

JA21000

Product Guide

JAZTEK Technology, Inc.

Features	2
Block Diagram	3
Memory	3
Power configuration.....	5
I/O configuration.....	5
Timer.....	7
Data Access (1Eh, 1Fh).....	9
Interrupt.....	12
Wake up.....	14
Oscillator.....	14
Reset	14
PA & 32768 generator.....	14
Mask Option.....	15
Register Initial Summarized	15
Application Circuit	16
Appendix	17
Macro Library (Sub-program).....	20
MAFLAG bit Definition	20
Sample Rate Table.....	21
Program Example	22
Revision History	24

Features

- Built-in an 8-bit CPU core
- Built-in 80~96 bytes of data RAM
- Built-in 0~1024KB user data ROM
- Built-in 16K~60KB program ROM
- On-chip RC oscillator
- Illegal address reset
- Maximum of 16 programmable I/O pins
- Power saving STOP & HALT modes
- Two 8-bit timers
- I/O State Change wake-up option for all of I/O port
- Operating voltage 2.4V – 5.2V
- Programmable R-option for PA4~7 & PB0~7
- Support 32768Hz crystal oscillator share with Port A
- 1ms interrupt (@ Fsys=2MHz)

General Description

JA21000 series is a series of 3 to 340 seconds (S.R.=6KHz, 4-bit ADPCM) single chip voice synthesizer IC with a PWM Direct Drive circuit or AUD output for transistor application, JA21000 series contains an 8-bit Micro-Controller Units (MCU), 8~16 programmable general I/Os, 16KB~60KB Program ROM, Special register, 0~1024KB Data ROM, Max. of two current type DAC or PWM output direct driving a speaker,

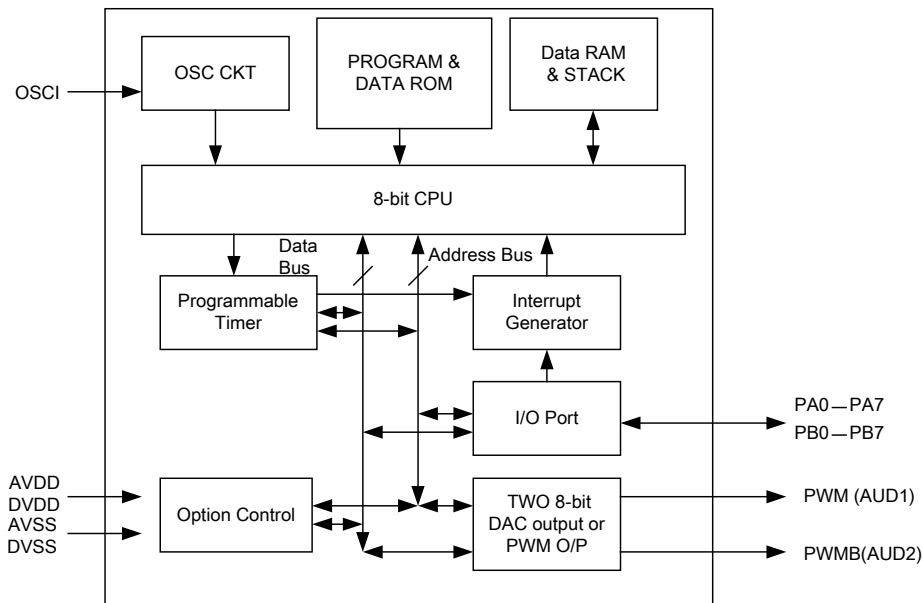
80~96 bytes of user data RAM and 2*8-bit Timers. The advanced sub-micron CMOS process technology ensures JA21000 series high performance, high reliability and sophisticated functionalities. In addition, this chip also provides high sink current port pins. JA21000 series offers the best cost/performance ratios, as a controller, for the industry applications.

Selection Table

Part No.	JA21003	JA21007	JA21014	JA21018	JA21032	JA21043
Voice Cap. (second)	3	7	14	21	32	43
I/O	8	8	8	8	12	12
PROM	16KB	28KB	48KB	60KB	8KB	8KB
DROM	0	0	0	0	96KB	128KB
RAM	80B	80B	80B	80B	96B	96B
Timer (Bits * No.)	8b*2	8b*2	8b*2	8b*2	8b*2	8b*2
PWM	V	V	V	V	V	V
DAC				V	V	V

Part No.	JA21064	JA21085	JA21128	JA21170	JA21256	JA21340
Voice Cap. (second)	64	85	128	170	256	340
I/O	12	12	16	16	16	16
PROM	8KB	8KB	8KB	8KB	8KB	8KB
DROM	192KB	256KB	384KB	512KB	768KB	1024KB
RAM	96B	96B	96B	96B	96B	96B
Timer (Bits * No.)	8b*2	8b*2	8b*2	8b*2	8b*2	8b*2
PWM	V	V	V	V	V	V
DAC	V	V	V	V	V	V

Block Diagram



Pin Assignment

Pin Name	I/O	Internal	Description
DVSS, AVSS	—	—	Negative power supply.
PA0 — PA7 PB0 — PB7	I/O	Pull Hi (*)	Port I/O pins can be configured as input or output. Some pins can configure for special function. Please refer to the I/O configuration description.
DVDD, AVDD	—	—	Positive power supply.
OSCI	I	—	It connects to an external oscillator resistor between OSCI and VDD.
PWM (AUD1), PWMB (AUD2)	O	CMOS or Open Drain	Current type output or PWM type output by mask option. For current type output, it must drive an external transistor and only for JA21018 — JA21340. For PWM type output, it can drive a speaker directly. (8 ohm – 32 ohm) The AUD1 output is provided to JA21018 — JA21340 The AUD2 output is provided to JA21064 — JA21340 only.

(*): The pull high with or without is optional by user program.

Memory

- Memory Mapping

Address	Definition
00h	POWERC (W)
01h	INTC (R/W)
02h	INTF (R/W)
03h	---
04h	---
05h	---
06h	TMR0 (R/W)
07h	TMR0C (R/W)
08h	---
09h	TMR1 (R/W)
0Ah	TMR1C (R/W)
0Bh	PA (R/W)

0Ch	PAC (R/W)
0Dh	PAR (R/W)
0Eh	PB (R/W)
0Fh	PBC (R/W)
10h	PBR (R/W)
11h	---
12h	---
13h	Not used
14h	Not used
15h	Not used
16h	Not used
17h	Not used
18h	Not used
19h	VOLC
1Ah	DA1 (WR)
1Bh	DA2 (WR)
1Ch	DA3 (WR)
1Dh	DA4 (WR)
1Eh	CROMCONT (W)
1Fh	CROMADD (R/W)
20h ~ 2Ah	Special RAM reserved for further expanding for setting internal as internal register
2Bh	---
2Ch	---
2Dh	---
2Eh	---
30h ~ FFh	General purpose Data Memory & Stack
100h ~ 0FFFh	Reserved used
1000h ~ FFFFh	User Program (8K~60KB) Note: Max 1KB for testing program

The low nibble of VOLC controls the AUD1 & the high nibble of VOLC controls the AUD2 (DAC output only.)

- **Data RAM**

Total of 256 bytes of RAM (including the stack, and special register) is available from \$00h to \$FFh.

The stack begins at the address \$FFh and proceeds down to \$00h.

For JA21003 – JA21018: 80 Bytes (B0h – FFh)

For JA21032 – JA21340: 96 Bytes (A0h – FFh)

The address of 00h – 2Fh are designed for special registers.

- **Program ROM**

The JA21000 series includes the Max. of 60K bytes of user ROM which located from \$1000h to \$FFFFh. The 1K bytes of internal test ROM is for testing program. The PROM location of various bodies is presented as follows:

For JA21003 (16KB): C400h – FFFFh (C000h – C3FFh for testing program)

For JA21007 (28KB): 9400h – FFFFh (9000h – 93FFh for testing program)

For JA21014 (48KB): 4400h – FFFFh (4000h – 43FFh for testing program)

For JA21018 (60KB): 1400h – FFFFh (1000h – 13FFh for testing program)

For JA21032 – JA21340 (8KB): E400h – FFFFh (E000h – E3FFh for testing program)

- **Data ROM**

To write the address expander control register of CROMCONT, CROMADD can access max. of 1024KB data. The address from \$00000 to \$FFFFFF. The DROM location of various bodies is illustrated as follows:

Body	JA21032	JA21043	JA21064	JA21085	JA21128	JA21170	JA21256	JA21340
DROM size	96KB	128KB	192KB	256KB	384KB	512KB	768KB	1024KB
DROM Address	00000h 17FFFh	00000h 1FFFFh	00000h 2FFFFh	00000h 3FFFFh	00000h 5FFFFh	00000h 7FFFFh	00000h BFFFFh	00000h FFFFFFh

- **Reset and IRQ vectors**

The address of RESET and IRQ are located from \$FFFC to \$FFFF. The interrupt vectors should be specified in the program as follows:

```

ORG $FFFC
JMP RESET_VECTOR ($FFFC, $FFFD)
JMP INT_VECTOR ($FFFE, $FFFF)

```

- **Stack Pointer**

The stack pointer is set from \$FFh after power on.

Power configuration

The JA21000 series provide two power saving mode, one is the HALT mode and the other is the STOP mode. When writing "1" to the HALT bit, the system will enter the HALT mode and the system clock will stop whereas the internal RC will free run continuously. The timer overflow will wakeup the system. When writing "1" to the STOP bit, the system will enter the STOP mode and the system clock, internal RC oscillator will be stop. Only the external interrupt (PX change state) can wakeup the system. When the system is overflow, the HALT and STOP bit will be cleared to "0" automatically.

POWERC

Register	Address	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
POWERC	00h	---	---	---	STOP	---	---	---	HALT

In the HALT mode, the 32k(PA) is On & Fosc is OFF. In the STOP mode, the 32k(PA) & Fosc are all OFF.

I/O configuration

Max. of 16 I/Os (grouped into 2 I/O ports, PA and PB) are provided. All of these I/O ports can be used as input and output operations. For input operation, these ports are non-latched and for output operation all the data are latched and remain unchanged till the output latch is re-written.

Each I/O line has its own control register (PAC and PBC) to control the input/output configuration. If the global interrupt(INTC.0) is disabled and the corresponding I/O interrupt(INTC.4 & INTC.5) is enabled, it will activate the system and its corresponding flag will be set when the corresponding I/O state is changed.

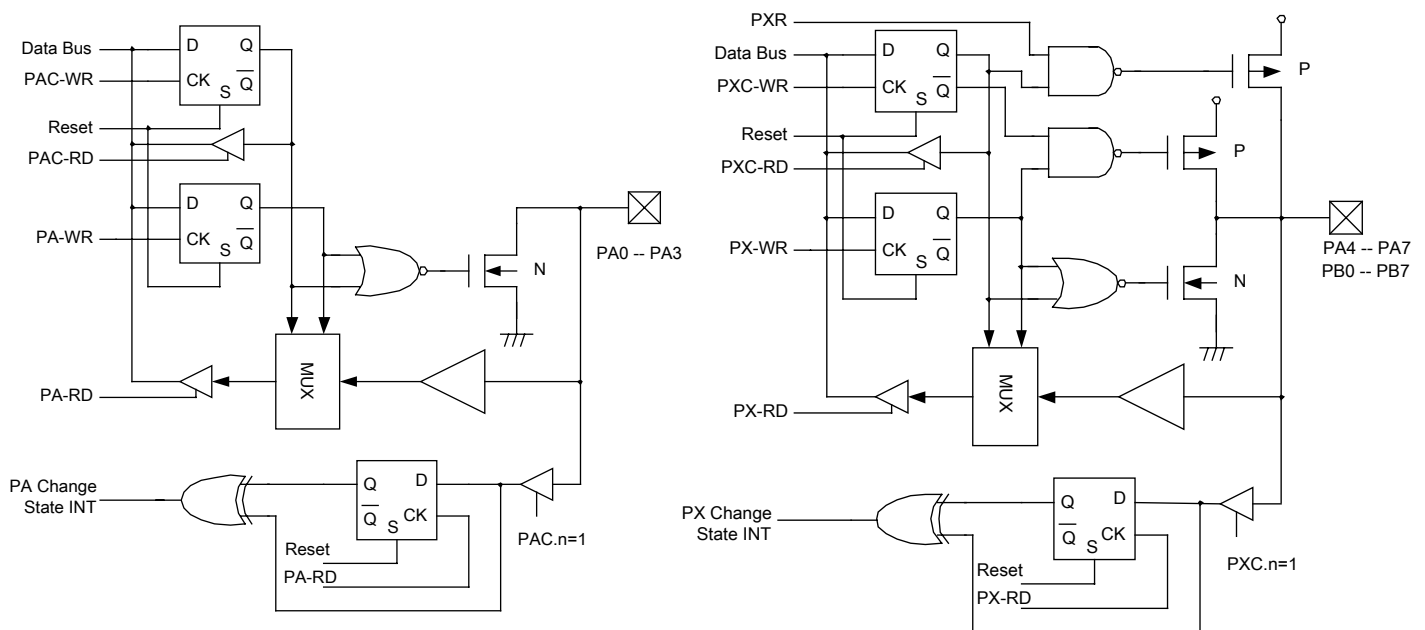
Part No.	I/O	Port A	Port B
JA21340	16	8	8
JA21256	16	8	8
JA21170	16	8	8
JA21128	16	8	8
JA21085	12	8	4
JA21064	12	8	4
JA21043	12	8	4
JA21032	12	8	4
JA21018	8	4	4
JA21014	8	4	4
JA23024	8	4	4
JA21003	8	4	4

PA Configuration

Label	Address	Function	R/W	Default
PA	0Bh	PA data input/output	R/W	FF
PAC	0Ch	PA direction control, 1=input 0=output	R/W	FF
PAR	0Dh	PA pull-high resistor option, 1=With, 0=Without	R/W	FX

1. The PA0 – PA3 they are defined as NMOS open drain output only when PAC is set as output mode.
2. The PA0 – PA3 without pull-high resistor in input mode.

For the others of the Port A, they can configure as follows :


PB Configuration

Label	Address	Function	R/W	Default
PB	0Eh	PB data input/output	R/W	FF
PBC	0Fh	PB direction control, 1=input 0=output	R/W	FF
PBR	10h	PB pull-high resistor option, 1=With, 0=Without	R/W	FF

1. Its diagram is shown above. The output mode (PB0 — PB7) is the same as PA4 — PA7.
2. The pull-high resistor will be disabled automatically when port is programmed as output mode.

Timer

There are 2 sets of programmable timer in JA21000 series, they are TMR0 (06h & 07h) and TMR1 (09h & 0Ah). Both TMR0 and TMR1 are 8-bit timer.

TMR0 (06h) & TMR0C (07h)

The TMR0 is a programmable 8-bit count-up counter. The counter registers is named as TMR0 (06h). The clock source may come from Fosc/4 or internal RC clock. The TMR0C is its control register and the default value is 00. The definition of TMR0C is listed below.

Labels	Bits	Function
TON0 (TMR0)	0	Timer0 enable/disable definition bit 0 = Disable; 1 = Enable
TS0, TS1, TS2	1 - 3	Timer clock source selection bits
—	4	Reserved, not used
TMR	5	To define the clock source is with pre-scale counter 0: Timer
TM0 TM1	6, 7	To define the operation mode 00= Timer mode (internal system clock; Fosc/4) 11 = Timer mode (internal system clock; Fosc or 32768 Hz option by X12PA)

Notes: 1. The event counter behaves from low to high transition.

2. When the X12PA=0 and (TM0, TM1)=(1, 1), then the clock source is 32kHz.

3. When the X12PA=1 and (TM0, TM1)=(1, 1), then the clock source is the Fosc(4MHz).

4. X12PA is a configuration option by user (code layer option).

In timer mode, user should write an initial value to the counter register (TMR0) and then to enable the TON0 (TMR0C.0). The TMR0 will count-up with the rate of TS0 – TS2 & TM0, TM1 setting. When the timer is overflow (ffh --> 00h), system will generate an interrupt and the corresponding flag (INTF.1) will be set to "1", if both the global interrupt bit (INTC.0) and the corresponding bit (INTC.2) are enabled. After timer is overflow, the initial value will reload to counter register automatically unless the TMR0 is disabled.

TS2	TS1	TS0	TMR Rate
0	0	0	1:2
0	0	1	1:4
0	1	0	1:8
0	1	1	1:16
1	0	0	1:32
1	0	1	1:64
1	1	0	1:128
1	1	1	1:256

TMR1 (09h) & TMR1C (0Ah)

The TMR1 is a programmable 8-bit count-up counter. The counter registers is named TMR1 (09h) respectively. The clock source is defined by the register of TMR1C. The TMR1C is its control register and the default value is 00. The definition of TMR1C is listed below.

Labels	Bits	Function
TON1	0	Timer1 enable/disable definition bit 0 = Disable; 1 = Enable
TS0, TS1, TS2	1 - 3	Timer clock source selection bits
—	4	Reserved, not used
---	5	---
TM0 TM1	6, 7	To define the operation mode 00= Timer mode (internal system clock; Fosc/4) 11= Timer mode (internal system clock; Fosc or 32768 Hz option by X12PA)

- Notes: 1. The event counter behaves from low to high transition.
 2. When the X12PA=0 and (TM0, TM1)=(1, 1), the clock source is 32kHz.
 3. When the X12PA=1 and (TM0, TM1)=(1, 1), the clock source is the Fosc(4MHz).
 4. X12PA is a configuration option by user (code layer option).

In timer mode, user should write an initial value to the counter registers (TMR1) and then to enable the TON1 (TMR1C.0). The TMR1 will count-up with the rate of TS0 – TS2 & TM0, TM1 setting. When the timer is overflow (ffffh --> 0000h), system will generate an interrupt and the corresponding flag (INTF.2) will be set to "1", if both the global interrupt bit (INTC.0) and the corresponding bit (INTC.3) are enabled. After timer is overflow, the initial value will reload to counter register automatically unless the TMR1 is disabled.

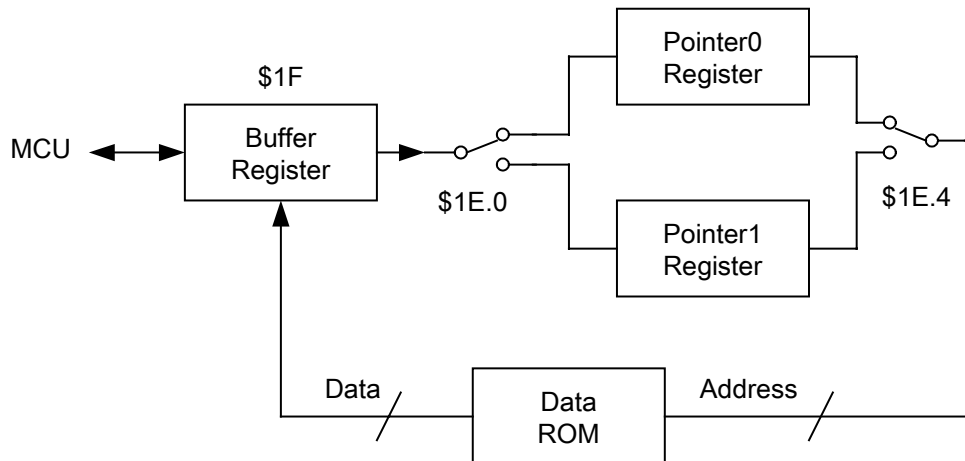
TS2	TS1	TS0	TMR Rate
0	0	0	1:2
0	0	1	1:4
0	1	0	1:8
0	1	1	1:16
1	0	0	1:32
1	0	1	1:64
1	1	0	1:128
1	1	1	1:256

Data Access (1Eh, 1Fh)

For JA21003~JA21018 bodies, data is accessed by index addressing instructions. So the CROMRCONT (1Eh) and CROMADD (1Fh) registers are useless.

For JA21032~JA21340 bodies, data is accessed by CROMCONT & CROMADD to addressing and read data.

ADDRESS	Name
1Eh	CROMCONT (W)
1Fh	CROMADD(R/W)



• **Write Initial Address**

There are 2 address pointer for the data memory (Pointer0, Pointer1). At the initial state, the host controller should select one of the pointers to write the start address, which is defined by \$1E.0. The start address must be 3 bytes (ADDR_L, ADDR_M, ADDR_H). If over 3 bytes of data written to the pointer, then the 1st position is written again. After 3 bytes of start address are written, then the host controller can read the data from \$1F register.

Code(\$1E.0)	Pointer	Default
0	0	V
1	1	

• **Read Data**

After the 3 bytes of initial address are written down, then the host controller can read the data from \$1F, which is defined by the \$1E.4. And the read command will reset the address counter of corresponding pointer to ADDR_L. After read the \$1F one time then the corresponding pointer address will increment automatically.

Code(\$1E.4)	Pointer	Default
0	0	V
1	1	

Example for JA21003 – JA21018 bodies :

The start address: \$3EC0 and using pointer0

Program:

```
ADDRL: equ $1A
```

```
ADDRH: equ ADDR+1
```

; To declare the ADDR, ADDRH location, and their location must be continuous.

...

```
lda #$C0
```

```
sta ADDL
```

```
lda #$3E
```

```
sta ADDH
```

ReadLoop:

```
ldx #0 ; Clear the X register to zero
```

```
lda (ADDRL,x) ; Read the data from the start address
```

```
sta BUFFER ; Save the read data to Buffer register
```

```
inc ADDRL
```

```
bne +2 ; If ADDRL 0, then escape the next instruction
```

```
inc ADDRH ; If ADDR=0, then generate a carry to ADDRH
```

```
... ; To process the read data
```

```
jmp ReadLoop
```

- **Example for JA21032 – JA21340 bodies :**

a. One pointer application

The start address: \$3EC0 and using pointer0

Program:

...

```
lda #00
```

```
sta $1E ; Set the pointer0 to addressing & read
```

```
lda $1F ; Reset the address counter
```

```
lda #$C0 ; ADD_L data
```

```
sta $1F
```

```
lda #$3E ; ADD_M data
```

```
sta $1F
```

```
lda #$00 ; ADD_H data
```

```
sta $1F
```

```
nop
```

```
nop ; To delay 2 nop instruction
```

```
lda $1F ; Read the $3EC0 data, and the address pointer change to $3EC1
```

```
.... ; Data process
```

```
lda $1F ; Read $3EC1 data, , and the address pointer change to $3EC2
```

...

b. Two pointers application

Section1 start address: \$003EC0 and using pointer0 to read

Section2 start address: \$01CF00 and using pointer1 to read

Program:

...

```
lda #00
```

```
sta $1E ; Set the pointer0 to addressing & read
```

```
lda $1F ; Reset the address counter
```

```
lda #$C0 ; ADD_L data of pointer0
```

```
sta $1F
```

```
lda #$3E ; ADD_M data of pointer0
```

```
sta $1F
```

```
lda #$00 ; ADD_H data of pointer0
```

```
sta $1F
```

```
.....
```

```
lda #01
sta $1E ; Set the pointer1 to addressing & pointer0 to read
lda $1F ; Reset the address counter
lda #$00 ; ADD_L data of pointer1
sta $1F
lda #$CF ; ADD_M data of pointer1
sta $1F
lda #$01 ; ADD_H data of pointer1
sta $1F
.*****
,
lda $1F ; Read the $3EC0 data, and the address pointer change to $3EC1
.... ; Data process
lda $1F ; Read $3EC1 data, , and the address pointer change to $3EC2
...
.*****
,
lda #$10
sta $1E ; Set pointer1 to read & pointer0 to addressing
lda $1F ; Read the $01CF00 data, and the address pointer change to $01CF01
.... ; Data process
lda $1F ; Read $01CF01 data, , and the address pointer change to $01CF02
....
```

Interrupt

When an interrupt is generated, the interrupt vector (\$FFFE) will be loaded to PCL and PCH (PCL, PCH: Program Counter) and the original data in PCL, PCH & Status register content will be saved in stack. The corresponding interrupt bit (INTF.X) will be set (Write in "1") Automatically. After the instruction of "RTI" is executed, the original data will be pulled out from stack and saved to original registers. When the interrupt program is executed, the corresponding bit of interrupt flag (INTF.X) should be cleared (Write in "0") by user program.

Interrupt request is generated by IRQ bar and will keep 9 ~ 16 cycles. If commands between "SEI" and "CLI" are over 9 instruction cycles, the interrupt may be lost but corresponding interrupt bit (INTF.X) will still be set ("1"). User can check INTF to avoid loss of interrupt.

The interrupt source include one of the following conditions:

- Timer overflow
- PA, PB change state input

INTC (R/W)

Register	Bit No.	Label	Function
INTC	0	INTE	Global interrupt enable bit (1= Enabled; 0 = Disabled)
	1	INT	External INT pin interrupt Enable bit (1= Enabled; 0 = Disabled)
	2	TMR0	TMR0 interrupt Enable bit (1= Enabled; 0 = Disabled)
	3	TMR1	TMR1 interrupt Enable bit (1= Enabled; 0 = Disabled)
	4	PAI	Port A change state interrupt Enable bit (1= Enabled; 0 = Disabled)
	5	PBI	Port B change state interrupt Enable bit (1= Enabled; 0 = Disabled)
	6	1ms	1ms interrupt Enable bit (1= Enabled; 0 = Disabled)
	7	---	---

INTF (R/W)

Register	Bit No.	Label	Function
INTF	0	INTF	External INT interrupt flag bit (1= Active; 0 = Inactive)
	1	TMR0F	TMR0 timer interrupt flag bit (1= Active; 0 = Inactive)
	2	TMR1F	TMR1 timer interrupt flag bit (1= Active; 0 = Inactive)
	3	---	---
	4	PAF	Port A change state interrupt flag bit (1= Active; 0 = Inactive)
	5	PBF	Port B change state interrupt flag bit (1= Active; 0 = Inactive)
	6	1ms	1ms interrupt flag bit (1= Active; 0 = Inactive)
	7	---	---

Note : 1. The time period of 1ms is generated when the system frequency is 2MHz.

2. The INTF can write-in "0" only. The data of "1" is ineffective. If want to clear INTF, don't use the command "AND", must write-in "0" to INTF immediately, see the Example for INTF process.
3. After the interrupt program is executed, the corresponding flag should be cleared by software.
4. If Port change into interrupt, Port must read again before leave interrupt. If not read again the Port interrupt will unstable.

Example for INTF process

```
INTREQUEST:
    sei
    pha
    txa
    pha
    lda    INTF                ;check TMR0 interrupt
    bit    #$02
    beq    OtherINT
    jsr    TMR0_interrupt
OtherINT:
    ...
    pla
    tax
    pla
    cli
    rti

TMR0_interrupt:
    lda    #$FD                ;clear TMR0 interrupt flag;INTF only write "0"
    sta    INTF
    ...
    rts
```

Example for PB interrupt process

```
Program:
    ...
    lda    PB                  ;before set PB interrupt flag must read PB
    lda    #$21                ;set PB interrupt flag
    sta    INTC
    ...

INTREQUEST:
    pha
    txa
    pla
    lda    #$01
    bit    INTF
    ...
    lda    #$20
    bit    INTF
    beq    OtherINT
    jsr    PB_interrupt
OtherINT:
    ...
    pla
    tax
    pla
    cli
```

```

rti
PB_interrupt:
    lda    #$DF          ;INTF only write 0, can't write 1
    sta    INTF
    lda    PB            ; Port interrupt must read again before leave
    ...
    rts

```

Wake up

The wake-up source include one of the following condition:

- Timer overflow
- PA and PB, change state input, if use port to wake up, must read the port before power down.

When the global interrupt bit (INTE) is disabled and the corresponding I/O port (PA and PB) interrupt bit is enabled, the wakeup source will generate wakeup without interrupt.

After the interrupt is generated, the system will be activated from HALT mode or the STOP mode and then the program counter will increase and the next instruction will be executed continuously. All of the content of registers will be unchanged.

Oscillator

The JA21000 SERIES supports RC type oscillation circuit. For the RC type, an external resistor connection between OSCI and VDD. The system will delay 64 clocks after power is turned on or the system is waken-up from the STOP mode

Reset

There are 3 conditions will reset the system

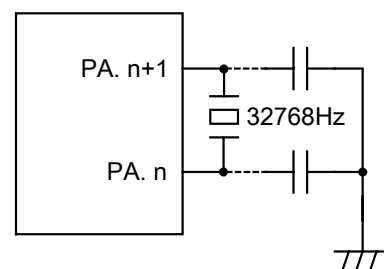
- Power on reset
- Illegal address generation
- VDD voltage lower than 1.8V

When the system is reset, the reset vector (\$FFFC, \$FFFD) will load to PCL, PCH and the instruction in the vector will be executed immediately. Only the power-on reset (cold reset) can initialize the internal register, the others reset (warm reset) can't change the content of all registers.

PA & 32768 generator

The PA port can share 2 pins to generate 32K frequency for internal timer by mask option. For various bodies, the pins of 32K generator would be different.

Part No.	Port A	Description
JA21340	PA6~7	I/O or Crystal mode is selectable by X12PA
JA21256	PA6~7	I/O or Crystal mode is selectable by X12PA
JA21170	PA6~7	I/O or Crystal mode is selectable by X12PA
JA21128	PA6~7	I/O or Crystal mode is selectable by X12PA
JA21085	PA6~7	I/O or Crystal mode is selectable by X12PA
JA21064	PA6~7	I/O or Crystal mode is selectable by X12PA
JA21043	PA6~7	I/O or Crystal mode is selectable by X12PA
JA21032	PA6~7	I/O or Crystal mode is selectable by X12PA
JA21018	PA2~3	I/O or Crystal mode is selectable by X12PA
JA21014	PA2~3	I/O or Crystal mode is selectable by X12PA
JA23024	PA2~3	I/O or Crystal mode is selectable by X12PA
JA21003	PA2~3	I/O or Crystal mode is selectable by X12PA



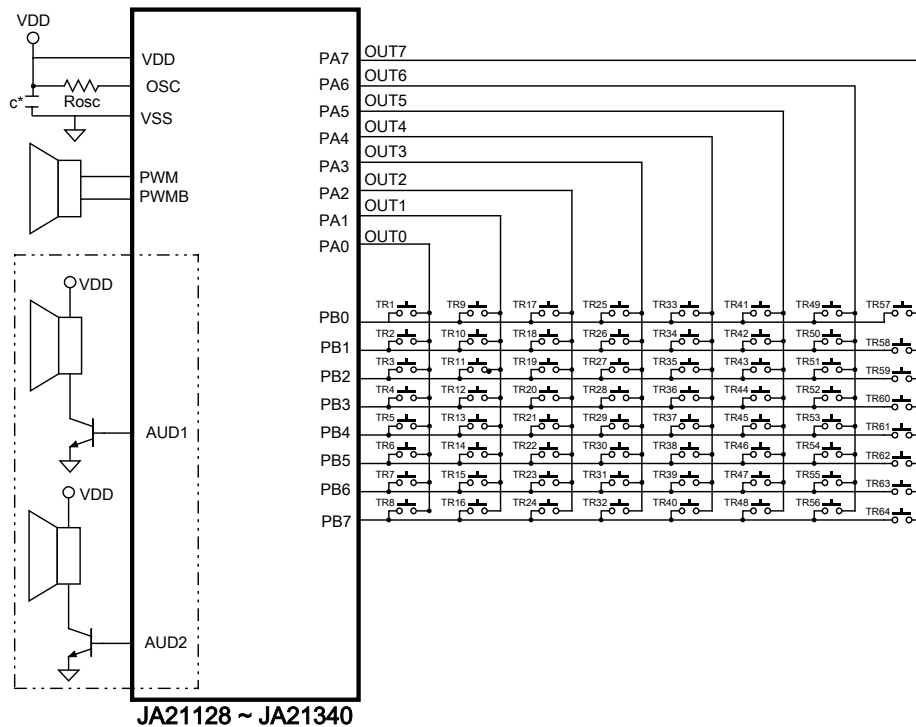
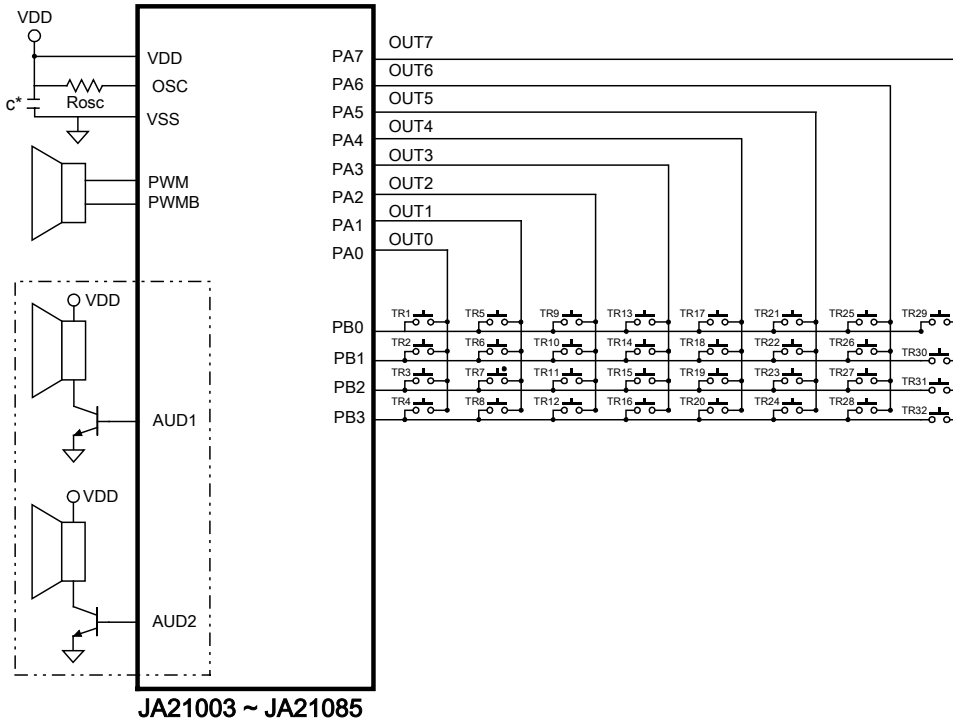
Mask Option

- a. TMR0 & TMR1 length: (8-bit)
- b. AUD1, AUD2 output: (DAC, PWM)
- c. PC pull high: (Without)
- d. PA pin option: (I/O, 32K oscillator)
- e. PC/1mS interrupt source : (1mS)

Register Initial Summarized

Register	Address	Power on reset	Reset (WDT Overflow, RESB Active, Illegal address)	Reset (WDT overflow from Halt)
INTC	01h	0000 0000	0000 0000	uuuu uuuu
INTF	02h	0000 0000	0000 0000	uuuu uuuu
TMRL	06h/09h	uuuu uuuu	uuuu uuuu	uuuu uuuu
TMRC	07h/0Ah	0000 0000	0000 0000	uuuu uuuu
PA, PB	0Bh/0Eh	1111 1111	1111 1111	uuuu uuuu
PAC, PBC	0Ch/0Fh	1111 1111	1111 1111	uuuu uuuu

Application Circuit



- Note :
1. For JA21003 JA21018, the max. of Easy N is Easy 16.
 2. For JA21032 JA21085, the max. of Easy N is Easy 32.
 3. AUD1 and AUD2 is shared with PWM and PWMB and PWM/DAC mode is selected by code option.
 4. The capacitor "c" is suggested as 0.1μF. (if with motor application, a 10-100μF capacitor should be added in addition)
 5. The IC substrate should be connected to VSS

Appendix
• Data Format

After JAICE compile the input data input and transfer to various format of data. The data format defines as follows.

a. The format of speech

1. 8-bit PCM data format

Addr0								Addr1							
b7	b6	b5	b4	b3	b2	b1	b0	b7	b6	b5	b4	b3	b2	b1	b0
System Clock				Type				Sample rate							
Addr2								Addr3							
b7	b6	b5	b4	b3	b2	b1	b0	b7	b6	b5	b4	b3	b2	b1	b0
Length (low byte)								Length (High byte)							
Addr4								Addr5							
b7	b6	b5	b4	b3	b2	b1	b0	b7	b6	b5	b4	b3	b2	b1	b0
Data0								Data1							

...

2. 4-bit ADPCM

Addr0								Addr1							
b7	b6	b5	b4	b3	b2	b1	b0	b7	b6	b5	b4	b3	b2	b1	b0
System Clock				Type				Sample rate							
Addr2								Addr3							
b7	b6	b5	b4	b3	b2	b1	b0	b7	b6	b5	b4	b3	b2	b1	b0
Length (low byte)								Length (High byte)							
Addr4								Addr5							
b7	b6	b5	b4	b3	b2	b1	b0	b7	b6	b5	b4	b3	b2	b1	b0
Data1				Data0				Data3				Data2			

...

3. 5-bit ADPCM

Addr0								Addr1							
b7	b6	b5	b4	b3	b2	b1	b0	b7	b6	b5	b4	b3	b2	b1	b0
System Clock				Type				Sample rate							
Addr2								Addr3							
b7	b6	b5	b4	b3	b2	b1	b0	b7	b6	b5	b4	b3	b2	b1	b0
Length (low byte)								Length (High byte)							
Addr4								Addr5							
b7	b6	b5	b4	b3	b2	b1	b0	b7	b6	b5	b4	b3	b2	b1	b0
Sign bits								Data1				Data0			

...

4. Silence data format

Addr0								Addr1							
b7	b6	b5	b4	b3	b2	b1	b0	b7	b6	b5	b4	b3	b2	b1	b0
System Clock				Type				Sample rate							
Addr2								Addr3							
b7	b6	b5	b4	b3	b2	b1	b0	b7	b6	b5	b4	b3	b2	b1	b0
Length (low byte)								Length (High byte)							

5. The format of raw data (binary code)

Addr0								Addr1							
b7	b6	b5	b4	b3	b2	b1	b0	b7	b6	b5	b4	b3	b2	b1	b0
System Clock				Type				Check Sum							
Addr2								Addr3							
b7	b6	b5	b4	b3	b2	b1	b0	b7	b6	b5	b4	b3	b2	b1	b0
Length (low byte)								Length (High byte)							
Addr4								Addr5							
b7	b6	b5	b4	b3	b2	b1	b0	b7	b6	b5	b4	b3	b2	b1	b0
Data0								Data1							

6. End of section

Addr0							
b7	b6	b5	b4	b3	b2	b1	b0
\$F				\$F			

The System Clock code (b7 – b4)

System clock	1MHz	2MHz	4MHz	6MHz	8MHz
Code	0001	0010	0100	0110	1000

The type code (b3 – b0)

Type	Silence	4-bit (ADPCM)	8-bit (PCM)	8-bit (Raw data)	5-bit (ADPCM)		
Code	0000	0001	0010	0100	1000		

• Instruction Set
For JA21003 – JA21340 bodies

Instruction	Operation	Flag	Addressing Mode	Note
ADC	A+M+C A, C	N, Z, C, V	IMM, ZP	
AND	A M A	N, Z	IMM, ZP	
BCC	Jump if C=0	—	REL	
BCS	Jump if C=1	—	REL	
BEQ	Jump if Z=1	—	REL	
BIT	A M, M7 N, M6 V	N, Z, V	ZP, ABS	
BMI	Jump if N=1	—	REL	
BNE	Jump if Z=0	—	REL	
BPL	Jump if N=0	—	REL	
BVC	Jump if V=0	—	REL	
BVS	Jump if V=1	—	REL	
CLC	0 C	C=0	IMP	
CLI	0 I	I=0	IMP	
CLV	0 V	V=0	IMP	
CMP	A – M	N, Z, C	IMM, ZP, ZPX	
CPX	X – M	N, Z, C	IMM, ZP	
DEC	M–1 M	N, Z	ZP, ZPX	
DEX	X–1 X	N, Z	IMP	
EOR	A.XOR. M A	N, Z	IMM, ZP	
INC	M+1 M	N, Z	ZP	
INX	X+1 X	N, Z	IMP	

JMP	(PC+1) (PC+2)	PCL, PCH	—	ABS, IND	
JSR	(PC+1) (PC+2) PC+2	PCL, PCH STACK	—	ABS	
LDA	M	A	N, Z	IMM, ZP, ZPX, ABS, ABSX, INDX	
LDX	M	X	N, Z	IMM, ZP	
NOP	—		—	IMP	
ORA	A.OR. M	A	N, Z	IMM, ZP	
PHA	A	STACK	—	IMP	
PHP	P (status)	STACK	—	IMP	
PLA	STACK	A	N, Z	IMP	
PLP	STACK	P (status)	ALL	IMP	
ROL	C	ACC or M	C	N, Z, C	ACC, ZP
ROR	C	ACC or M	C	N, Z, C	ACC, ZP
RTI	STACK STACK-1 STACK-2	P PCH PCL	ALL	IMP	
RTS	STACK PC+1	PC PC	—	IMP	
SBC	A-M- C	A	N, Z, C, V	IMM, ZP	
SEC	1	C	C=1	IMP	
SEI	1	I	I=1	IMP	
STA	A	M	—	ZP, ZPX, INDX	
STX	X	M	—	ZP, ABS	
TAX	A	X	N, Z	IMP	
TSX	S	X	N, Z	IMP	
TXA	X	A	N, Z	IMP	
TXS	X	S	—	IMP	

Note: For the JA21003 – JA21018 bodies, the instructions of Y index or address mode with Y index are useless.

Note:

B: Break flag
 N: Negative flag
 Z: Zero Flag
 C: Carry flag
 I: Interrupt disable flag
 D: Decimal flag

V: Overflow flag
 A (ACC): Accumulator
 X: Index X register
 Y: Index Y register
 S: Stack pointer
 M: Memory

Address Mode:

ACC: Operation in ACC; A
 IMM: Immediate data; #dd
 ZP: Zero page address; aa
 ZPX: Zero page address with index X; aa, X
 ZPY: Zero page address with index Y; aa, Y
 ABS: Absolute address; aaaa
 ABSX: Absolute address with index X; aaaa, X
 ABSY: Absolute address with index Y; aaaa, Y
 REL: Relative address; aa
 IMP: Implied address;

Note:

- #dd: A 8-bit data, the range can be #00 — #FF
- aa: An 8-bit address; the range can be 00 — FF
- aaaa: A 16-bit address; the range can be 0000 — FFFF

- **JA21XXX series Macro Instruction definition**

a. Macro Library (Sub-program)
MAPPING.ASM (LMAPPING.ASM)

Description:

Set the special register or RAM location.

INTREQUEST.ASM (L INTREQUEST.ASM)

Description:

Set the AUD output initialization and start address, voice length, sample rate, and enable interrupt function and Implement the voice synthesis for ADPCM or PCM format

Note:

1. The macro files of MAPPING, INTREQUEST are used for JA21032 – JA21340 bodies.
2. The macro files of LMAPPING, LINTREQUEST are used for JA21003 – JA21018 bodies.

User Program Definition
Data RAM & Register Definition (mapping.asm)

```

...
DAC0BUF equ $E0
VOICE equ DAC0BUF+1 ($E1)
CODETEMP equ VOICE+1 ($E2)
LENGL equ CODETEMP+1 ($E3)
LENGH equ LENGL+1 ($E4)
GNBUF equ LENGH+1 ($E5)
BUFFER equ GNBUF+1 ($E6)
ADDRL equ BUFFER+1 ($E7)
ADDRM equ ADDRL+1 ($E8)
ADDRH equ ADDR+1 ($E9)
MAFLAG equ ADDR+1 ($EA)
SYSCLOCK equ MAFLAG+1 ($EB)
REPEAT equ SYSCLOCK+1 ($EC)
...

```

Note:

1. User should reserve \$FF -- \$F8 for interrupt stack at least.
2. The RAM address of \$E0 -- \$EC is reserved for the sub-program of voice synthesizer.

MAFLAG bit Definition

End/Normal				Stop/Play	SR		PWM/DAC
------------	--	--	--	-----------	----	--	---------

- a. B7 (END/Normal): When voice is end, B7 (End/Normal) will be set to "1", user can check this bit to know if voice is end or not.
- b. B3 (Stop/Play): Stop the voice (B3=1) or Start the voice (B3=0)
- c. B2 (SR): To set the sample rate is User defined (B2=1) or Default (B2=0)
- d. B0 (PWM/DAC): To define the audio output is PWM (B0=1) or DAC (B0=0) type

Sample Rate Table

SR	CODE	SR	CODE	SR	CODE
3.0K	AC	5.2K	CF	9.5K	E5
3.1K	AF	5.3K	D0	9.8K	E6
3.2K	B1	5.4K	D1	10.2K	E7
3.3K	B4	5.6K	D3	10.7K	E8
3.4K	B6	5.7K	D4	11.1K	E9
3.5K	B8	5.8K	D4	11.6K	EA
3.6K	BA	6.0K	D6	12.2K	EB
3.7K	BC	6.1K	D7	12.8K	EC
3.8K	BE	6.2K	D7	13.5K	ED
3.9K	BF	6.4K	D8	14.2K	EE
4.0K	C1	6.6K	DA	15.1K	EF
4.1K	C3	6.7K	DA	16.0K	F0
4.2K	C4	6.9K	DB	17.1K	F1
4.3K	C5	7.1K	DC	18.3K	F2
4.4K	C7	7.3K	DD	19.7K	F3
4.5K	C8	7.5K	DE	21.3K	F4
4.6K	C9	7.8K	DF	23.3K	F5
4.7K	CA	8.0K	E0	25.6K	F6
4.8K	CB	8.3K	E1	28.4K	F7
4.9K	CC	8.5K	E2	32.0K	F8
5.0K	CE	8.8K	E3		
5.1K	CE	9.1K	E4		

Program Example

1. Special register and data RAM setting

```
.include mapping.asm (User RAM definition)
```

```
...
```

2. Program start setting and include sub-program

```
org $xxxx ; User program start
```

```
START:
```

```
.include INTREQUEST.asm; Include the voice processing sub-program
```

```
...
```

```
...
```

3. Voice play setting for user program

```
*****
```

```
; This program is used to call the voice synthesizer  
lda #$aa ; Set the volume  
sta VOLC  
lda #$00  
sta VOICE ; Initialize the VOICE flag  
lda #$02 ; Start play bit, default sample rate, mono & DAC output  
sta MAFLAG ; Set the MAFLAG register  
  
lda #$XX ; Look up sample rate table to set sample  
sta TMR1L ; If have set SR flag in MAFLAG  
  
lda #$XX  
sta ADDR1  
lda #$XX  
sta ADDR2  
lda #$XX ; The start address can get from *.idx file  
sta ADDRH ; To set the voice start address,  
lda #$nn  
sta REPEAT  
jsr INITIAL ; To call the voice initial sub-program  
; If JA21003 – JA21018 bodies INITIAL will be replaced by LINITIAL  
jsr PLAY ; To call the voice play sub-program  
; If JA21003 – JA21018 bodies PLAY will be replaced by LPLAY
```

```
*****
```

```
;
```

```
...
```

```
...
```

4. Check the voice play is terminated or not

```
; User to confirm the voice whether is terminated or not?
```

```
CheckVoice:
```

```
lda $80
```

```
bit MAFLAG
```

```
beq    CheckVoice    ; If voice is playing
jmp    OtherProgram  ; If voice is end
...
```

5. Interrupt request declaration

INTREQUEST:

```
    pha
    txa
    pha
lda   #$01
bit   INTF
...
lda   #$02
bit   INTF
beq   OtherINT
jsr   VOICEINT    ; Voice synthesizer sub-program label
; If JA21003 – JA21018 bodies VOICEINT will be replaced by LVOICEINT
OtherINT:
...
    pla
    tax
    pla
    cli
    rti
```

6. Include the voice synthesizer

```
.include INTREQUEST.ASM
```

7. Power-on and Interrupt vector declaration

```
org    $ffc
dw     START    ; Power-on or reset start vector
dw     INTREQUEST
end
```

Revision History

Date	Reversion #		Page
2003.5.8	0.0	Original	---
2003.7.1	0.1	1. Revised Operating Voltage 2. Modified MAFLAG bit Definition	1 21
2004.5.12	0.2	1. Removed the "DADIS" flag 2. Modify JA21032~JA21340 Instruction Set 3. Modify Sample Rate Table	4 17 20
2005.4.15	0.3	1. Add ADPCM 5 bit format	16