W529XX Application Note

Winbond Electronics Corp.

ADPCM/PCM VOICE SYNTHESIZER
(*PowerSpeech*ä *II*)

GENERAL DESCRIPTION

Fabricated using Winbond's advanced CMOS process, the W529XX *PowerSpeech*ä *II* series of voice synthesizers now offers more power than ever. The *PowerSpeech*ä *II* series includes the following range of ICs with different built-in ROM memory capacities:

BODY	W52902	W52904	W52905	W52906	W52910	W52915	W52920
*Second	6 sec.	12 sec.	20 sec.	30 sec.	40 sec.	60 sec.	80 sec.

*Note: All playback lengths are estimated for typical applications.

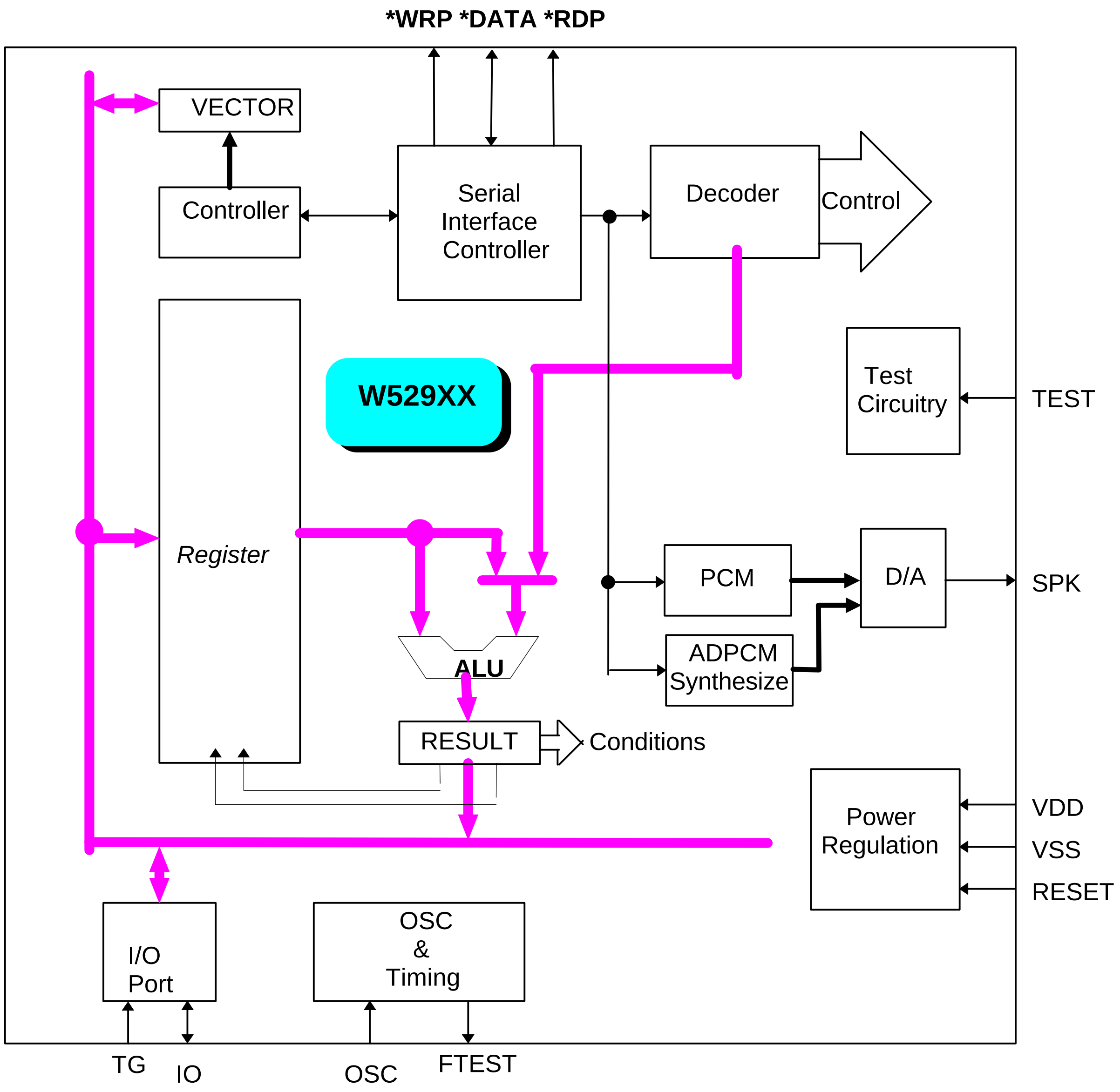
FEATURES

- Programmable speech synthesizer
- Wide operating voltage range: 2.4 - 5.5 volts
- Both 4-bit ADPCM and 8-bit PCM synthesis technique can be used
- Four trigger pins with separate control of falling/rising edge
- Two trigger input debounce times can be set: Long/ Short
- Eight multiplexed pins can be set as SCAN, LED(or FTEST), STOP, and INPUT
- Supports ALU operations, including
  – Branch decisions
  – Logic operations
  – Binary addition and subtraction
  – Data move
  – Bit operand
- Eight general purpose registers: R0 ~ R7
- Four special registers: EN, MODEn (n:1,2), OUTPUT, and ACC
- A maximum of 32 matrix keys can be defined by H/W or S/W
- Random number generation (H/W)
- Section control for each GO instruction
  – Variable frequency: 4.8/6/8/12 KHz
  – LED: ON/OFF
- Three LED flash types: 3Hz/ Circular/ Random

Version C
Release Date:May. 1998

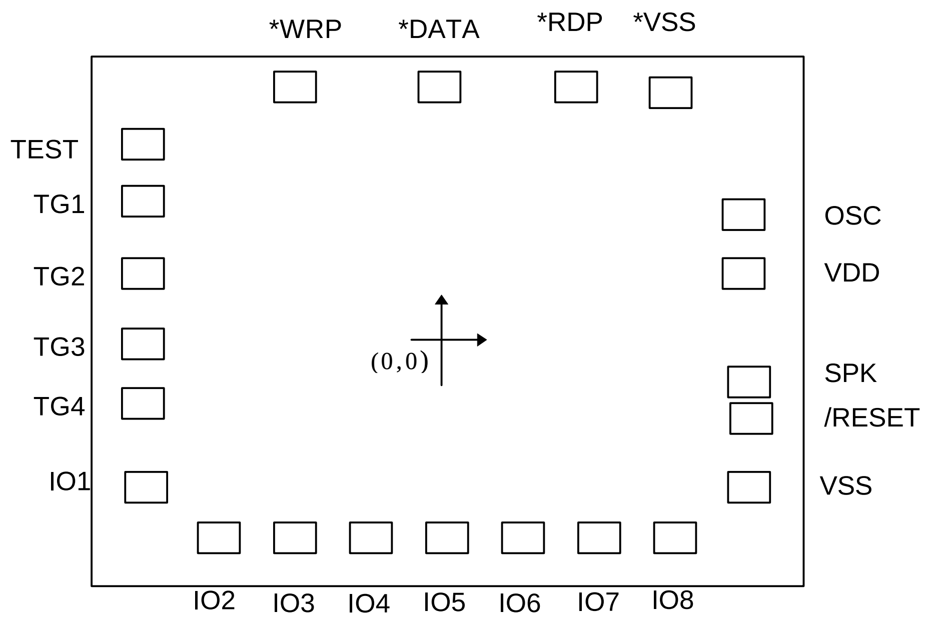
- System clock: 1.5 MHz
- Instruction cycle time: 200uS
- A total of 256 label entries available for programming

3. BLOCK DIAGRAM



*: These pins exist only on the W52900 demo chip.

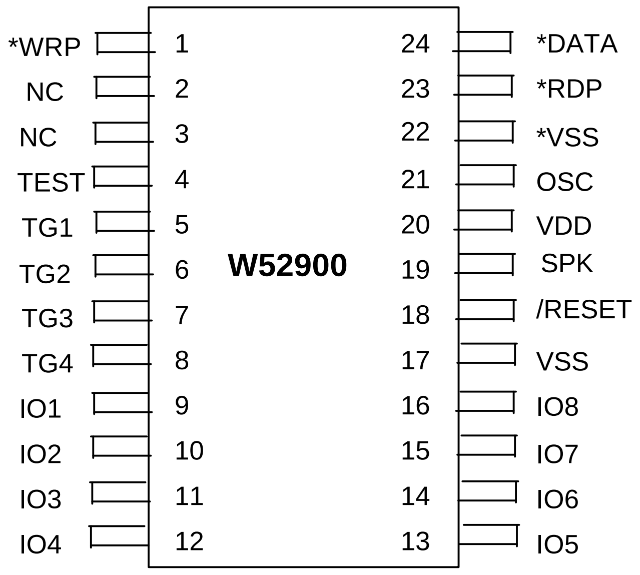
PAD DIAGRAM



*: These pins exist only on the W52900 demo chip.

PIN CONFIGURATION

The 24-pin P-DIP skinny package is used for evaluation purposes



*: These pins exist only on the W52900 demo chip.

PIN DESCRIPTION

No.	Name	I/O	Description
1	*WRP	O	Write clock output for serial interface
2	NC	-	Not connected
3	NC	-	Not connected
4	TEST	I	Test pin, internally pulled low
5	TG1	I	Direct trigger input 1, internally pulled high
6	TG2	I	Direct trigger input 2, internally pulled high
7	TG3	I	Direct trigger input 3, internally pulled high
8	TG4	I	Direct trigger input 4, internally pulled high
9	IO1	I/O	SCAN/LED/STOP/INPUT multiplexed pin 1
10	IO2	I/O	SCAN/LED/STOP/INPUT multiplexed pin 2
11	IO3	I/O	SCAN/LED/STOP/INPUT multiplexed pin 3
12	IO4	I/O	SCAN/LED/STOP/INPUT multiplexed pin 4
13	IO5	I/O	SCAN/LED/STOP/INPUT multiplexed pin 5
14	IO6	I/O	SCAN/LED/STOP/INPUT multiplexed pin 6
15	IO7	I/O	SCAN/LED/STOP/INPUT multiplexed pin 7
16	IO8	I/O	FTEST/LED/STOP/INPUT multiplexed pin 8
17	VSS	-	Negative power supply
18	/RESET	I	Reset all; functions as POR, internally pulled high
19	SPK	O	Current type output for speaker
20	VDD	-	Positive power supply
21	OSC	I	Oscillator input, connect with ROSC to VDD
22	*VSS	-	Negative power supply
23	*RDP	O	Read clock output for serial interface
24	*DATA	I/O	Bi-directional data for serial interface

*: These pins exist only on the W52900 demo chip.

1. TG1 - TG4

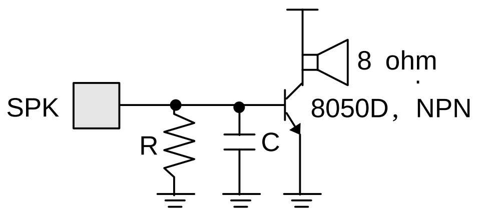
TG1 through TG4 are direct trigger input pins with internal pull high resistance of around 1 MΩ. After a 45 mS or 350 µS debounce time (determined by the MODE1 register chosen long or short debounce time), these direct trigger inputs execute their corresponding voice groups. These groups are 0/2/4/6 for falling edge triggers and groups 1/3/5/7 for rising edge triggers TG1 - TG4, respectively. Priority is set by internal H/W as TG1F > TG1R > TG2F > TG2R > TG3F > TG3R > TG4F > TG4R (F: Falling edge, R: Rising edge).

For keypad matrix applications, these four direct trigger inputs can be used together with VSS and pins IO1~IO7 pins to obtain a 32 key matrix.

2. SPK

The SPK pin is a current-type voice output, connected to the internal D/A converter output. The full scale output of the D/A converter is 5 mA. This output when amplified with a low-power NPN transistor with a β of around 120 - 160 is able to drive an external 8-Ω speaker.

The SPK output application circuit is shown as follows,



The shunt resistor Rs in the figure above is used to adjust the input base current of the NPN transistor in order to drive the external speaker without distortion. Distortion results from two factors: one is due to saturation of the transistor due to a large base current, and the other is the through the introduction of Rs which can cut small signals out of the original waveform. The value of the shunt resistor Rs, can be adjusted to get the desired voice. A value of around 310Ω - 750Ω is recommended. Note that the smaller the resistance, the smaller the input current to the transistor.

Capacitor C in the figure above is used for low-pass filtering of unwanted high-frequency noise generated from the D/A converter during sample transitions. Users may adjust the value of C to improve sound quality. Note that this capacitor can also be omitted without having a major affect on the voice quality.

3. IO1 - IO8

The eight I/O pins can have four functions: SCAN, LED, STOP, and INPUT. All these functions are defined by the MODE1 and MODE2 registers according to the following I/O function table.

Group	IO8	IO7	IO6	IO5	IO4	IO3	IO2	IO1
0	INPUT/STOP				INPUT/LED			
1	FTEST/LED	SCAN/LED			SCAN/LED			
2	INPUT/LED				SCAN/STOP			
3	LED/STOP				LED/STOP			

The MODE1 register selects the group (0~3), and the MODE2 register defines the function of each selected I/O pin. A "0" for one of these bits selects the first pair of modes indicated; a "1" selects the second pair.

To select a SCAN output, the selected output signal is initially set to "0" for keypad scan purposes. After a key is depressed and debounced successfully, the SCAN outputs return to the "1" state. A low pulse will then start to ripple according to a pre-defined H/W pattern to identify which key is being pressed. For more information, see the FUNCTIONAL DESCRIPTION section.

For an LED output, the selected pin flashes with a frequency of 3 Hz or flashes with circular/random effects. See the FUNCTIONAL DESCRIPTION section for more details.

For a STOP output, the user may define different STOP outputs at certain timing slots by programming the *PowerSpeech*ä synthesizer *.out files. After compilation, the STOP signals will be controlled by the internal OUTPUT register. The OUTPUT register can be manipulated through register operations. The STOP signals are typically used to drive external components.

For INPUT purposes, the status of the selected input pins are latched into the designated register by masking unselected pins to zeros. In this way, customers may then decide what to do depending on the port status of the input pins. Actions are invoked through bit manipulation commands like the JP N@reg.m=1 instruction (i.e. This instruction says that… if the mth bit of the register is equal to 1, then jump to voice group N to execute program.).

Those IO pins that are declared as INPUT pins could be connected to "1", "0", or just left floating. The floating INPUT pins will be recognized as "0" by the W529XX. The W529XX latches the input status only at the execution time of the command "MV erg, IPORT" and tri-states the input port elsewhere.

Pin IO8 can also be selected as a frequency test pin (FTEST), provided for some customers to facilitate the back-end sample rate test at the factory site. The nominal output frequency, FTEST, is fixed at 6 KHz on the condition that the oscillation frequency is equal to 1.5 MHz.

4. OSC

The OSC pin is connected with an ROSC resistor to VDD.

5. TEST

The TEST pin is used solely for test purposes.

6. /RESET

Active low reset input with an internal pull-high resistance of 500 KΩ. A falling edge on the /RESET pin will fully reset the W529XX in the same way as a power on reset (POR). The /RESET pin was originally used as a last resort to reset the W529XX in situations where the internal POR circuit of the W529XX does not operate properly. If customers fail to discharge the VDD to ground and repower up the W529XX, the chip may function abnormally, causing unpredictable operations. Users may then reset the W529XX by sending a pulse through the /RESET pin.

7. VDD

Positive power supply.

8. VSS

Negative power supply.

9. WRP/RDP/DATA (Provided only on W52900)

These three pins provide a serial interface to external memory devices, like serial SRAM and Flash EPROM. For customer evaluation purposes, the ROM file can be put into any of the serial memory devices mentioned above to get the effect of "what you see is what you get", without the need of mask tooling procedures. Of course, customers may request volume production after a successful evaluation based on Winbond's coding environment. The **WRP** pin is used to send out addresses to the serial memory devices for accessing data. The **RDP** pin is used to read back this data from the addressed memory space. The **DATA** pin is bidirectional: an input pin during read operation, and an output pin during address writing.

FUNCTIONAL DESCRIPTION

The W529XX family is a derivative of Winbond's *PowerSpeech*ä synthesizers, devices which are prevailing in the consumer market, especially for toy applications. The W529XX family adopts the same architecture of the *PowerSpeech*ä synthesizers, while enhancing its power by expanding the command set, providing a keypad matrix of up to 32 keys, PCM/ADPCM synthesis selection, etc.

The command set of the W529XX family is expanded from simply Jump (JP) and Load (LD) commands into four kinds of commands: ALU, Data Move, Branch, and others. In addition, the general-purpose registers are expanded from just one, R0, to eight, R0 - R7. The addition of these extra commands enables users to produce more efficienct code.

To allow for the needs of more flexible applications, IO1 - IO8 can be selected to have a SCAN, LED, STOP or INPUT function. The IO1 - IO8 pins have to be first configured by the customer for their particular applications by programming the MODE1 and MODE2 registers before operating.

In order to smoothly connect different voice reproductions synthesized by the ADPCM algorithm, a new mechanism is used to generate the ramp up (i.e., the head, H4) and the ramp down (i.e., the tail, T4) automatically by H/W to eliminate the needs of H4 and T4 in previous coding for the W528x series. Users do not have to consider H4/T4 in the programming of the *.out file for the W529XX.

Owing to the increased complexity of the *.out file, the need for a label-based symbolic compiler for the W529XX is increasing. This sort of symbolic compiler supports label addressing instead of group numbering, constant definition for special usages, etc.

The W529XX enters the POI (Power On Initialization) process when powering up. The voice group 64 is allocated for this special event and its priority is above all, i.e., no triggers can override the POI process if they all happen simultaneously. If more than two events happen simultaneously, the priority that is set by the internal H/W is: POI > TG1F > TG1R > TG2F > TG2R > TG3F > TG3R > TG4F > TG4R > JP commands.

1. POWER ON INITIALIZATION

The W529XX will automatically execute the POI (Power On Initialization) process upon power up or depression of the RESET pin. The voice group dedicated for POI is allocated at "64".

Upon power up, the W529XX resets itself to initial values by internally generating a POR (Power On Reset) pulse to start the POI process.

During the POI process, certain triggers can override to take over if the EN register is set to properly enable those triggers. Take the following example (for reference only); after the MODE1 & MODE2 registers are set to their appropriate values according to the customers' configuration, the EN register is set to enable the falling-edge triggers of TG1 and TG2. Therefore, during the reproduction of voice segments, like Voice1, Voice2, Voice3, and Voice4, triggers TG1F and TG2F may terminate the normal operation of the POI process and start a new trigger later on.

```
W52906
        FREQ2
        LED1
POI:
        LD MODE1, 0x03
        LD MODE2, 0xF2
        LD EN, 0x03         ;Enable TG1 and TG2 falling edge
        LD OUTPUT, 0xC5
        Voice1 + 2*Voice2 + 8*Voice3
        JP EXIT @TG3_L
        Voice4
EXIT:
        END
```

See also "Register definition and control" for default settings during a power up.

2. GROUP NUMBER ALLOCATION

For direct trigger inputs, there are 8 voice groups allocated to four TG pins: voice groups 0, 2, 4, and 6 are for falling edge triggers, and voice groups 1, 3, 5, and 7 are used for the rising edge triggers.

Voice Group	Description
0	TG1F
1	TG1R
2	TG2F
3	TG2R
4	TG3F
5	TG3R
6	TG4F
7	TG4R
2N, 0≤N≤31	Keypad matrix for falling edges
2N+1, 4≤N≤31	User defined groups
64	POI
65 - 255	User defined groups

For a keypad scan matrix, the voice groups are listed in the following table for quick reference. Each key is composed of the connection of a trigger (TG1 - TG4) and a ground source (VSS, IO1 - IO7). Upon depression of a key, the trigger input senses a low pulse during a H/W scan period to determine which key is pressed. In order to avoid possible ambiguities during multiple key depressions, only falling edge triggers are allocated for keypad applications. Rising edge triggers are only available for direct trigger pins, i.e. TG1 - TG4.

Keypad	TG1	TG2	TG3	TG4
VSS	0 (1)	2 (3)	4 (5)	6 (7)
IO1	8 (NA)	10 (NA)	12 (NA)	14 (NA)
IO2	16 (NA)	18 (NA)	20 (NA)	22 (NA)
IO3	24 (NA)	26 (NA)	28 (NA)	30 (NA)
IO4	32 (NA)	34 (NA)	36 (NA)	38 (NA)
IO5	40 (NA)	42 (NA)	44 (NA)	46 (NA)
IO6	48 (NA)	50 (NA)	52 (NA)	54 (NA)
IO7	56 (NA)	58 (NA)	60 (NA)	62 (NA)

Note: The number in parenthesis (n) stands for the voice group of the rising edges, which are Not Available (NA) except for direct triggers to avoid possible ambiguities in keypad scan applications by H/W.

3. GLOBAL REPEAT

The W529XX family provides a maximum four times global repeat that can give the user more programming convenience for repetitious playback effects. An example (for reference only) is shown below,

```
Label:  4
        LD MODE1, 0x07
        LD EN, 0x34
        voice1 + voice2
        END
```

The above example demonstrates the usage of global repeat. The voice group that is indicated by "Label" will be executed for 4 times.

4. SECTION CONTROL

Variable Frequency Section Control

The W529XX provides four sampling frequencies, 4.8 KHz, 6 KHz, 8 KHz, and 12 KHz, which can be set by the section control function in the speech equations. These sampling frequencies are suitable for both ADPCM and PCM synthesis.

Sample Rate (KHz)	FREQ
4.8	0
6	1
8	2
12	3

LED ON/OFF Section Control

The LED section control allows the user set the LED output state, On or Off as shown in the following table, after the LED mode (3-Hz, circular, or random) has been defined.

ON/OFF	LED
OFF	0
ON	1

5. KEYPAD MATRIX SCAN

In order to minimize programming effort to detect keys in keypad matrix applications, the W529XX provides customers with another alternative for a H/W automatic scan, other than the previous (W528X) S/W programming approach. Customers may select the H/W approach by configuring the IO pins to be scan outputs.

For the H/W automatic scan mechanism, the internal H/W starts to scan the defined keypad matrix with the priority of VSS > IO1 > IO2 > IO3 > IO4 > IO5 > IO6 > IO7 after any TG pin is debounced successfully. During the H/W scan, the W529XX ripples a low pulse through the scan outputs IO1 - IO7, causing the depressed key to be detected within a period of no longer than 1 mS. Owing to the priority setting in the H/W scan mechanism, the time used to detect a depressed key may vary somewhat depending on which column it belongs to. To avoid possible ambiguities, all TG pins used for keypad scan purposes should be disabled for their corresponding rising edges by software programming. The trigger debounce time must be set to a long time (around 45mS).

For a S/W controlled keypad scan, customers may program their own scan patterns for specific purposes. In order for the W529XX to correctly detect a depressed key, the *.out file should be programmed based on the JP N @TG_status commands with the IO pins being configured as STOP outputs.

A typical example for a 12-key keypad is as follows: (for reference only)

```
W52906
LED1
FREQ1                   ; Sample rate is 6KHz

POI:
        LD MODE1, 0x03  ; 45mS debounce time must be selected for keypad scan
        LD MODE2, 0xFF
        LD EN, 0x0F
        LD OUTPUT, 0x00 ; Reset OUTPUT to arm keys for triggering.
        END

TG1F:                   ; For TG1F edge
        LD OUTPUT, 0xFF
        JP key1 @TG1_L
        LD OUTPUT, 0xFE
        JP key5 @TG1_L
        LD OUTPUT, 0xFD
        JP key9 @TG1_L
        END

TG2F:                   ;For TG2F edge
        LD OUTPUT, 0xFF
        JP key2 @TG2_L
        LD OUTPUT, 0xFE
        JP key6 @TG2_L
        LD OUTPUT, 0xFD
        JP key10 @TG2_L
        END

TG3F:                   ; For TG3F edge
        LD OUTPUT, 0xFF
        JP key3 @TG3_L
        LD OUTPUT, 0xFE
        JP key7 @TG3_L
        LD OUTPUT, 0xFD
        JP key11 @TG3_L
        END

TG4F:                               ; For TG4F edge
        LD OUTPUT, 0xFF
        JP key4 @TG4_L
        LD OUTPUT, 0xFE
        JP key8 @TG4_L
        LD OUTPUT, 0xFD
        JP key12 @TG4_L
        END

key1:
        ...
key2:
        ...
        ...
key12:
        ...
        END
```

For saving the extra bits needed for assigning multiplexing selections for the IO1 - IO8 pins, the W529XX imposes some restrictions on the possible combinations: two bits in the MODE1 register are first decoded into four possible groups, then the 8-bit MODE2 register is used to further decode the IO1 - IO8 for their corresponding selections.

See also **IO1 - IO8** of Pin Description for further details.

6. REGISTER DEFINITION AND CONTROL

The register file of the W529XX is composed of 13 registers, designed for various operations. Each register can be arithmetically operated on with immediate data or with any register, be bit-manipulated by logical operations, like AND, OR and XOR or can be modified by LD or MV commands.

All the 13 registers share the common 8-bit wide bus for data exchange and command execution control. The related register information is shown in the following table.

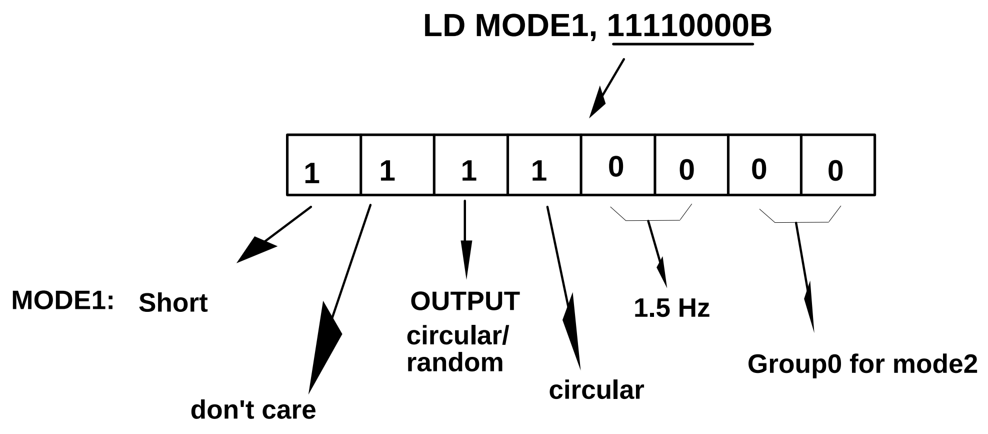
Name	Purpose	Bit definition B7, B6, B5, B4, B3, B2, B1, B0	Initial value upon power up
EN	To hold enable/disable control bits for TG1 - TG4	TG4R, TG3R, TG2R, TG1R, TG4F, TG3F, TG2F, TG1F	1111 1111
R0 - R7	General purpose registers	-	0100 0000
MODE1	To hold various mode controlled bits for different configurations	Debounce time, reserved, LED source, LED type, LED clock 1, LED clock 0, IO group 1, IO group 0	0000 0000
MODE2	To hold various mode controlled bits for different configurations	IO8, IO7, IO6, IO5, IO4, IO3, IO2, IO1	0000 0000
ACC	Accumulator	-	0000 0000
OUTPUT	To hold output values that are reflected at IO1 - IO8 pads, or initial values for LED effects	-	0000 0000

MODE1

Bit	Name	0	1
0	IO group selection 0	*00: group 0	*01: group 1
1	IO group selection 1	*10: group 2	*11: group 3
2	LED clock 0	**00: 1.5 Hz	**01: 3 Hz
3	LED clock 1	**10: 6 Hz	**11: 12 Hz
4	LED type	synchronous or circular	alternate or random
5	LED source	3 Hz	OUTPUT
6	Reserved	-	-
7	Debounce time	Long (arround 45 mS)	Short (arround 350 µS)

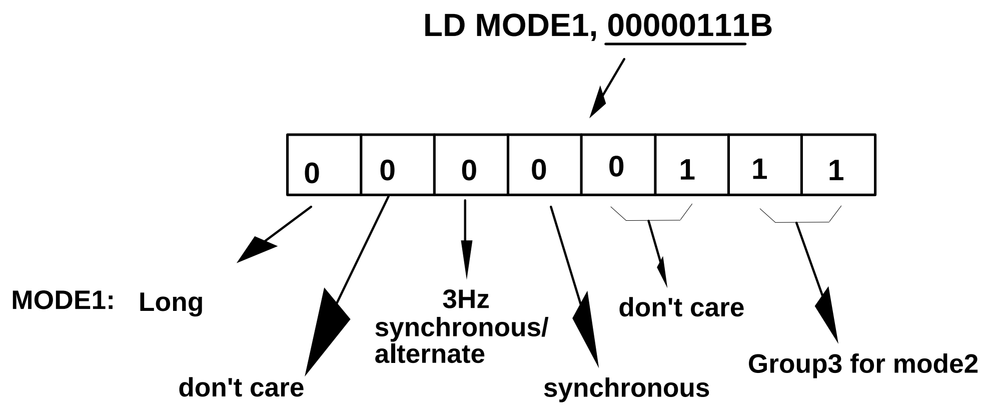
Note: " * " bit1 and bit0. The first bit defined as bit1 and the second bit defined as bit0.
" ** " bit3 and bit2. The first bit defined as bit3 and the second bit defined as bit2.

Example1:



a. The debounce time of the trigger inputs is set as a short time. (around 350 µS)
b. The operation of the LEDs is circular with a 1.5Hz frequency; from bit2 to bit5 definitons.
c. Bit1 and bit0 define the group0 for IO1-IO8 pins option.

Example2:



a. The debounce time of the trigger inputs is set as a long time. (around 45 mS)
b. The operation of the LEDs is synchronous with a frequency of 3Hz; from bit4 and bit5 definitons.
c. Bit1 and bit0 define the group3 for IO1-IO8 pins option.

The group selection for the IO pins are defined by bit 1 and bit 0 altogether. See **IO1 - IO8** of Pin Description for more details.

MODE2

The MODE2 register is dedicated for IO configuration definition after group selection, which is defined in the MODE1 register. There are two options that can be selected for each bit of the MODE2 register. A "0" selects the former option, while a "1" picks the latter option. See also Pin Description for further details.

EN

A "1" means "enabled", while a "0" means "disabled" for that edge of the particular TG pin. For example, the command "LD EN, 0x0F" enables all the falling edge triggers of TG1 - TG4, while disabling the rising edge triggers of TG1 - TG4. Users may modify the EN register during operation of the W529XX to achieve various kinds of trigger functions, like retriggerable or not, one shot or level hold play modes, etc.

In other words the user may change the contents of the EN register during synthesis to determine which TG is to be enabled/disabled and triggered on its falling/rising edge.

ACC

This is the register that is called the accumulator. For **ALU** operations (e.g., ADD, AND, OR, and XOR), the ACC is modified implicitly with the same result of the destination register. For example, assume R6 equals 5(Dec), the command "ADD R6, 12" when executed gives R6 and **ACC** the same result of 17(Dec) at the end of command processing.

OUTPUT

The OUTPUT register is used to control the status of the IO1-IO8 pins. These can be configured to be STOP outputs by setting the proper bits in the MODE1/MODE2 registers. For example, if the IO2 pin is selected as a STOP output, the corresponding bit, that is bit 1 of the OUTPUT register is set to drive the output buffer, an inverted stage, to show its logic status.

This register is also used as a shift register for LED circular/random effects. See also LED Operation for more details.

R0 - R7

These eight registers function as general purpose registers. They can be used to hold group numbers, values for number counting, event flags, etc.

IPORT

The IPORT is not a physical register in the W529XX, but rather a register-mapped input port that shares the 8-bit bus with other registers to facilitate reading of the port status. Pins IO1 - IO8 can be selected to act as an input port, provided the proper settings are made to the MODE1 and MODE2 registers. Unselected bits of the IPORT register are masked as zeros. Only one command " MV Rn, IPORT" (n: 0 - 7) can be used in programming.

7. PORT READING

At most, 8 bits can be acquired on the data bus if the IO pins are all configured as INPUTs. The command "MV reg, IPORT" will put the status of the port into the register "reg" for further processing. The user has to first configure the MODE1 and MODE2 registers for the required IO selections before the proper port status can be acquired. The following example (for reference only) shows how the port status can be manipulated.

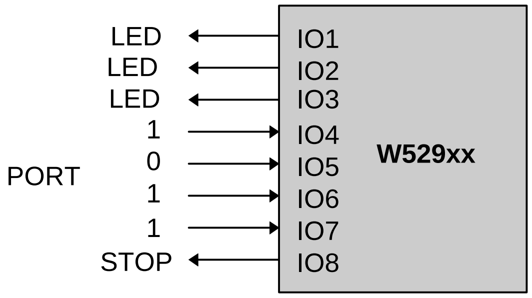
W52905

LED1

```
            FREQ1

    POI:
            LD MODE1, 0x00          ; IO group 0 selected
            LD MODE2, 0x87          ; IO1 - IO3 -> LED, IO4 - IO7 -> INPUT, IO8 -> STOP
            END

    port:
            MV R2, IPORT            ; To move port status into register R2
            JP case1 @R2.3=1        ; If bit3 of R2 register is equal to 1, jump to case1 voice group.
            JP case2 @R2.5=1        ; If bit5 of R2 register is equal to 1, jump to case2 voice group.
            END
```



Only those bits that are declared as INPUT pins can be acquired by the "MV reg, IPORT" command. Others bits, which are not INPUT pins, are masked by zeros. For example, the above program selects IO group 0 and chooses IO4 - IO7 as the INPUT pins. Assuming that the status of 4 external bits is "1101", after completion of the command "MV R2, IPORT", R2 holds the result of 0***1101***000(B).

8. SYMBOLIC COMPILER

In order to reduce the workload of the *.out programming for the W529XX, a symbolic compiler has been designed. This accepts not only the commonly used group numbers, but also the symbolic labels that are adopted in most commercial µP compilers.

9. DIRECT TRIGGER

The W529XX provides a total of 4 trigger inputs to interface directly with the outside world. After debouncing, the W529XX responds to each edge of a valid trigger with a voice group, provided that the associated enable flag of that particular edge has been previously set. The triggered input then starts to execute the voice group by overwriting current operations, if any.

10. D/A CONVERTER

When a voice GO instruction is executed, the D/A converter is activated until an "END" or "PSAVE" instruction is encountered. The W529XX provides a unique instruction "PSAVE" to turn off the D/A converter for power saving purposes. The role that the "PSAVE" plays in the W529XX will be described in more detail in the following section.

11. RAMP UP/DOWN

In the past Winbond's speech synthesizers will encode a "head" and "tail" during code preparations to avoid the pop sounds that may occur at the start or end of a voice. The following example is a typical *.out file for previous *PowerSpeech*ä products:

```
W5282
LED1
FREQ2
LED1_S_CTRL

0: 2
        H4 + voice1 + 2 * voice2 + T4
        END

1:
        H4 + Voice3 + T4
        END
```

In the W529XX, a new H/W mechanism automatically inserts the head (H4) and tail (T4), to eliminate the pop sounds. The D/A converter is disabled and the PCM value and quantization pointer of the ADPCM stage are reset to 0. This initial status will not be changed until a voice GO instruction is executed. Both the head (H4) and the tail (T4) periods are 10.67mS.

A special point concerning power consumption and use of the "PSAVE" instruction should be mentioned. As the "PSAVE" instruction turns off the D/A converter, it can be used to disable the output of the D/A converter when running the "Command" instructions. With the new mechanism for ramp up/down, it is now possible to control the playing voice with more flexibility than before. For example, if we want two voices to be played more continuously between their connection point, we only need to omit the "PSAVE" instruction between them. Doing this will ensure no insertion of "head" and "tail", which may take about 21.34 ms between these two voices. If voice continuity is more important than power consumption, then it is possible that no "PSAVE" instructions are written into the program.

12. ADPCM/PCM SYNTHESIS

The W529XX offers an alternative to the ADPCM algorithm for voice reproduction, direct 8-bit PCM coding. This PCM direct coding may be used for better reproductions especially for applications using high quality music sources.

The following voice combinations show the mixing of ADPCM and PCM synthesis.

```
W52905
LED1
FREQ2
```

```
Synthesis: 3
                Voice1 + Voice2 + Voice3
                END
```

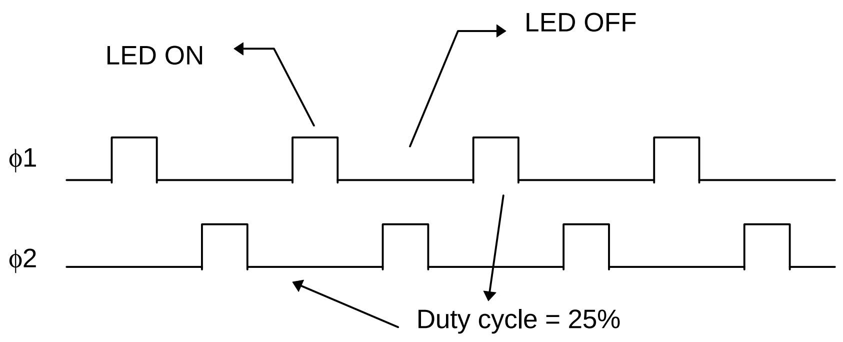
Voice1 and Voice3 are ADPCM converted files with *.WAM extensions, while Voice2 is a direct PCM format with an *.SRC extension. The source (*.SRC) files, which can be acquired from Winbond's sound card or transformed from *.WAV files, have to be converted into ADPCM format before compilation can proceed. During compilation, the W529XX compiler will first check for the existence of Voice1.WAM, before checking if Voice1.SRC exists. In either case, the compiler can proceed normally with ADPCM/PCM synthesis. The same check procedures continue for the Voice2 and Voice3 files. The compiler will show an error message if both the *.WAM and *.SRC files do not exist.

13. LED OPERATION

The LED pins, selected from IO1 - IO8 by configuring the MODE1/MODE2 registers, can be used to enhance the product's attractiveness. The LEDs for the W529XX may have the following possible selections: 3 Hz synchronous or alternate flashing, flashing circularly or randomly. The LEDs can also be section-controlled by each GO instruction, together with the frequency control feature.

FLASH WITH 3 Hz

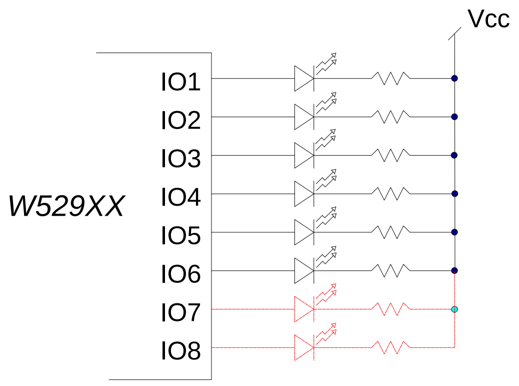
The 3 Hz clock is phase divided into two LED sources, which are 180° out of phase, in order to facilitate the synchronous/alternate features.



The ϕ1 clock is applied to the LED1, LED3, LED5, and LED7 output pins. For synchronous flashing of the LEDs, the ϕ1 clock is connected to LED2, LED4, LED6, and LED8. For alternate flashing, the ϕ2 clock is connected to LED2, LED4, LED6, and LED8. In other words, all the LED outputs flash synchronously with the ϕ1 clock or flash alternately in two groups, one is LED1/3/5/7 with ϕ1 and the other is LED2/4/6/8 with ϕ2.

CIRCULAR/RANDOM OPERATION

The W529XX provides special LED circular and random flashing functions.



The basic operating principles are described below:

a. The W529XX provides only 6 or 8 LEDs for circular/random operations. If the user uses 6 LEDs, then the two MSBs are left unchanged during operations. Under such conditions only LD commands can be used to modify bit7/bit6. During circular/random operations, the LD command receives a higher priority than the shifting clock. The user can therefore always modify the LED patterns without problems.

OUTPUT register	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
IO1 - IO8	***IO8**	***IO7**	**IO6**	**IO5**	**IO4**	**IO3**	**IO2**	**IO1**
**LD OUTPUT, 00111000b	0	0	1	1	1	0	0	0
LED on/off	*on*	*on*	*off*	*off*	*off*	*on*	*on*	*on*

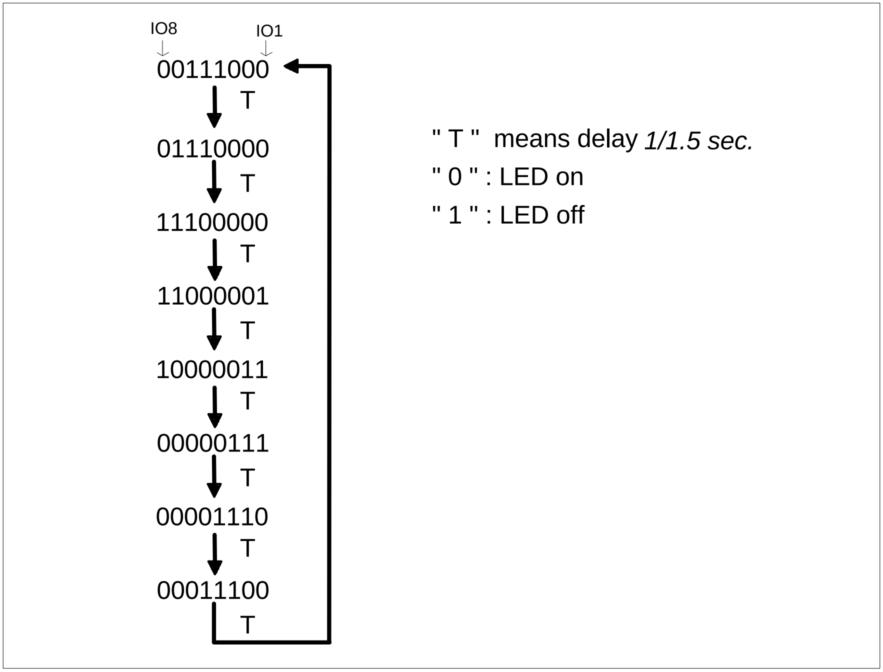
* If the 6 LEDs are defined for the circular/random function, the IO8 and IO7 will not shift during operation. The IO8 and IO7 will be fixed in low.

** "LD OUTPUT,00111000b " defines the initial value for circular or random

b. For circular/random LED operation to take effect, the MODE1 register has to be configured properly: MODE1.5 should be defined as an OUTPUT. Then the LED type, circular or random, should be selected by resetting or setting MODE1.4, respectively. The LED clock can be defined as 12/6/3/1.5 Hz by configuring (MODE1.3, MODE1.2) to be (1,1)/(1,0)/(0,1)/(0,0) respectively. When MODE1.5 is defined as an OUTPUT for the LED source, the LED clock starts to shift out the OUTPUT register as long as the chip is operating. To stop its shift operation, just re-program the value of MODE1.5 to 0 for the selection of 3-Hz source.

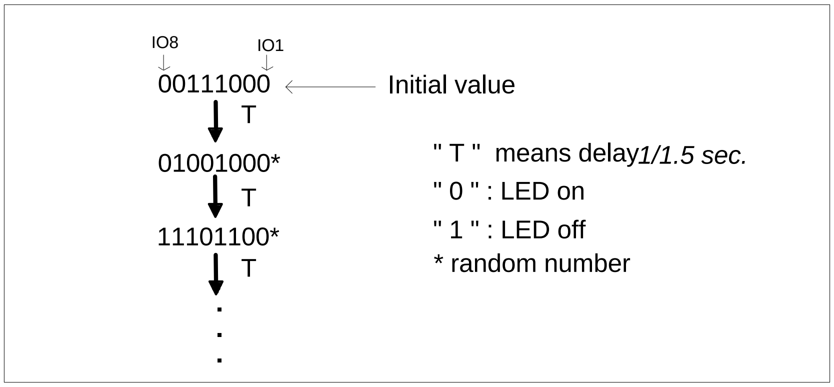
8 LEDs circular operation:

```
LD mode1, 00100011b  ; IO group3, LED source: output
                     ; LED type:circular, LED clock:   1.5Hz
LD mode2, 00000000b  ; IO1 - IO8 as LEDs
LD output, 00111000b ; for random initial value
```



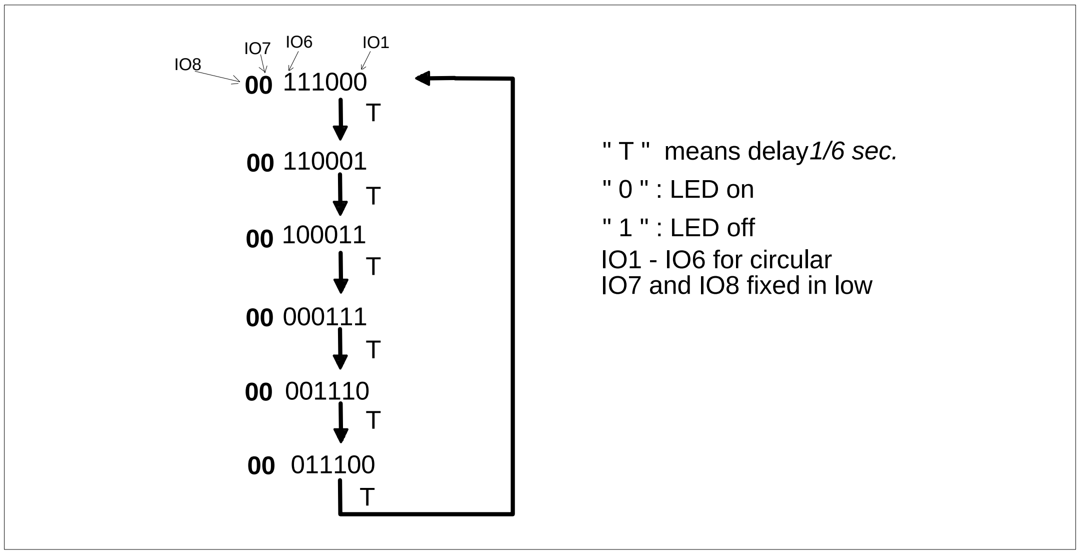
8 LEDs random operation:

```
LD mode1, 00110011b  ; IO group3, LED source: output
                     ; LED type:random, LED clock:    1.5 Hz
LD mode2, 00000000b  ; IO1 - IO8 as LEDs
LD output, 00111000b ; for random initial value
```



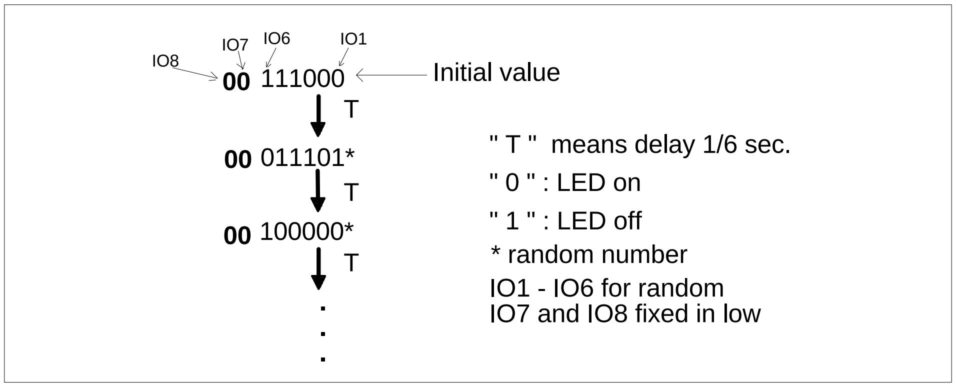
6 LEDs circular operation:

LD mode1, 00101011b	; IO group3, LED source: output ; LED type:circular, LED clock: *6 Hz*
LD mode2, 11000000b	; IO1 - IO6 as LEDs, IO7 and IO8 as STOPs
LD output, 00111000b	; for circular initial value



6 LEDs random operation:

LD mode1, 00111011b	; IO group3, LED source: output ; LED type:random, LED clock: *6 Hz*
LD mode2, 11000000b	; IO1 - IO6 as LEDs, IO7 and IO8 as STOPs
LD output, 00111000b	; for circular initial value



c. To achieve the circular/random operation, the W529XX has some constraints.

1> Those selections with a different number of LEDs, which are other than 6 or 8, may suffer from the drawbacks of discontinuity of LED flashes, disruption of STOP outputs, etc.

2> The H/W key matrix scan mode cannot be used as it will produce abnormal operation; i.e. the IO pins cannot be chosen as the "SCAN" for circular/random functions.

The user has to take into consideration these limitations.

The following example (for reference only) shows a program which displays powerful LED effects for the W529XX.

```
W52906
        LED1                    ; LED section control is ON as default.
        FREQ2                   ; SR = 8 KHz.
POI:
        LD MODE1, 00000011b             ; IO group 3, LED source: 3 Hz, LED clock: 1.5 Hz,
                                        ; LED type: synchronous.
        LD MODE2, 11000000b             ; 2 STOP outputs and 6 LEDs. IO8/7: STOP, IO6 -
                                        ; IO1: LED.
        END
Demo0:
        JP 3Hz_Syn                      ; To setup MODE1 register
Demo1:
        Voice1 + Voice2_20 + Voice3     ; LED is OFF during Voice2 playback
        JP 3Hz_Alt                      ; To setup MODE1 register
Demo2:
        Voice4 + Voice5
        JP Circular1                    ; To setup MODE1 & OUTPUT registers
Demo3:
        Voice6
        JP Circular2                    ; To setup MODE1 register
Demo4:
        Voice6
        JP Circular3                    ; To setup MODE1 register
Demo5:
        Voice6
        JP Circular4                    ; To setup MODE1 register
Demo6:
        Voice6
        JP Random                       ; To setup MODE1 & OUTPUT registers
Demo7:
        Voice7
        END
3Hz_Syn:
        CLRB MODE1.5                            ; LED source: 3 Hz
```

```
            CLRB MODE1.4                    ; LED type: synchronous
            JP Demo1
3Hz_Alt:
            CLRB MODE1.5                    ; LED source: 3 Hz
            SETB MODE1.4                    ; LED type: Alternate
            JP Demo2
Circular1:
            LD OUTPUT, 11000001b            ; Load 00_0001 for circular operation.
            SETB MODE1.5                    ; LED source: OUTPUT
            CLRB MODE1.4                    ; LED type: circular
            SETB MODE1.3                    ; LED clock: 12 Hz
            SETB MODE1.2
            JP Demo3
Circular2:
            SETB MODE1.5                    : LED source: OUTPUT
            CLRB MODE1.4                    : LED type: circular
            SETB MODE1.3                    ; LED clock: 6 Hz
            CLRB MODE1.2
            JP Demo4
Circular3:
            SETB MODE1.5                    ; LED source: OUTPUT
            CLRB MODE1.4                    ; LED type: circular
            CLRB MODE1.3                    ; LED clock: 3 Hz
            SETB MODE1.2
            JP Demo5
Circular4:
            SETB MODE1.5                    ; LED source: OUTPUT
            CLRB MODE1.4                    ; LED type: circular
            CLRB MODE1.3                    ; LED clock: 1.5 Hz
            CLRB MODE1.2
            JP Demo6
Random:
            LD OUTPUT, 11000010b            ; Load 000010 as the starting point for random
                                            ; operation.
            SETB MODE1.5                    ; LED source: OUTPUT
            SETB MODE1.4                    ; LED type: random
            SETB MODE1.3                    ; LED clock: 6 Hz
            CLRB MODE1.2
            JP Demo7
```

14. SAMPLE RATE

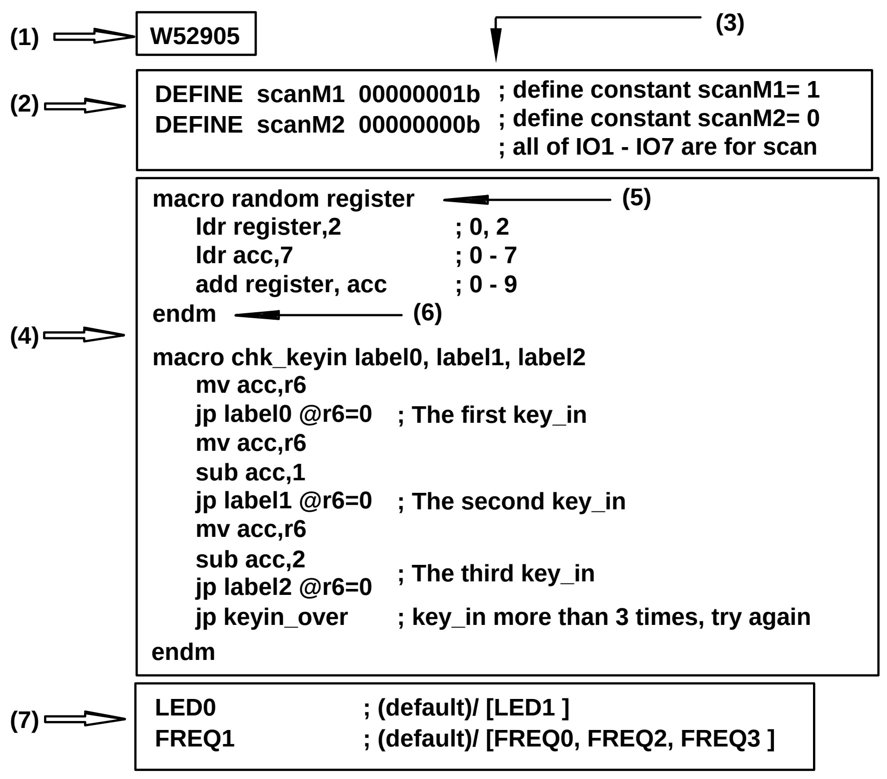
The sample rate for voice reproductions can be determined by two methods. One method adjusts the external resistor ROSC, and the other method changes the control bits for variable frequency effects. The former method adjusts the master frequency of the whole chip, while the latter method divides the master frequency by different divisors to get the different 4.8/6/8/12 KHz sample rate clocks for the typical condition of FOSC = 1.5 MHz.

Normally, the user has to reproduce the voices at the same sample rate as recordings in order to get the original voice outputs. The maximum sample rate allowed for the W529XX is 12 KHz, a limitation that is mainly due to the access time of the ROM production chips.

During the development phase of application code, the user may use Winbond's speech download board or OTP chip to evaluate the functions and voice quality. However it should be noted that the sample rate cannot be set above 12 KHz, which is the maximum limitation of the W529XX, even though in practice it can work normally. This is due to a fundamental difference of ROM structure between the evaluation tools and real chips.

15. W529XX PROGRAM FORMAT

The W529XX *PowerSpeech*ä *II* enables users to define the functions of their products using the W529XX *PowerSpeech*ä programming language. An example of the W529XX *PowerSpeech*ä program format is shown below. (Explanatory notes follow the example.)



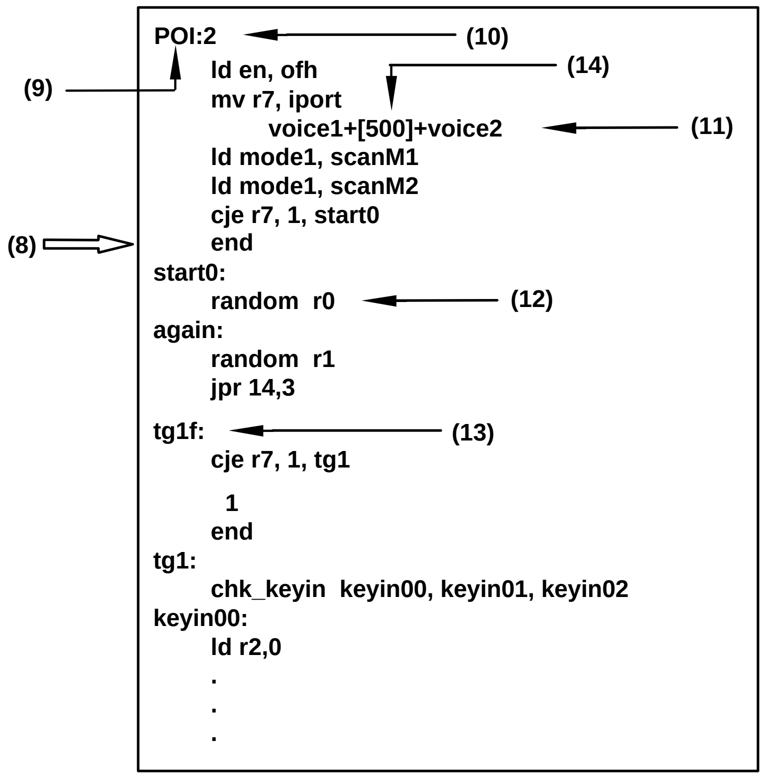
```
W52905

DEFINE  scanM1  00000001b  ; define constant scanM1= 1
DEFINE  scanM2  00000000b  ; define constant scanM2= 0
                           ; all of IO1 - IO7 are for scan

macro random register
    ldr register,2          ; 0, 2
    ldr acc,7               ; 0 - 7
    add register, acc       ; 0 - 9
endm

macro chk_keyin label0, label1, label2
    mv acc,r6
    jp label0 @r6=0     ; The first key_in
    mv acc,r6
    sub acc,1
    jp label1 @r6=0     ; The second key_in
    mv acc,r6
    sub acc,2
    jp label2 @r6=0     ; The third key_in
    jp keyin_over       ; key_in more than 3 times, try again
endm

LED0               ; (default)/ [LED1 ]
FREQ1              ; (default)/ [FREQ0, FREQ2, FREQ3 ]
```



```
POI:2
        ld en, ofh
        mv r7, iport
                voice1+[500]+voice2
        ld mode1, scanM1
        ld mode1, scanM2
        cje r7, 1, start0
        end
start0:
        random  r0
again:
        random  r1
        jpr 14,3

tg1f:
        cje r7, 1, tg1

          1
        end
tg1:
        chk_keyin  keyin00, keyin01, keyin02
keyin00:
        ld r2,0
        .
        .
        .
```

(1) Bodies: The user must first define the W529XX body to be used, or else an error message will appear during compiling. The W529XX bodies include the following: W52902, W52903, W52904, W52905, W52906, W52910, W52915 and W52920

(2) Define: The user can define constants in order to ease programming. Both upper and lower case are considered the same by the compiler, in other words "A" and "a" are the same after compilation. Hexadecimal (0x00 or 0ffh), binary (11010101b) or decimal (0, 100..) can be used as parameter types. So the following examples are the same:

```
define  cycle 16
DEFINE Cycle 0fh
Define Cycle 0x0f
Define CYCLE 00001111b
```

The format of the "define" statement is shown below:

first item	second item	third item
Define	constant	parameter

Space must be inserted between these three items in order to divide them.

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

(4) Macro: If the program possesses many repetitive sections, a "Macro" may be used to provide a more efficient structure. In a macro program, the "label:" is illegal because it isn't a subroutine. After compilation the macro will be extended and inserted into the main program. So although a macro can offer more convenience in program editing it cannot reduce ROM size. It is also possible to call a macro within another macro program. Eight levels of depth can be used in programming. Note that all of the unknown parameters must be transferred from the main program. The "ENDM" command will end the macro program.

Illegal writing:

```
macro chk_keyin label0, label1, label2
    mv acc,r6
    jp label0 @r6=0
    mv acc,r6
    sub acc,1
~~label0~~:
    :
    :
endm
```

macro name: chk_keyin
transmitted paramaters: label0, label1, label2

(5) Macro begin line: This includes the keyword "Macro" to declare "macro start", a user given name for the macro and the transmitted parameters. The format is as follows:

first item	second item	next
Macro	Name	parameters

The individual transmitted parameters have a " , " between them.

(6) Macro end line: "endm" must be added at the end of the macro.

(7) Declarations: The output frequency and LED on/off state.

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

Output frequency:
--FREQ0: 4.8 KHz
--FREQ1: 6 KHz (default)
--FREQ2: 8 KHz
--FREQ3: 12 KHz

(8) program body: Write application program and speech operations, including the following:

Define entry point of speech group.
Determine number of global repeats.
Describe speech equations.

Define the register values and command sets.

Call macro ... etc.

The maximum program memory size of the W529XX is 256K bits. Each GO instruction occupies 48 bits, and each group entry (0 - 255) occupies 16 bits. A total of around 5,400 GO instructions can therefore be used in the program. The GO instruction includes the following contents:

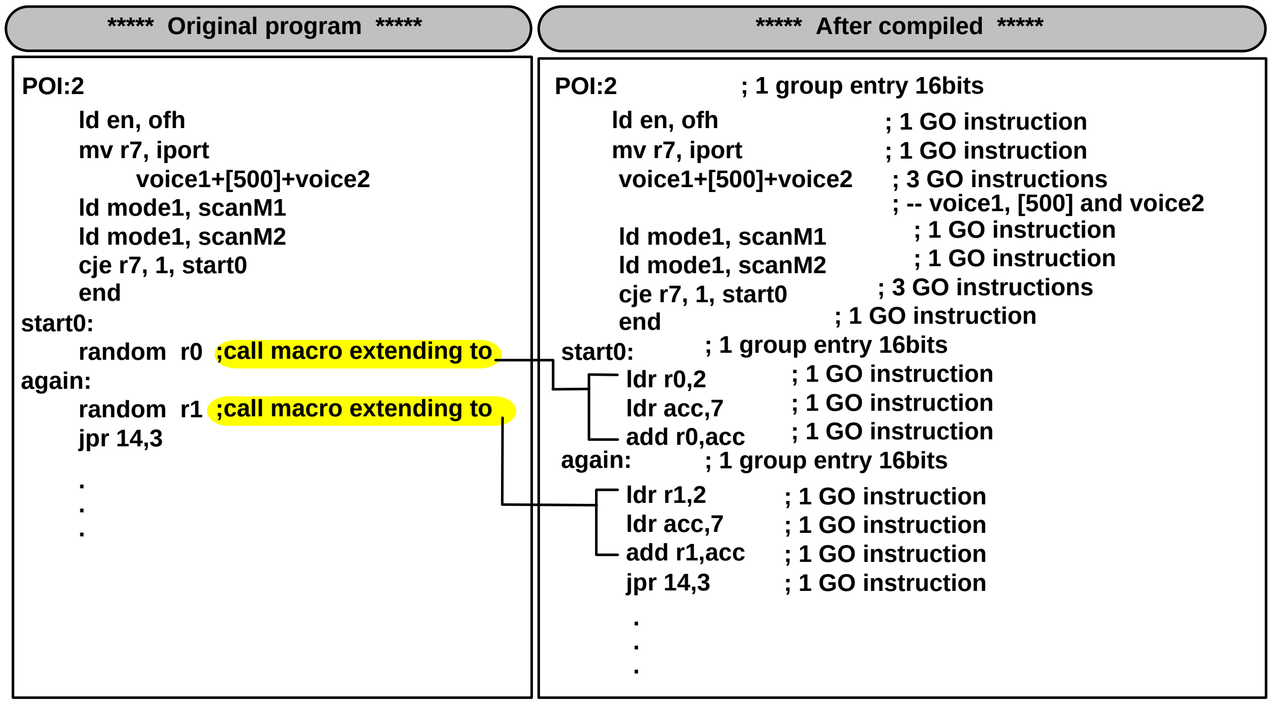
(1) command set with one GO instruction: ADD, OR, XOR, AND, LD, LDR, MV, JP, JPR, NOP, END, INS, DEC, SETB, CLRB, NOT

(2) command set with two GO instructions: COMP, DJNZ

(3) command set with three GO instructions: CJNE, CJE

(4) Each ADPCM and PCM file of the speech equation with one GO instruction: For example in the speech equation: A+B_21+C+[300]+D, there are a total of 5 GO instructions, including A, B_21, C, [300] and D.

The above example (for reference only) will be compiled and extended as follows:



```
*****  Original program  *****

POI:2
        ld en, ofh
        mv r7, iport
                voice1+[500]+voice2
        ld mode1, scanM1
        ld mode1, scanM2
        cje r7, 1, start0
        end
start0:
        random  r0  ;call macro extending to
again:
        random  r1  ;call macro extending to
        jpr 14,3
        .
        .
        .
```

```
*****  After compiled  *****

POI:2                   ; 1 group entry 16bits
        ld en, ofh                  ; 1 GO instruction
        mv r7, iport                ; 1 GO instruction
         voice1+[500]+voice2         ; 3 GO instructions
                                     ; -- voice1, [500] and voice2
         ld mode1, scanM1              ; 1 GO instruction
         ld mode1, scanM2              ; 1 GO instruction
         cje r7, 1, start0          ; 3 GO instructions
         end                   ; 1 GO instruction
 start0:         ; 1 group entry 16bits
          ldr r0,2          ; 1 GO instruction
          ldr acc,7         ; 1 GO instruction
          add r0,acc        ; 1 GO instruction
 again:          ; 1 group entry 16bits
          ldr r1,2          ; 1 GO instruction
          ldr acc,7         ; 1 GO instruction
          add r1,acc        ; 1 GO instruction
          jpr 14,3          ; 1 GO instruction
          .
          .
          .
```

In the above program, there are a total of 18 GO instructions and 3 group entries.

(9) Group name: Define the voice group entry point. This is different from the previous *PowerSpeech*ä W525XX and W528X devices which had to use a number to define the group name. In the W529XX, a label can be used to define the group name, as in the above program which used "start", "again", "keyin00",...etc. Note that the W529XX can also use a number to define the group name, but reserved words can't be used as normal labels. These reserved words are listed in the last page. Only letters and numbers can be used as the group name. Symbols such as "~", " #", " - ", ...etc are forbidden with the exception of the underline "_".

(10) Global repeat: The global repeat instruction is "n" which is from 1 to 4. This instruction must be placed on the same line as the group name.

(11) Speech equation: The program can accept ADPCM (*.WAM) files or PCM (*.SRC) files during compiliation and it will first look for ADPCM files. Therefore, if PCM files are used for better sound quality the "*.WAM" file should not exist in the same directory. See the above description. In the speech equation, using "*" for the local repeat and using "+" for combinative voices occupy around 160µS each.

(12) Call macro: To call a macro within the program, the macro name and the transmitted parameters must be in the same line and the number of the transmitted parameters must be equal. See the following example:

```
        Macro chk_keyin label0, label1, label2
        .
        .
        endm
        .
tg1:
        chk_keyin keyin00, keyin01, keyin02
        .
        .
```

The "chk_keyin" is the macro name, keyin00, keyin01 and keyin02 are the transmitted parameters.

During compilation, the statement below the "tg1" label will call a macro "chk_keyin keyin00, keyin01, keyin02" which will be extended and placed in the macro contents. When extending the macro "chk_keyin", "label0" is equal to "keyin00", "label1" is equal to "keyin01", and "label2" is equal to "keyin02". So the "chk_keyin keyin00, keyin01, keyin02" will be compiled to:

```
mv acc, r6
jp keyin00 @r6=0
mv acc, r6
sub acc, 1
jp  keyin01 @r6=0
mv acc, r6
sub acc, 2
jp  keyin02 @r6=0
jp keyin_over
```

Macros do not reduce the program size.

(13) Blank: A voice group name must be followed by one full blank line without any instructions or speech equations.

(14) Silence: In this regard, the W529XX is different from the other *PowerSpeech*ä ICs such as the W525XX, W528X, W523X and W581XX. The silence length in the W529XX will be automatically left shifted by 4 bits. The silence length of the [123] in the W529XX will be equal to [1230] in the other *PowerSpeech*ä ICs. So in the above example, [500] is equal to [5000] in others ICs and its period is about 0.853 sec. (sample rate is 6kHz). The maximum silence in the W529XX is [FFFF] and its period is about 43.69 sec. (sample rate is 6khz)

COMMAND SET

In the W528x and other predecessors, the *PowerSpeech*ä synthesizers simply provide customers with two types of commands: LD & JP, which are actually SIL in previous products. In the W529XX, more effort has been given to greatly expand the particular role of SIL into different kinds of commands, like ALU, move, subroutine call, etc.

During command execution, the internal sample rate selection will be set to 12 KHz by means of H/W & compiler maneuvering in order to reduce the cycle time. Each command will be executed within one cycle, which is defined to be less than 200 µS, except for those pseudo mnemonics that may require more than one cycle for the sake of user friendliness.

1. CONDITION CODE (~@cc)

@Trigger status

Upon execution of the command with such a condition, the W529XX will check the pad status of the specified TG1 - TG4 right at the execution time and then decide whether the command is to be executed, depending on whether the condition is met or not.

There are eight trigger status that are associated with the four TG pins as follows,

@TG1_H or @TG1_HIGH
@TG1_L or @TG1_LOW
@TG2_H or @TG2_HIGH
@TG2_L or @TG2_LOW
@TG3_H or @TG3_HIGH
@TG3_L or @TG3_LOW
@TG4_H or @TG4_HIGH
@TG4_L or @TG4_LOW

@LAST (Last global play time)
The maximum global repeat time of W529xx is four times. The condition @LAST is met only if the W529XX is running its last global repeat for a certain voice group.

@Flag
The @Flag includes four conditions: @ZERO, @~ZERO, @CARRY, @~CARRY.

- @ZERO (The condition when the result is zero.)
- @~ZERO (The condition when the result is not zero.)
The zero flag is modified accordingly upon execution of an ALU command. It is set to "1" when the result equals to zero, otherwise it is set to "0". For other conditions, such as data move commands, where the result may also equal zero, the zero flag is NOT affected.
Commands that will affect the zero flag are ADD, OR, XOR, AND, INC, DEC, SUB, SETB, CLRB, COMP, NOT, DJNZ, CJNE, and CJE.

- @CARRY (The condition when the carry flag is 1.)
- @~CARRY (The condition when the carry flag is not 1.)

When the ALU operation overflows, the CARRY flag will be set to "1". The carry flag is modified accordingly upon the execution of ADD, INC, DEC, SUB, and DJNZ commands.

Command	Flags affected	
	Carry	Zero
ADD	√	√
OR	-	√
XOR	-	√
AND	-	√
INC	√	√
DEC	√	√
SUB	√	√
SETB	-	√
CLRB	-	√
COMP	-	√
NOT	-	√
DJNZ	√	√
CJNE	-	√
CJE	-	√

Table: Relationship between ALU commands and processor flags

The processor's flags can only be modified by ALU commands, which are listed above, other commands will not modify the status of these flags. For example, the command "LD ACC, 0x00" also causes the result to become zero after the completion of command execution, but fails to alter the content of the ZERO flag.

@reg.n = 1:
This condition represents bitn of the register equal to 1.

@reg.n = 0:
This condition represents bitn of the register equal to 0.

@reg = 0:
This condition represents the value of the register equal to 0.

@reg! = 0:
This condition represents the value of the register not equal to 0.

8.2 INSTRUCTION DESCRIPTION

8.2.1 ALU

Instruction Set	Description	Example
ADD reg, n [@cc]	*ACC, reg ‹ reg + n, if cc met* Add a constant "n" to register "reg" [if condition is met]. The result is kept in "reg" and ACC. Flag affected: CF, ZF	**ADD R0, 00101011b** Memory. / Before exec. / After exec. n / 0010 1011b / 0010 1011b R0 / 7Fh / AAh ACC / --- / AAh
ADD reg2, reg1 [@cc]	*ACC, reg2 ‹ reg2 + reg1, if cc met* Add contents of register "reg1" to register "reg2" [if condition is met]. The result will be kept in "reg2" and ACC. Flag affected: CF, ZF	**ADD R0, R1** Memory. / Before exec. / After exec. R1 / 1 / 1 R0 / 4 / 5 ACC / --- / 5
OR reg, n [@cc]	*ACC, reg ‹ reg | n, if cc met* Bitwise OR operation of register "reg" and immediate value "n" [if condition is met]. The result will be placed in destination "reg" and also in ACC. Flag affected: ZF	**OR OUTPUT, 10101010b** Memory. / Before exec. / After exec. n / 1010 1010b / 1010 1010b OUTPUT / 0101 0101b / 1111 1111b ACC / --- / 1111 1111b
OR reg2, reg1 [@cc]	*ACC, reg2 ‹ reg2 | reg1, if cc met* Bitwise OR operation of register2 "reg2" and register1 "reg1" [if condition is met]. The result will be placed in destination "reg2" and ACC. Flag affected: ZF	**OR R0, R2** Memory. / Before exec. / After exec. R2 / 0Ah / 0Ah R0 / A0h / AAh ACC / --- / AAh
XOR reg, n [@cc]	*ACC, reg ‹ reg ^ n, if cc met* Bitwise XOR operation of register "reg" and value "n" [if condition is met]. The result will be placed in destination "reg" and ACC. Flag affected: ZF	**XOR R7, 10101010b** Memory. / Before exec. / After exec. n / 0011 1010b / 0011 1010b R7 / 77h / 0100 1101b ACC / --- / 0100 1101b

XOR reg2, reg1 [@cc]	*ACC, reg2 ‹ reg2 ^ reg1, if cc met* Bitwise XOR operation of register2 "reg2" and register1 "reg1" [if condition is met]. The result will be placed in destination "reg2" and ACC. Flag affected: ZF	**XOR R4, R5** Memory. Before exec. After exec. R5 55h 55h R4 AAh FFh ACC --- FFh
AND reg, n [@cc]	*ACC, reg ‹ reg & n, if cc met* Bitwise AND operation of register "reg" and value "n" [if condition is met]. The result will be placed in destination "reg" and ACC. Flag affected: ZF	**AND R2, 5EH** Memory. Before exec. After exec. n 0101 1110b 0011 1010b R2 0100 0101b 0100 0100b ACC --- 0100 0100b
AND reg2, reg1 [@cc]	*ACC, reg2 ‹ reg2 & reg1, if cc met* Bitwise AND operation of register2 "reg2" and register1 "reg1" [if condition is met]. The result will be placed in destination "reg2" and ACC. Flag affected: ZF	**AND R4, R5** Memory. Before exec. After exec. R5 55h 55h R4 AAh 00h ACC ---- 00h

8.2.2 DATA MOVE

Instruction Set	Description	Example
LD reg, n [@cc]	*reg ‹ n, if cc met* Load register "reg" with immediate value "n" [if condition is met]. Flag affected: none	**LD OUTPUT, 01011100b** Memory. / Before exec. / After exec. n / 0101 1100b / 0101 1100b OUTPUT / --- / 0101 1100b
LDR reg, RND [@cc]	*reg ‹ random, if cc met* Load register "reg" with a random number, which is determined by the following formula: $reg_i = r \cdot random_i$, $0 \le i \le 3$. $reg_i = 0$, $4 \le i \le 7$. where reg_i = bit i of register "reg" in binary representation, r = random bit(0 or 1), which is generated by H/W, $random_i$ = bit i of "random"(4 bit, 0 - 15) in binary representation. Flag affected: none	**LDR R2, 7** Memory. / Before exec. / After exec. random / 0000 0rrrb / 0000 0rrrb R2 / --- / 0000 0rrrb (0~7) r: random bit (0 or 1), which is generated by H/W.
MV reg2, reg1 [@cc]	*reg2 ‹ reg1, if cc met* Move the contents of register1 to register2 [if condition is met]. Flag affected: none	**MV R2, IPORT** Memory. / Before exec. / After exec. IPORT / 0100 1110b / 0100 1110b R2 / --- / 0100 1110b

8.2.3 BRANCH

Instruction Set	Description	Example
JP label (or group no.) [@cc]	*VECTOR ‹ group no., if cc met* Direct jump to the label entry or group no entry point [if condition is met], 0 ≤ Group No ≤ 255. Flag affected: none	**JP EXIT** Memory. Before exec. After exec. group no. EXIT 255 VECTOR --- 255 (In this example, the EXIT label is compiled to 255.)
JP reg [@cc]	*VECTOR ‹ [reg]., if cc met* Indirect jump to the entry value that is stored in the register [if condition is met]. Flag affected: none	**JP R7** Memory. Before exec. After exec. R7 56 56 VECTOR --- 56
JP reg+pointer [@cc]	*VECTOR ‹ [reg]+pointer, if cc met* Indirect jump to a location that is relative to immediate value "pointer" by [reg] [if condition is met]. Flag affected: none	**JP R3+128** Memory. Before exec. After exec. pointer 128 128 R3 20 20 VECTOR --- 148
JP reg2+reg1 [@cc]	*VECTOR ‹ [reg2]+[reg1], if cc met* Indirect Jump to a location that is relative to [reg1] [if condition is met]. Flag affected: none	**JP R3+R5** Memory. Before exec. After exec. R5 22 22 R3 36 36 VECTOR --- 58
JPR page, RND [@cc]	*VECTOR ‹ page.random, if cc met* Jump to a random location, which is jointly determined by one of a total of 16 pages and hardware generation. (see description below) Flag affected: none	**JPR 3,7** Memory. Before exec. After exec. random 7 (0111b, LSB) 0rrrb, LSB page 3 (0011b, MSB) 0011b, MSB VECTOR --- 0011 0rrrb The random number will be generated randomly between the range of voice group 00110000b ~ 00110111b, that is 48~55 group entry.

More information about the **JPR page, random [@cc]** instruction:

This instruction can let the user direct Jump to a 1-out-of-2^i random locations [if condition is met], under the range of $0 \le i \le 4$. The randomness, which is controlled by non-predictable H/W nodes, for this kind of command supports from 1 to 1/16. The page, defined by four MSB bits, is used to confine the range of random locations and it can be 0 - 15. The randomness, defined by four LSB bits, is used to generate a random number by hardware located on the corresponding page.
The pages list and corresponding voice groups are shown as following table.

Page	Voice Group
0	0 - 15
1	16 - 31
2	32 - 47
3	48 - 63
4	64 - 79
5	80 - 95
6	96 - 111
7	112 - 127
8	128 - 143
9	144 - 159
10	160 - 175
11	176 - 191
12	192 - 207
13	208 - 223
14	224 - 239
15	240 - 255

8.2.4 OTHERS

Instruction Set	Description	Example
NOP [@cc]	No operation [if condition is met]. This command can be used to slow down the W529XX as a delay.	**NOP**
END [@cc]	The W529XX stops immediately and enters into a standby state [if condition is met], provided there are no more global repeats. Otherwise, the W529XX will repeat the whole voice group again.	**END**
PSAVE	Ramp down and D/A converter off.	**PSAVE**

DEC reg [@cc]	*ACC, reg ‹ [reg2]-1, if cc met*	**DEC R2**		
	Decrement register by 1 [if condition is met].	Memory.	Before exec.	After exec.
		R2	5	4
	Flag affected: ZF, CF	ACC	---	4
INC reg [@cc]	*ACC, reg ‹ [reg]+1, if cc met*	**INC R2**		
	Increment register by 1 [if condition is met].	Memory.	Before exec.	After exec.
		R2	5	6
	Flag affected: ZF, CF	ACC	---	6
SUB reg, n [@cc]	*ACC, reg ‹ [reg]-n, if cc met*	**SUB R1, 25**		
	Subtract immediate data n from the register [if condition is met].	Memory.	Before exec.	After exec.
		n	25	25
		R2	56	31
	Flag affected: ZF, CF	ACC	---	31
SETB reg.bit [@cc]	*ACC, [reg]i ‹ 1, if cc met*	**SETB R4.5**		
	Set one of the register bits to 1 without altering the values of other bits [if condition is met].	Memory.	Before exec.	After exec.
		R4	0100 1100b	0110 1100b
		ACC	---	0110 1100b
	Flag affected: ZF, CF			
CLRB reg.bit [@cc]	*ACC, [reg]i ‹ 0, if cc met*	**SETB R4.3**		
	Set one of the register bits to 0 without altering the values of other bits [if condition is met].	Memory.	Before exec.	After exec.
		R4	0100 1100b	0100 0100b
		ACC	---	0100 0100b
	Flag affected: ZF, CF			
COMP reg, n [@cc]	*If [reg] equal to n, ACC ‹ 0, if cc met*	**COMP R4, 54**		
	Compare the data and place the result in ACC, but do not change the value stored in the register [if condition is met].	Memory.	Before exec.	After exec.
		n	54	54
		R4	54	54
		ACC	---	0
	Flag affected: ZF, CF			

COMP reg2, reg1 [@cc]	*If [reg2] = [reg1], ACC ‹ 0, if cc met* Compare the data between two registers and place the result in ACC, but do not change the value stored in the register [if condition is met]. Flag affected: ZF, CF	**COMP R4, 54** Memory. Before exec. After exec. R5 54 54 R4 54 54 ACC --- 0
NOT reg [@cc]	*ACC, reg ‹ [reg]' , if cc met* Performs 1's complement of the register's contents "reg" [if condition is met]. Flag affected: ZF, CF	**NOT R7** Memory. Before exec. After exec. R7 0111 0010b 1000 1101b ACC --- 1000 1101b
DJNZ reg, group no	*ACC, reg ‹ [reg]-1,' if ~ZERO=1 then VECTOR ‹ group no.* Decrement register by 1, then jump to the group number if the Not Zero flag is 1. Flag affected: ZF, CF	**DJNZ R6, start** Memory. Before exec. After exec. start 25 25 R6 4 3 ACC --- 3 VECTOR --- 25
DJNZ reg2, reg1	*ACC, reg2 ‹ [reg2]-1, if ~ZERO=1 then VECTOR ‹ [reg1]* Decrement register2 by 1, then jump to the voice group entry contained in the contents of register1, if the Not Zero flag is 1. Flag affected: ZF, CF	**DJNZ R6, R0** Memory. Before exec. After exec. R0 43 43 R6 7 6 ACC --- 6 VECTOR --- 43
CJNE reg, n, group no.	*If [reg] not equal to n, then VECTOR ‹ group no.* Compare the register value with the immediate data n, if not equal then jump to the voice group number. Flag affected: ---	**CJNE R0, 10, 20** Memory. Before exec. After exec. n 10 10 R0 15 15 VECTOR --- 20 ACC --- 5

CJNE reg1, reg2, group no	*If [reg1] not equal to [reg2], then VECTOR ‹ group number* Compare the register1 value with the register2 value, if not equal then jump to the voice group number. Flag affected: ---	**CJNE R0, R1, 20** Memory. Before exec. After exec. R0 10 10 R1 15 15 VECTOR --- 20 ACC --- 5
CJNE reg1, n, reg2	*If [reg1] not equal to n, then VECTOR ‹ [reg2]* Compare the register1 value with the immediate value n, if not equal then jump to the voice group number contained in register2. Flag affected: ---	**CJNE R0, 10, R1** Memory. Before exec. After exec. n 10 10 R0 15 15 R1 20 20 VECTOR --- 20 ACC --- 5
CJNE reg1, reg2, reg3	*If [reg1] not equal to [reg2], then VECTOR ‹ [reg3]* Compare the register1 value with the value of register2, if not equal then jump to the voice group number contained in register3. Flag affected: ---	**CJNE R0, R1, R2** Memory. Before exec. After exec. R0 10 10 R1 15 15 R2 20 20 VECTOR --- 20 ACC --- 5
CJE reg, n, group no.	*If [reg] equal to n, then VECTOR ‹ group no.* Compare the register value with the immediate data n, if equal then jump to the voice group number. Flag affected: ---	**CJE R0, 10, 20** Memory. Before exec. After exec. n 10 10 R0 10 10 VECTOR --- 20 ACC --- 0

CJE reg1, reg2, group no	*If [reg1] equal to [reg2], then VECTOR ‹ group number* Compare the register1 value with the register2 value, if equal then jump to the voice group number. Flag affected: ---	**CJE R0, R1, 20** Memory. Before exec. After exec. R0 10 10 R1 10 10 VECTOR --- 20 ACC --- 0
CJE reg1, n, reg2	*If [reg1] equal to n, then VECTOR ‹ [reg2]* Compare the register1 value with the immediate value n, if equal then jump to the voice group number that is contained in register2. Flag affected: ---	**CJE R0, 10, R1** Memory. Before exec. After exec. n 10 10 R0 10 10 R1 20 20 VECTOR --- 20 ACC --- 0
CJE reg1, reg2, reg3	*If [reg1] equal to [reg2], then VECTOR ‹ [reg3]* Compare the register1 value with the value of register2, if equal then jump to the voice group number that is contained in register3. Flag affected: ---	**CJE R0, R1, R2** Memory. Before exec. After exec. R0 10 10 R1 10 10 R2 20 20 VECTOR --- 20 ACC --- 0

8.3 RESERVED WORDS

There are several reserved words for the W529xx that should not be used as a normal label names. The following tables lists them.

ALU	**ADD, OR, XOR, AND, SUB, DEC, INC, NOT, SETB, CLRB, COMP**
Data Move	**LD, LDR, MV**
Branch	**JP, JPR, DJNZ, CJE, CJNE**
Other Instructions	**NOP, END, PSAVE, DEFINE, MACRO, ENDM**

Register	**R0, R1, R2, R3, R4, R5, R6, R7, ACC, MODE1, MODE2, OUTPUT, EN**
Keypad	**POI, TG1F, TG1R, TG2F, TG2R, TG3F, TG3R, TG4F, TG4R** **KEY10, KEY11, KEY12, KEY13, KEY14, KEY15, KEY16, KEY17** **KEY20, KEY21, KEY22, KEY23, KEY24, KEY25, KEY26, KEY27** **KEY30, KEY31, KEY32, KEY33, KEY34, KEY35, KEY36, KEY37** **KEY40, KEY41, KEY42, KEY43, KEY44, KEY45, KEY46, KEY47** **(i.e., The KEYmn means that this matrix tey is composed of the Trigger_m and the IO_n.)**
Conditional	**@LAST, @ZERO, @~ZERO, @CARRY, @~CARRY,** **@TG1_H, @TG2_H, @TG3_H, @TG4_H** **@TG1_L, @TG2_L, @TG3_L, @TG4_L** **@TG1_HIGH, @TG2_HIGH, @TG3_HIGH, @TG4_HIGH** **@TG1_LOW, @TG2_LOW, @TG3_LOW, @TG4_LOW** **@reg.n=1, @reg.n=0, @reg=0, @reg!=0 (!= means not equal to)** **(The reg means all kinds of registers of W529xx, and n is number 0~7)**
Others	**LED0, LED1, FREQ0, FREQ1, FREQ2, FREQ3, IPORT, ENTRY**