aPR33A3

CPU Serial Mode (C1.1)

Datasheet

Recording voice IC

APLUS INTEGRATED CIRCUITS INC.

Address:

3 F-10, No. 32, Sec. 1, Chenggung Rd., Taipei, Taiwan 115, R.O.C.

TEL:

886-2-2782-9266

FAX:

886-2-2782-9255

WEBSITE :

http://www.aplusinc.com.tw

Technology E-mail:

service@aplusinc.com.tw

Sales E-mail:

sales@aplusinc.com.tw

■ FEATURES

- Operating Voltage Range: 3V ~ 6.5V
- Single Chip, High Quality Audio/Voice Recording & Playback Solution
  - ◆ No External ICs Required
  - ◆ Minimum External Components
- User Friendly, Easy to Use Operation
  - ◆ Programming & Development Systems Not Required
- 680 sec. Voice Recording Length in aPR33A3
- Powerful 16-Bits Digital Audio Processor.
- Nonvolatile Flash Memory Technology
  - ◆ No Battery Backup Required
- External Reset pin.
- Powerful Power Management Unit
  - ◆ Very Low Standby Current: 1uA
  - ◆ Low Power-Down Current: 15uA
  - ◆ Supports Power-Down Mode for Power Saving
- Built-in Audio-Recording Microphone Amplifier
  - ◆ No External OPAMP or BJT Required
  - ◆ Easy to PCB layout
- Configurable analog interface
  - ◆ Differential-ended MIC pre-amp for Low Noise
  - ◆ High Quality Line Receiver
- High Quality Analog to Digital and PWM module
  - ◆ Resolution up to 16-bits
- Up To Maximum 1024 Voice Sections controlled through 5 pins only
- Built-in Memory-Management System

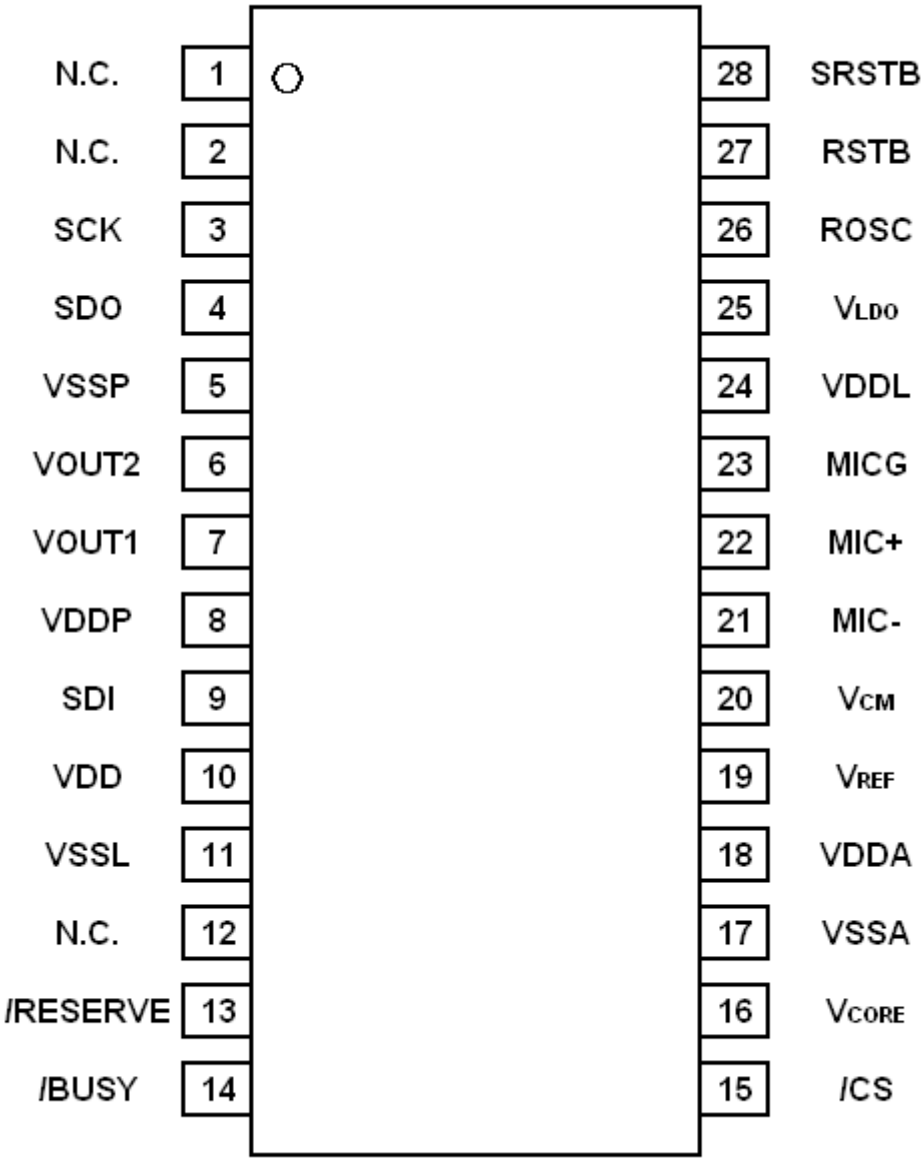
■ DESCRIPTION

Today's consumers demand the best in audio/voice. They want crystal-clear sound wherever they are in whatever format they want to use. APLUS delivers the technology to enhance a listener's audio/voice experience.

The aPR33A series are powerful audio processor along with high performance audio analog-to-digital converters (ADCs) and digital-to-analog converters (DACs). The aPR33A series are a fully integrated solution offering high performance and unparalleled integration with analog input, digital processing and analog output functionality. The aPR33A series incorporates all the functionality required to perform demanding audio/voice applications. High quality audio/voice systems with lower bill-of-material costs can be implemented with the aPR33A series because of its integrated analog data converters and full suite of quality-enhancing features such as sample-rate convertor.

The aPR33A series C1.X is specially designed for simple CPU interface, user can record or playback up to 1024 voices by 5 I/Os only. This mode built in one complete memory-management system. The control side doesn't need to be burdened complicated memory distribution problems and it only needs to be through a simple instruction to proceed the audio/voice recording & playback so it largely shorten the developing time. Meanwhile, Chip provides the power-management system too. Users can let the chip enter power-down mode when unused. It can effectively reduce electric current consuming to 15uA and increase the using time in any projects powered by batteries.

■ PIN CONFIGURATION

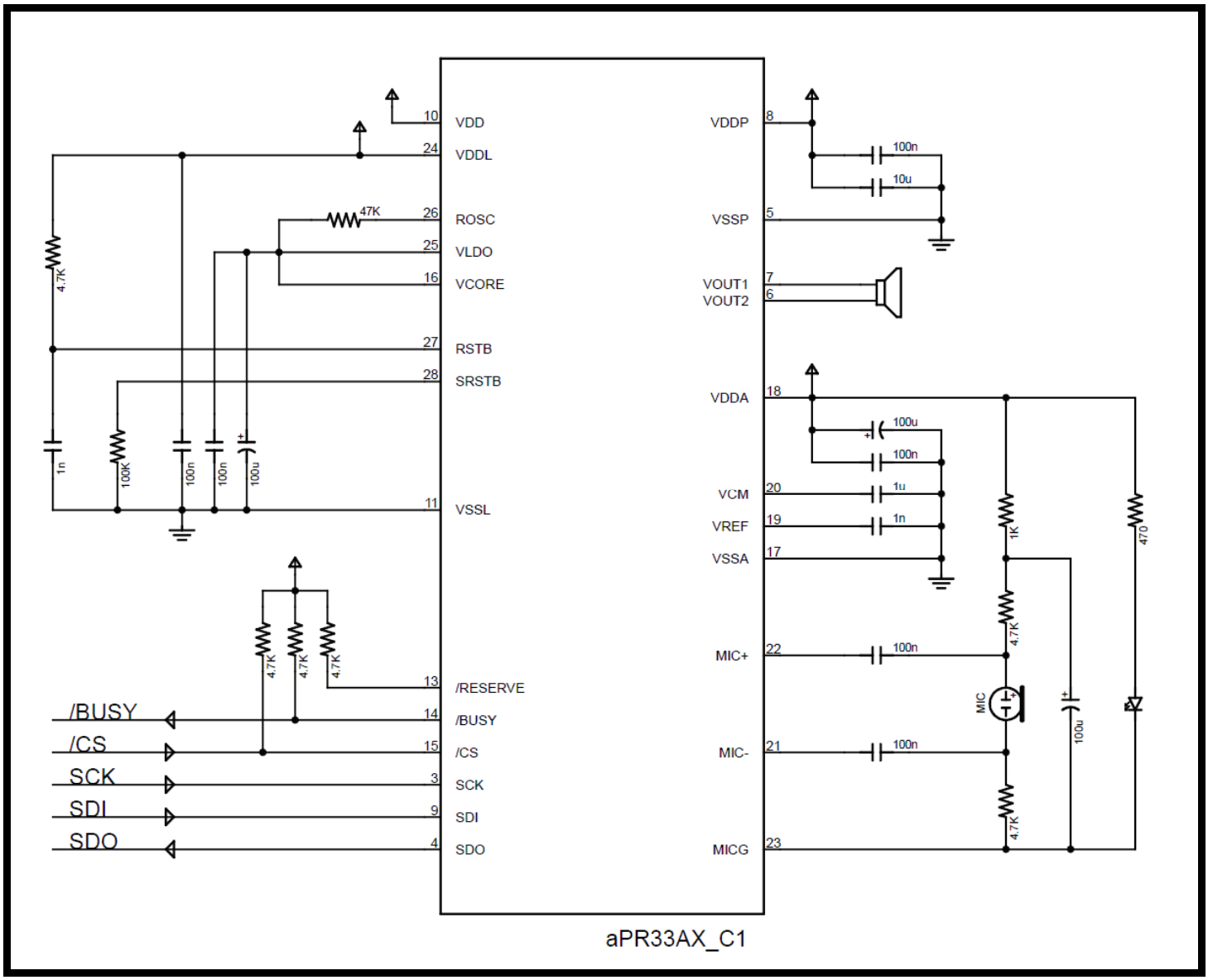


DIP / SOP Package

■ PIN DESCRIPTION

Pin Names	Pin No	TYPE	Description
VDDP VDD VDDA VDDL	8 10 18 24		Positive power supply.
VSSP VSSL VSSA	5 11 17		Power ground.
V_{LDO}	25		Internal LDO output.
V_{CORE}	16		Positive power supply for core.
V_{REF}	19		Reference voltage.
V_{CM}	20		Common mode voltage.
Rosc	26	INPUT	Oscillator resistor input.
RSTB	27	INPUT	Reset. (Low active)
SRSTB	28	INPUT	System reset, pull-down a resistor to the VSSL.
MIC+ MIC-	21 22	INPUT	Microphone differential input.
MICG	23	OUTPUT	Microphone ground.
VOUT2 VOUT1	6 7	OUTPUT	PWM output to drive speaker directly.
/CS	15	INPUT	Chip select. (Low active)
SCK	3	INPUT	Serial clock.
SDI	9	INPUT	Serial data input.
SDO	4	OUTPUT	Serial data output.
/BUSY	14	OUTPUT	System busy output.
/RESERVE	13	OUTPUT	Output reserve.

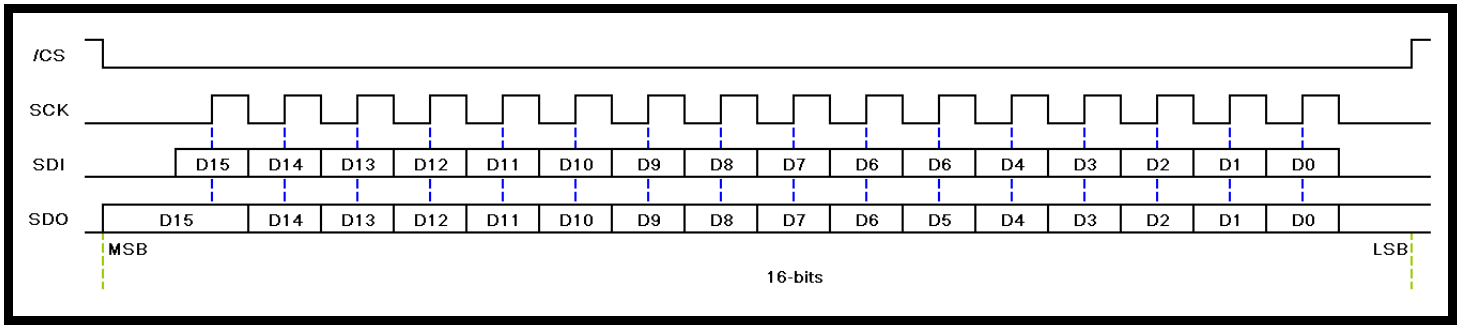
■ CONNECTION DIAGRAM



■ SERIAL COMMAND

The aPR33A3 series C1.X is specially designed for simple CPU interface. Chip is controlled by command sent to it from the host CPU. The /CS pin is used to select chip. The SCK and SDI pin are used to input command word into the chip while SDO and BUSY as output from the chip to the host CPU for feedback response.

Command input into the chip contains 16-bit data and list the command format & summarize the available commands as below:



Command	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
STOP	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
DELETE	0	0	0	1	0	0	Voice No In Binary									
REC	0	0	1	0	0	0	Voice No In Binary									
PLAY	0	0	1	1	0	0	Voice No In Binary									
PUP	1	0	1	1	0	0	0	1	0	0	0	0	0	0	0	0
PDN	1	0	1	1	0	0	1	0	0	0	0	0	0	0	0	0
FORMAT	1	0	1	0	0	1	0	1	1	0	1	0	0	1	0	1

Note: First time use this chip, please execute FORMAT instruction first.

- REC

The REC command is used to start record the voice to the specified voice number.

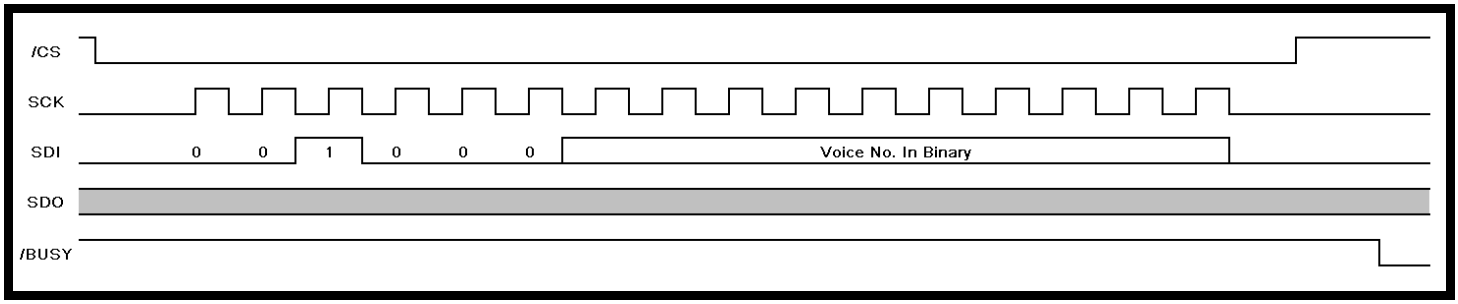
In the REC command, the bit-15 ~ bit-10 is 001000 in binary, and the bit-9 ~ bit-0 is the voice number in binary. Up to 1024 voice numbers user can specify.

After the REC command sent, the /BUSY pin will be drove low and playback “beep” tone to indicate the record operation starting.

During the record operating, the /BUSY pin will keep driving low, and any command except STOP will be ignored.

The record operation will continue until users send STOP command or full of memory, the /BUSY pin will be released and playback “beep” tone 2 times to indicate the record operation finished.

If the specified voice number already exist voice data or the memory is full, the /BUSY pin will not drive to low and execute REC operating. User can use the DELETE command to clear specified voice number before REC command.



- PLAY

The PLAY command is used to start playback the voice in the specified voice number.

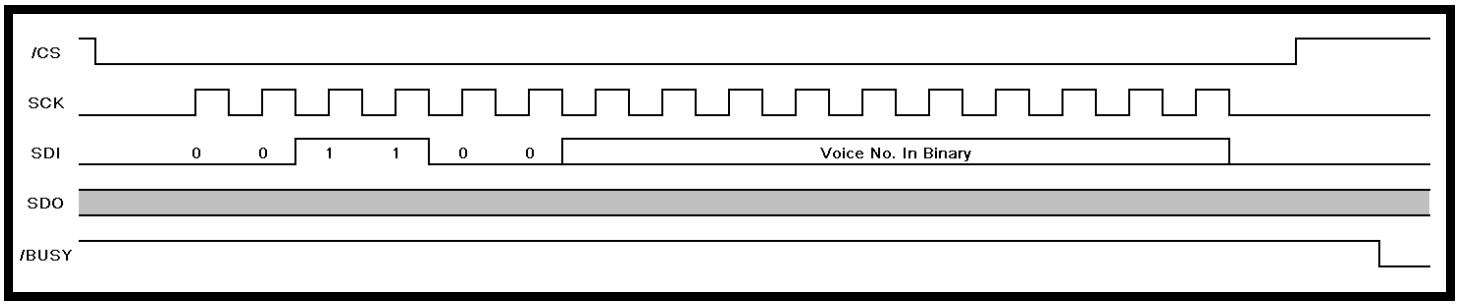
In the PLAY command, the bit-15 ~ bit-10 is 001100 in binary, and the bit-9 ~ bit-0 is the voice number in binary. Up to 1024 voice numbers user can specify.

After the PLAY command sent, the /BUSY pin will be drove low to indicate the playback operation starting.

During the playback operating, the /BUSY pin will keep drive low, and any command except STOP will be ignored.

The playback operation will continue until users send STOP command or end of voice, the /BUSY pin will be released to indicate the record operation finished.

If the specified voice number is empty, it will not drive /BUSY to low and playback.

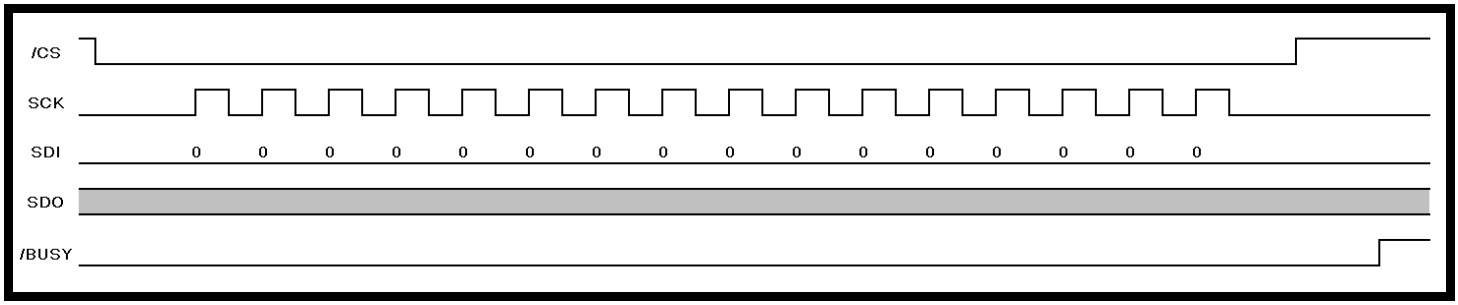


- STOP

  The STOP command is used to stop current operation.

  After the STOP command sent, the /BUSY pin will be released to indicate end of the current operation.

  The STOP command is effective only in playing or recording.



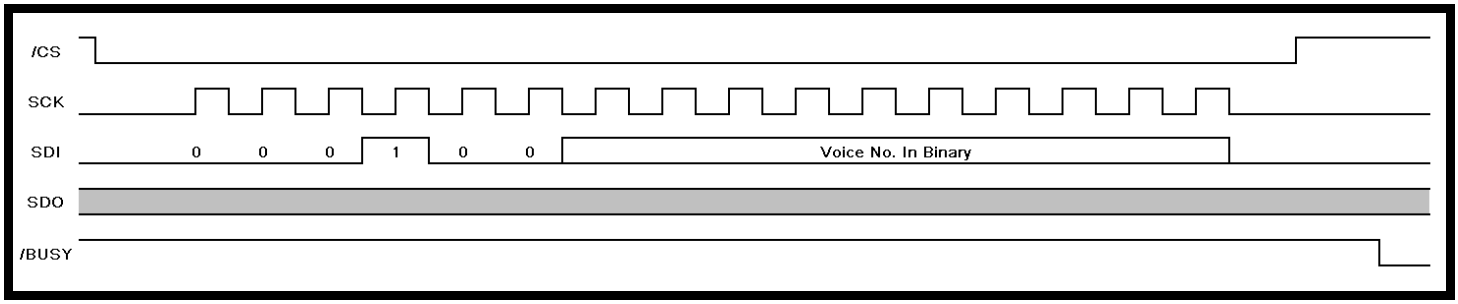
- DELETE

The DELETE command is used to delete the voice in the specified voice number.

In the DELETE command, the bit-15 ~ bit-10 is 000100 in binary, and the bit-9 ~ bit-0 is the voice number in binary. Up to 1024 voice numbers user can specify.

After the DELETE command sent, the /BUSY pin will be drove low to indicate the delete operation starting. When delete operation is finished, the /BUSY pin will be released.

The memory space in the specified voice number will be release after delete operation, user can get more free space by delete unused voice.

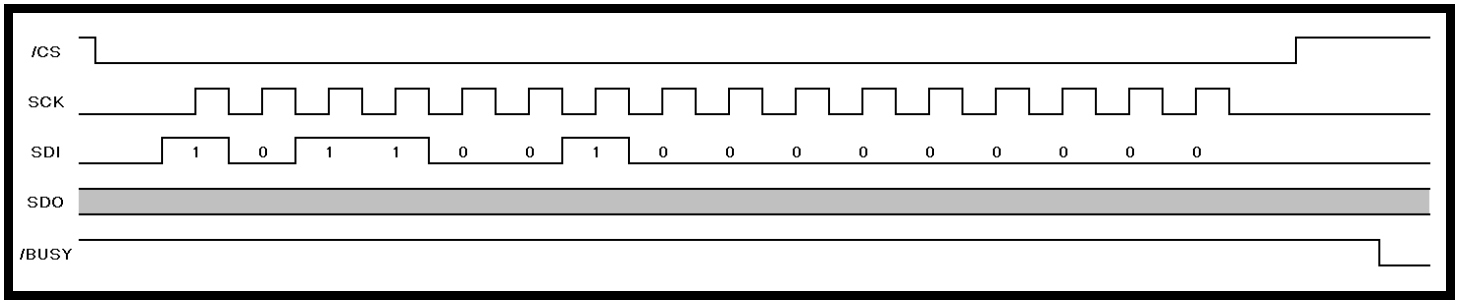


- PDN

The PDN command is used to enter the power-down mode.

After the PDN command sent, the /BUSY pin will be drove low to indicate the power-down operation starting. When chip is in the power-down mode, the /BUSY pin will be released.

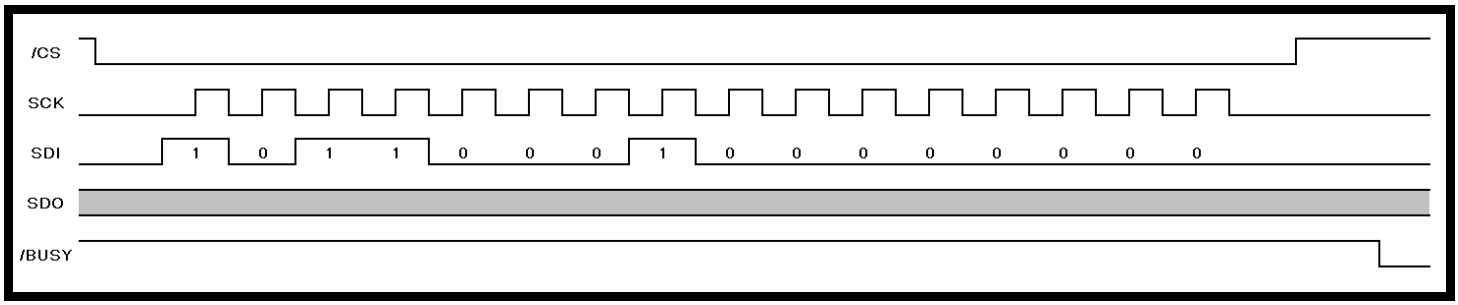
During chip in the sleep mode, the current consumption is reduced to I_{PDN} and any command except PUP will be ignored.



- PUP

The PUP command is used to power up from sleep mode.

After the PUP command sent, the /BUSY pin will be drove low to indicate the power up operation starting. When chip is in the idle mode, the /BUSY pin will be released. User can execute REC, PLAY or DELETE, or other command in idle mode.

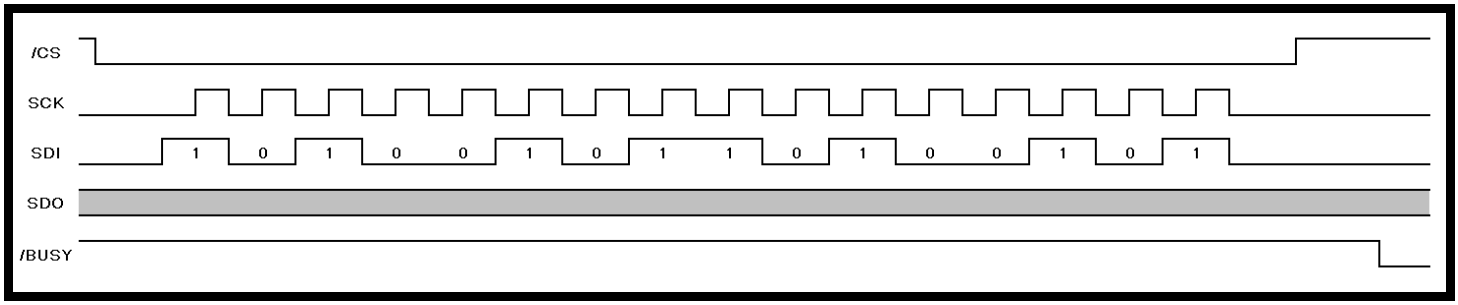


- FORMAT

The FORMAT command is used to restore memory to factory state.

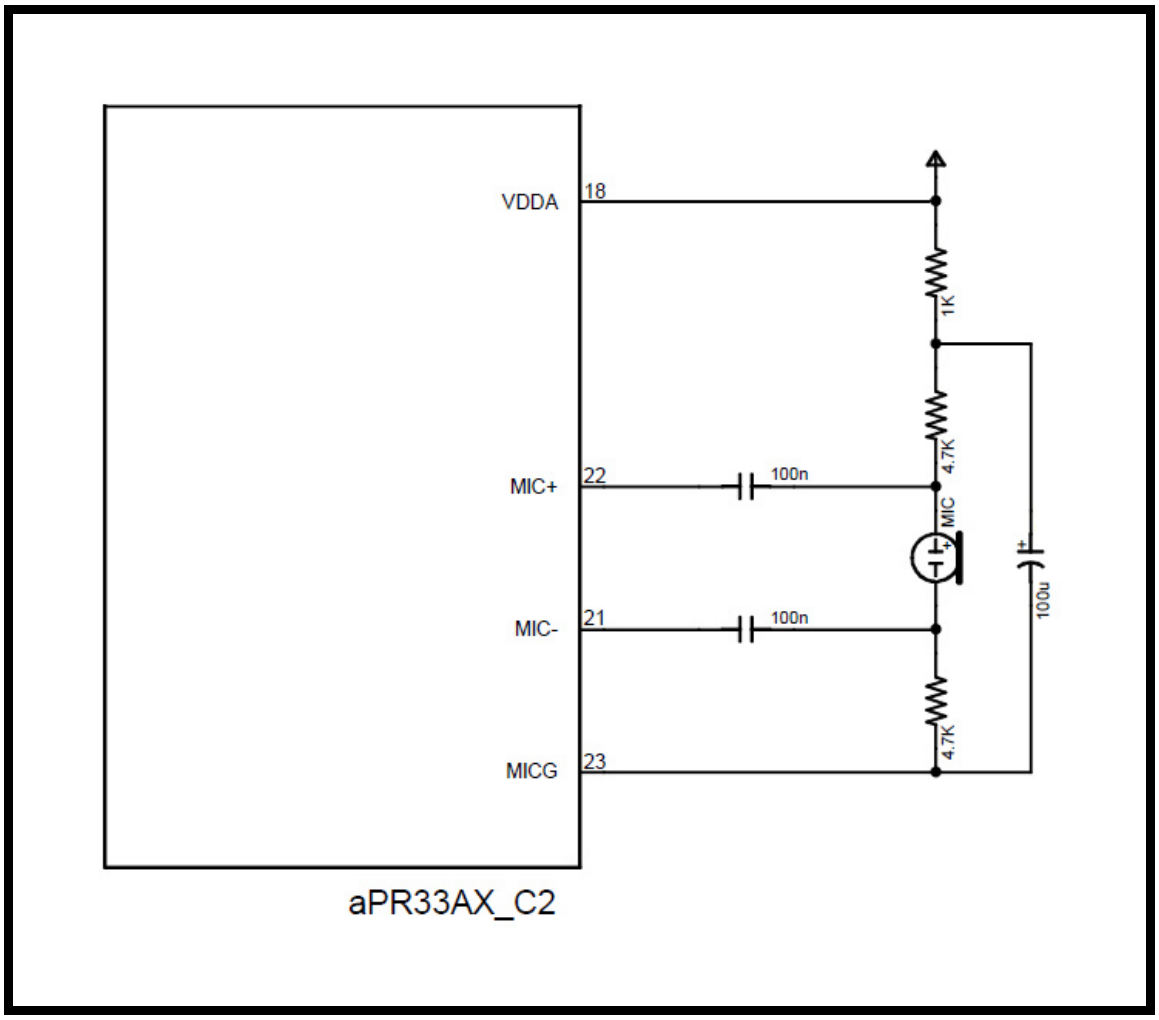
After the FORMAT command sent, the /BUSY pin will be drove low to indicate the format operation starting. When format operation is finished, the /BUSY pin will be released.

All of the voice in the memory will be clear after execute format operation.

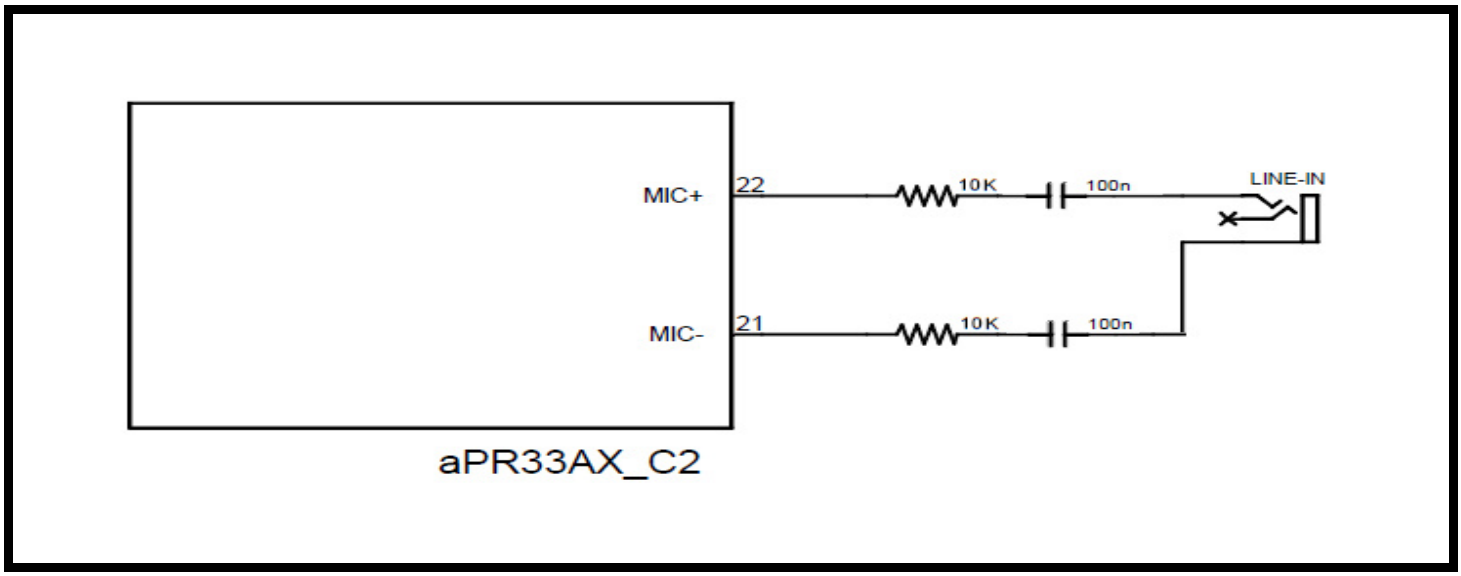


■ VOICE INPUT

The aPR33A series supported single channel voice input by microphone or line-in. The following fig. showed circuit for different input methods: microphone, line-in and mixture of both.

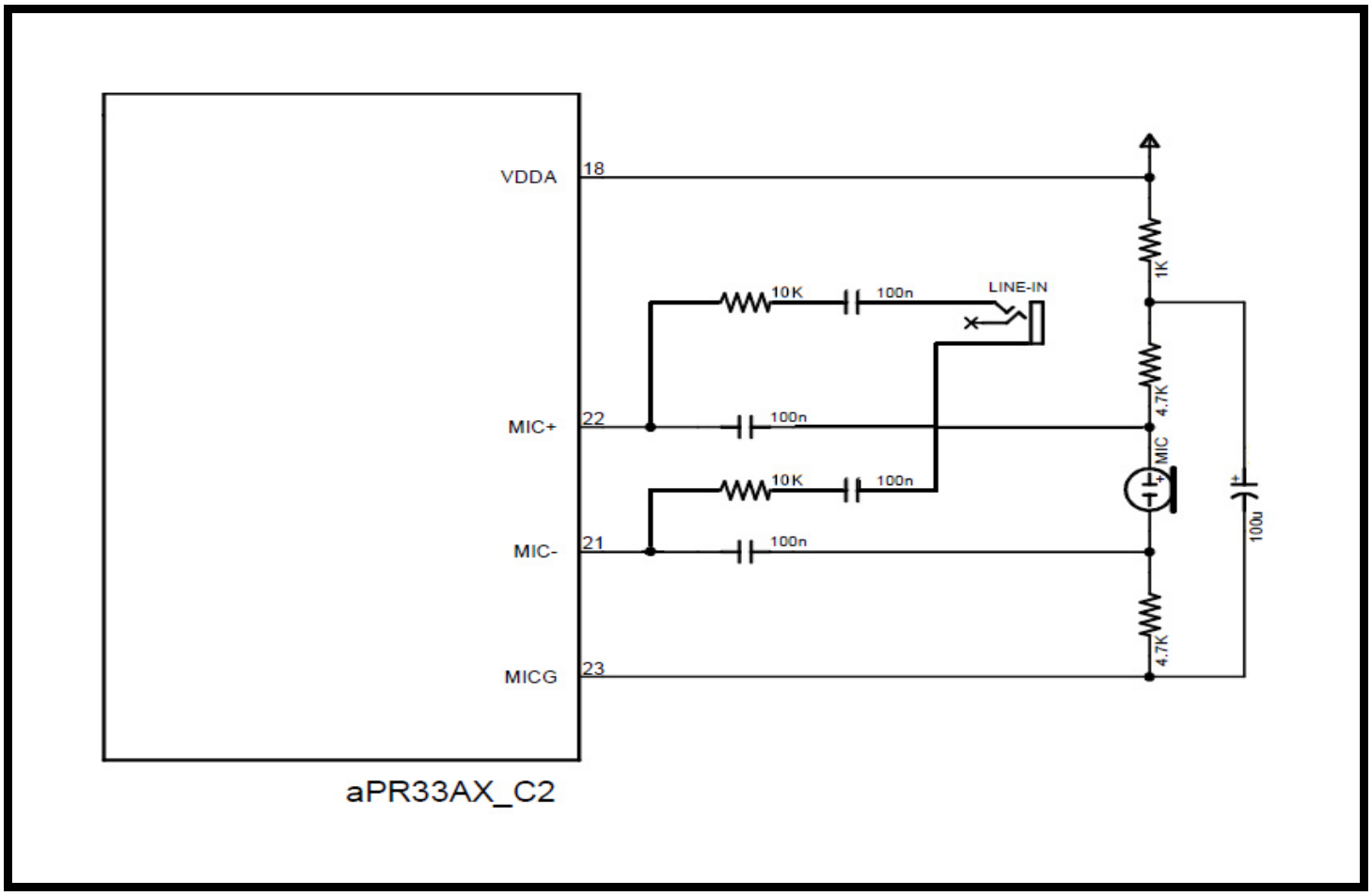


(A) Microphone



Note: The 10K resistor used for input signal adjust, and the value just for reference.

(B) Line-In



Note: The 10K resistor used for input signal adjust, and the value just for reference.

(C) Microphone + Line-In

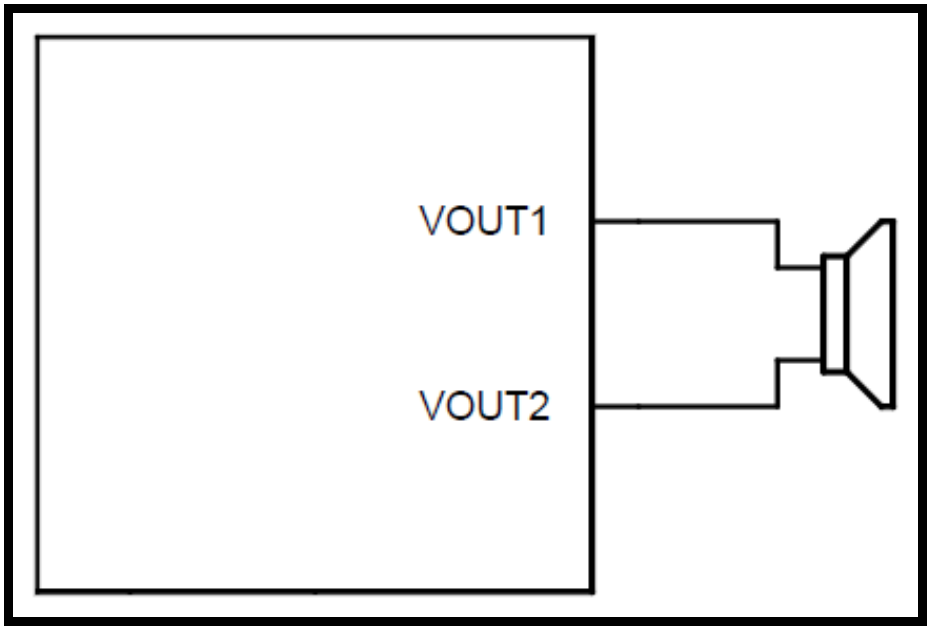
■ VOICE OUTPUT

The aPR33A series support 2 voice output mode, PWM and DAC.

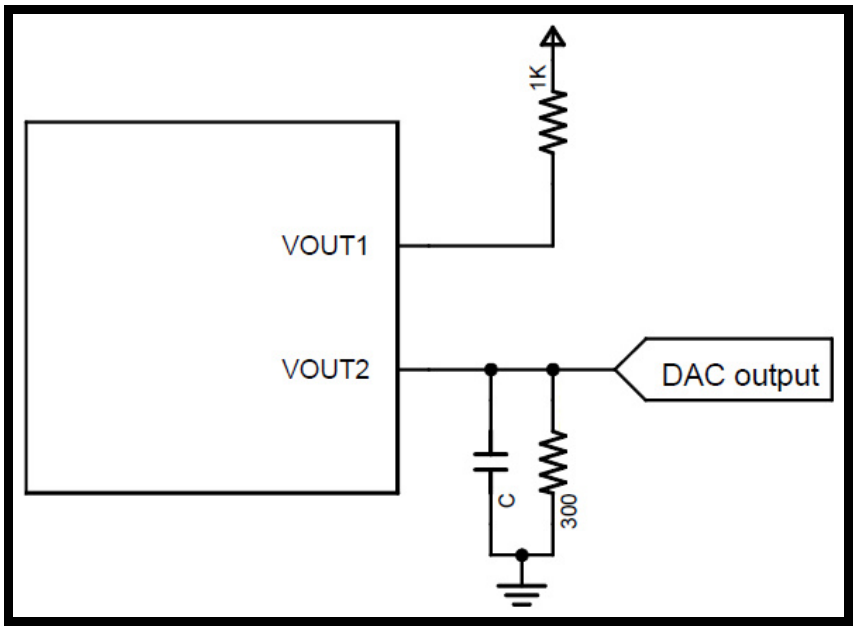
The PWM mode use VOUT1 and VOUT2 pin to drive speaker directly without external components to save cost.

The DAC mode use VOUT2 pin to output current signal. User can use the signal to drive audio amplifier or mix with other components in their applications to provide larger voice volume.

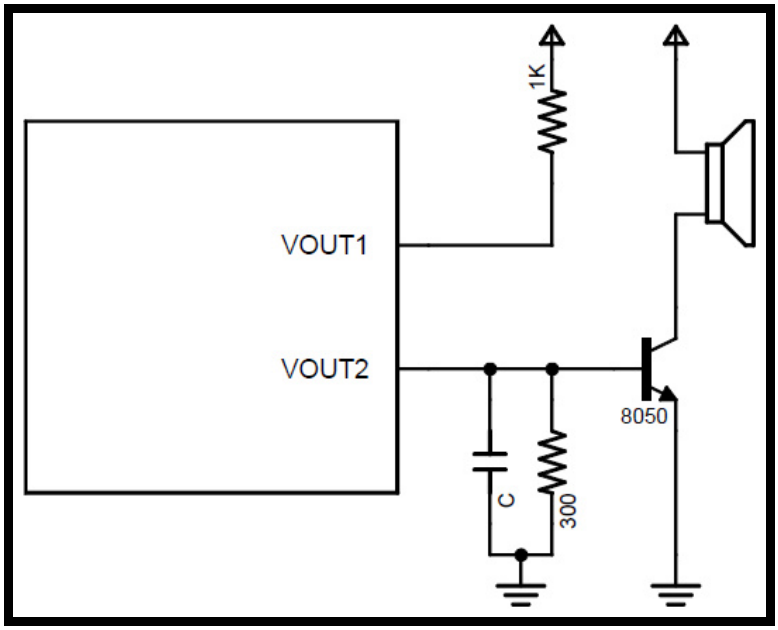
The following fig. show circuit for different output methods: PWM, DAC, DAC with transistor, DAC with audio amplifier AP4890B.



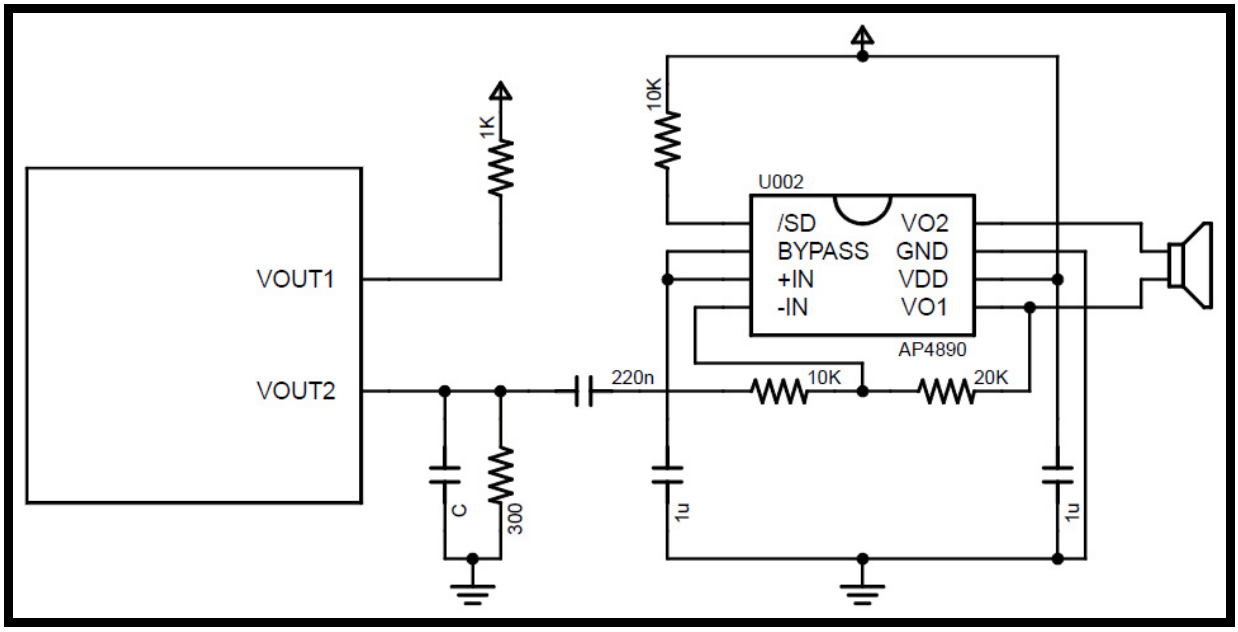
(A) PWM



(B) DAC



(C) DAC with transistor



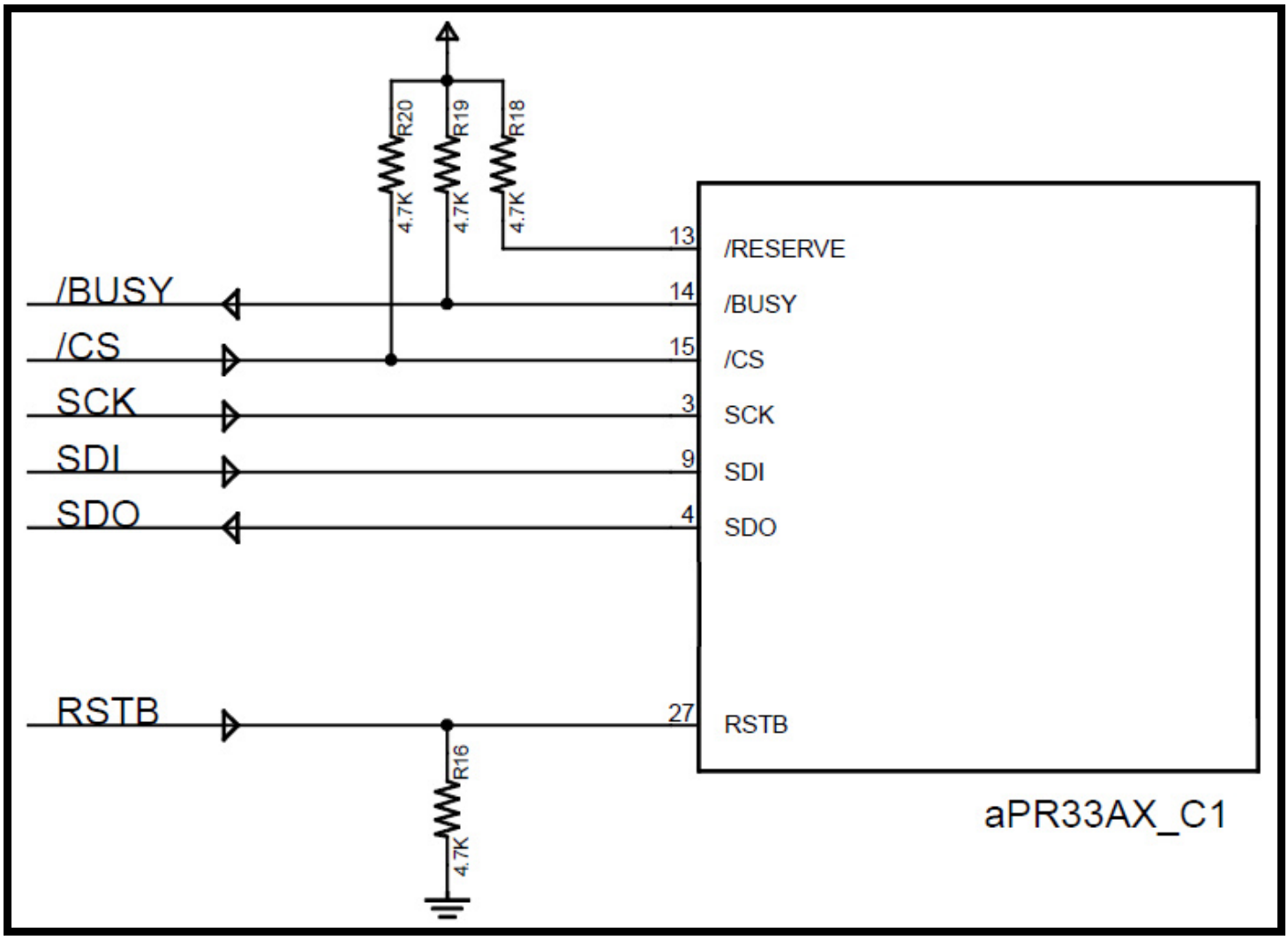
(D) DAC with audio amplifier AP4890B

■ RESET

aPR33A series can enter standby mode when RSTB pin drive to low. During chip in the standby mode, the current consumption is reduced to I_{SB} and any operation will be stopped, user also can not execute any new operate in this mode.

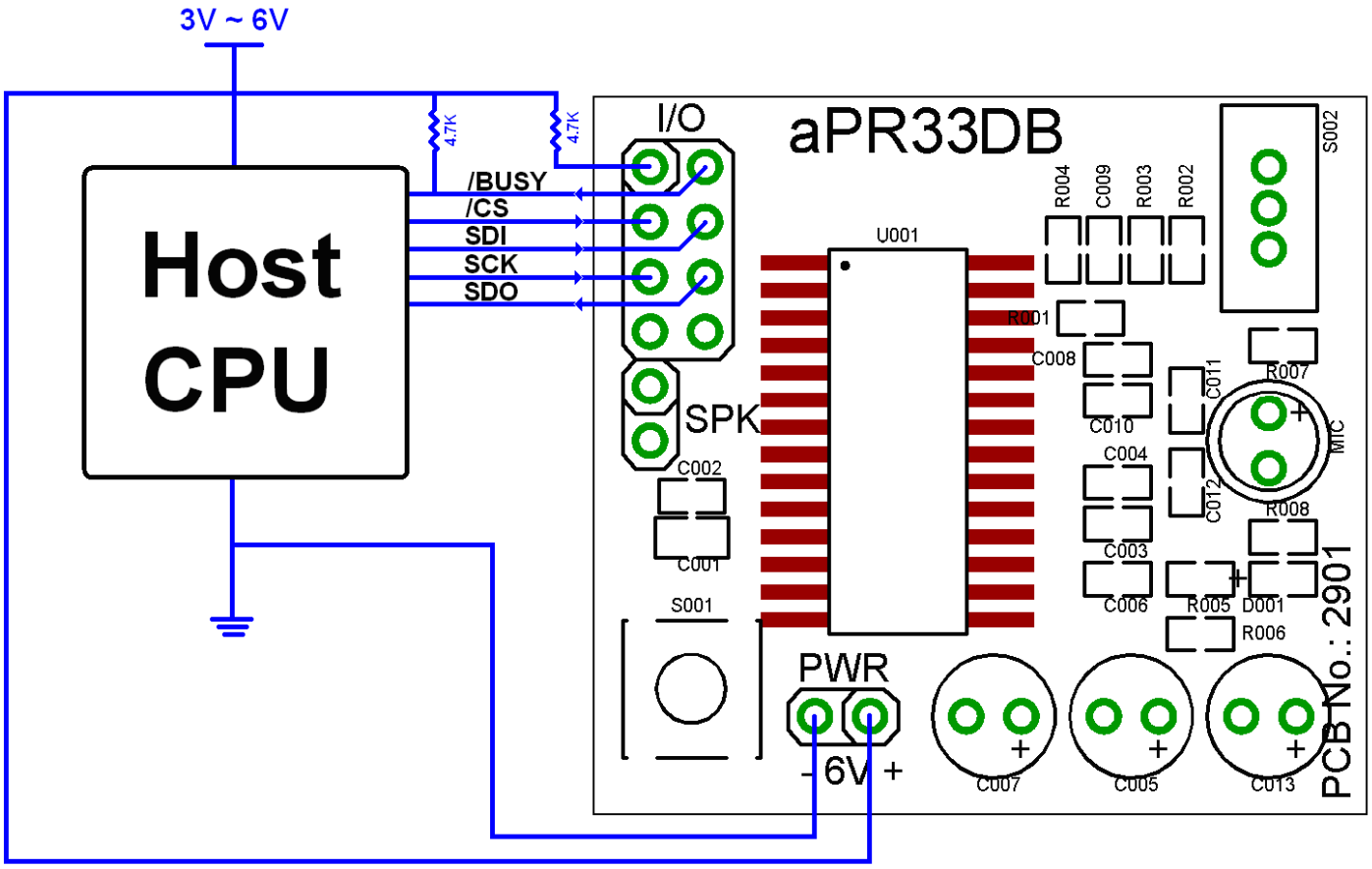
The standby mode will continue until RSTB pin goes to high, chip will be started to initial, and playback “beep” tone to indicate enter idle mode.

User can get less current consumption by control RSTB pin specially in some application which concern standby current.



■ EXAMPLE

The aPR33DB is one of the simplest solutions for achieve serial command mode demo. The circuit board already includes the peripheral circuit which containing microphone. Developers only need to notice how to connect with their development environment. It can effectively decrease the time of circuit connecting & any possible mistakes. Below figure shows how to connect aPR33DB with external host board in serial command mode :



The below example C code display all the operating instructions. Developers can evaluate any possible operating ways and voice quality in the shortest time :

```
//=============================================================================================
// I/O Define
//=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
// P_CS    : aPR33Ax /CS pin.
// P_SCK   : aPR33Ax SCK pin.
// P_DO    : aPR33Ax DO pin.
// P_DI    : aPR33Ax DI pin.
// P_BUSY: : aPR33Ax /BUSY pin.
// KEY_REC : Key for record, high active.
// KEY_PLAY: Key for playback, high active
//=============================================================================================
// Type Define
//=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
// unsigned char: 1-byte.
// unsigned int : 2-bytes.
typedef union           UTYPE
{
    unsigned int        Tunsigned int;                      // Dual-byte
    struct
    {
        unsigned        TBIT0:1;
        unsigned        TBIT1:1;
        unsigned        TBIT2:1;
        unsigned        TBIT3:1;
        unsigned        TBIT4:1;
        unsigned        TBIT5:1;
        unsigned        TBIT6:1;
        unsigned        TBIT7:1;
        unsigned        TBIT8:1;
        unsigned        TBIT9:1;
        unsigned        TBIT10:1;
        unsigned        TBIT11:1;
```

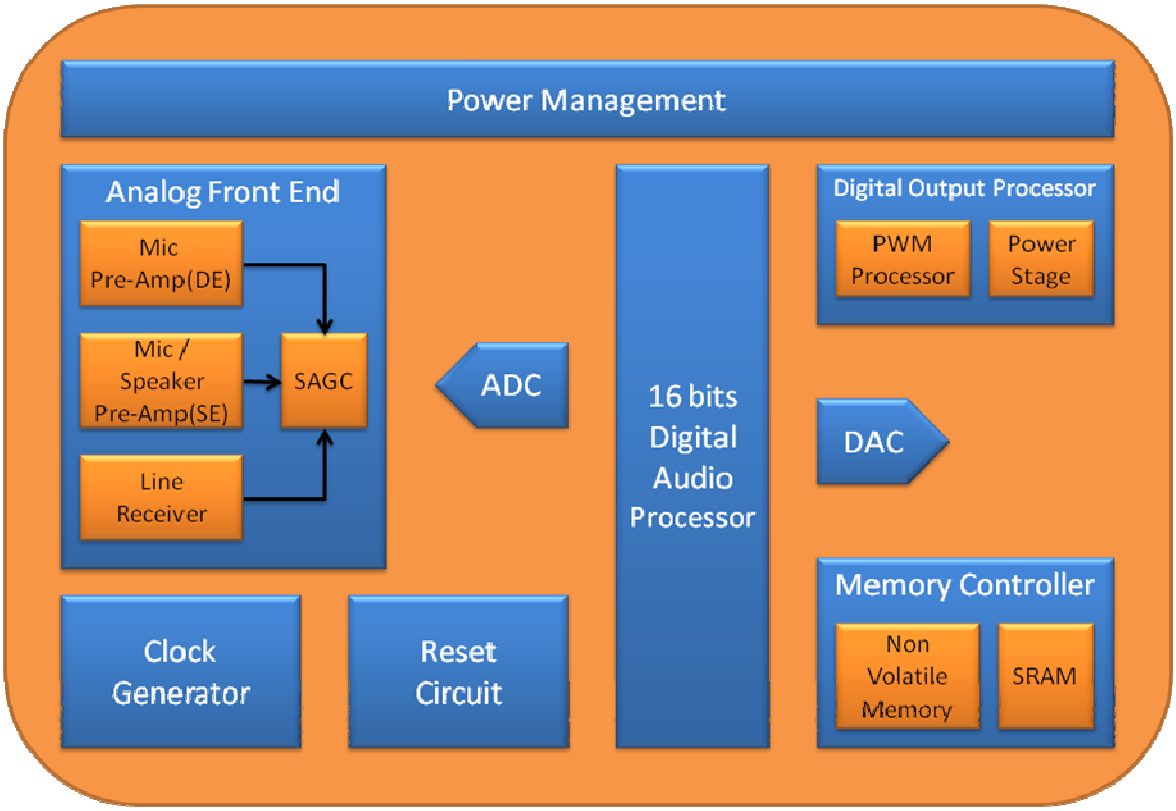
```
        unsigned        TBIT12:1;
        unsigned        TBIT13:1;
        unsigned        TBIT14:1;
        unsigned        TBIT15:1;
    };
}UTYPE;
//==========================================================================================
// Prototype
//=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
void                    CS(BOOL Value)          {   P_CS =Value;                        }
void                    SCK(BOOL Value)         {   P_SCK=Value;    Delay_500nS();      }
void                    SDO(BOOL Value)         {   P_DO =Value;    Delay_500nS();      }
BOOL                    SDI()                   {    return(P_DI);                      }
//=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
unsigned int            SendCmd(unsigned int Value)
{
    UTYPE               TxData, RxData;
    //--------------------------------------------------------------------------------------
    TxData.Tunsigned int = Value;
    //--------------------------------------------------------------------------------------
                SDO(TxData.TBIT15);         RxData.TBIT15=SDI();        SCK(1);
    SCK(0);     SDO(TxData.TBIT14);         RxData.TBIT14=SDI();        SCK(1);
    SCK(0);     SDO(TxData.TBIT13);         RxData.TBIT13=SDI();        SCK(1);
    SCK(0);     SDO(TxData.TBIT12);         RxData.TBIT12=SDI();        SCK(1);
    SCK(0);     SDO(TxData.TBIT11);         RxData.TBIT11=SDI();        SCK(1);
    SCK(0);     SDO(TxData.TBIT10);         RxData.TBIT10=SDI();        SCK(1);
    SCK(0);     SDO(TxData.TBIT9);          RxData.TBIT9=SDI();         SCK(1);
    SCK(0);     SDO(TxData.TBIT8);          RxData.TBIT8=SDI();         SCK(1);
    SCK(0);     SDO(TxData.TBIT7);          RxData.TBIT7=SDI();         SCK(1);
    SCK(0);     SDO(TxData.TBIT6);          RxData.TBIT6=SDI();         SCK(1);
    SCK(0);     SDO(TxData.TBIT5);          RxData.TBIT5=SDI();         SCK(1);
    SCK(0);     SDO(TxData.TBIT4);          RxData.TBIT4=SDI();         SCK(1);
    SCK(0);     SDO(TxData.TBIT3);          RxData.TBIT3=SDI();         SCK(1);
    SCK(0);     SDO(TxData.TBIT2);          RxData.TBIT2=SDI();         SCK(1);
    SCK(0);     SDO(TxData.TBIT1);          RxData.TBIT1=SDI();         SCK(1);
    SCK(0);     SDO(TxData.TBIT0);          RxData.TBIT0=SDI();         SCK(1);
    SCK(0);
    //--------------------------------------------------------------------------------------
```

```
    return (RxData.Tunsigned int);
}
//=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
void    PUP(void)                     {   CS(0);  Delay_10mS();   SendCmd(0xB100);    CS(1);  }
void    PDN(void)                     {   CS(0);  Delay_10mS();   SendCmd(0xB200);    CS(1);  }
void    FORMAT(void)                  {   CS(0);  Delay_10mS();   SendCmd(0xA5A5);    CS(1);  }
void    STOP(void)                    {   CS(0);  Delay_10mS();   SendCmd(0x0000);    CS(1);  }
void    DELETE(unsigned int VoiceNo)  {   CS(0);  Delay_10mS();   SendCmd(0x1000|(VoiceNo&0x03FF));   CS(1);  }
void    REC(unsigned int VoiceNo)     {   CS(0);  Delay_10mS();   SendCmd(0x2000|(VoiceNo&0x03FF));   CS(1);  }
void    PLAY(unsigned int VoiceNo)    {   CS(0);  Delay_10mS();   SendCmd(0x3000|(VoiceNo&0x03FF));   CS(1);  }
//==================================================================================================
// Main
//=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
void    main(void)
{
    //----------------------------------------------------------------------------------------------
    Init_IO();                                    // Initial I/O: /CS=0, SCK=0, SDI=0, SDO=input, /BUSY=input.
    //----------------------------------------------------------------------------------------------
    while(!P_BUSY);                               // Check for /BUSY pull-up.
    //----------------------------------------------------------------------------------------------
    // Format chip
    FORMAT();
    while(P_BUSY);   while(!P_BUSY);              // Wait for format operation start & finished.
    //----------------------------------------------------------------------------------------------
    while(1)
    {
        if(KEY_REC)
        {
            DELETE(0x0000);
            while(P_BUSY);  while(!P_BUSY);       // Wait for delete operation start & finished.

            REC(0x0000);    Delay_10mS();         // Record the No.0 voice.
            while((KEY_REC)&(!P_BUSY));           // Wait for release record key or full of memory.

            STOP();
            while(!P_BUSY);                       // Wait for record finished.
        }
        if(KEY_PLAY)
```

```
        {
            PLAY(0x0000);   Delay_10mS();              // Play the No.0 voice.
            while((KEY_PLAY)&(!P_BUSY));               // Wait for release play key or end of voice.
            STOP();
            while(!P_BUSY);                            // Wait for play finished.
        }
    }
    //-------------------------------------------------------------------------------------------
    return ;
}
//===========================================================================================
```

■ BLOCK DIAGRAM

Figure 1. Block Diagram



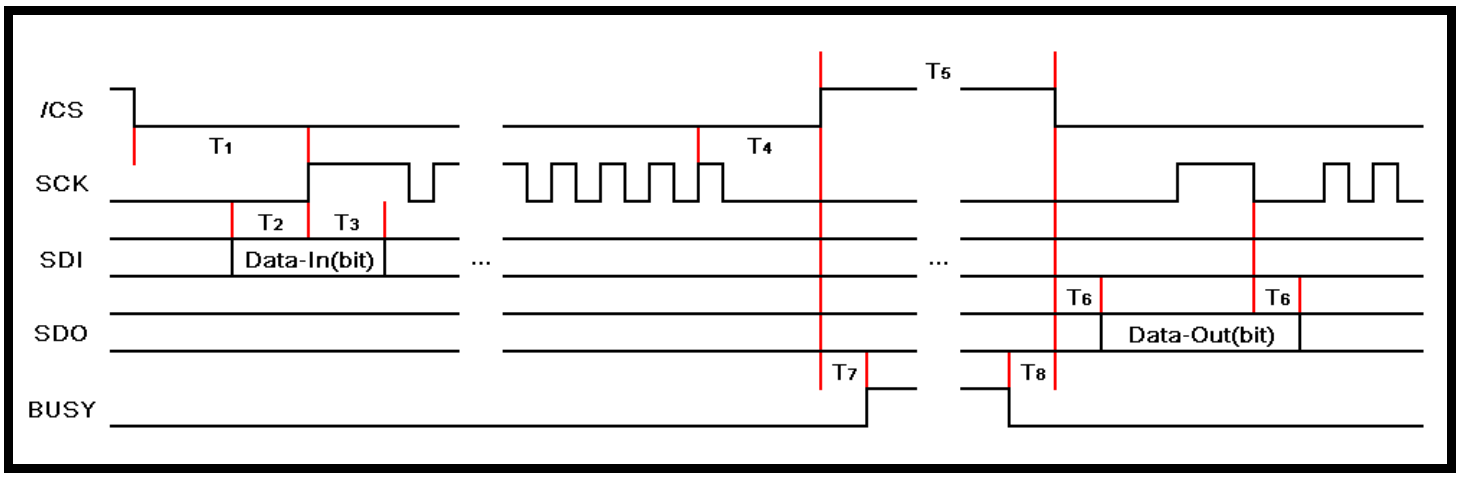
■ ABSOLUTE MAXIMUM RATINGS

Symbol	Rating	Unit
VDD – VSS	-0.3 ~ +10.0	V
V_{IN}	VSS-0.3 < V_{IN} < VDD+0.3	V
V_{OUT}	VSS < V_{OUT} < VDD	V
T(Operating)	-40 ~ +85	℃
T(Junction)	-40 ~ +125	℃
T(Storage)	-40 ~ +125	℃

■ DC CHARACTERISTICS

Symbol	Parameter	Min.	Typ.	Max.	Unit	Conditions
VDD	Operating Voltage	3.0		6.5	V	
I_{SB}	Standby Current			1	μA	
I_{PDN}	Power-Down Current		15	20	μA	
$I_{OP(IDLE)}$	Operating Current (Idle)		20		mA	VDD = 5V
$I_{OP(REC)}$	Operating Current (Record)		35		mA	VDD = 5V
$I_{OP(PLAY)}$	Operating Current (Playback)		25		mA	VDD = 5V
V_{IH}	"H" Input Voltage	2.5			V	
V_{IL}	"L" Input Voltage			0.6	V	
I_{VOUT}	VOUT Current		185		mA	
I_{OH}	O/P High Current		8		mA	VDD = 5V / VOH=4.5V
I_{OL}	O/P Low Current		14		mA	VDD = 5V / VOH=0.5V
R_{NPIO}	Input pin pull-down resistance		300		KΩ	External floating or drive low.
			1		MΩ	External drive high.
R_{UPIO}	Input pin pull-up resistance		4.7		KΩ	
△Fs/Fs	Frequency stability			5	%	VDD = 5V ± 1.0V
△Fc/Fc	Chip to chip Frequency Variation			5	%	Also apply to lot to lot variation.

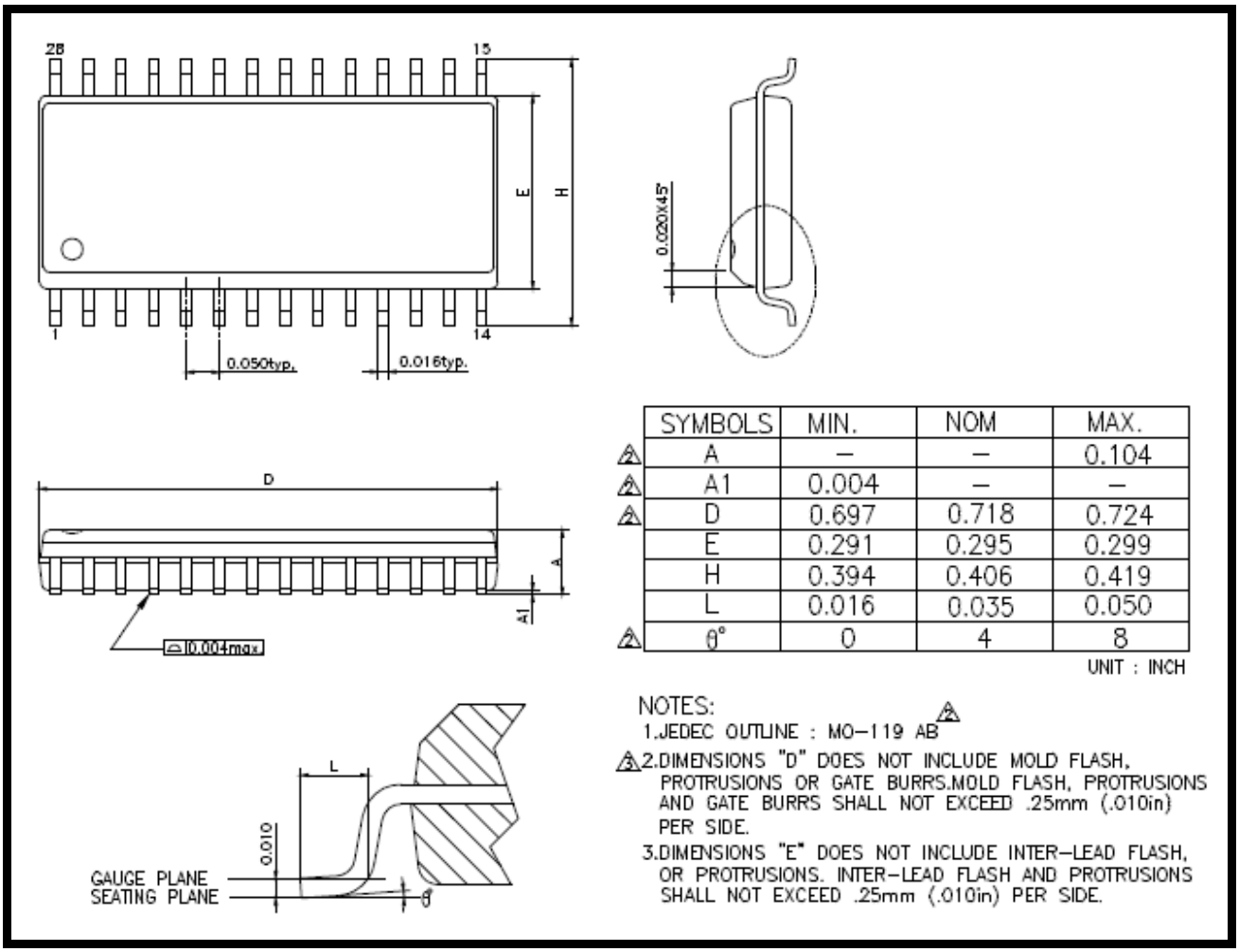
■ AC CHARACTERISTICS



Symbol	Parameter	Min.	Typ.	Max.	Unit	Conditions
T1	CS Setup Time	10	--	--	mS	VDD=5.0V
T2	Data-In Setup Time	500	--	--	nS	VDD=5.0V
T3	Data-In Hold Time	500	--	--	nS	VDD=5.0V
T4	/CS Hold Time	500	--	--	nS	VDD=5.0V
T5	/CS High Time	10	--	--	mS	VDD=5.0V
T6	Data-Out Setup Time	--	--	500	nS	VDD=5.0V
T7	BUSY Setup Time	--	--	10	mS	VDD=5.0V
T8	BUSY Hold Time	--	--	10	mS	VDD=5.0V

■ PACKAGE INFORMATION

28Pin 300mil SOP Package



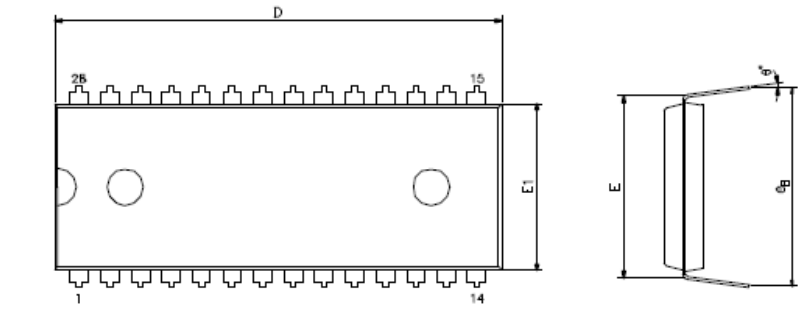
	SYMBOLS	MIN.	NOM	MAX.
2	A	–	–	0.104
2	A1	0.004	–	–
4	D	0.697	0.718	0.724
	E	0.291	0.295	0.299
	H	0.394	0.406	0.419
	L	0.016	0.035	0.050
2	θ°	0	4	8

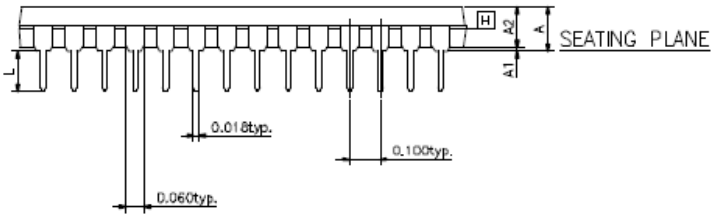
UNIT : INCH

NOTES:

1. JEDEC OUTLINE : MO-119 AB 4
3 2. DIMENSIONS "D" DOES NOT INCLUDE MOLD FLASH, PROTRUSIONS OR GATE BURRS. MOLD FLASH, PROTRUSIONS AND GATE BURRS SHALL NOT EXCEED .25mm (.010in) PER SIDE.
3. DIMENSIONS "E" DOES NOT INCLUDE INTER-LEAD FLASH, OR PROTRUSIONS. INTER-LEAD FLASH AND PROTRUSIONS SHALL NOT EXCEED .25mm (.010in) PER SIDE.

28Pin 600mil DIP Package





SYMBOLS	MIN.	NOR.	MAX.
A	–	–	0.220
A1	0.015	–	–
A2	0.150	0.155	0.160
D	1.455	1.460	1.470
E		0.600 BSC	
E1	0.540	0.545	0.550
⚠ L	0.115	0.158	0.200
e_B	0.630	0.650	0.670
θ°	0	7	15

UNIT : INCH

NOTE:
1.JEDEC OUTLINE : MS-011 AB
2.E1 DOES NOT INCLUDE MOLD FLASH.

■ HISTORY

Ver. E (2014/01/06)

- Delete aPR33A1 data

Ver. D (2013/11/26)

- Modify page.14-15 line in circuit

Ver. C (2013/06/07)

- Add DIP Package and outline
- Delete data of aPR33A2
- Modify the name of the pins : DI→SDI , DO→SDO

Ver. B (2013/05/07)

- Add section “VOICE OUTPUT”.
- Modify code version to C1.1.

Ver. A (2012/11/08)

- Original version data sheet for aPR33Ax C1.0.