Winbond Electronics Corp.

ADPCM VOICE SYNTHESIZER
(Enhanced *PowerSpeech*ä)

INTRODUCTION

The W581XX family is an enhanced version of the W528X *PowerSpeech*ä synthesizer series. The W581XX family features a ADPCM synthesizer and an 8-bit D/A converter to generate various kinds of voice effects. It also provides Load and Jump commands, eight general and three specific purpose registers, and all other necessary control and timing logic circuit to implement more advanced applications.

The W58100 is an emulation chip used for the purpose of demonstrating the W581XX series enhanced *PowerSpeech*ä products.

The W581XX family includes the W58101, W58102, W58103, W58104, W58105, W58106, W58110, W58115, and the W58120. The ROM size of each of these products is shown below:

BODY	W58101	W58102	W58103	W58104	W58105	W58106	W58110	W58115	W58120
ROM Size	80K	160K	240K	320K	512K	768K	1M	1.5M	2M

FEATURES

- Wide operating voltage range: 2.4 to 5.5 volts
- 4-bit ADPCM synthesis
- Four direct trigger inputs
- Two trigger input, long and short, debounce times
- Up to two LED and five STOP outputs
- Easily software extendable to 24 matrix trigger inputs
- Flexible functions programmable through the following:
  - LD (load), JP (jump) commands
  - Eight general purpose registers: R0~R7
  - Three specific registers: EN, STOP, and MODE
  – Conditional instructions
  – Speech equations
  – END instruction
  – Output frequency and LED flash type setting
- Fading effect (patent pending), and eight levels of SPK volume control can be set easily by section control
- Programmable power-on initialization (POI) (can be interrupted by trigger inputs)
- Every LED pin can simultaneously drive a maximum of three LEDs
- LED can be set as a volume control mode output
- LED flash frequency: 3 Hz

Version E
Release Date: April 1999

- AUD output current: 5 mA
- POI delay time of 160 mS ensures stable voltage during chip power on
- Can be programmed for the following functions:

– Interrupt or non-interrupt for each trigger pin; rising or falling edge (this feature determines retriggerable, non-retriggerable, overwrite, and non-overwrite features of each trigger pin)

– Four playing modes:

One Shot (OS)

Level Hold (LH)

Single-cycle level hold (S_LH)

Complete-cycle level hold (C_LH)

– Stop output signal setting

– Serial, direct, or random trigger mode setting

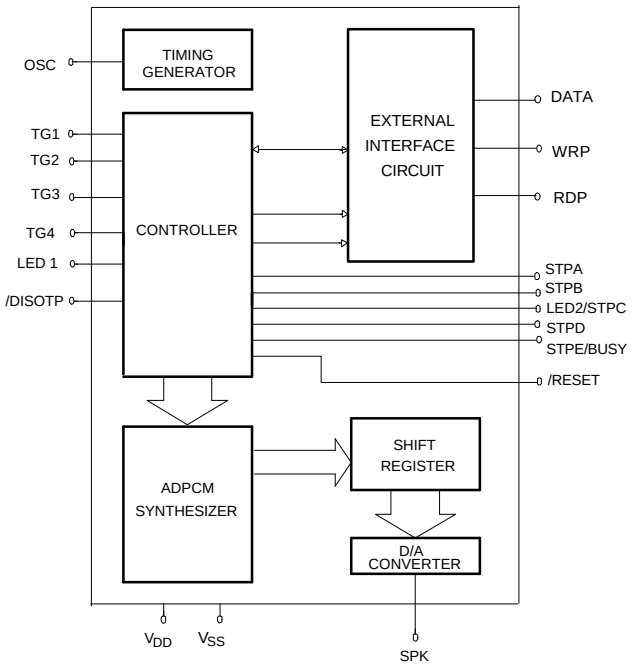
- Four frequency options (4/4.8/6/8 KHz) and LED On/Off control can be set independently in each speech equation GO instruction
- Independent control of LED 1 and LED 2
- Total of 256 voice group entries available for programming
- Provides the following mask options:

- LED flash type: synchronous/alternate
- LED 1 section-controlled: Yes/No
- LED 2 section-controlled/STPC-controlled
- LED volume-controlled: No/Yes
- FTEST/STPD (Note: The FTEST function is not provided on the W58100, only on W581XX.)
- STPE/BUSY
- NORMAL/CPU

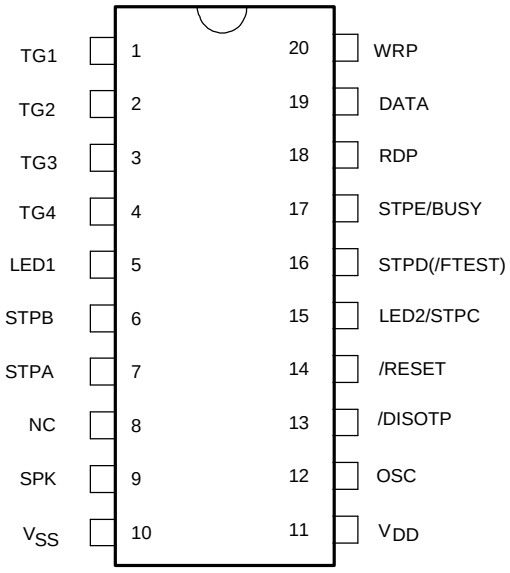
- Provides the following program declarations:

- FREQ0, FREQ1, FREQ2, FREQ3: Frequency variable
- LED0, LED1: LED on or off
- VOL0~VOL7: Fading effect (patent pending)

BLOCK DIAGRAM (W58100)



PIN CONFIGURATION (W58100)



* FTEST: not provided on W58100, but provided on W581xx

PIN DESCRIPTION (W58100)

NO.	NAME	I/O	FUNCTION
1	TG1	I	Trigger Input 1
2	TG2	I	Trigger Input 2
3	TG3	I	Trigger Input 3
4	TG4	I	Trigger Input 4
5	LED1	O	LED 1
6	STPB	O	Stop Signal B
7	STPA	O	Stop signal A
8	NC	O	No connected
9	SPK	O	Current output for speaker
10	VSS	-	Negative power supply
11	VDD	-	Positive power supply
12	OSC	I	Oscillation frequency control, connect Rosc to VDD
13	/DISOTP	I	Disable all of the serial interface pins (low active)
14	/RESET	I	Reset pin (low active)
15	LED2/STPC	O	LED2 or Stop C output
16	STPD(/FTEST)	O	Stop D output (FTEST not provided on W58100)
17	STPE/BUSY	O	Stop E or Busy signal output
18	*RDP	O	Read pulse clock output for serial interface
19	*DATA	I/O	Bidirectional Data Pin for the serial interface
20	*WRP	O	Write pulse clock output for serial interface

*: The RDP, WRP, and DATA pins are only provided on W58100.

1. TG1~TG4

These pins are pulled high internally by an equivalent resistance of around 1M ohm. After being activated these direct trigger inputs start executing the corresponding voice groups, which are located at 0/1/2/3 for falling edge triggers and at 4/5/6/7 for rising edge triggers.

The priority is set as: TG1F > TG1R > TG2F > TG2R > TG3F > TG3R > TG4F > TG4R. When more than one trigger is activated, only the one with the highest priority is serviced; all other trigger pins are suppressed.

2. LED1, LED2

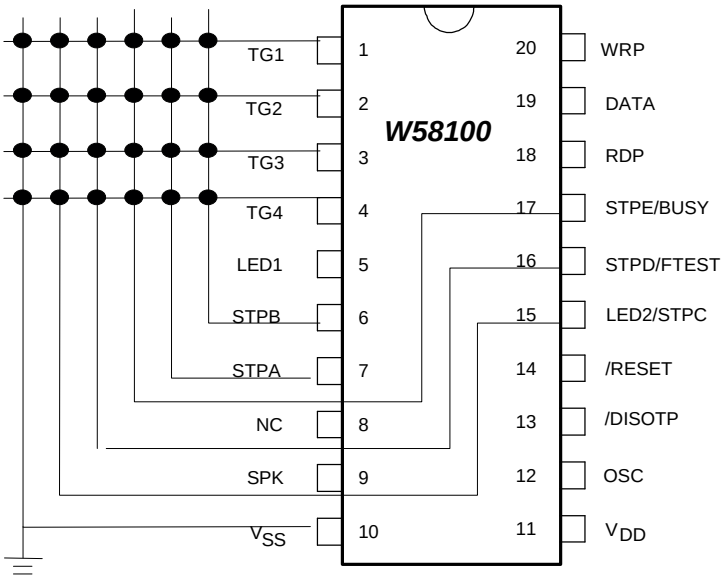
These pins are open-drain outputs to sink current through the LEDs.

3. STPA~STPE

These are inverter-type stop output signals that can be used to drive external peripherals such as transistors, motors, or lamps. They are also useful in keypad scan applications with up to 24 keys.

In keypad matrix applications, TG1~TG4 function as the trigger inputs to initiate the software scan routine. These inputs determine which key has been pressed by determining which row (TG1 ~ TG4) and which column (Vss, STPA ~ STPE) has been connected.

The 24 key matrix does not require additional components and is shown below:



4. STPE/BUSY

This pin is configured as STPE or BUSY depending upon the program declaration of this pin. If this pin is not declared as either STPE or BUSY then it will default to STPE in NORMAL mode and as BUSY in CPU mode.

The BUSY signal goes high whenever a successful interrupt occurs, whether the interrupt comes from POI, the micro-controller (CPU), or a trigger input. When an interrupt occurs, the BUSY signal goes high and stays high until the end of the operation. i.e. after encountering an "END" instruction the BUSY signal will be low.

5. STPD/FTEST

This pin is configured as a STPD output or as a frequency output test pin, FTEST, depending on the program declaration of this pin.

After being declared as an FTEST pin at the beginning of the program, this pin will generate a constant frequency output during playback. A counter can then be used to test this constant frequency in order to check whether the frequency deviation of this code is normal or abnormal. The constant frequency will be kept at 6K Hz, under the condition of the typical master frequency - for example, 3M Hz.

6. OSC

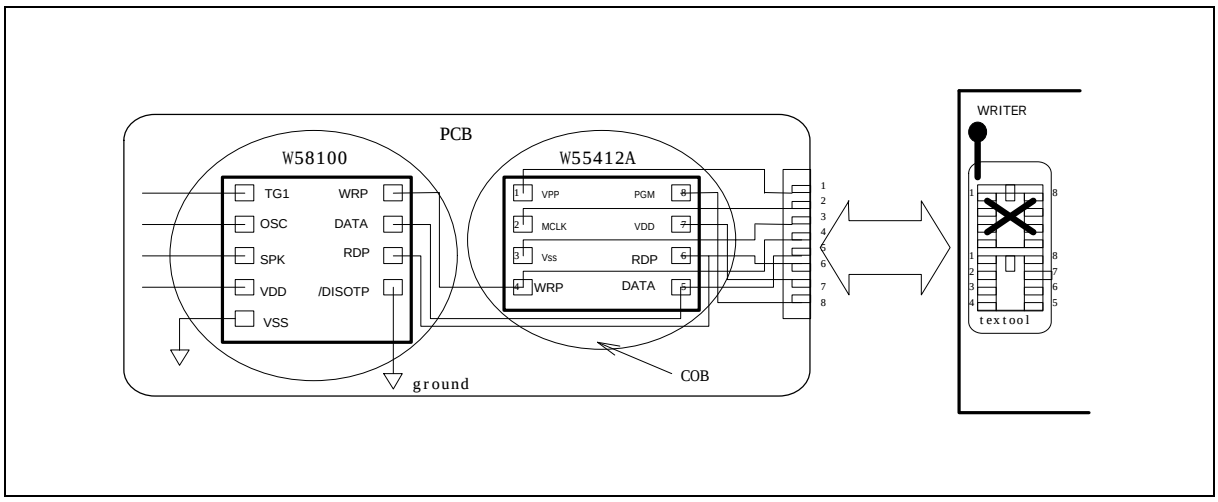
A ring oscillator is used to generate the master frequency of about 3 MHz. This pin is connected with an Rosc resistor to VDD providing a bias current for the ring oscillator.

7. WRP, RDP, and DATA

These three pins are dedicated to the W58100 emulation chip and are used to communicate with external serial memory devices such as flash EPROMs. Using these three pins together with an W58100 chip allows for 100% emulation of code to be used in mass produced chips.

8. /DISOTP

This pin is used to disable the three serial interface pins (WRP, RDP, and DATA) on the W58100 emulation chip. This allows the programming of a flash EPROM without disconnecting the interface pins in advance. This pin can be grounded to program the flash EPROM which is bonded on the COB and W58100. This concept is shown in the following diagram:



After programming, the /DISOTP pin must be floated in order to enable the WRP, DATA and RDP pins on the W58100. The sound effect of the flash EPROM can then be heard.

9. /RESET

This is an active-low reset input with an internal pull-high resistance of 500K ohm. The falling edge of this pin resets the W581XX IC completely just as if a power-on reset had been performed. The W581XX begins the POI process on the rising edge of this pin.

The device may function abnormally and cause unpredictable operation if VDD is not discharged to ground and the W581XX is powered up again. This pin can be used to reset the W581XX.

10. SPK

The SPK pin is a current type voice output connected to the internal D/A converter. The full-scale output of the 8-bit D/A converter is 5mA. This output is sufficient to drive an external 8 ohm speaker by using an external low-power NPN transistor with a 120~160 beta value, such as an 8050D, or its equivalent.

FUNCTIONAL DESCRIPTION

1. Instruction Sets

The W581XX family *PowerSpeech*ä program instruction sets include unconditional and conditional instructions. Most of these instructions are programmed by writing "LD (Load)" and "JP (Jump)" commands and by modifying the contents of the R0~R7, EN, STOP, and MODE registers.

Registers

A. R0~R7 Register

Rn (n:0~7) is an 8-bit register that stores the entry values from 0 to 255 voice groups. The structure of this register is shown below:

Rn:

Bit:	7	6	5	4	3	2	1	0

The default value of Rn is 0.

B. EN Register

EN is an 8-bit register that stores the rising/falling edge enable or disable status information for all trigger pins. This information determines whether each trigger pin is retriggerable, non-retriggerable, overwrite, or non-overwrite. The 8-bit structure of this register and the rising or falling edge of the triggers corresponding to each bit are shown below:

EN:

Bit:	7	6	5	4	3	2	1	0
Trigger:	4r	3r	2r	1r	4f	3f	2f	1f

The digits 1 to 4 represent TG1 to TG4, respectively; "r" represents the rising edge; and "f" represents the falling edge. When any one of the eight bits is set to "1," the rising or falling edge of the corresponding trigger pin can be enabled, interrupting the current state.

The default value of EN register is "1111 1111"".

C. STOP Register

The STOP register stores stop output status information to determine the voltage level of each stop output pin. The 8-bit structure of this register and the stop output pin corresponding to each bit are shown below:

STOP:

Bit:	7	6	5	4	3	2	1	0
STOP:	X	X	X	STPE	STPD	STPC	STPB	STPA

"X" indicates a "don't care" bit.

The default value of the STOP register is "1111 1111"

D. MODE Register

The MODE register is used to store operand information to select the various operating modes as shown below.

MODE:

Bit:	7	6	5	4	3	2	1	0
MODE:	Flash/DC	LED2/STPC	X	Long/Short debounce time	X	X	X	X

A "1" for one of these bits selects the first of the pair of modes indicated; a "0" selects the second of the pair.

Bit 7 is used to determine the output status of LED1 and/or LED2: Flash alternate or synchronous output (by mask option), or DC (LED will be lit constantly without flashing).

Bit 6 determines whether pin15 acts as a LED output pin or STOP output pin.

Bit 4 is used to determine whether the debounce time for all trigger inputs is Long (around 45 mS) or Short (around 350μS) time.

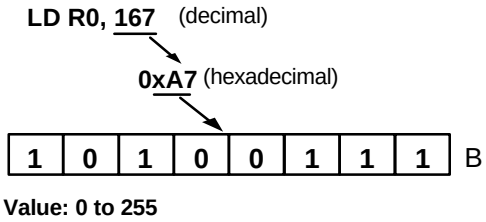
The default value of the MODE register is "11X1 XXXX", that is "Flash, LED2, X, Long debounce time, X,X,X,X"

Commands

A. Unconditional Instructions

Load (LD) command:
This command can load value or operand data into the Rn (n:0~7), EN, STOP, or MODE register.

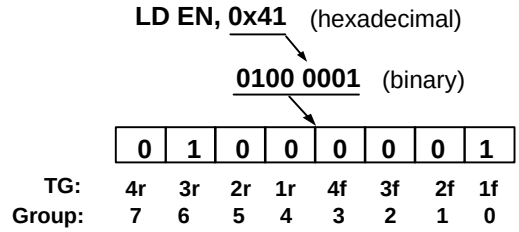
LD Rn, value:
This instruction is used to load a voice group entry value into register Rn (n:0~7), as shown in the following example.
Example:



LD EN, operand:

This instruction is used to define the trigger interrupt settings by loading the operand message into register EN. The following example illustrates how the settings are defined.

Example:



a. When the rising edge of TG3 (3r) is activated, the EN register will cause TG3 to interrupt the current playing state and jump immediately to voice group 6, the voice group that corresponds to 3r.

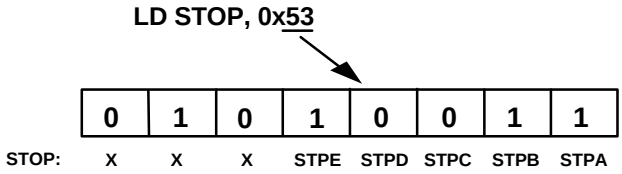
b. When the falling edge of TG1 goes active, the EN register will cause TG1 to interrupt the current playing state and jump immediately to voice group 0, the voice group that corresponds to 1f.

c. No action will be taken when the other trigger pins are pressed, because the corresponding bits are set to "0."

LD STOP, operand:

This instruction loads the operand message into the STOP register to set the output levels of the stop signals. When a particular STOP bit is set to "0," the corresponding stop signal will be an active low output.

Example:



a. The STPA, STPB and STPE output signals will be high outputs.

b. The STPC and STPD output signals will be low outputs.

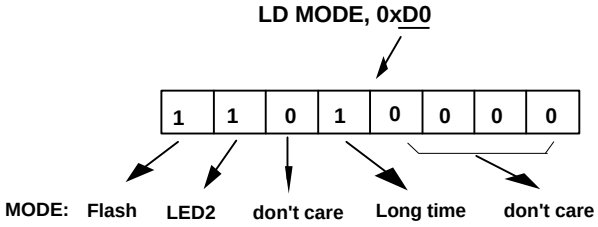
c. The bit6 "1" is a "don't care" bit and so has no effect on the stop signal output setting.

LD MODE, operand:

This instruction is used to select among various operating modes. It loads an operand message into the MODE register to select one mode from each of several pairs of modes.

A "1" for one of these bits selects the first of the pair of modes indicated; a "0" selects the second of the pair. The following example describes the MODE setting of the W581XX product.

Example:



a. The LED is set as a flash type, with the flash frequency 3 Hz.

b. Pin 15 (LED2/STPC) is configured as the LED2 output pin.

c. The bit5 is a "do not care" bit because the 4th pin of W581XX is fixed as TG4 pin.

d. The debounce time of the trigger inputs is set to long time (around 45 mS)

JUMP (JP) Command:

JP value:

Instructs the device to jump directly to the voice group corresponding to the value indicated. The voice group value may range from 0 to 127.

JP Rn (n:0~7):

Instructs the device to jump to whatever voice group is indicated by the value currently stored in register Rn.

B. Conditional Instructions:

Conditional instructions are executed only when the conditions specified in the instructions hold. The conditional instructions are listed below. An explanation of the notation used in the instructions follows.

(Note: There are no conditional instructions for LD MODE.)

Load (LD) command:

LD Rn (n:0~7), value @LAST:

Load the voice group entry value into Rn when the last global repeat sound cycle is finished.

LD Rn (n:0~7), value @TGm_HIGH (or_LOW):

If the m-th (m: 1 to 4) trigger pin status is kept at "High" (or "Low") voltage level, then load the value into Rn register.

LD EN, operand @LAST:

Load the operand message into the EN register when the last global repeat sound cycle is finished.

LD STOP, operand @LAST:

Load the operand message into the STOP register when the last global repeat sound cycle is finished.

Jump (JP) command:

JP value @LAST:

When the last global repeat sound cycle is finished, jump to the group entry value indicated (range: 0 to 127) and begin execution.

JP Rn (n:0~7) @LAST:

When the last global repeat sound cycle is finished, jump to the group entry value indicated by the Rn register and begin execution.

JP value @TGm_HIGH (or _LOW):

If the m-th (m: 1 to 4) trigger pin is kept at "High" (or "Low") voltage level, then jump to the indicated value (range: 0 to 127) and begin execution.

JP Rn (n:0~7) @TGm_HIGH (or _LOW):

If the m-th (m: 1 to 4) trigger pin is kept at "High" (or "Low") voltage level, then jump to the group entry value indicated by the Rn register and begin execution.

C. End Instruction:

END:

This command instructs the chip to immediately cease all activity.

D. Instruction Set List:

	INSTRUCTION	RANGE	DESCRIPTION	DEFAULT VALUE
Unconditional	LD Rn, value (n:0~7)	0–255	Rn ¡ö value	0000 0000
	LD EN, operand	–	EN ¡ö operand	1111 1111
	LD STOP, operand	–	STOP ¡ö operand	xxx1 1111
	LD MODE, operand	–	MODE ¡ö operand	11x1 xxxx
	JP value	0–127	Jump to the group entry value indicated	–
	JP Rn (n:0~7)	0–255	Jump to the group entry indicated by Rn	–
Conditional	LD Rn, value @LAST (n:0~7)	0–255	If last global repeat finished, Rn ¡ö value	–
	LD Rn, value @TGm_HIGH (n:0~7)	0–255	If TGm (m:1- 4) status is high level, Rn ¡ö value	–
	LD Rn, value @TGm_LOW (n:0~7)	0–255	If TGn (m:1-4) status is low level, Rn ¡ö value	–
	LD EN, operand @LAST	–	If last global repeat finished, EN ¡ö operand	–
	LD STOP, operand @LAST	–	If last global repeat finished, STOP ¡ö operand	–
	JP value @LAST	0–127	If last global repeat finished, jump to the group entry value indicated	–
	JP Rn @LAST (n:0~7)	0–255	If last global repeat finished, jump to the group entry value indicated in Rn	–
	JP value @TGm_HIGH	0–127	If TGm (m: 1- 4) status is high level, jump to the group entry value indicated	–
	JP value @TGm_LOW	0–127	If TGm (m: 1- 4) status is low level, jump to the group entry value indicated	–
	JP Rn @TGm_HIGH (n:0~7)	0–255	If TGm (m: 1- 4) status is high level, jump to the group entry value indicated in Rn.	–
	JP Rn @TGm_LOW (n:0~7)	0–255	If TGm (m: 1– 4) status is low level, jump to the group entry value indicated in Rn	–
END	END		Stop all activity and enter standby state	–

2. Mask Option Description

The mask options of the W581XX Enhanced *PowerSpeech*ä are used to select features that cannot be programmed through the chip's registers. The W581XX provides seven mask options, which are listed in the following table:

MASK OPTION	INSTRUCTION	DEMO CHIP OPTION
LED flash type (Asynchronous/Synchronous)	LED_ASYN; (default) LED_SYN	–
LED volume controlled (No/Yes)	LED_VOL_OFF; (default) LED_VOL_ON	If LED_VOL_ON is set, the other mask options will be redundant
LED1: section-controlled (Yes/No)	LED1_S_CTL; (default) LED1_S_OFF	–
LED2: section-controlled /STPC-controlled	LED2_S_CTL; (default) LED2_STC_CTL	–
The 16th pin defined	STPD ; (default) FTEST	For W58100 only STPD is provided
The 17th pin defined	STPE; (default) BUSY	If in CPU mode, the default item is BUSY.
Normal/CPU mode setting	NORMAL; (default) CPU	–

Notes:

1.The demo chip for the W581XX series is the W58100.

2. Mask options can be configured automatically by the W58100.

3. Speech Equation Description

Speech equations are used to define the combination of playback sounds. The following is an example of a speech equation format:

```
i:N

H4+m1*(A_flv+m2*(B_flv+m3*(C_flv+m4*D_flv)))+...T4

m4*[1FFFF]

END
```

where

(1). i defines the group number (from 0 to 255).

(2). N defines the number of global repeats (from 1 to 16).

(3). m1, m2, m3 and **m4** define the number of local repeats (from 1 to 7).

(4). A, B, C, and **D** are files containing ADPCM converted voice data.

(5). _flv is the section control setting, for which the parameters f, l and v are as follows:
f: Sampling Frequency define (default value: f=2)

f	3	2	1	0
Frequency	8 KHz	6 KHz	4.8 KHz	4 KHz

l: LED status define (default value: l=0)

l	1	0
LED status	On	Off

v: Fading Effect define (default value: v=7)

The max. value of the voice is v=7, that is the full scale of the sound. The min. value is v=0, that is 0.1 times of the full scale of the sound. The volume value from large to small is 7> 6> 5> 4> 3> 2> 1> 0.

Note 1: If the section control setting is not defined, the default values are used. For example, in the speech equation H4+A+T4, if the A sound is played, the default values are used. These values are: 6K Hz sampling frequency (f=2), LED off (l=0), and maximum volume output (v=7).

Note 2: If the section control setting is defined, both the f and l digits must be defined together. The v digit is optional.

For example:

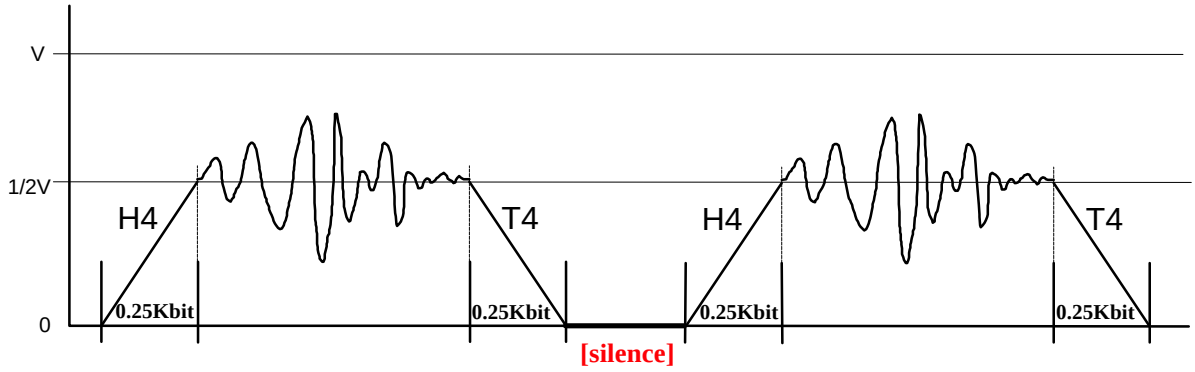
1. H4+A_2+T4: This is a wrong speech equation since only one digit (f) of section control is defined.

2. H4+A_21+T4: This is a correct speech equation with section control. The sampling frequency is 6K Hz (f=2), the LED is ON (l=1), and the volume is maximum value (v=7, the default value).

3. H4+A_210+T4: This is a complete speech equation with section control. The sampling frequency is 6K Hz (f=2), the LED is ON (l=1), and the volume is minimum value (v=0).

(6). [1FFFF] is a period of silence of length 1FFFF (around 5.46 seconds) under the 6K Hz sampling frequency condition. The maximum silence length in one "[ ]" is [FFFFF], that is 1M bits, around 43.69 seconds under the 6K Hz sampling frequency condition. **The silence can't be placed between voices and H4/T4 that will generate a little noise. It is necessary to place it on one line as above example that will gain a good silence.**

(7). H4 and **T4** are the Head file and Tail file with 4-bit ADPCM data format. These two files can be used to eliminate the popping sound which may occur when the sound starts and stops. The following is a sample waveform:



If the voice (H4+....+T4) doesn't play smoothly, then NH4 and NT4 can be used to displace H4 and T4. By using NH4 and NT4 smooth playing voices can be created, however more ROM memory is required (around 0.66Kbits).

(8). Flexible combination of unlimited "( )": This feature can be used in writing speech equations. This feature reduces program writing time but does not save program memory.

For example:

EQ1: H4+2*(A+2*(B+2*(C+3*D)))+T4

EQ2: H4+A+B+C+3*D+C+3*D+B+C+3*D+C+3*D+A+B+C+3*D+C+3*D+B+C+3*D+C+3*D+T4

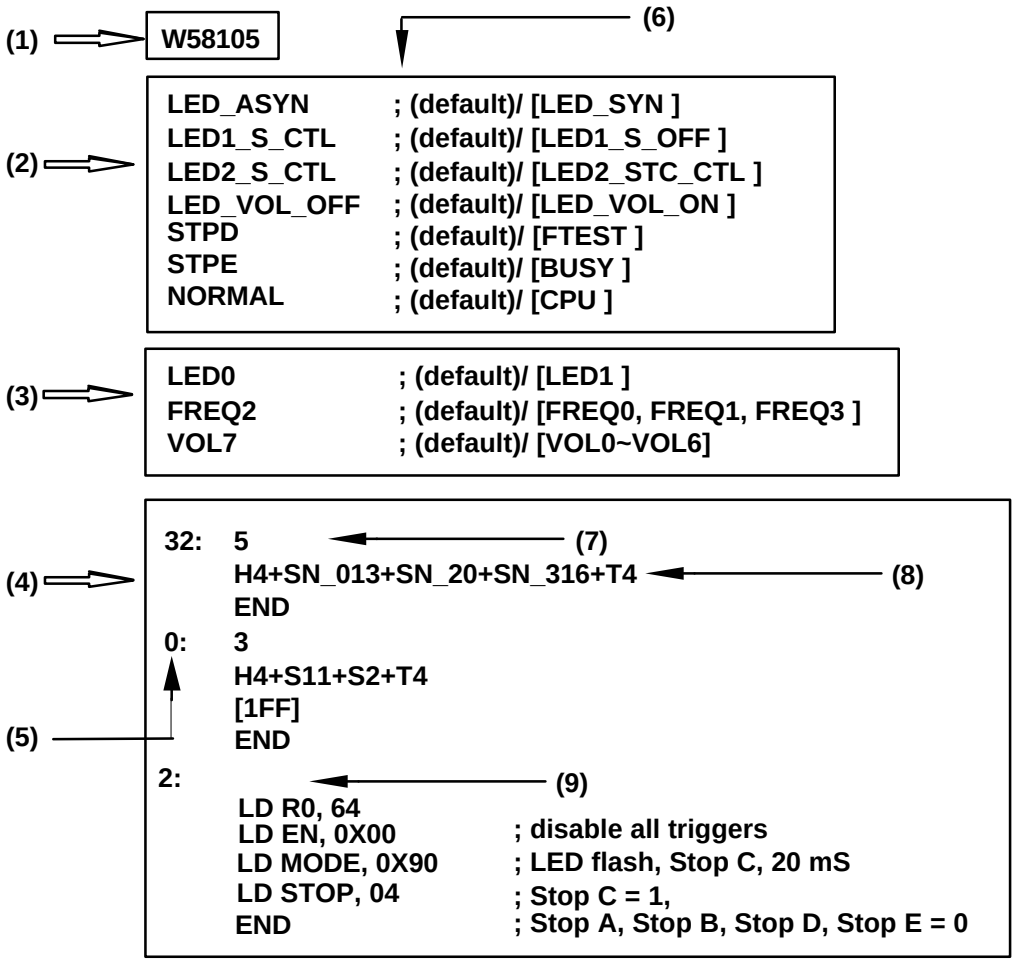
These two speech equation are the same. After compiling, these two speech equations will occupy the same program memory size. Note that EQ1 is much easier to write and understand than EQ2. There is no limit in the number of "( )". However, each pair of "( )"" must be written on the same line.

4. Programmable Power-on Initialization

Whenever the W581XX Enhanced *PowerSpeech*ä is powered on, the programs contained in the 32nd voice group will be executed immediately. Programs can therefore be written into this group to set the initial power-on state. If the user does not wish to execute any programs at power-on, an "END" instruction should be entered in group 32.

5. *PowerSpeech*ä Program Format

The W581XX *PowerSpeech*ä has a programming language to define the product functions. An example of the W581XX *PowerSpeech*ä program format is shown below. Explanatory notes follow the example (for reference only)



Notes:

(1) **Bodies:** The user must first define the enhanced *PowerSpeech*ä body to be used, or else an error message will appear during compiation. The enhanced *PowerSpeech*ä bodies include the following:

W581XX: W58101, W58102, W58103, W58104, W58105, W58106, W58110, W58115, and W58120

(2) **Mask Options:** See above description.

(3) **Declarations:** State the output frequency, LED on/off state and volume variable, as follows:

LED on/off:

-- LED0: LED off (default)
-- LED1: LED on

Output frequency:

-- FREQ0: 4KHz
-- FREQ1: 4.8 KHz
-- FREQ2: 6 KHz (default)

-- FREQ3: 8 KHz

Volume Variable:

-- VOL0: (minimum) 0.1 times volume of the full scale of the sound
-- VOL1
-- VOL2
-- VOL3
-- VOL4
-- VOL5
-- VOL6
-- VOL7: (maximum) The full scale of the sound (default)

(4) **Program body**: Write application program and speech operations, including the following:

Define entry point of speech group.

Determine the number of global repeats.

Describe speech equations.

Define the register values and instructions.

Note 1: The maximum program memory size that can be used by the W581XX is 64K bits.

Note 2: Every GO instruction occupies 48 bits, and every group entry value (0~255) occupies 16 bits. Therefore, a total of around 1,300 GO instructions can be used in a program.

Note 3: The GO instruction includes the following contents:

(a). Instruction set: Like "LD R0, XX", "JP R3 @TG2_LOW" ...etc.

(b). END instruction: END

(c). Every ADPCM file for the speech equations. For example, in the speech equation: H4+A+B_217+C+T4, there are a total of 5 GO instructions: H4, A, B_217, C, and T4.

These GO instructions do not include the Mask Option and Declaration instructions.

For example: (for reference only)

```
W58105                  ; Body define, 0 GO instruction

LED_SYN                 ; Mask Option, 0 GO instruction
LED1_S_CTL              ; Mask Option, 0 GO instruction
FREQ3                   ; Declaration, 0 GO instruction
VOL5                    ; Declaration, 0 GO instruction

32:                     ; POI group entry, 16 bits
    LD EN, 0X0F             ; 48 bits, 1 GO instruction
    LD STOP, 0X10                   ; 1 GO instruction
    END                     ; END instruction, 1 GO instruction

0:3                     ; Group entry, 16 bits.
```

```
                ; The global repeat 3 occupied 4 bit ROM size included in the Group entry.
        LD EN, 0X00             ; 1 GO instruction
        LD STOP, 0X1F           ; 1 GO instruction
        H4+3*A+B_217+C+T4       ; Speech equation, total with 6 GO
        [1FFF]                  ; instructions. Include:
                                ; H4, 3*A, B_217, C, [1FFF], and T4.
        LD STOP, 0X10                   ; 1 GO instruction
        LD EN, 0X0F             ; 1 GO instruction
        END                     ; 1 GO instruction

1:                                      ; 16 bits
        H4+A+B+C_30+D+E+T4              ; 8 GO instructions:
        [1FFF]                          ; H4, A, B, C_30, [1FFF], D, E, and T4
        END                     ; 1 GO instruction

2:                                      ; 16 bits
        JP 100 @TG4_LOW         ; 1 GO instruction
        END                     ; 1 GO instruction
100:                    ; 16 bits
        H4+3*(A+2*B)+T4 ; Equal to: H4+A+2*B+A+2*B+A+2*B+T4, total with 8 GO
                        ; instructions.
        END             ; 1 GO instruction
```

In the program example shown above, there are a total of 34 GO instructions and 101 group entries (= 100+1, because the maximum group entry to be used is 100). The total program memory size for this program is 3,248 bits (34*48+101*16=3248).

Note 4: To use the 64K bits program memory size more efficiently, the following suggestions can be followed:

(a). The voice groups should be used consecutively and without skipping. For example: 0, 1, 2, 3....255, is better than 0, 1, 100, 150... 255.

(b). If the program includes several voice groups, then the voice group with the most GO instructions should be placed at the end of the program.

(5) **Group body:** Define the voice group entry point.

Product	Group entry points	TG H/W entry points	Power-on entry point
W581XX	0-255	0-7	32

(6) **Note:** A semicolon (";") is used to distinguish characters that are not part of the program. Characters written to the right of the semicolon are not considered part of the program.

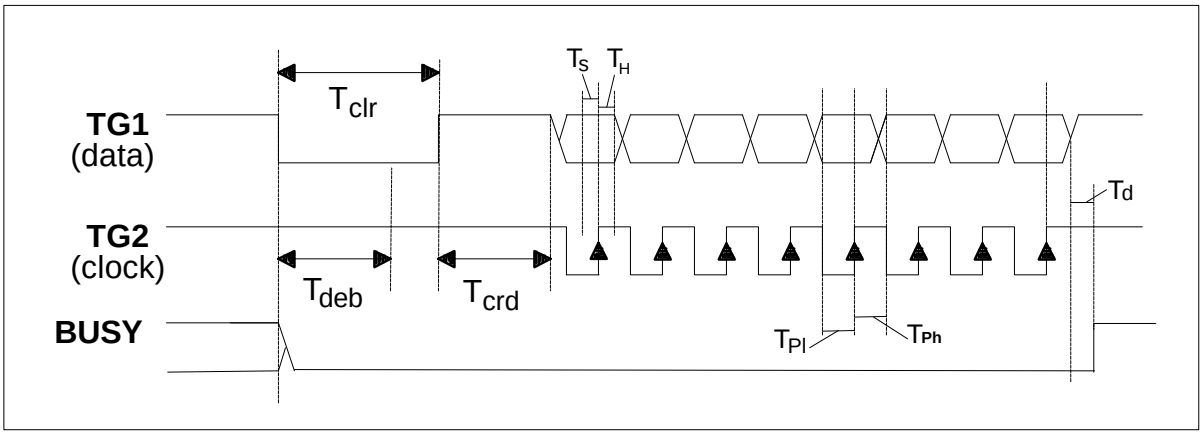
(7) **Global Repeat:** The global repeat instruction is "n" where n is from 1 to 16. This instruction must be placed on the same line as the group entry point.

(8) **Speech equation:** These are used to define the combination of playback sounds.

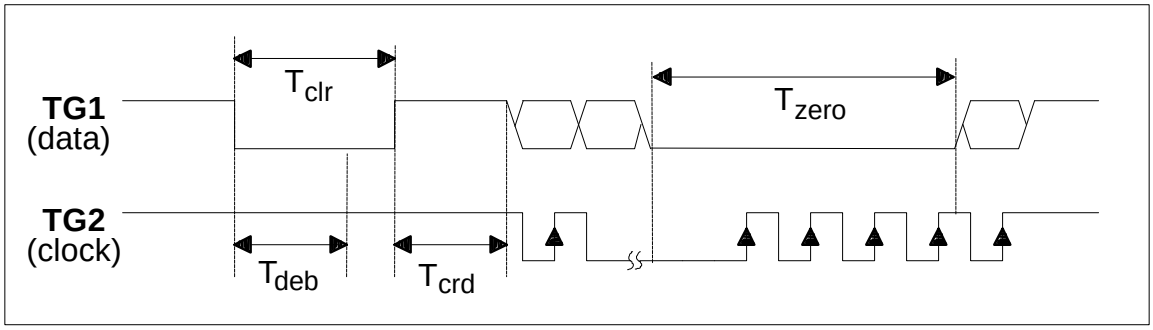
(9) **Blank:** A voice group entry point must be followed by one full blank line without any instructions or speech equations.

7. CPU interface

The W581XX can communicate with an external microprocessor through a simple serial CPU interface. **The voice group transmitted from CPU must between 128 and 255.** It is shown below:



Tclr > Tdeb, Tdeb = 256/Fs or 2/Fs depend on long or short debounce, which Fs is the speech sampling rate of last synthesized speech. Fs default value is 6kHz when power on or after reset.
Tcrd > 5uS
Ts > 500nS
Th> 50nS
Tpl > 500ns
Tph > 500 ns
Tzero < 128/Fs or 1/Fs depend on long or short debounce
Td < 50 ns



Note:

1. Tdeb means the "Debounce time" which can be "Long" or "Short", depending on bit4 of the MODE register. If the MODE register has a "1" in bit4, then the debounce time will be set to "Long" If bit4 is "0"," then the debounce time will be set to "Short". In case of last synthesized speech sampling rate is 6kHz, for long debounce Tdeb = 256/6k = 42.7ms, for short debounce Tdeb = 2/6k = 333us.

2. Only when Tclr is longer than Tdeb, the receiving data counter of W581xx can be clear to zero. Considering about the speech sampling frequency may shift due to the deviation of component or voltage, at least 20% mark up,i.e. 42.7ms * (1 + 20%) = 51.2 ms for long debounce when the last synthesized speech sampling rate is 6kHz, is suggested to guarantee the success of debounce.
For short debounce at 6kHz speech sampling rate, 333us * (1 + 20%) = 400 us were recommended.

3. Tcrd is the "CPU Reset Delay" time.

4. During data transfer phase, Tzero, the consecutive low state on TG1(data), could not be longer than half of debounce time Tdeb, otherwise it may be treated as a debounce and then reset the receiving data counter. For the same reason of possible frequency shift just like Tclr, 20% margin were suggested. At Fs = 6kHz and long debounce, the consecutive low state on TG1(Tzero) can not more than 0.5 * Tdeb * (1 - 20%) = 0.5 * 42.7 ms * 0.8 = 17 ms for long debounce and 0.5 * 333us * (1-20%) = 133 us for short debounce.

To program W581XX in CPU mode, the ***CPU*** keyword must be added and TG1F/ TG1R/ TG2F/ TG2R must be disabled including the directly interrupt and condition commands. So In the CPU mode the "@TG1_high/low" and "@TG2_high/low" are illegal and bit0, bit1, bit4 and bit5 of the EN register must be set to "0". The following example shows this: (for reference only)

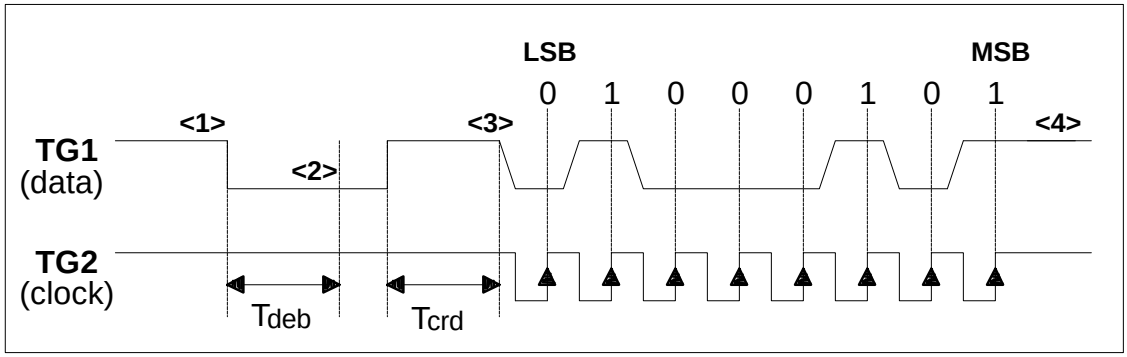
```
W58105

CPU             ; Reserved word, for CPU mode mask option
LED1_S_CTL
LED2_STC_CTL

32:
        LD EN,0X0C          ; disable TG1F/TG1R/TG2F/TG2R
        LD MODE,0X8F        ; bit4 of the MODE register is 0
                            ; so the debounce time is selected "Short" (arround 350 μS)
        END
0:
1:
4:
5:
        END
162:                                    ; CPU interrupt
        H4+voice1+T4
        END
```

The waveform must be sent from the external μC through TG1 and TG2 to control the playing of the voice group. The waveform looks like this:

In the program example (for reference only) shown above, the μC will transfer the number 162 to interrupt W581XX. The number 162 (decimal) is equal to 10100010b (binary).



<1> When TG1 is pulled low, the W581XX stops playing voice or instructions and waits for data from the external μC.

<2> When TG1F debounces OK, the W581XX clears the CPU receiving buffer.

<3> 8-bit data are transferred by TG1 (data) and TG2 (clock). The first bit of the data to be sent is the *LSB* .

<4> TG1 returns high and starts the CPU interrupt. In this case W58105 will play the *H4+voice1+T4* sections and the BUSY pin is pulled high until the "END" instruction is reached.

Example:

<1> Micro controller program (used W74C250 as the controller)

```
; W581 CPU mode test program
; W581 chosen the long debounce time
; W74C260 with the RC oscillator: 11 Kohm Rosc
; CLOCK pin connect to RE0
; DATA pin connect to RE1
; BUSY pin connect to RC3

addr_h          equ     ram20           ; store the W581CPU data, high byte
addr_l          equ     ram21           ; low byte
;................................................
                org     000h
                mov     IEF,#00000000b          ; disable all interrupt
                mov     HEF,#00000000b
                mov     PEF,#0000B
                jmp     begin
                org     004h                    ; divider0
                rtn
                org     008h                    ; timer0
                rtn
                org     00ch
                jmp     portrc
                org     014h                    ; divider1
                rtn
                org     020h                    ; timer1
```

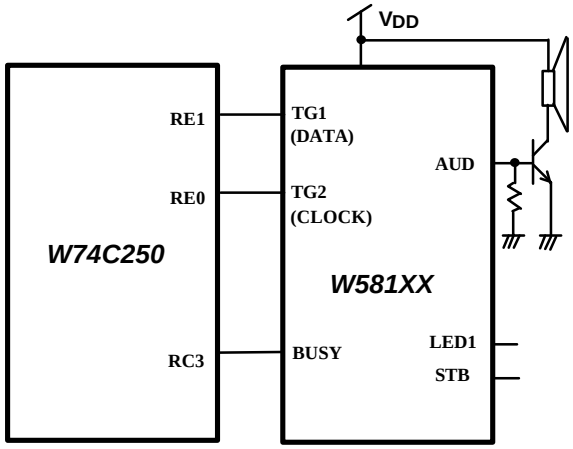
```
                rtn
;.....................................................
begin:
                mov     PM0,#1100B          ; RC.3 as W581 busy pin
                mov     PM1,#1111b          ; RA is input pins
                mov     PM2,#1111b          ; RB is input pins
                mov     wre,#1111b          ; data and clock pins pulled high
                mov     RE,wre
                mov     RA,wre
                mov     RB,wre
;.....................................................
                mov     ram10,#2H           ; power on waiting 200ms in order to
DDD5b:          mov     ram11,#0FH          ; complete W581 setup procedure
DDD4b:          mov     ram12,#0FH
DDD3b:          mov     ram13,#0FH
DDD2b:          mov     ram14,#0FH
DDD1b:          dec     ram14
                jnz     DDD1b
                dec     ram13
                jnz     DDD2b
                dec     ram12
                jnz     DDD3b
                dec     ram11
                jnz     DDD4b
                dec     ram10
                jnz     DDD5b
;.....................................................
                mov     addr_h,#0010b       ; send group 32 for POI
                mov     addr_l,#0000b
                call    playsound           ;serial output subroutine
                mov     IEF,#00000100b      ; enable interrupt port RC
                mov     HEF,#00000100b
                mov     PEF,#1000B
set128:
                mov     addr_h,#1000b
                mov     addr_l,#0000b
sleep:
                en      int
                Hold
                inc     addr_l          ; increment W581 address data
                mov     wrf,addr_h      ; from 128 to 255
                adcr    wrf,#0
                mov     addr_h,wrf
                jz      set128
                jmp     sleep
;.....................................................
portrc:
                mova            wre,rc
                xrl     wre,#1111b
                jb3     W581CPU
```

```
over_rc:
                clr     PSR0
                rtn
W581CPU:
                call    playsound       ; W581 play sound
                jmp     over_rc
;.....................................
playsound:
                mov     wr8,#3          ; retransmit 3 times
send_again:
                mov     wr0,#1          ; make tg1 low
                mov     RE,wr0
                mov     ram10,#08H      ; tg1 low for >70mS to fit the long debounce time spec.
DDD4a:          mov     ram11,#0FH
DDD3a:          mov     ram12,#0FH
DDD2a:          mov     ram13,#0FH
DDD1a:          dec     ram13
                jnz     DDD1a
                dec     ram12
                jnz     DDD2a
                dec     ram11
                jnz     DDD3a
                dec     ram10
                jnz     DDD4a
                mov     wr0,#3          ; Tcrd -> CPU reset time > 5µS
                mov     RE,wr0
                nop
                nop
                nop
                nop
;.....................................
                mov     acc,addr_l      ; LSB first send
                jb0     bit0_h
                mov     wr0,#0          ; bit0 <-- low
                mov     RE,wr0
                mov     wr0,#1
                nop
                mov     RE,wr0
                jmp     bit1
bit0_h:                                 ; bit0 <-- high
                mov     wr0,#2
                mov     RE,wr0
                mov     wr0,#3
                nop
                mov     RE,wr0
bit1:
                mov     acc,addr_l
                jb1     bit1_h
                mov     wr0,#0          ; bit 1 <-- low
                mov     RE,wr0
```

```
                mov     wr0,#1
                nop
                mov     RE,wr0
                jmp     bit2
bit1_h:                                 ; bit 1 <-- high
                mov     wr0,#2
                mov     RE,wr0
                mov     wr0,#3
                nop
                mov     RE,wr0
bit2:
                mov     acc,addr_l
                jb2     bit2_h
                mov     wr0,#0          ; bit 2 <-- low
                mov     RE,wr0
                mov     wr0,#1
                nop
                mov     RE,wr0
                jmp     bit3
bit2_h:                                 ; bit 2 <-- high
                mov     wr0,#2
                mov     RE,wr0
                mov     wr0,#3
                nop
                mov     RE,wr0
bit3:
                mov     acc,addr_l
                jb3     bit3_h
                mov     wr0,#0          ; bit 3 <-- low
                mov     RE,wr0
                mov     wr0,#1
                nop
                mov     RE,wr0
                jmp     bit4
bit3_h:                                 ; bit 3 <-- high
                mov     wr0,#2
                mov     RE,wr0
                mov     wr0,#3
                nop
                mov     RE,wr0
bit4:
                mov     acc,addr_h
                jb0     bit4_h
                mov     wr0,#0          ; bit 4 <-- low
                mov     RE,wr0
                mov     wr0,#1
                nop
                mov     RE,wr0
                jmp     bit5
bit4_h:                                 ; bit 4 <-- high
```

```
                mov     wr0,#2
                mov     RE,wr0
                mov     wr0,#3
                nop
                mov     RE,wr0
bit5:                                           ; for next nibble addr_h
                mov     acc,addr_h
                jb1     bit5_h
                mov     wr0,#0          ; bit 5 <-- low
                mov     RE,wr0
                mov     wr0,#1
                nop
                mov     RE,wr0
                jmp     bit6
bit5_h:                                 ; bit 5 <-- high
                mov     wr0,#2
                mov     RE,wr0
                mov     wr0,#3
                nop
                mov     RE,wr0
bit6:
                mov     acc,addr_h
                jb2     bit6_h
                mov     wr0,#0          ; bit 6 <-- low
                mov     RE,wr0
                mov     wr0,#1
                nop
                mov     RE,wr0
                jmp     bit7
bit6_h:                                 ; bit 6 <-- high
                mov     wr0,#2
                mov     RE,wr0
                mov     wr0,#3
                nop
                mov     RE,wr0
bit7:
                mov     acc,addr_h
                jb3     bit7_h
                mov     wr0,#0          ; bit 7 <-- low
                mov     RE,wr0
                mov     wr0,#1
                nop
                mov     RE,wr0
                jmp     oversend
bit7_h:                                 ; bit 7 <-- high
                mov     wr0,#2
                mov     RE,wr0
                mov     wr0,#3
                nop
                mov     RE,wr0
```

```
oversend:
                mov     wr0,#3
                mov     RE,wr0
                mov     ram12,#0FH      ; delay >200uS
D2a:            mov     ram13,#0FH
D1a:            dec     ram13
                jnz     D1a
                dec     ram12
                jnz     D2a
                mova            wre,rc          ; check BUSY pin high or not
                skb3    wre
                jmp     next_time       ; if BUSY pin still low send the data again
                rtn
next_time:
                dskz    wr8
                jmp     send_again
                rtn
;.....................................................
                end
```



<2> W581xx CPU mode program

```
w58104

CPU

POI:
   ld  mode,0xff
   ld  en,0x00
  END

128:
   h4+1+2+8+t4
   END
129:
   h4+1+2+9+t4
   END
130:
   h4+1+3+0+t4
   END
131:
   h4+1+3+1+t4
   END
132:
   h4+1+3+2+t4
   END
133:
   h4+1+3+3+t4
```

END
134:
h4+1+3+4+t4
END
135:
h4+1+3+5+t4
END
136:
h4+1+3+6+t4
END
137:
h4+1+3+7+t4
END
138:
h4+1+3+8+t4
END
139:
h4+1+3+9+t4
END
140:
h4+1+4+0+t4
END
141:
h4+1+4+1+t4
END
142:
h4+1+4+2+t4
END
143:
h4+1+4+3+t4
END
144:
h4+1+4+4+t4
END
145:
h4+1+4+5+t4
END
146:
h4+1+4+6+t4
END
147:
h4+1+4+7+t4
END
148:
h4+1+4+8+t4
END
149:
h4+1+4+9+t4
END
150:
h4+1+5+0+t4
END
151:
h4+1+5+1+t4
END
152:
h4+1+5+2+t4
END
153:
h4+1+5+3+t4
END
154:
h4+1+5+4+t4
END
155:
h4+1+5+5+t4
END
156:
h4+1+5+6+t4
END
157:
h4+1+5+7+t4
END
158:
h4+1+5+8+t4
END
159:
h4+1+5+9+t4
END
160:
h4+1+6+0+t4
END
161:
h4+1+6+1+t4
END
162:
h4+1+6+2+t4
END
163:
h4+1+6+3+t4
END
164:
h4+1+6+4+t4
END
165:
h4+1+6+5+t4
END
166:
h4+1+6+6+t4
END

167:
h4+1+6+7+t4
END
168:
h4+1+6+8+t4
END
169:
h4+1+6+9+t4
END
170:
h4+1+7+0+t4
END
171:
h4+1+7+1+t4
END
172:
h4+1+7+2+t4
END
173:
h4+1+7+3+t4
END
174:
h4+1+7+4+t4
END
175:
h4+1+7+5+t4
END
176:
h4+1+7+6+t4
END
177:
h4+1+7+7+t4
END
178:
h4+1+7+8+t4
END
179:
h4+1+7+9+t4
END
180:
h4+1+8+0+t4
END
181:
h4+1+8+1+t4
END
182:
h4+1+8+2+t4
END
183:
h4+1+8+3+t4
END
184:
h4+1+8+4+t4
END
185:
h4+1+8+5+t4
END
186:
h4+1+8+6+t4
END
187:
h4+1+8+7+t4
END
188:
h4+1+8+8+t4
END
189:
h4+1+8+9+t4
END
190:
h4+1+9+0+t4
END
191:
h4+1+9+1+t4
END
192:
h4+1+9+2+t4
END
193:
h4+1+9+3+t4
END
194:
h4+1+9+4+t4
END
195:
h4+1+9+5+t4
END
196:
h4+1+9+6+t4
END
197:
h4+1+9+7+t4
END
198:
h4+1+9+8+t4
END
199:
h4+1+9+9+t4
END
200:

h4+2+0+0+t4
END
201:
h4+2+0+1+t4
END
202:
h4+2+0+2+t4
END
203:
h4+2+0+3+t4
END
204:
h4+2+0+4+t4
END
205:
h4+2+0+5+t4
END
206:
h4+2+0+6+t4
END
207:
h4+2+0+7+t4
END
208:
h4+2+0+8+t4
END
209:
h4+2+0+9+t4
END
210:
h4+2+1+0+t4
END
211:
h4+2+1+1+t4
END
212:
h4+2+1+2+t4
END
213:
h4+2+1+3+t4
END
214:
h4+2+1+4+t4
END
215:
h4+2+1+5+t4
END
216:
h4+2+1+6+t4
END
217:
h4+2+1+7+t4
END
218:
h4+2+1+8+t4
END
219:
h4+2+1+9+t4
END
220:
h4+2+2+0+t4
END
221:
h4+2+2+1+t4
END
222:
h4+2+2+2+t4
END
223:
h4+2+2+3+t4
END
224:
h4+2+2+4+t4
END
225:
h4+2+2+5+t4
END
226:
h4+2+2+6+t4
END
227:
h4+2+2+7+t4
END
228:
h4+2+2+8+t4
END
229:
h4+2+2+9+t4
END
230:
h4+2+3+0+t4
END
231:
h4+2+3+1+t4
END
232:
h4+2+3+2+t4
END
233:
h4+2+3+3+t4

```
    END
234:
    h4+2+3+4+t4
    END
235:
    h4+2+3+5+t4
    END
236:
    h4+2+3+6+t4
    END
237:
    h4+2+3+7+t4
    END
238:
    h4+2+3+8+t4
    END
239:
    h4+2+3+9+t4
    END
240:
    h4+2+4+0+t4
    END
241:
    h4+2+4+1+t4
    END
242:
    h4+2+4+2+t4
    END
243:
    h4+2+4+3+t4
    END
244:
    h4+2+4+4+t4
    END
245:
    h4+2+4+5+t4
    END
246:
    h4+2+4+6+t4
    END
247:
    h4+2+4+7+t4
    END
248:
    h4+2+4+8+t4
    END
249:
    h4+2+4+9+t4
    END
250:
    h4+2+5+0+t4
    END
251:
    h4+2+5+1+t4
    END
252:
    h4+2+5+2+t4
    END
253:
    h4+2+5+3+t4
    END
254:
    h4+2+5+4+t4
    END
255:
    h4+2+5+5+t4
    END
```

8. Programming Examples (for reference only)

This section presents several programming examples for the W581XX enhanced *PowerSpeech*ä chips. User programs should be written in ASCII code using a text editor. After compiling, the sound effects resulting from the programs can be tested by using a Winbond demo board.

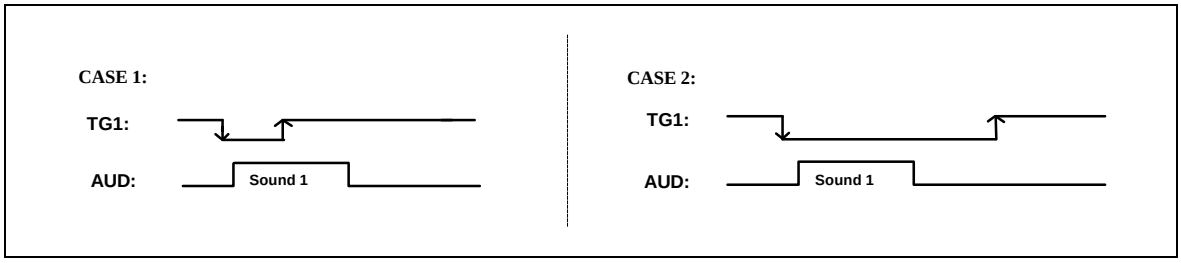
Example1: Four playing mode settings:

a. One-shot Trigger Mode

```
0:                      ; TG1 falling edge group entry point
   LD EN, 0X01          ; Enable TG1 falling edge input only
   H4+sound+T4
```

END

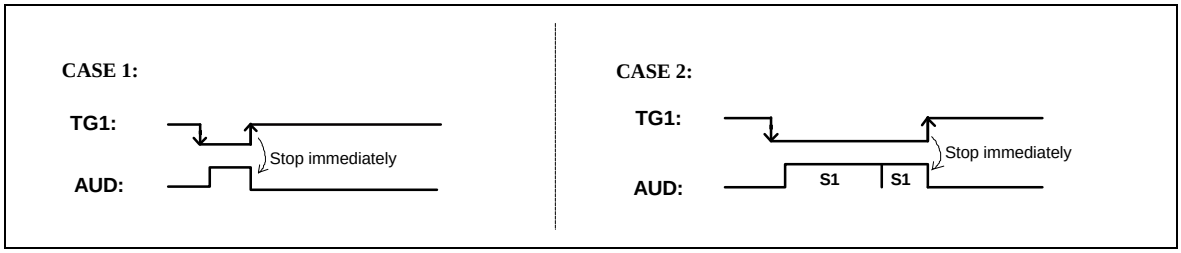
The timing diagram for this example is shown below:



b. Level Hold Trigger Mode

```
0:                          ; TG1 falling edge group entry point
    LD EN, 0X11             ; Enable TG1 falling and rising edge input
    H4+sound1+T4
    JP 0
4:                          ; TG1 rising edge group entry point
    END
```

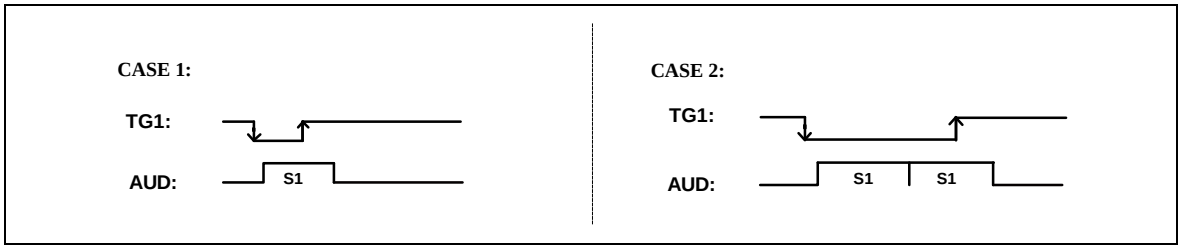
The timing diagram is shown below:



c. Completed Cycle Level Hold

```
0:                          ; TG1 falling edge group entry point
    LD EN, 0X01             ; Enable TG1 falling edge input only
    H4+sound1+T4
    JP 0 @TG1_LOW           ; If TG1 state is low, jump to 0 entry point
    END
```

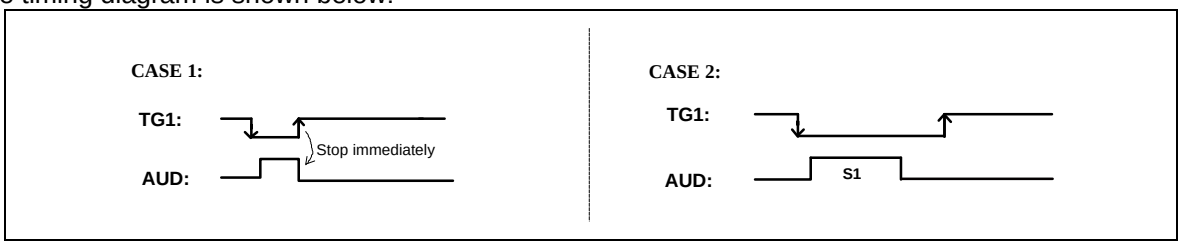
The timing diagram is shown below:



d. Single Cycle Level Hold

```
0:                      ; TG1 falling edge group entry point
   LD EN, 0X11          ; Enable TG1 falling and rising edge input
   H4+sound1+T4
   END
4:
   END
```

The timing diagram is shown below:

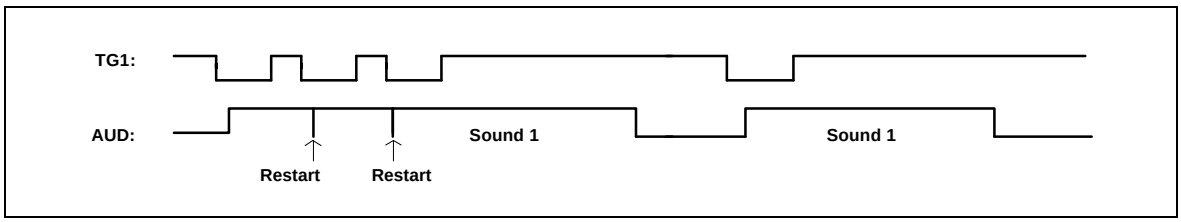


Example 2: Retriggerable and Non-retriggerable setting

a. Retriggerable:

```
0: LD EN, 0x01
        .
        .
        .
    END
```

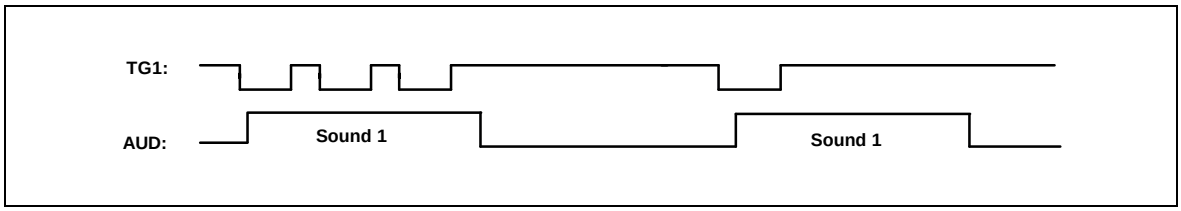
The timing diagram is shown below:



b. Non-retriggerable:

```
0: LD EN, 0x00
       .
       .
       .
   LD EN, 0x11
   END
```

The timing diagram is shown below:



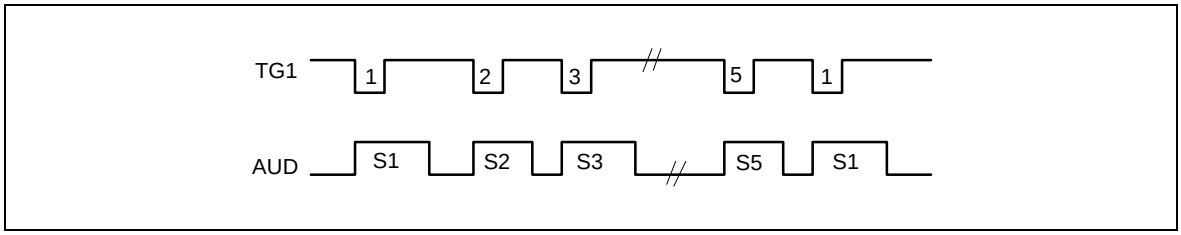
Example 3: Serial Playing Mode (5 segments)

W58105	

32:
LD R0, 8
LD EN, 0X01
END
	10:
	LD R0, 11
	H4+S3+T4
	END

0: JP R0	11: LD R0, 12 H4+S4+T4 END
8: LD R0, 9 H4+S1+T4 END	12: LD R0, 8 H4+S5+T4 END
9: LD R0, 10 H4+S2+T4 END	

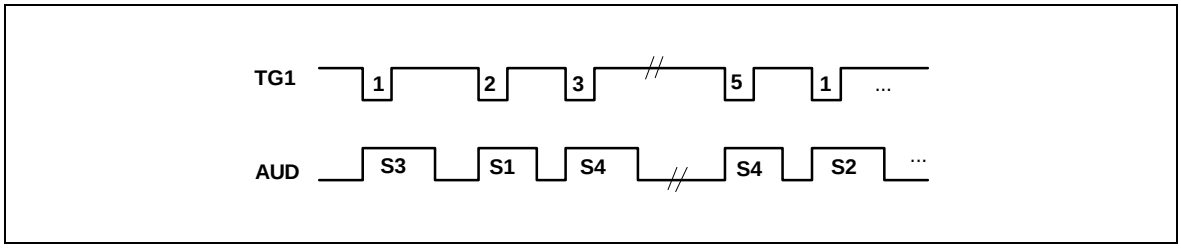
The timing diagram is shown below:



Example 4: Random (1)

W58105
32:
LD EN, 0X01
LD R0, 8
END
0:
LD EN, 0X00
JP R0
8:
JP 18 @TG1_HIGH
9:
JP 19 @TG1_HIGH
10:
JP 20 @TG1_HIGH
11:
JP 21 @TG1_HIGH
JP 8
	18:
	H4+S1+T4
	LD R0, 9
	JP 31
	19:
	H4+S2+T4
	LD R0, 8
	JP 31
	20:
	H4+S3+T4
	LD R0, 11
	JP 31
	21:
	H4+S4+T4
	LD R0, 10
	31:
	LD EN, 0X01
	END

The timing diagram is shown below:

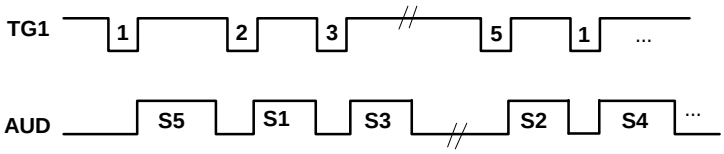


Example 5: Random (2)

W58105
32:
LD EN, 0X11
END
0:
LD R0, 8
[300]
LD R0, 9
[300]
	8:
	H4+S4+T4
	END
	9:
	H4+S1+T4
	END
	10:
	H4+S5+T4

```
    LD R0, 10
    [300]
    LD R0, 11
    [300]
    LD R0, 12
    [300]
    JP 0
4:
    JP R0
    END
11:
    H4+S3+T4
    END
12:
    H4+S2+T4
    END
```

The timing diagram is shown below:



7. Application Examples (for reference only)

The following paragraph presents several special application examples.

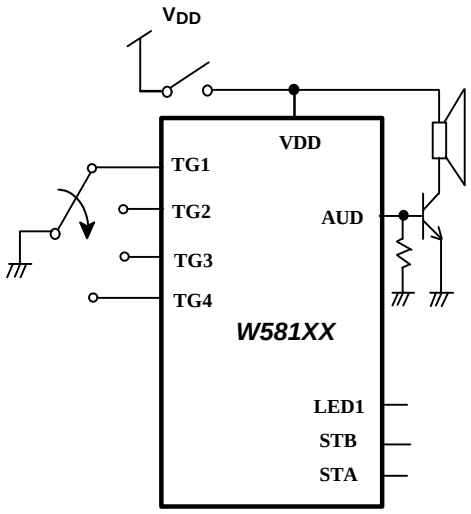
Example 1: Power-on Trigger:

If one of the trigger pins is grounded, then the sound corresponding to that trigger will be played out at power-on.

Program:

```
W58105
32:
    LD EN, 0X00
    JP 8 @TG1_LOW
    JP 9 @TG2_LOW
    JP 10 @TG3_LOW
    JP 11 @TG4_LOW
    LD EN, 0X0F
 0:
 8:
    H4+S1+T4
    LD EN, 0X0F
    END
 1:
 9:
    H4+S2+T4
    LD EN, 0X0F
    END
 2:
10
    H4+S3+END
    LD EN, 0X0F
    END
 3:
11:
    H4+S4+T4
    LD EN, 0X0F
    END
```

Application Circuit:



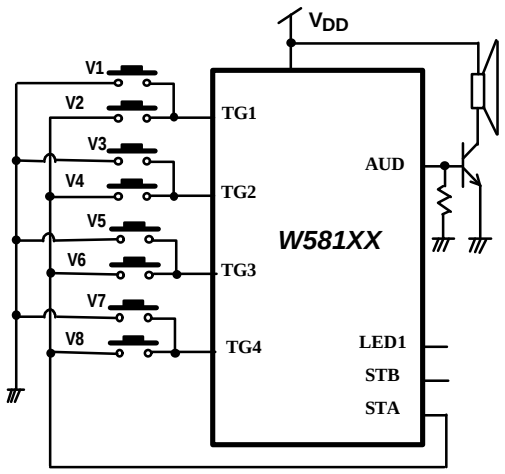
Example 2: 8 TG Input Application:
In this application, the 4 trigger inputs are expanded to 8 trigger inputs.
Program:

W58105

```
32:
    LD MODE, 0XA0
    LD STOP, 0X00          ; STPA set to low level
    LD EN, 0X0F            ; One Shot play mode
    END
0:
    LD STOP, 0X01          ; STPA set to high level
    JP 8 @TG1_HIGH         ; check high
    H4+V1+T4               ; play V1
    LD STOP, 0X00          ; STPA set to low level
    END
8:                         ; pseudo trigger pin
    H4+V2+T4               ; play V2
    LD STOP, 0X00          ; STPA set to low level
    END
1:
    LD STOP, 0X01          ; STPA set to high level
    JP 9 @TG2_HIGH         ; check high
    H4+V3+T4               ; play V3
    LD STOP, 0X00          ; STPA set to low level
    END
9:                         ; pseudo trigger pin
```

```
        H4+V4+T4                ; play V4
        LD STOP, 0X00           ; STPA set to low level
        END
2:
        LD STOP, 0X01           ; STPA set to high level
        JP 10 @TG3_HIGH         ; check high
        H4+V5+T4                ; play V5
        LD STOP, 0X00           ; STPA set to low level
        END
10:                             ; pseudo trigger pin
        H4+V6+T4                ; play V6
        LD STOP, 0X00           ; STPA set to low level
        END
3:
        LD STOP, 0X01           ; STPA set to high level
        JP 11 @TG4_HIGH         ; check high
        H4+V7+T4                ; play V7
        LD STOP, 0X00           ; STPA set to low level
        END
11:                             ; pseudo trigger pin
        H4+V8+T4                ; play V8
        LD STOP, 0X00           ; STPA set to low level
        END
```

Application Circuit



Delay time: V1~V8: 2mS + debounce time

Example 3: Fading Effect Application

This program will play the "voice" with a reducing volume. The STOP signals will be turned off (1: on, 0: off) one by one.

Program:

```
W58105

32:
    LD MODE, 0X10          ; pin15 set as STPC output
    LD STOP, 0X00          ; set STPA, STPB, and STPC = 0
    LD EN, 0X0F
    END
0:
   LD STOP,0X1F            ; STPA, B, C, D, E turn ON
   H4+voice_217            ; sample rate= 6K, LED1 turn ON, volume= 7 (the biggest)
   LD STOP,0X1E            ; STPB, C, D, E turn ON, STPA turn OFF
   voice_306               ; sample rate= 8K, LED1 turn OFF, volume= 6
   voice_215               ; sample rate= 6K, LED1 turn ON, volume= 5
   LD STOP,0X1C            ; STPC, D, E turn ON, STPA, B turn OFF
   voice_304               ; sample rate= 8K, LED1 turn OFF, volume= 4
   voice_213               ; sample rate= 6K, LED1 turn ON, volume= 3
   LD STOP,0X18            ; STPD, E turn ON, STPA, B, C turn OFF
   voice_302               ; sample rate= 8K, LED1 turn OFF, volume= 2
   LD STOP,0X10            ; STPE turn ON, STPA, B, C, D turn OFF
   voice_211               ; sample rate= 6K, LED1 turn ON, volume= 1
   LD STOP,0X00            ; STPA, B, C, D, E turn OFF
   voice_300+T4            ; sample rate= 8K, LED1 turn OFF, volume= 0
   END
```

Example 4: Register Application

This program combines one sentence when TG1 is triggered before. It will then play a sentence with the TG2 voice and TG3 voice. TG2 and TG3 are sequence functions. Words for TG2 and TG3 can be chosen and trigger TG1 will play the combined sentence.

Program:

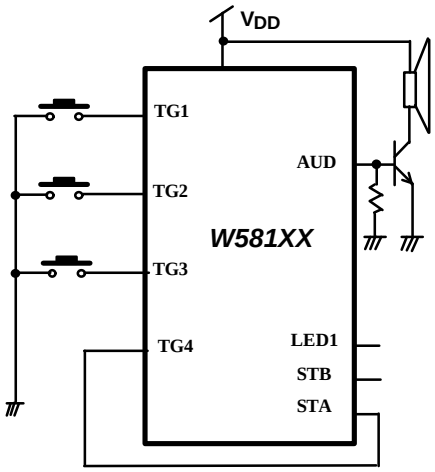
```
W58105

32:
    LD EN,0x00
    LD MODE, 0X10          ; pin15 set as STPC output
    LD STOP, 0xff          ; set STPA, STPB, STPC, STPD, STPE = 1
    LD R1,10               ; set R1, R2, R3, R4 initial data
    LD R2,20
    LD R3,10
```

```
      LD R4,20
      LD EN, 0X07                ; enable TG1, TG2, TG3
      END
0:                               ; TG1 plays the combined sentence
       LD STOP,0xfe              ; set TG4 low as the combine function flag
      JP R3                      ; play the R3+R4 voice
1:
      JP R1                      ; play sequentially Voice1 --> Voice2 --> Voice3 --> Voice1...
2:
      JP R2                      ; play sequentially Voice4 --> Voice5 --> Voice6 --> Voice4...
10:
      LD R1, 11
      LD R3, 10
      H4+I+am+T4                 ; Voice1
      JP R4 @TG4_LOW
      END
11:
      LD R1, 12
      LD R3, 11
      H4+you+are+T4              ; Voice2
      JP R4 @TG4_LOW
      END
12:
      LD R1, 10
      LD R3, 12
      H4+she+is+T4               ; Voice3
      JP R4 @TG4_LOW
      END
20:
      LD R2, 21
      LD R4, 20
      LD STOP, 0xff              ; set STPA, STPB, STPC, STPD, STPE = 1
      H4+a+pretty+woman+T4       ; Voice4
      END
21:
      LD R2, 22
      LD R4, 21
      LD STOP, 0xff              ; set STPA, STPB, STPC, STPD, STPE = 1
      H4+an+ugly+girl+T4         ; Voice5
      END
22:
      LD R2, 20
```

```
    LD R4, 22
    LD STOP, 0xff              ; set STPA, STPB, STPC, STPD, STPE = 1
    H4+a+fat+lady+T4           ; Voice6
    END
```

Application Circuit



Revision History

Version	Date	Writer	Reasons for change
E	April 22nd, 1999	Sophia Ho	• Modify CPU mode and add an example of 4bit control program