ブレーク割り込みの設定

SW1-7 は、MULTI では使用していません。

SW1 番号	8	ブート時のバスサイズ（BT16B）及び ROM
設定	ON	32bit（BT16B=0） Flash ROM からブート
	OFF	16bit（BT16B=1） EPROM からブート （出荷時の設定）

ブート時のバスサイズ及び ROM の設定

13.2.2. SW2 の設定

SW2 は、ISA バスの I/O アドレス選択のスイッチです。スイッチの番号 1～8 が ISA バスのアドレス A4～A11 に対応しています（A12～A15 は０固定）。したがって、I/O アドレスとして 000xH～03FxH が選択できます。なおスイッチは、OFF で "1"、ON で "0"の値となります。一般的には、20xH～3FxH の間で設定します。

SW2 番号	1	2	3	4	5	6	7	8	
アドレス	A4	A5	A6	A7	A8	A9	A10	A11	I/O アドレス
ON/ OFF	0	0	0	0	0	1	0	0	020xH （出荷時の設定）

I/O アドレスの対応

13.3. MULTI モニタ

ボードに実装されている ROM には、MULTI 用のモニタが組み込まれています。ホストの MULT デバッガと接続して使用する場合の注意点について説明します。

13.3.1. モニタ・ワークRAM

モニタでは、SRAM の最上位の 64KB をワーク用の RAM として使用しています。したがって、FE07-0000H～FE07-FFFFH はユーザ・プログラムでは使用できません。

13.3.2. 割り込み

ユーザプログラムで割り込みを使用する場合には、「12 マスカブル割り込みを使用したアプリケーションの開発」を参照してください。

13.3.3. 強制ブレーク用の割り込み

モニタが強制ブレーク（MULTI デバッガでの Halt ボタン）及び、タイマ（プロファイラ等）で使用する割り込みは、NMI と INTP03 が選択できます。選択基準としては、DMA を使用するプログラムを実行する場合（音声入出力等）には INTP3 を使用し、そうでない場合は NMI を使用することをお薦めします。以下にそれぞれの場合の相違点を示します。

NMI を使用する場合

ユーザプログラム実行中いつでもブレークできますが、DMA が動作中にブレークした場合は、DMA も中断してしまいます（再実行時にも自動的に再開されることはありません）。

INTP03 を使用した場合

ユーザプログラムで INTP03 より高い優先順位の割り込みを使用している場合や、割り込みが許可されていない状態ではブレークできません。但し、DMA はブレーク中も継続して実行されます。

13.3.4. _INIT_SP の設定

モニタで_INIT_SP（スタック・ポインタの初期値）は、FE06-FFFCH（SRAM の最上位）に設定されています（MULTI デバッガの環境で_INIT_SP を変更することもできます）。

13.3.5. リモート接続

MULTI デバッガでモニタとの接続は、シリアル接続と ISA バス接続が選択できますが、一度接続後、他方に切り替える場合には、モニタをリセット（リアパネルのリセット・スイッチを押す）してから、RTE for Win32 のユーティリティ、Check RTE32.exe で通信路の接続を変更してください。

13.3.6. モニタの実行領域

搭載している ROM には、128K バイトの前半にキャッシュ領域で実行するコードが、後半にアンキャッシュ領域で実行するコードの二つが入ってます。

プログラムのダウンロードを高速に行いたい場合は、キャッシュ側を選択してください。

但し、モニタの介入（プロファイルタイマなど）によって、ユーザプログラムで実行中に使用していたキャッシュが、一時的にフラッシュされてしまいますので、プロファイル測定中においては獲得データの誤差が大きくなる場合があります。このような場合は、アンキャッシュ側を指定してください。

出荷時の設定では、アンキャッシュ（JP2　1-2 ショート）の設定となっています。

13.4. RTE コマンド

サーバと接続すると TARGET ウィンドウが開かれ、ここで RTE コマンドを発行することができます。表に RTE コマンドの一覧を示します。

コマンド名	内容
HELP, ?	ヘルプ表示
INIT	イニシャライズ
VER	バージョン表示
INB, INH, INW	I/O リード
OUTB, OUTH, OUTW	I/O ライト
DCTR,INTR,PLLCR, CMCR	内部レジスタの変更、表示
SFR	内部 I/O 表示／設定

RTE コマンド一覧

各コマンドには、パラメータを必要とするものがあります。アドレスやデータなど、数値のパラメータは、全て 16 進数とみなされます。以下の数値指定は誤りです。

0x1234　1234H　$1234

13.4.1. HELP(?)

<書式>　HELP [コマンド名]

HELP は、RTE コマンドの一覧や書式を表示します。また、”HELP”と入力するかわりに”?”としても同様です。コマンド名を省略すると、使用できるコマンド一覧を表示します。

<例>　HELP SFR

SFR コマンドのヘルプを表示します。

13.4.2. INIT

<書式>　INIT

INIT は、RTE 環境の初期化を行ないます。通常、このコマンドを使用しないでください。

13.4.3. VER

<書式>　VER

VER は、RTE 環境のバージョンを表示します。

13.4.4. INB,INH,INW

<書式>　INB [アドレス]
INH [アドレス]
INW [アドレス]

INB,INH,INW は、I/O リードを行ないます。INB はバイト、INH はハーフ・ワード、INW はワード単位でアクセスします。アドレスが省略すると、前回のアドレスが指定されたものとみなします。

＜例>　INB 1000

1000H からバイトで I/O リードします。

13.4.5. OUTB,OUTH,OUTW

<書式> OUTB [[アドレス] データ]
OUTH [[アドレス] データ]
OUTW [[アドレス] データ]

OUTB,OUTH,OUTW は、I/O ライトを行ないます。OUTB はバイト、OUTH はハーフ・ワード、OUTW はワード単位でアクセスします。アドレスとデータが省略すると、前回の指定と同じものとみなされます。

＜例> OUTH 2000 55AA
2000H 番地にデータ 55AAH をハーフ・ワードで I/O ライトします。

13.4.6. DCTR コマンド

<書式> DCTR [ALL]

DCTR レジスタを表示します。レジスタは 256 個ありますが、バリッドビットが有効になっているレジスタのみ表示します。ただし、all を指定した場合は、全てのレジスタ値を表示します。DCTR レジスタは、I/O 空間 f2000000h-f2000fffh にマップされています。

13.4.7. ITCR コマンド

<書式> ITCR [ALL]

ICTR レジスタを表示します。レジスタは 128 個ありますが、バリッドビットが有効になっているレジスタのみ表示します。ただし、all を指定した場合は、全てのレジスタ値を表示します。ICTR レジスタは、I/O 空間 fa000000h-fa000fffh にマップされています。

13.4.8. CMCR コマンド

<書式> CMCR [=]VALUE

CMCR（キャッシュメモリ・コントロールレジスタ）に値を設定します。

13.4.9. SFR コマンド

<書式> SFR [レジスタ名 [=データ]]

レジスタ名を指定してデータを省略した場合は、そのレジスタからリードしたデータを表示します。レジスタ名と"="の後にデータを指定した場合には、そのレジスタにデータをライトします。データのサイズは、指定したレジスタの有効サイズで自動的に決定されます。内部 I/O レジスタの詳細については、V831-CPU のマニュアルを参照してください。

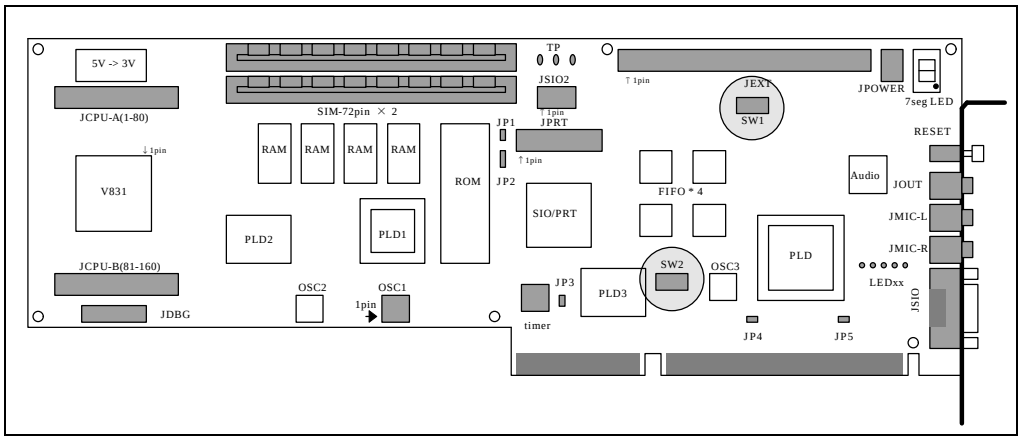
<例 1> SFR
レジスター覧を表示します。
<例 2> SFR IMR
レジスタ IMR の内容を表示します。
<例 3> SFR IMR=55AA
レジスタ IMR にデータ 55AAH をライトします。

14. APPEDIX.B PARTNER モニタ

PARTNER 用のモニタ ROM を使用して、ホストのデバッガと接続して使用する場合の設置方法と使用上の注意事項について説明します。

14.1. スイッチの設定

ボードには、２つの設定用ディップ・スイッチが設けられています。スイッチの箇所を図に示します。



ボードのスイッチの位置(SW1,SW2)

14.1.1. SW1 の設定

SW1 は、汎用の入力ポートのスイッチですが、実装されている MULTI 用のモニタでは、以下の通り使用しています。正しく設定してご使用ください。

SW1 番号	1	2	ボーレート
設定	ON	ON	115200 boud
	OFF	ON	38400 baud
	ON	OFF	19200 baud
	OFF	OFF	9600 baud （出荷時の設定）

ボーレートの設定

SW1 番号	3	4	プロファイラ周期
設定	ON	ON	タイマを使用しない （出荷時の設定）

プロファイラ周期の設定

SW1 番号	5	デバッグモード
設定	ON	7segLED をモニタが使用
	OFF	通常の使用状態 （出荷時の設定）

デバッグモードの設定

SW1 番号	6	ブレーク割り込み
設定	ON	INTP3 を使用 （出荷時の設定）
	OFF	NMI を使用

ブレーク割り込みの設定

SW1-7 は、モニタでは使用していません。

SW1 番号	8	ブート時のバスサイズ（BT16B）及び ROM
設定	ON	32bit（BT16B=0） Flash ROM からブート
	OFF	16bit（BT16B=1） EPROM からブート （出荷時の設定）

ブート時のバスサイズ及び ROM の設定

14.1.2. SW2 の設定

SW2 は、ISA バスの I/O アドレス選択のスイッチです。スイッチの番号 1～8 が ISA バスのアドレス A4～A11 に対応しています（A12～A15 は０固定）。したがって、I/O アドレスとして 000xH～03FxH が選択できます。なおスイッチは、OFF で ”1”、ON で ”0”の値となります。一般的には、20xH～3FxH の間で設定します。

SW2 番号	1	2	3	4	5	6	7	8	
アドレス	A4	A5	A6	A7	A8	A9	A10	A11	I/O アドレス
ON/ OFF	0	0	0	0	0	1	0	0	020xH （出荷時の設定）

I/O アドレスの対応

14.2. PARTNER モニタ

14.2.1. モニタ・ワークRAM

モニタでは、SRAM の最上位の 64KB をワーク用の RAM として使用しています。したがって、FE07-0000H～FE07-FFFFH はユーザ・プログラムでは使用できません。

14.2.2. 割り込み

ユーザプログラムで割り込みを使用する場合には、「12 マスカブル割り込みを使用したアプリケーションの開発」を参照してください。

14.2.3. 強制ブレーク用の割り込み

モニタの通信及び、強制ブレーク（ESC ボタン）で使用する割り込みは、NMI と INTP03 が選択できます。選択基準としては、DMA を使用するプログラムを実行する場合（音声入出力等）には INTP3 を使用し、そうでない場合は NMI を使用することをお薦めします。以下にそれぞれの場合の相違点を示します。

NMI を使用する場合

ユーザプログラム実行中いつでもブレークできますが、DMA が動作中にブレークした場合は、DMA も中断してしまいます（再実行時にも自動的に再開されることはありません）。

INTP03 を使用した場合

ユーザプログラムで INTP03 より高い優先順位の割り込みを使用している場合や、割り込みが許可されていない状態ではブレークできません。但し、DMA はブレーク中も継続して実行されます。

14.2.4. SP の設定

モニタのスタック・ポインタの初期値は、FE06-FFFCH（SRAM の最上位）に設定されています。

14.2.5. リモート接続

デバッガとの接続は、シリアル接続と ISA バス接続が選択できます。設定は、RTSETUP.exe で行います。詳細は、PARTNER のマニュアルを参照下さい。

14.2.6. モニタの実行領域

搭載している ROM は、128K バイトの前半にキャッシュ領域で実行するコードが、後半にアンキャッシュ領域で実行するコードの二つが入ってます。

キャッシュ領域でモニタを動作させた場合の方が、モニタの実行が速いためプログラムのダウンロードも速くなります。通常は、キャッシュ側の設定（出荷時の状態）でご使用下さい。

出荷時の設定では、キャッシュ（JP2 2-3 ショート）となっています。

- Memo -