

INSTRUCTION MANUAL FOR SIM3022

1. Introduction

SIM3022 is a NC3022 software development tool that simulates the hardware devices using the software, such as NC3022, a custom CPU developed based on CASIO's V30MZ and associated LSI. This simulator provides not only the simulation environment, but also the debugging environment necessary to perform the symbolic debug of the hardware level BIOS and source code debug environment for C-language applications. This makes it possible to reduce application development costs and improve the development efficiency very much.

2. System Requirements

It is recommended to run SIM3022 on the system listed below. Particularly, use of a higher-speed CPU is strongly recommended since the simulation speed is in proportion to the capability of the CPU.

OS used	Windows 95/98 or Windows NT 4.0SP3 or later
CPU	Pentium 200 MHz or higher
On board memory	64 Mbytes or more
HDD	A free capacity of 10 Mbytes or more

3. Scope of Support

3.1 Scope of support

Presently (as of April 1999), SIM3022 supports the simulation of the following hardware devices.

CPU	NC3022
Memory	RAM / ROM FLASH ROM(MBM29LV160B/MBM29LV800B/MBM29DL322/3/4B) MEMORY I/O
D/D	OC3015/16, OC3025/26, T6C23/4, Generic LCD (Provided by Plug-in module)
Keys	General KI/KO keys and pull-up keys (Provided by Plug-in module)

3.2 Plug-in module

SIM3022 uses a plug-in module structure for D/D and key inputs to make flexibly applicable to a wide variety of target system environments. By changing the plug-in module, the simulator is applicable to other D/D. Additionally, to make the simulator applicable to a new D/D, only necessary plug-in modules need to develop.

3.3 Limitations and differences from actual machine

SIM3022 provides the hardware simulation, which is very close to the actual machine. However, there are some operations different from those of the actual machine due to some problems, such as simulation speed.

3.3.1 Interrupt timing

In NEC V30 based CPU, interrupts, which are input while the segment register load instruction is being executed, do not jump to a vector of the interrupt input previously after executing the next interrupt, but jump to a position after this interrupt has been executed, different from Intel 80x86 based CPU. However, SIM3022 does not check this interrupt position strictly and always puts the interrupt before the instruction to be always executed first after the interrupt has been input (in the same manner as 80x86).

Note that the interrupt cannot be input immediately after STI and I/O port write instructions (OUT/OUTSB/W).

3.3.2 About limitations of SIM86

The limitations on the discontinuity of bank when running the program and on the EMS management that have been observed on the SIM86 simulator for NC3020 are completely cleared.

4. Projects

A project file contains the hardware and software information necessary to simulate an application in SIM3022. By changing this project file, SIM3022 can perform the simulation and debugging of several systems with different structures.

4.1 Setting directories [Project]-[Directories]

4.1.1 Directory structure predicted

In SIM3022, all files used by the simulator use the relative directory designation in order to keep the transportability of the project files. Therefore, it is absolutely necessary to specify the reference directory, in order to calculate the actual file positions. This reference directory must be specified in the project.

Caution: Relative directories to be accessed must be located under the reference directory. That is, the directory structure must be configured so that all environments can be copied as the reference directory is copied.

4.1.2 Project current directory [Project Current Directory]

This project current directory is a reference directory for projects described above. All the files used for the project other than project file itself must be placed under this directory.

4.1.3 C-source base directory [C-source Base Directory]

This directory is used for the source code level debug described later. If the source code level debug is not made, an appropriate directory is used. For details, see the Chapter, Source code level debug.

4.2 General [Project]-[Configuration]-[General]

This menu is used to perform the basic hardware settings for simulation.

Project name [Project Name]

A project name is set. This name is displayed in the LCD window.

Base clock [Base Clock]

A frequency of the ceramic oscillator is set to run the project. As a larger value is set, the register update and interrupt, which depend on the ceramic oscillator, become faster (HTL becomes short).

INT and BLD signal logic setting [INT, I/O Logic Setting]

The input logic of the INT0, 1, and BLD signals is set.

4.3 Chip [Project]-[Configuration]-[Chips]

The operating status of the memory and D/D connected to target hardware devices are set. According to this connection information, the memory manager may load data from the memory chip file to a virtual 1M area.

4.4 Display [Project]-[Configuration]-[Display]

Items related to the display of the target are set. Window size, actual display size, actual display offset position, and magnification factor of the display can be specified.

4.5 Tablet [Project]-[Configuration]-[Tablet]

Parameters necessary to convert the mouse position in the display area into the A/D information, and the background of the LCD window (hard-icons) are set.

4.6 Keyboard [Project]-[Configuration]-[Keyboard]

The keyboard plug-in is set. Windows key assignments to KI/KO bits are set.

4.7 Source [Project]-[Configuration]-[Source]

Source codes for the symbolic debug and source level debug, analysis of the disassembly codes, and display settings are made. These settings can be omitted when the source code level debug and symbolic debug are not made.

4.8 Addin [Project]-[Configuration]-[Addin]

This is used to perform settings to run addin programs for the domestic models.

4.9 Open project [File]-[Open Project]

An existing project is opened. At this time, the contents in the SRAM and flash memory are restored to those when the project has been closed last, and the virtual CPU and simulating hardware are initialized.

4.10 Close project [File]-[Close Project]

An existing project is closed. At this time, the system prompts you whether or not the project is saved. Clicking [OK] will save the information, such as the contents in the SRAM and flash memory, and window positions. (This information will affect the simulator when opening the project for the next time.)

5. Build Project

The detail descriptions to build a project are provided in this chapter.

5.1 Creating new project

To create a new project, select **[File]-[New Project]**, or restart SIM3022. Basically, when selecting **[File]-[New Project]**, the project file name must be determined first.

When creating a new project by restarting SIM3022, the project file name is determined when saving the project.

5.2 Setting directories ([Project]-[Directories])

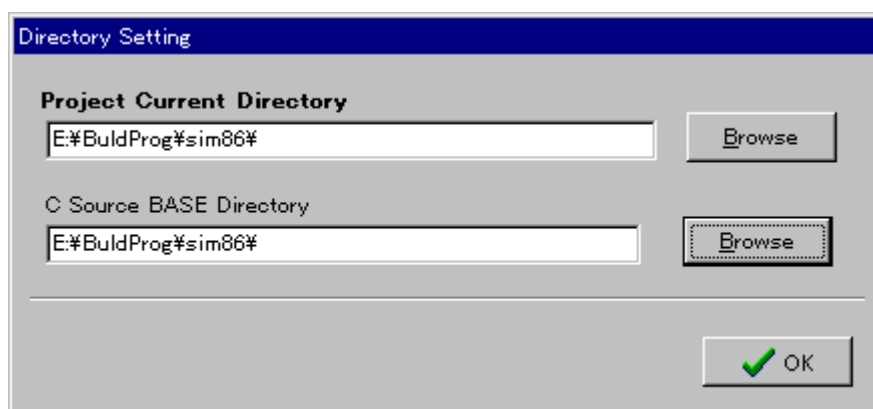


Figure-1: Directory setting dialog

As it is described in the chapter 4, it is necessary to determine a directory (**[Project]-[Directories]-[Project Current Directory]**) first for setting up the project.

As the simulator uses the relative directory to access the related files, the reference directory is needed. That is, all setting files or files for the chip must be stored in this reference directory.

In addition, it is necessary to specify the base directory (**[Project]-[Directories]-[C Source Base Directory]**), in order to perform a C-source code level debug. Normally, a directory where the map file locates is specified as a base directory. (For details, see the chapter 9.)

This directory must be set before **[Project]-[Configuration]** is set. Any available directory can be temporarily specified as *[C Source Base Directory]*. However, *[Project Current Directory]* must be set in advance since it is used for the initialization file settings internally in *[Configuration]*.

5.3 Setting general items ([Project]-[Configuration]-[General])

This section defines the general operations of the simulator.

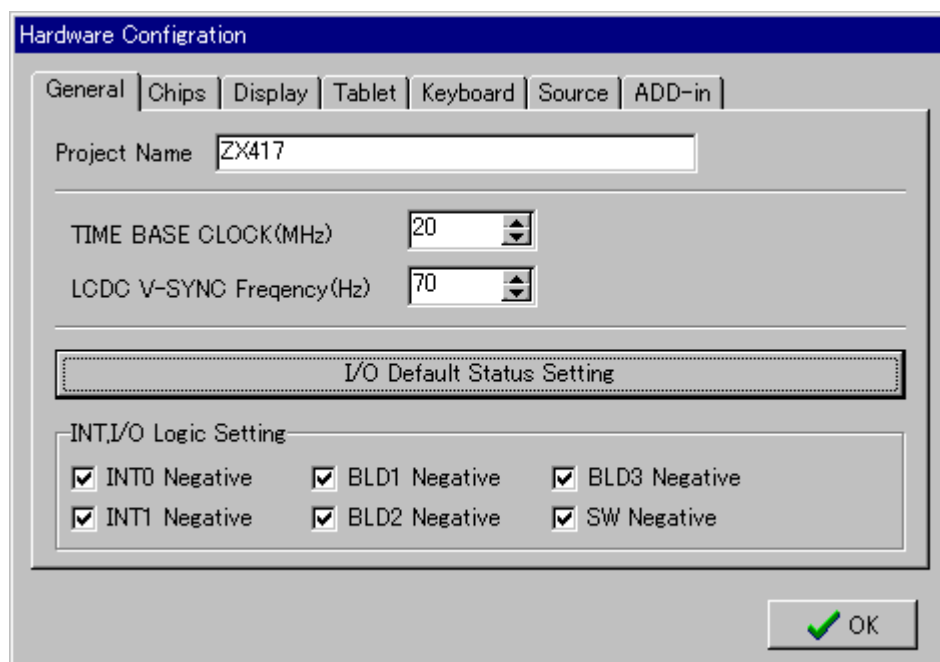


Figure-2: Hardware Configuration dialog

[Project Name]

This is used to set a project name. This project name is displayed in the program window.

[Time Base Clock]

The ceramic oscillator is used to set the pseudo operation speeds for counters. The precondition of the CPU operation is that the ceramic resonator is operated at 10 MHz. However, if this is set to 40 MHz, the simulative counting speed of the counters that synchronize with the ceramic resonator becomes four times faster (wait times become shorter when the wait operations are performed by the HLT command.)

[LCDC V-SYNC Frequency]

This setting is used to specify the frequency that generates the LCDC common interrupts. The interval of the LCDC common interrupts can be uniquely determined by this setting (no dependence on the setting of the I/O port 0x7D).

[I/O Default Status Setting]

This is used to set the default logic of the general-purpose ports. When this button is pressed, the setting dialog shown right is displayed. Each cell in the dialog indicates a bit for each port. When the input setting is made to the specified port, the value(s) specified in this dialog is read. The green colored cell indicates "L" (low), and the yellow one indicates "H" (high). So, click to select (toggle) a bit in the specified port.

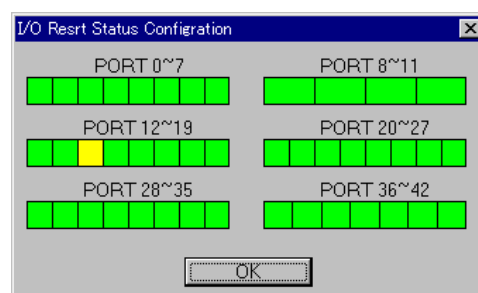


Figure-3: I/O Default Status Setting

[INT, I/O Logic Setting]

The input signals for the normal state of INT0, 1 and BLD pins are determined by this setting. As default, the negative logic is set since all pins are checked. That is, normally all pins are set to "H" level.

5.4 Setting chips ([Project]-[Configuration]-[Chips])

This section describes how memories or I/Os are connected to the CPU, and how those connections can be set.

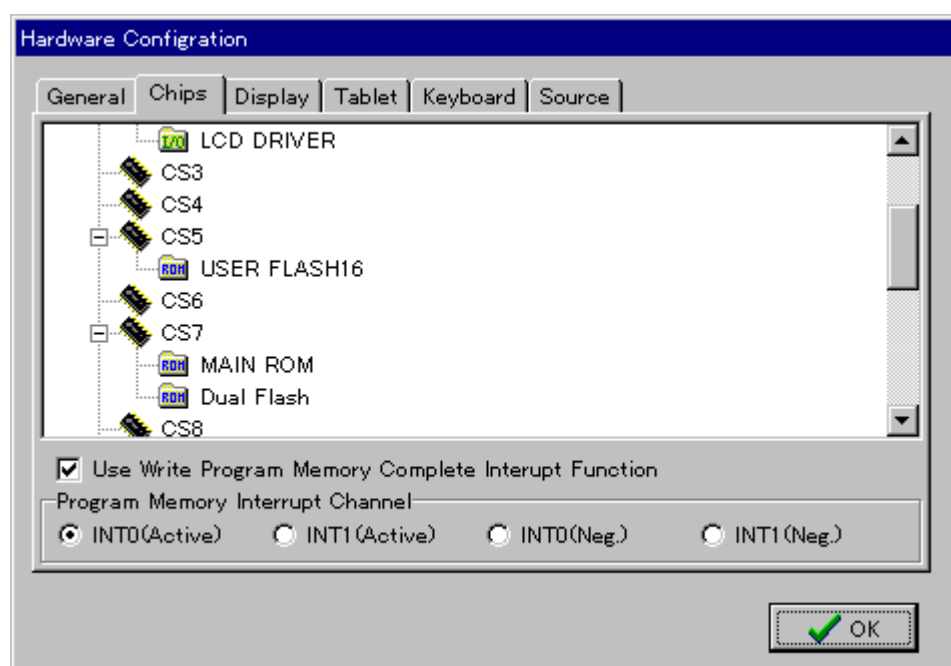



Figure-4: Hardware Configuration - Chips Tab

5.4.1 Limitations

- (1) Except the program device, the bus size is not considered in the memory-related simulations. In addition, the bus connection system (8-bit high/low, etc.) is not considered for all devices.
- (2) When the memory I/O is selected using the Chip Kind, other devices cannot be allocated to that CS.

5.4.2 Item settings

When opening "Chips" tab by selecting **[Project]-[Configuration]-[Chips]**, you can view the directory trees showing *[CS0] ~ [CS11]*, *[CSF]*, and *[INSIDE]*. *[CSn]* shows the CS pins of the NC3022, and indicates the actually connected chips by hardware. *[INSIDE]* indicates the internal ROM and RAM areas of the NC3022.

When performing the settings, selecting the CS item  (LSI icon) and clicking it provide you [Add] popup menu that enables you to open the chip selection screen to add items to each CS. In the screen, describe the chip settings properly in order to add a folder icon under the LSI icon.

Additionally, selecting a chip already registered, clicking it with the right mouse button, and selecting *[Properties]* brings you to the edition of the settings; selecting *[Del]* deletes the registered chip.

When the addition or edition of the chips is performed, the chip setting screen is displayed.

5.4.3 ROM settings

The program binary file that is actually operated is specified in the ROM settings. Generally, programs are stored in the Flash memory in the most actual applications. However, you must specify the program area as "ROM" in the SIM3022.

5.4.3.1 Actual setting example:

Figure-5: Chip File Property

Practical example of the settings is shown above. This display shows a setting for the chip that is displayed as MAIN ROM in CS7 in the Chip setting screen (Hardware Configuration - Chips tab) shown at the top of the section 5.4.

Details are described step by step below. Note that the basic parts of the explanation for RAM and program device are the same.

[Chip Caption]

This is a caption display item to be displayed at a side of the folder icons shown in the Chips tab of the Hardware Configuration dialog. This is a label to show simple explanation of the chips. So, you may use this to put a name to distinguish from other icons.

[Chip Filename]

This is used to specify a chip file to be used. As the ROM setting has been made in this case, so the file to be configured has to exist.

[Load Offset Address]

This is used to specify the CS address where the specified chip file is allocated.

[Chip Size]

This setting is not required for the ROM configuration. The SIM3022 automatically calculates this from the file size.

[Chip Kind]

This is used to select a kind of chip for a specified file that can be referred by the SIM3022 when handling it. In this explanation, it is a ROM. So, the ROM is checked.

[Program Chip Name], [Program Area Size] and [Connection]

Those items are not necessary to configure for ROM. This will be described in the section for the program device.

5.4.3.2 Error display when ROM is set.

When the settings are performed as described in the section 5.4.3.1, the target file is displayed as a folder icon in the Chips tab window if there is no setting error. If the settings are not correctly performed, the error display appears. For the ROM settings, the following points should be cared. If those are not kept, an error may occur when performing the settings.

- (1) The ROM file must exist in the specified location.
- (2) The areas to be occupied by *[Load Offset]* and *[Chip Size]* should not be overlapped with other Chip Files.
- (3) The CS where the program device or Memory I/O has already existed cannot be used for the ROM.

5.4.4 RAM settings

Basically, the same setting methods with ROM can be used for the RAM settings. However, the following two points are different:

(1) Set *[Chip Kind]* to RAM.**(2) Non existing file can also be specified at *[Chip File]*.**

If non-existing file is specified, the warning message is displayed during the setting operation, then a dialog appears and prompts you to enter the file size.

At this point, if you specify any file size, then the SIM3022 creates an empty file with the specified size.

Obviously, if this setting overlaps with the configured area for other chip files, then the warning message appears and the registration cannot be complete.

If the file exists, the RAM is configured with that file size.

5.4.5 Program device settings

The program device in this document means a device such as the Flash memory that can be erased or written by the command issued by the special address bus/data bus operation. As the simulation codes for such devices exist in the SIM3022 independently, the devices that have various accessing methods can be specified to individual files in the SIM3022.

In the SIM3022, the user memory space is used to place those devices, and therefore, you cannot place and run the programs in the space. However, as it is possible to allocate the specific space size by specifying *[Program Area Size]*, you can run the application program by configuring the ROM device to that space.

[Chip File]

The file that does not exist can also be used for this setting as well as the RAM. If non-existing file is specified, the file is created when simulation is executed.

[Chip Kind]

The Chip Kind for the program devices is either one from NOR FLASH/NAND FLASH/EEPROM. Though only NOR FLASH (MBM29LV800B/160B/29DL322/3/4B) is supported at the moment.

[Program Chip Name]

When selecting one from NOR FLASH and NAND FLASH/EEPROM at *[Chip Kind]*, the device names that the SIM3022 supports for the selected device is displayed. So, you can select a device that suits for your application.

[Chip Size]

The capacity of the specified device is automatically placed.

[Program Area Size]

When Dual Flash memory is used, this is used to specify the area size for storing the program file. Once this is specified, when the program accesses the same area, this Flash memory is not accessed, but the file that is configured as an overlapping ROM is accessed. That is, if the Flash memory command is issued to the reserved area by *Program Area Size*, the data bus never acts in the same manner as the Flash memory. So, this must be noted.

[Connection]

This specifies the connection type of the data bus to the specific program device. Either BYTE connection or WORD connection can be selected.

5.4.6 Memory I/O settings

The memory I/O specifies CS to be accessed in order to execute the memory port write API of Display Plug-in. Thus, [Chip File] and [Load Offset] are ignored. Furthermore, the possible settings that you can perform are to specify [Chip Caption] and specify MEMORY I/O to [Chip Kind].

If D/D that is used utilizes the memory mapped I/O, the CS must be reserved by this setting. Even if one MEMORY I/O is allocated to CS, no more Chips can be registered to that CS.

5.4.7 Program memory execution end interrupt settings

(Use Write Program Memory...)

The program memories including the flash memory indicate the processing statuses for data write or sector erase using the READY/BUSY signal.

To perform a wait for processing, connect this signal to the interrupt pin and stop the CPU using the HLT command.

This setting is to choose if simulating this or not. If this setting is checked, the status of the Flash memory is reflected to the pin that is used for the interrupt setting described later.

In the simulation, a busy flag of this READY pin is valid for 10-instruction execution time. (It is about 10 μ sec since one instruction takes 10 clocks and the tester is operated by 10 MHz.)

The toggle-bit emulation is also performed.

5.4.8 Program memory execution end interrupt signal settings

(Program Memory Interrupt Channel)

Sets the interrupt pin that is used for the memory access end interrupt. Where, "Active" indicates the positive logic and normally the signal level is "High"; the device that outputs "Low" during "Busy" is connected to the specified INT pin.

"Negative" indicates the negative logic and normally it is "Low"; the device that outputs "High" during "Busy" is connected to the specified INT pin.

5.5 Setting display ([Project]-[Configuration]-[Display])

The LCDC or display related simulation settings are performed.

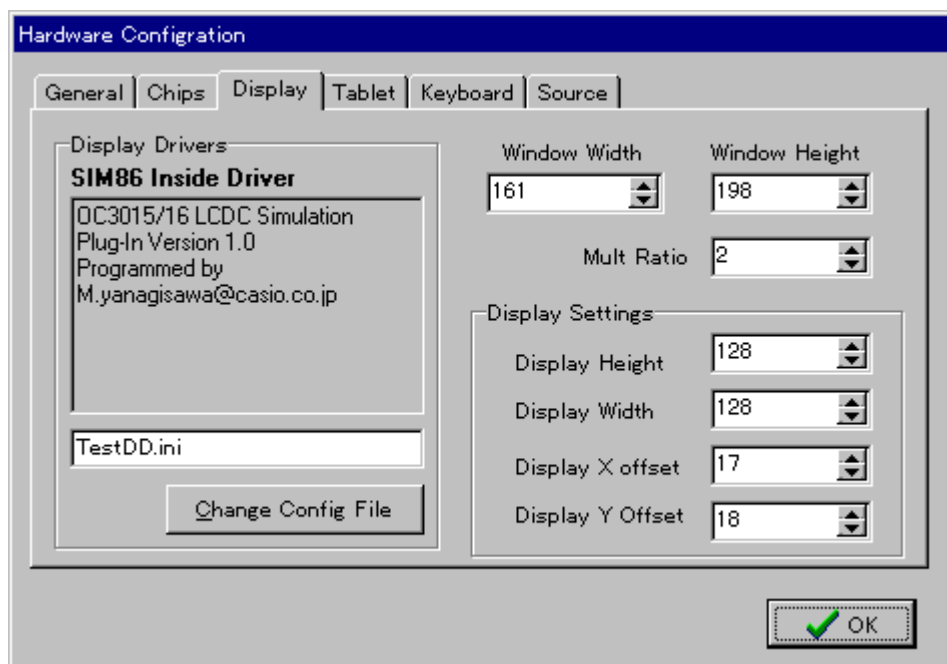


Figure-6: Hardware Configuration - Display Tab

5.5.1 Window size (Window Width / Window Height)

This setting specifies the window size that is used to display a target screen in the simulation. The setting size must be larger than the actual display size.

See the figure shown right to know more about the size description.

5.5.2 Zoom ratio (Mult Ratio)

A zoom ratio to view the displayed item is defined. Set the magnification ratio depending on the desktop size of the Windows.

5.5.3 Actual display size

(Display Height / Display Width)

This is an actual display area size to display in the window. By this setting, the VRAM size in the display plug-in module is set. So, it must be the same with the actual unit (tester).

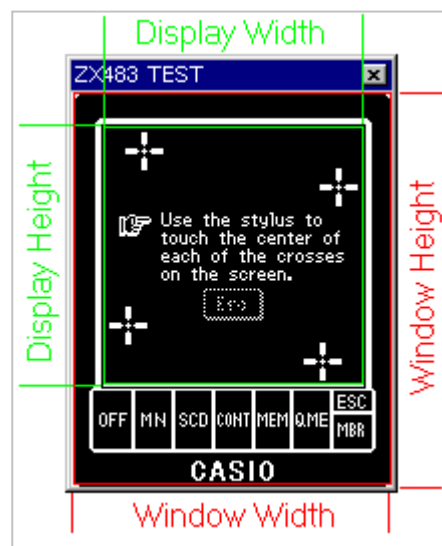


Figure-7: Display size

5.5.4 Display offset (Display X Offset / Display Y Offset)

This specifies the position to display the actual display area. The top left of the window coordinate is (0, 0), toward the right increases X, and toward the down increases Y.

5.5.5 Display plug-in

The SIM 3022 employs the plug-in display block. Therefore, it is possible to apply the simulation system to various target devices by replacing the plug-in file.

5.5.5.1 Configuration file

The display plug-in has a configuration file. Specifying this file can perform the special settings for each plug-in module.

This file can be selected by clicking the *[Change Config File]* button. The SIM3022 transfers this selected file to the plug-in module, and the plug-in initializes according to that information.

5.5.5.2 OC3015/16 Plug-in settings

The OC3015/16 plug-in can set one configuration file for OC3015 single unit and two files for the combination use of OC3015 and OC3016.

The plug-in reads the configuration file at the initialization, and the simulation codes for the internal D/D are initialized based on the contents of the configuration file.

This configuration file is a plain text file and has the following structure.

Example of Configuration file (ZX483)

```
DRIVERS=C[0,127,0,0],S[0,127]..... Segment/common setting of the driver.
DRV_CS=2,2 ..... Sets CS used by the driver.
DRV_MASK=0010,0020 ..... Sets address line used by the driver.
DRV_BUS=L_BYTE ..... Sets bus used by the driver.
CLKDIR=LEFT ..... Sets dot clock supply direction.
```

(1) DRIVERS

This is used to set a layout of the common/segment driver OC3015 and the segment driver OC3016, and a wire direction to the LCD.

In the layout, "C" indicates OC3015 and "S" indicates OC3016. "C" should be placed at the top or the end of the configuration. As shown in the example (Figure-8), when "C" is placed at the top of the configuration, OC3015 is placed on the left side of the panel. When "C" is placed at the end of the configuration, OC3015 is placed on the right side of the panel.

In the Figure-8, range values inside square brackets [] show common or segment array direction. For example, [A, B] shows the start point "A" and the end point "B". That is, if [0, 127] is given, the segments are wired to the LCD pins from 0th pin to 127th pin in normal order. However, note that wires to be commons or to be segments must be specified respectively when using OC3015. Since the OC3015 is a driver that supports both common and segment.

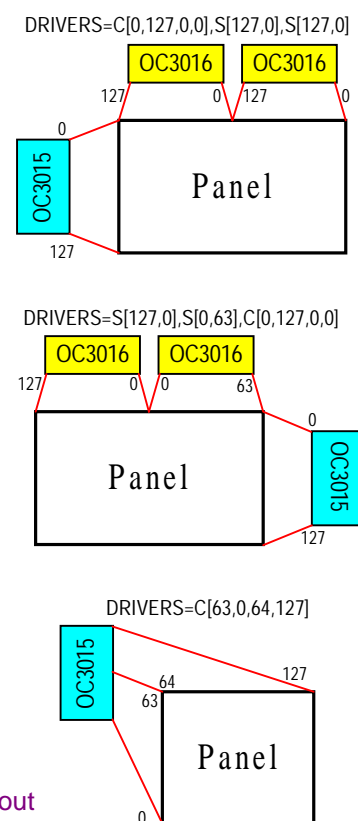


Figure-8: Examples of OC3015/16 layout

(2) DRV_CS

This setting is used to specify CS where the driver is connected. The setting is performed for each driver, and the setting order depends on the array order that has been specified in *DRIVERS*.

(3) DRV_MASK

Specifies the address lines to perform driver selections. The access is performed to the driver where the CS is selected and the specified bit is set.

(4) DRV_BUS

This is to select if connecting to the high-order 8-bit of the bus or the low-order 8-bit of the bus. Currently, this setting has no mean. Always select *L_BYTE*.

(5) CLK_DIR

This is used to set how the dot clock is transferred to the segments. This setting determines that the VRAM to be allocated to the lower side of the address is either right one or left one.

For example, in the Display circuit shown right, if the clock drifts from No.-1 to No.-2, then "RIGHT" is written in this setting. On the contrary, if the clock drifts from No.-2 to No.-1, then it is "LEFT".

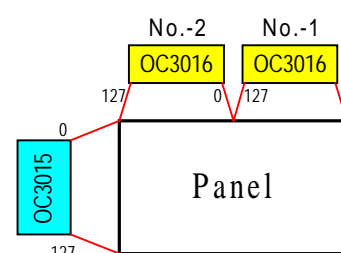


Figure-9: CLK_DIR

5.5.5.3 T6C23/4 plug-in settings

In the same manner as for OC3015/16 plug-in, T6C23/4 also reads a plain-text file transferred from SIM3022 as its configuration file, and initializes the plug-in internally.

Example of configuration file

```
DRIVERS=C[0,239,0,0],S[159,0],S[159,0]
DRV_CS=2,2,2
DRV_MASK=8000,2000,4000
DIR=L
```

(1) DRIVERS, DRV_CS, DRV_MASK

These settings are the same with OC3015/16. So, see the descriptions for OC3015/16.

(2) DIR

This setting has the same meaning with CLKDIR for OC3015/16. However, in this setting, just enter one letter "L" or "R". (Not like "LEFT" or "RIGHT".)

5.5.5.4 Generic LCDC plug-in settings

Generic LCDC plug-in is a general-purpose display simulation plug-in that uses the specific physical addresses as the steal VRAM addresses. Using this makes it possible to perform the development that is free from the type of LCDC to be used.

Normally, this plug-in does not refresh the screen display. Writing any value to the I/O port specified in the configuration file performs the refresh. Free addresses where the actual unit does not use should be specified to this I/O port.

Example of configuration file

```
SIZE=128,128  
MEMADDR=F0000  
REFRESH_PORT=7E
```

(1) SIZE

This sets the screen size. Specify the width first and then the height in dot units.

(2)MEMADDR

This is used to set a place for the steal VRAM. The addresses are set by the physical address in hexadecimal.

(3)REFRESH_PORT

This specifies the I/O port address to execute the screen refresh.

5.5.5.5 OC3025/26 LCDC plug-in settings

The OC3025/26 plug-in supports two commons and two segments as its maximum. Both OC3025 and OC3026 support the common and segment, differently from OC3015/16. Thus, they can be placed to any side, the common side or the segment side. The both plug-ins are internally recognized as the common and segment multiplexed type LCDC that has the different number of output pins.

Example of configuration file (ZX417)

DRIVERS=C6[0,119],C6[0,119],S5[0,159],S5[0,159]..... Segment/common setting of the driver.

DRV_CS=2 Sets CS used by the driver.

BUS=REVERSE Sets bus connected to the driver.

CLKDIR=LEFT Method to supply segment advance clocks.

CCLKDIR=DOWN Method to supply common advance clocks.

PALETTE_B=120,135,75 Sets background color (R, G, B).

PALETTE_W=0,0,0 Sets font color (R, G, B).

(1) DRIVERS

This is used to set the layout of DD. The basic methods to set the segment array are the same with the methods described for OC3015/16. However, the setting methods for the common and segment are different.

In the parameters, C and S indicate Common (C) and Segment (S) respectively, and following figure (5/6) indicates OC3025 (5) or OC3026 (6). The difference between OC3025 and OC3026 is the number of pins to be able to set; OC3025 has 120 pins (pin setting from 0 to 119), OC3026 has 160 pins (pin setting from 0 to 159).

(2) DRV_CS

This is used to set the CS where the DD is connected. The plug-in works properly only when the external LCD accessing CS is set to this CS.

(3) BUS

This sets the weight direction of the data bus where the DD is connected. Normally, NOMAL is specified. However, if the weight of the data bus is reversed, then specify "REVERSE".

(4) CLKDIR, CCLKDIR

This specifies the direction of the advance clock when multiple DDs are connected to the commons or segments. The CLKDIR is for segments. When "LEFT" is set to CLKDIR, the advance is performed from left to right. When "RIGHT" is set, then it is performed from right to left. The CCLKDIR sets the common side. When it is set to "UP", the advance clock drifts from down to up. When "DOWN" is set, then it drifts from up to down.

(5) PALETTE_B, PALETTE

The background and font colors to be displayed are set. This setting is performed using RGB, three primary colors. Each color can vary between 0 and 255, and white color is comprised when all three colors have the level of 255.

5.6 Setting tablet ([Project]-[Configuration]-[Tablet])

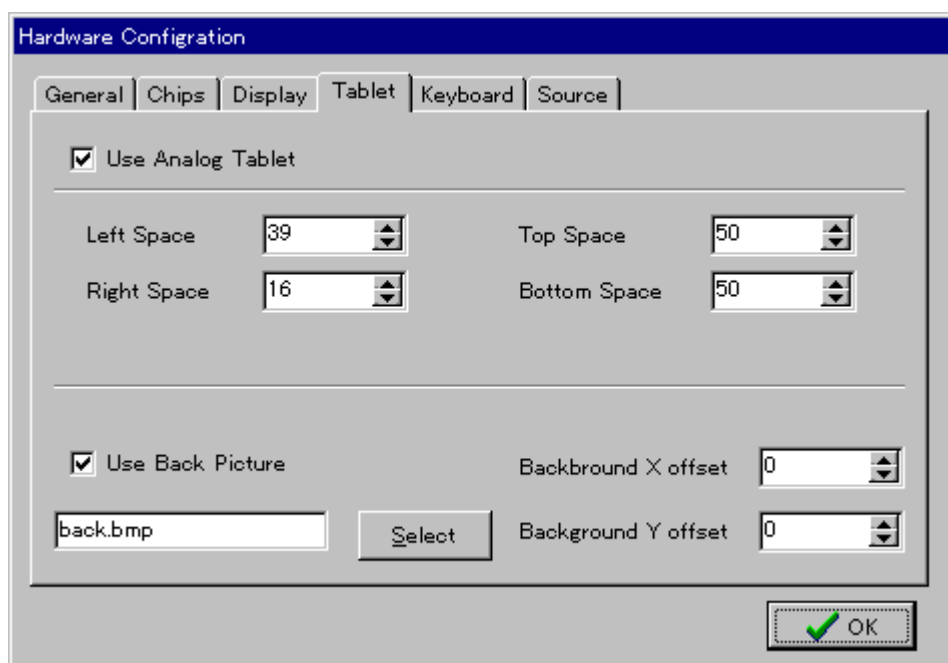


Figure-10: Tablet tab

5.6.1 Tablet settings

This sets parameters that specify values to output to the A/D conversion input for tablet using the mouse input. Each value specifies the extra tablet area against the actual display size of the LCD (*Display Width, Display Height*). By this way, the A/D conversion outputs for the touch location can be adjusted.

"*Left Space*" indicates the left side space, "*Right Space*" indicates the right side space, "*Top Space*" indicates the up side space, and "*Bottom Space*" indicates down side space of the screen respectively.

In case of the actual unit, the values vary depending on the characteristics of tablet to be used. For example, when determining the input range for the position setting symbol that is used for the calibration at startup, first adjust it on the actual unit so that it works properly. Then adjust the parameters on the simulator (when BIOS is set properly, it should work even if all values are set to "0").

Note that the "Use Analog Tablet" entry has no mean at the moment. However, it must be checked when the tablet is used.

5.6.2 Hardware icon settings

This is a setting to output the background screen for the LCD display window. When the hardware icons are drawn in this background screen, they may be used as a mark for the touch operation. This file is valid when the "Use Back Picture" entry is checked. At this time, if the "Select" button is clicked, the selection dialog of the BMP files for the background appears. So, select a file from the dialog.

Both "*BackGround X Offset*" and "*BackGround Y Offset*" are parameters to specify the location for outputting a selected file.

5.7 Setting keyboard ([Project]-[Configuration]-[Keyboard])

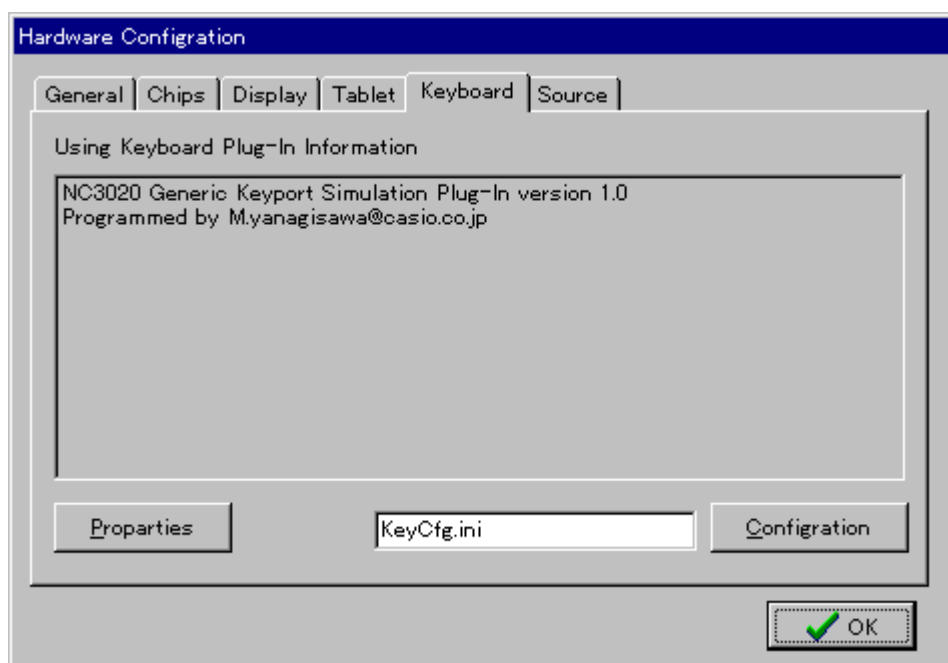


Figure-11: Keyboard tab

In the SIM3022, controlling ON/OFF of the internal KI/KO bits simulates the key operation. This is achieved by operating the corresponding KI bit based on the operation of the key where the keyboard simulation code is input or the operation of the Windows control.

5.7.1 Keyboard plug-in

In the SIM3022, the keyboard also uses a plug-in module as well as the display plug-in module. By this way, the keyboard can be used not only to convert the key input by the Windows but also to input by the software keyboard.

Currently, "Generic keyboard" plug-in that can simulate the Windows key inputs is released.

5.7.1.1 Configuration file

The keyboard plug-in can also specify the configuration files individually as well as the display plug-in. This file can be specified by clicking the **[Configuration]** button. When the button is clicked, the selection dialog box appears. So, a configuration file to be used can be selected in the dialog.

5.7.1.2 Property specification

If the keyboard plug-in supports some settings for the operation, it is possible to change the setting by clicking the **[Properties]** button.

5.7.1.3 Generic keyboard plug-in settings

This plug-in gets KI/KO information from a key to be input when the focus is in the plug-in window or when the focus is in the LCD window, and transfers the information to the SIM3022. The information to proceed the conversion from the input key to KI/KO information is described in the configuration file.

(1) Windows key codes

The virtual key code that corresponds to each key is set in the Windows. Using this virtual key code, the Windows key that corresponds to the target KI/KO is specified. See "APPENDIX A. Windows Virtual Key Code List" about the values.

Note that the list shows the code values in hexadecimal. However, when you set the values, it must be in decimal.

(2) Section

In the NC3022, the key port is composed of KO1-12, and KI1-8 that correspond to each KO. Therefore, when allocating a key, it can be set as follow: When key "X" is pressed, then KIy that corresponds to KOx is turned on.

As KI is set to each KO, writes the configuration file so that KI1-8 is set to a certain KO. See following examples to know more about this.

- When Windows key code "34" is pressed, KI2 of KO8 is turned on:

[KO_PIN8]

KI2=34

- When Windows key code "60" is pressed, KI3 of KO10 is turned on:

[KO_PIN10]

KI3=60

That is, when KOx is used for the explanation, it is written under **[KO_PINx]**. Like this character string is called the "section". It is possible to write multiple numbers of KIy=n under the section [KO_PINx].

(3) Pull-up key

In the NC3022 system, a key that turns on a specific KI regardless of the state of KO can be exist. To set such key, make a section as **[KO_ALL]** and write a corresponding KI to there. Once if the KI is specified in the [KO_ALL] section, other key cannot use that KI. The great care should be taken on this point.

5.8 Source code symbol handling ([Project]-[Configuration]-[Source])

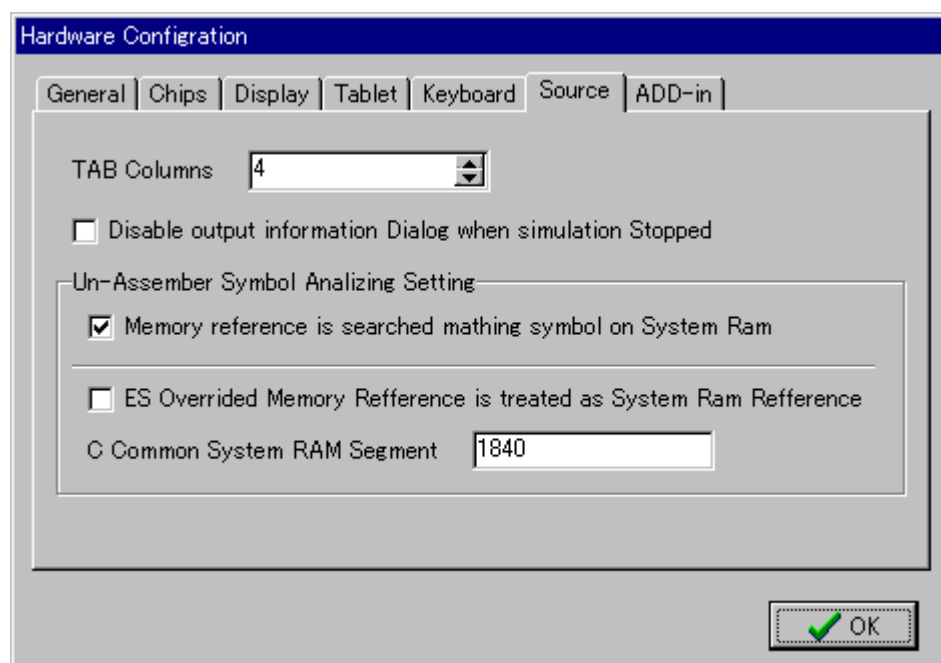


Figure-12: Source tab

5.8.1 TAB setting (*TAB Columns*)

This sets the number of columns to display source codes of the C language in the source code window.

5.8.2 Dialog display when the simulation is stopped

(Disable output information Dialog when simulation Stopped)

This is used to set whether the dialog is displayed or not when the simulator is stopped. When this item is checked, the address display at break will not be performed.

5.8.3 Symbol analyzing setting when disassembling

(Un-Assembler Symbol Analyzing Setting)

If the symbol file is loaded to the simulator, the memory reference labels cannot be set correctly at disassembling, while the destination labels for the CALL and JMP instructions can be determined correctly. Because of this, considering the characteristic of the output codes of the LSI-C86 compiler that is a standard development tool for NC3022, the symbols are determined based on the characteristic. The premises are set at this setting. (See the chapter 9.)

(1) Searching the system RAM symbol by the memory reference.

(Memory Reference is searched...)

Performs a process to replace the memory references with the symbols. As default, only the references for the DGROUP segment (referred by DS) are replaced.

(2)The memory references that are overridden by the ES segment is handled as the references of the common C system RAM area.

(ES Overrided Memory Reference is treated as System Ram Reference)

In the LSI-C86, the symbols other than those of DGROUP generate the codes to override with the ES segment. Because of this, the reference to the system common RAM area that is shared by applications is performed through ES, and therefore only the symbols for the common system RAM area are searched.

(3) C language common system RAM segment setting

(C Common System Ram Segment)

This sets the segments for the C application common RAM area. When (2) is not checked, this setting is not effective.

5.9 Addin program related setting

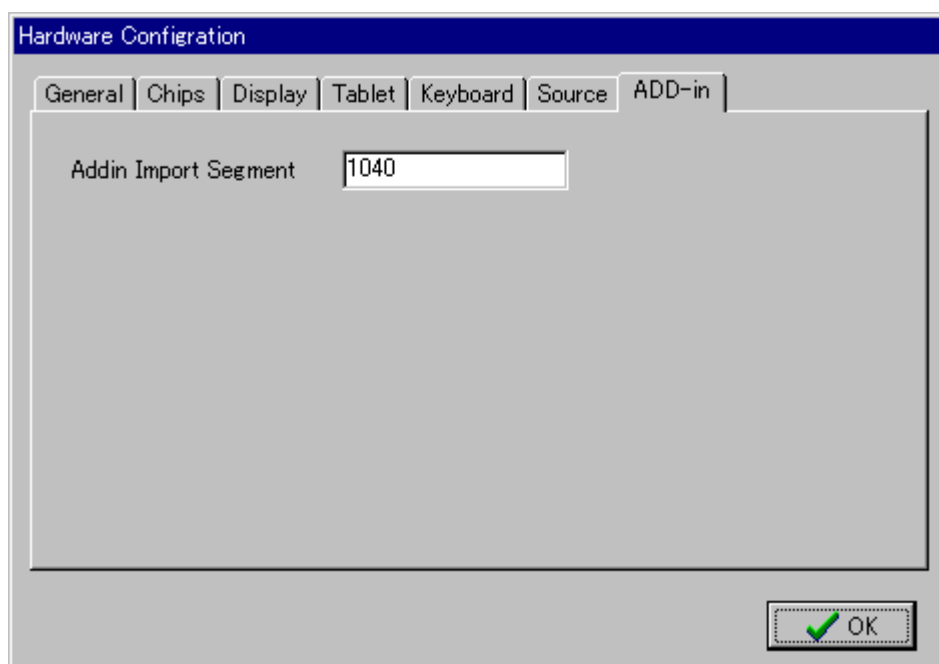


Figure-13: Add-in tab

5.9.1 Load segment setting of ADDIN program (*Addin Import Segment*)

This is used to set the load segment of the addin programs for the domestic version. The load and boot functions in the [Addin] menu is performed to this segment.

5.10 Saving project

When the setting of all items in the Configuration tab is complete, the project is closed by **[File]-[Close Project]**. At this time, if the project has not been created by **[File]-[New Project]** but by the startup state, the project file name is asked. So, enter the file name and save the project.

5.11 Updating project

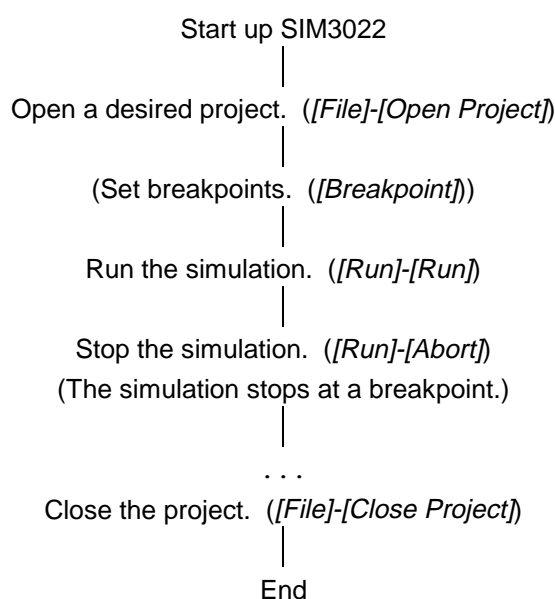
When the parameters for either [Project]-[Configuration] or [Project]-[Directories] are updated, the project must be closed by [File]-[Close Project] and saved in order to make the changes effective. Except [Project]-[Configuration]-[General]-[BaseClock], other parameters will not become effective instantaneously.

6. Run Simulation

When SIM3022 loads the project correctly, it resets the simulation kernel, making it possible to start the simulation. In the simulation, it is possible to operate the memory window. At this time, however, the display is not updated at real-time. Therefore, if any operation is made while the simulation is running, a fatal error may occur. In principle, do not operate while the simulation is running. Particularly, never change any values during running of the simulator.

If the focus is in the C-source code window in the C-source code level debug enable state, the step run and trace run are made at the source code level. Great care shall be taken.

[Running procedure]



6.1 Run ([Run]-[Run]) ShortCut=F9

The simulation is started from CS:IP displayed in the register window. The simulation is run until the user breaks it, breakpoint setting is activated, or undefined instruction is run.

6.2 Run trace ([Run]-[Trace]) ShortCut=F7

The trace run is started. Instructions are run one-by-one in the trace mode.

6.3 Run step ([Run]-[Step]) ShortCut=F8

The step run is started. Different from the trace run, the step run is stopped by the next instruction belonging to a series of addresses for instructions to be run.

For example, when the CALL instruction is run, the step run is stopped after the subroutine called up by this CALL has been run completely.

Therefore, pay special attention to the loop escaping conditions if the step run is stopped when escaping the LOOP instruction.

Additionally, the step run is always stopped every time one instruction is run when running a JMP instruction or RET instruction.

6.4 Run area ([Run]-[Here])

The program is run with breakpoints set at a specified position temporarily. This break information is different from the breakpoints. Therefore, this command is used even though the maximum number of breakpoints has already been set.

6.5 Run animate ([Run]-[Animate])

Instructions are run one-by-one and the registers and memory information are always updated every time one instruction is run.

6.6 Reset ([Run]-[Reset])

Simulating hardware devices and virtual CPU are reset.

6.7 Abort ([Run]-[Abort]) ShortCut= ESC

Simulation that is being run is aborted.

6.8 Open/Close log file ([Run]-[Open Log File])

Changes of CS:IP are recorded in a text file while the simulation is being run. This command menu becomes "[Run]-[Close Log File]" when recording the log.

6.9 I/O port operation ([I/O])

(1) Write to I/O port ([I/O]-[Write Word]/[I/O]-[Write Byte])

Data is written to a specified port. The data size is either 16 bits (Word) or 8 bits (Byte). When writing Word data, Byte data is written twice internally (lower byte and upper byte in that order). The address is masked by 16 bits, and therefore does not exceed \$FFFF.

(2) Read from I/O port ([I/O]-[Read Word]/[I/O]-[Read Byte])

Data is read from a specified port. 16 bits and 8 bits are read when specifying "Read Word" and "Read Byte", respectively.

(3) Displaying and setting the generic port values [I/O]-[Generic Port Control])

This is used to display the settings of the generic ports and set the statuses. The upper row of each port display is for the input/output setting, the lower row is for the display of the read value. The blue color shows the input, the red color shows the output in the upper row; the yellow color shows that "1" is input (output), the light green shows that "0" is input (output). Clicking the desired data box (toggle) can change the setting.

If the input/output setting is shown in the gray color, the port is used for other purpose. For instance, the PORT 36-42 is commonly used with the CRADLE PORT.

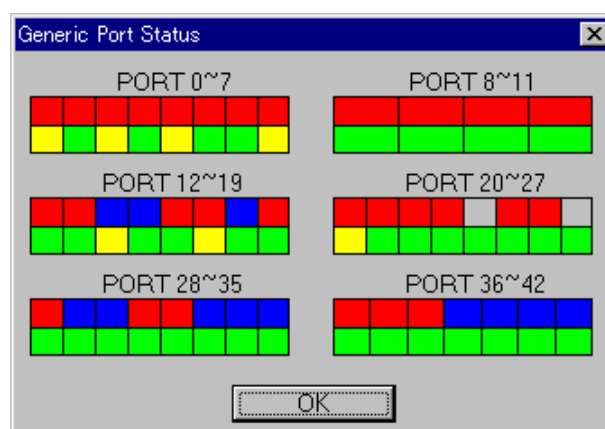


Figure-14: Generic Port Status

When the CRADLE port is selected, all boxes in the upper row of the PORT becomes gray. When it is not selected, the input/output settings are displayed as shown in Figure-14.

When the input/output setting is in the gray color, the output data is not actually output to a target that is set to other purpose. This should be noted.

5.10 INT/BLD signal operation

Pressing a button (Figure-15) located at the top of SIM3022 makes it possible to input an external interrupt signal, such as INT signal and BLD signal. The buttons from the left have the functions, NMI, INT0, INT1, BLD1, BLD2, BLD3, and SW.



Figure-15: Signal buttons

Click a desired button with the left mouse button to make relevant signal TRUE while keeping the button pressed.

Clicking a desired button with the right mouse button will lock the button in the TRUE state. To unlock the signal state, click the same button with the right mouse button.

This TRUE signal level is set using *INT, I/O Logic Setting of [Project]-[Configuration]-[General]*. If this setting is not checked ON, the H level of relevant signal becomes TRUE. On the contrary, if this setting is checked ON, the L level of relevant signal becomes TRUE.

NMI generates only NMI interrupt (INT 02h), and the values for the other ports are not affected. In addition, BLD3 is reflected to the bits in the 52h-th port, but no interrupt (NMI) is generated.

When BLD1 is set to TRUE, at the same time the A/D input value of the CH3 is changed to 0x100 (normally 0x3ff).

6.11 Import/Export from/to memory ([File]-[Memory Import]/[File]-[Memory Export])

Data is read or written from/to the virtual 1M byte area of the simulator. Data is read and written directly from/to relevant chip. This allows reading and writing from/to the program memory.

When clicking a desired item, the system asks you a target file name. After setting a file name, the system asks you a start address. You may specify a start address, from which data is read or written. Additionally, if you wish to write data, the system asks you a size of data to be written. When all necessary parameters are input completely, reading/writing of data is then started.

6.12 Copy and past memory contents ([Edit]-[Copy]/[Edit]-[Paste]/[Edit]-[Paste To File])

When the dump window is selected, performs copy and paste operations of the memory data. To perform a copy, select a copy area in the dump window using a mouse, and execute [Edit]-[Copy]. Then, place a cursor to the location where you want to paste the copied data, and execute [Edit]-[Paste] to paste the data. [Edit]-[Paste To File] writes the copied data to a file. The area to be copied is up to 32 Kbytes.

It is possible to paste a data copied not only from the dump window of the simulator but also from the text data of the general application. However, the text data copied is pasted as a Null termination character string.

7. Window functions

7.1 Disassembly window (UnAsm)

This disassembly window displays disassembly codes from a desired address. When clicking the right mouse button in this window, a pop-up menu will open, allowing you to select a desired command. At this time, the simulation with an address specified, such as running of a specified area or addition of breakpoints can be made. Additionally, when returning the pointer a line using the ↑ key, the system disassembles the codes and displays the disassembly results from an address, at which the longest valid instruction is located. Therefore, the address conversion may fail. If this occurs, moving the pointer using the Page Up or Down key will solve such trouble in most cases.




Key operation

Page Up	Moves the address backward by one page.
Page Down	Moves the address forward by one page.
↑	Moves the address backward by one line.
↓	Moves the address forward by one line.

7.1.1 Program counter (CS:IP) and breakpoint display

Some icons appear at the left end of the disassembly window. These icons have specific meanings as shown below. If conditions for more than one icon are met, these marks are overlapped.

Icons

 Red arrow	A position specified by the program counter (CS:IP).
 mark	An address at which the breakpoint (valid) is set.
 mark and X	An address at which the breakpoint (invalid) is set.

7.1.2 If the symbol file (MAP file) has already been read:

If the symbol information already exists, the address information is replaced with appropriate symbol information using the setting of *[Project]-[Configuration]-[Source]*. Additionally, if the C-language line number information exists, C-language source codes corresponding to addresses being displayed in the disassembly window are also displayed. (For details, see later sections.)

7.2 Register window (Regs)

This register window displays the internal register statuses in the simulator. Register values shown in this window are correctly displayed only when the simulation is stopped or when the animate is run. If this window is operated while the simulation is running, this may cause a fatal error. Never operate any register values while the simulation is running. When clicking the right mouse button in this window, a pop-up menu will open, providing you with several operations, increment, decrement, and zero-clear. Additionally, a value is input though double-clicking of the mouse button or numeric keys.

7.3 Dump window (Dump)

This dump window displays dump data at a desired logical address. Data displayed in the window can be edited in the hexadecimal notation. Additionally, data in the program memory can also be changed only in the dump window. The display area is one segment (64 Kbytes) and the pointer moves between offset addresses 0000 and FFFF during scroll operation.

It is recommended to avoid data change while the simulation is running. However, if the address, at which you wish to change data, is obviously beyond operation area of the currently running program, it is allowed to change such data.

Additionally, the display contents are not updated at real-time. Therefore, note that data displayed is that obtained immediately after the dump display routine of SIM3022 reads the data.

A pop-up menu appears when clicking the right mouse button in the dump window. This pop-up menu makes it possible to set a memory break to the memory in the selected place, specify display addresses, or perform the property display. By the property display, you can know the CS for the displayed position, the offset within the CS, and the physical addresses.

By the data search, it is possible to search a specified character string or a hexadecimal data within the segment that is displayed in the dump window.

Searching a character string is the same with the usual search. To perform the hexadecimal data search, check the "Hexa Data" item and write hexadecimal codes as a search string. For instance, if you wish to search a 3 byte data "0x00 0x20 0xAF", then enter '0020AF'.

Key operations

↑	Moves backward by one line (subtracts 16 bytes from the offset.)
↓	Moves forward by one line (adds 16 bytes to the offset.)
PageUP	Moves backward by display lines (subtracts 16 bytes x the number of display lines from offset).
Page Down	Moves forward by display lines (adds 16 bytes x the number of display lines to offset).
←, →	Scrolls left and right.
0 - 9, A - F	Inputs a numeric value.
S	Data search.

7.4 Stack window (Stack)

This stack window displays the word dump data while it is synchronized with values at SS:SP of the virtual CPU inside the simulator. It is possible to display the stack frame based on the BP register.

Additionally, the same key operations apply to this window, but the left and right scroll operations are not provided.

7.4.1 Switching of display format

As described above, it is possible to switch to the stack frame display in the stack window. This window switch is made through the pop-up menu that appears by clicking the right mouse button in the stack window. The command, **Stack Frame**, is provided on the 2nd line of the pop-up menu. Clicking this command will change the display. This display setting functions like a toggle switch. Therefore, to return to the previous status, click the menu item again. (This menu item is checked ON when the stack frame is displayed.)

7.5 Bank/Frame Window

This window displays the bank status and EMS frame setting status. Basically, this window displays only bank registers and EMS frame registers in the address format in order to provide an easily recognizable view. A pop-up menu is displayed by clicking the right mouse button during the bank or EMS is selected. Using this pop-up menu, the frame setting of the bank and EMS can be changed easily. If the window displays the data in blue, this shows that the banks and/or EMS are valid. On the contrary, if the window displays the data in red, it shows that they are invalid. (For banks, the status of BIT 7 is shown.)

EMS is also similar. However, NC3022 can disable the entire EMS function. At this time, a red strikeout is drawn on the FRAME display.

Though the addresses A27-A12 are displayed, the lower 2 bits are always "0" at the EMS frame display. EMS/BANK address setting also uses the same address expression. Note that the lower 6 bits of the bank address and the lower 2 bits of the EMS address is cleared (specification of NC3022).

7.5.1 Bank setting

In the bank setting, a pop-up menu that has *[Toggle]* and *[Set]* menu is displayed. The Enable/Disable setting of the selected bank is performed by clicking *[Toggle]*, and the value can be set by clicking *[Set]*.

Double-clicking with the left mouse button has the same meaning with the *[Set]* menu.

7.5.2 EMS settings

A pop-up menu appears when clicking the EMS frame with the right mouse button. *[Toggle]* and *[Set]* in the menu work in the same manner with those for the bank setting.

Additionally, the EMS setting has an item to disable the EMS function that is described above. This is "EMS Enable" in the menu. Each time this item is clicked, the selected state is toggled.

When this item has a check mark, it shows that EMS is enabled.

Furthermore, the positions of the EMS frame 8 to 11 can be set. The NC3022 makes it possible to move the frame address to either of C0000, D0000, or E0000 by software using the EMS setting. Because of this ability of the NC3022, the simulator can also perform the same operation. *[C0000-CFFFF]*, *[D0000-DFFFF]*, and *[E0000-EFFFF]* are available in the menu, and the currently selected frame has a check mark.

To switch this selection, click the desired frame address on the display.

7.6 Watch window (Watch)

Using this watch window, you may monitor specific values. The monitoring system you can set is very simple. Additionally, values displayed in this window are updated after the simulation has exited or the evaluation has been run.

7.6.1 Syntax of evaluation expression

The syntax of the evaluation expression is shown below.

Memory evaluation *(SegmentOverride)(Symbol)[Method] , (SegmentOverride)Symbol*
Value evaluation *Method*

An arithmetic expression with brackets, including registers and direct values applies to *Method*.

Example

```
GLOBAL_INDEX
ES:SYS_DATE_FORMAT[BX+5]
(SI+7)*10
```

7.6.2 Adding, deleting, and editing the evaluation expression

An evaluation expression can be added using **[Source]-[Add Watch]** or through the pop-up menu, that appears by clicking the right mouse button in the watch window. To add an evaluation expression, click **[Add]**. To delete or edit an evaluation expression, select a desired evaluation expression to be deleted or edited, and click the right mouse button. The menu will appear. Clicking **[Delete]** will delete the expression while clicking **[Edit]** will allow you to edit it.

When adding or editing an evaluation expression, the monitoring method setting dialog box (Watch Edit) will appear. You may put an expression in the Watch input part in this dialog box and set the evaluation type.

The watch window displays values based on the evaluation type you have set. At this time, if the expression you have set is incorrect, an error occurs, prompting you to input an expression again.

Evaluation type

Character	Character
Byte	1-byte hexadecimal display
Word	2-byte hexadecimal display
DWord	4-byte hexadecimal display
Integer	2-byte decimal display
String	Character string divided by NULL

7.6.3 Re-evaluation of evaluation expression

To re-evaluate an expression, select **[Source]-[Evaluate]** or **[Evaluate]** in the pop-up menu.

Normally, the expression is evaluated when the simulation is aborted or when you edit any items in the watch window. Additionally, if you wish to refresh the display, you may execute the re-evaluation of the evaluation expression. This may update the display with the latest information.

7.7 Source window (Source)

SIM3022 reads relevant C-source code file into the memory if the C-source code line number information is contained in the MAP file when SIM3022 reads the symbol data. This source window is used to display the C-source codes.

The source window is always synchronized with the simulator. This allows you to display the lines at the C-source code level, which are being run, set or cancel the breakpoints in the source window when an application is run.

7.7.1 Meanings of display items in source window

In the source code display, line numbers are shown on the left portion. In response to these line numbers, there are lines, on which numeric data is shown and those on which numeric data is deleted with red line.

Lines, on which numeric data is shown, are specified by the line number information in the MAP file. You may put breakpoints on these lines, but you cannot put breakpoints on lines, which are deleted with red lines.

Additionally, icons may appear on the left end in the same manner as described for the disassembly window. The meanings of these icons are the same as those shown in the disassembly window. For details, see the section, Disassembly window.

Key operations

Page Up	Moves backward by one page.
Page Down	Moves forward by one page.
↑	Moves backward by one line.
↓	Moves forward by one line.

7.8 Key and LCD display windows

These windows simulate the system screen and keys, which are to be simulated. When touching the LCD display window using a mouse, this information is simulated to process the tablet in the same manner as if the actual tablet is touched.

The shape of the key window is not the same as that on the actual machine. However, this window functions to transfer key operations made in the Windows to the simulator.

Note that the key data is transferred to the simulator only when the focus is in the key window or LCD display window. Great care should be taken during simulation.

8. Breakpoints

A breakpoint is used to stop the program at a desired execution address during simulation. SIM3022 provides three kinds of breakpoints. A program breakpoint is used to stop the program before it executes an instruction at a specified address. A memory read breakpoint is used to stop the program before it reads data from a memory. A memory write breakpoint is used to stop the program before it writes data to a memory.

8.1 Program breakpoint

There are two kinds of designation methods to express a program breakpoint, one uses a logical address and the other uses a physical address. When using a logical address designation, a breakpoint is specified by an address in the 1 Mbytes area, an actual execution address area of V30MZ, which is a core part of NC3022. When using a physical address designation, a breakpoint is specified by a physical address, to which the memory management unit of NC3022 can access directly.

Among these two designation methods, the breakpoint is basically determined by the physical address. In NC3022, a memory in the physical area is attached to a bank in the logical address area. Therefore, if a logical address is used, this may cause the program to stop in an unexpected place or not to stop continuously depending on the bank register setting.

However, when debugging the C-source codes, only physical address information exists in the map file. At this time, breakpoints are checked by a logical address.

In principle, the source code level debug is intended to debug the application. Additionally, the memory for the application does not change the physical address corresponding to the current logical address. Therefore, there are no problems to use the logical address for breakpoints.

8.2 Memory breakpoint

SIM3022 also allows you to set breakpoints for the memory access. That is, it is possible to stop the simulation when the simulator attempts to read data at a specific address of the memory (memory read break) or write data to the memory (memory write breakpoint).

In principle, a memory breakpoint is set by physical address. If it is specified by logical address, breakpoints are set for the physical address corresponding to the logical address.

8.3 Breakpoint operations, such as addition and deletion

A breakpoint is added through the menu *[Breakpoint]* in the main window. To add a program breakpoint, memory read breakpoint, and memory write breakpoint, click *[Break At...]*, *[Memory Read Break]*, and *[Memory Write Break]*, respectively. After that, a dialog box will appear where you can set a breakpoint address.





The program is set only by the logical address. The memory breakpoint is set by the logical and physical addresses. You may input a desired breakpoint address meeting your purpose.

Both *[Delete At...]* (Delete) and *[Toggle At...]* (status change) are processed regardless of the type of breakpoint if the specified address is the same.

8.4 Breakpoint window (Break)

This breakpoint window lists up currently set breakpoints. The breakpoint contains the items, State, Kind, Real Addr, Physical Addr, and Information. Their meanings are shown below.



Display example

State	Kind	Real Addr	Physical Addr.	Information
 EN	PROGRAM	8000:0000	-----	C\CLKMAIN.C at 55 Line
 DI	PROGRAM	8220:95C0	8457C0	
 EN	M-READ	8000:0000	800000	
 EN	M-WRITE	-----	040000	

State

The state provides two kinds of attributes, enable and disable. If EN is set to the state, the breakpoint is enabled. If DI is set to the state, the breakpoint is disabled.

State indications

 N (Green circle)	Breakpoint is enabled.
 DI (Red circle)	Breakpoint is disabled.

Kind

This is used to specify the kind of breakpoint. Three kinds of indications are available.

PROGRAM	shows the program breakpoint.
M-READ	shows the memory read breakpoint.
M-WRITE	shows the memory write breakpoint.

Real Address, Physical Address

The real address means a logical address of the break point while the physical address shows the its physical address. If any address is invalid for a specific breakpoint, "-----" is shown. For example, "-----" is shown when setting C-source code level breakpoints. Additionally, when specifying a physical address directly for memory read/write breakpoints, "-----" is also shown. If both logical and physical addresses are set for program breakpoints, this logical address display shows a logical address when the breakpoint is set. Therefore, if relevant physical address is put in other bank, the simulation is stopped even though the logical address is different. Always pay special attention to this point.

Information

This is additional information. If a breakpoint is added through the pop-up menu that appears by clicking the right mouse button in the C-source code window, line numbers set by the source file name and breakpoints are added as a comment.

8.5 Editing breakpoints

It is also possible to add or delete a breakpoint through the pop-up menu that appears by clicking the right mouse button in the breakpoint window.

To add program, memory read, and memory write breakpoints, select **[Add Program Break]**, **[Add Memory Read Break]**, and **[Add Memory Write Break]**, respectively from the menu that appears by clicking the right mouse button in the breakpoint window.

When a breakpoint item is selected in the list of the window, you may delete the selected breakpoint or change its status using **[Delete]** or **[Toggle]** (enable or disable).

If no breakpoints are selected, operation is made on all the breakpoints (Delete: **[All Disable]** and status setting: **[All Toggle]**).

The status setting is toggled. If the current status is enabled, clicking [Toggle] will make the status disabled. On the contrary, if the current status is disabled, clicking [Toggle] will make the status enabled.

8.6 Checking the breakpoint position

If you wish to check the breakpoint positions in the program or memory, you may click **[View Unasm]**, **[View Dump]**, and **[View Source]** to update the display addresses (or lines) displayed in the disassembly, dump, and source windows, respectively.

Each window shows only data at logical addresses. Therefore, if there is no entity of data at that logical address (data at relevant physical address), the window display becomes meaningless.

9. Symbol debug

This symbol debug is a useful function that allows you to embed symbols in the disassembly list or to use symbols for address designation, such as jump.

9.1 Preparations for use of symbol debug

There must be a correspondence list between the symbol and address in order to use symbols. A MAP file with Microsoft specifications is used for this correspondence list, which is generated when linking objects. When SIM3022 reads a MAP file, the symbol - address conversion list is made. After that, you may use symbols for address designation. Additionally, the address substitution is also made for the disassembly source.

9.2 Public symbol window (Public Symbol)

This window lists up the symbol information read from the MAP file. The list is sorted, and the sort order can be changed. Data is to be sorted by *[Address]* and *[Symbol Name]*. Clicking the top column will sort data contained in that column. Additionally, the sort order is toggled between ascending and descending sort order by clicking the same column.

This window also provides a pop-up window that appears by clicking the right mouse button. The pop-up window contains several function menus, such as adding of a symbol to Watch, or moving of a disassembly or source code to a display position of the window.

To add a specified symbol to Watch, click *[Add Watch]*.

Additionally, when you wish to display the position of a desired symbol, select *[Jump Unasm View]* for disassembly code, *[Jump Dump View]* for dump code, or *[Jump Source View]* for source code. To change all data, select *[Jump All Views]*.

9.3 Expanding the disassembly window

By reading the symbol file (MAP file), the disassembly window will replace the addresses, such as JMP or CALL destination, with symbols based on the symbol information.

In addition to these symbols with fixed addressed, symbols to be used for the memory reference are also available. Even though it is ideal to replace such symbols, x86 CPU may cause the segment to switch during segment override in the memory reference. Therefore, it is not possible to replace the symbols while checking only the offsets.

Such symbol replacement can be made under specific conditions in the C-source code level debug. This setting is made by ***[Project]-[Configuration]-[Source]***. (For details, see the source code level debug in the next chapter.) However, the above symbol replacement cannot be used in the BIOS debug. The settings are made carefully so that this replacement is not made in such debug.

10. Source code level debug

This source code debug provides break and step operations at the C-language source code level to support debugging of an application which is being developed by the C language applicable to NC3022. Using this debug function, you may understand how the program runs in the C-language, making it possible to easily verify or debug the logical operations. This section describes the source code level debug by using the LSI-C86 compiler, a standard NC3022 application development tool, as an example.

10.1 Preparations for executing the source code level debug

Before executing the source code level debug, it is necessary to get necessary information containing how the C-language program is located in the memory. This information is called “line number information” and can be created when the MAP file described in the previous section, Symbol debug, is written according to the C-language compile settings. When loading the MAP file containing this line number information into SIM3022, relevant C-language source codes are also read into SIM3022 and displayed in the source window.

10.1.1 Directory structure of target

The MAP file used in this debug contains the address information corresponding to each line of the source file. A directory specified during link may affect the directory position of the source file. To avoid such problems, the base directory for the source file must be specified clearly in SIM3022 in order to correctly read the source codes.

Another word, C-source codes must be put under this base directory and can be accessed by the directory designation described in the MAP file.

In SIM3022, such directory is called base directory of the target and specified using **[Project]-[Directories]-[C Source Base Directory]**.

Before starting the source code level debug, it is absolutely necessary to specify this directory in a valid place.

10.1.2 Creating a MAP file and MAKEFILE

When creating a MAP file, always pay special attention to the following cautions.

1. It must be necessary to access to the C-source code of the target through a directory set using [C Source Base Directory] by means of the relative directory designation.

(Example)

```
lcc86 -ms -g -a -c -cs c\mainfile.c
```

2. The line number information must be put. (“-g” option must be put in LSIC-86.)

(Example)

```
lcc86 -ms -g -a -c -cs c\mainfile.c
```

To create a makefile, the above two points must be taken into consideration. When reading a MAP file created by compiling a makefile using **[Source]-[Load Symbol]**, the symbol information, source codes, and line number information are read together and displayed in the source code window.

10.2 Source debug settings

To execute the source code level debug properly, it is absolutely necessary to fully understand the settings of **[Project]-[Configuration]-[Source]**.

In principle, these settings are not important when checking only the C codes. However, if you wish to check the disassembly display, you must make these settings properly.

(1) Searching the system RAM for symbol using memory reference (Memory Reference is searched ...)

This replaces the memory reference with symbols. With the default settings, only the reference to the DGROUP (DS reference) segment is replaced.

(2) Using the memory reference overridden by ES segment as common C-system RAM area reference (ES Overrided Memory Reference is treated as System Ram Reference)

In LSI-C86, the symbols other than DGROUP generate codes to be overridden by the ES segment. Therefore, the system common RAM area shared by the applications is referred to through ES. At this time, only symbols in the common system RAM area are searched for.

(3) C-language common system RAM segment settings (C Common System Ram Segment)

This sets segments in the C-application common RAM area. This setting is invalid unless item (2) is checked ON.

10.3 Source code window

This source code window displays only correctly read files of those specified by the MAP file.

Tabs showing file names are provided under the window caption. Clicking a desired tab will change the file display. The program counters and breakpoints are displayed at the left end in the same manner as described in the section, Disassembly window. Since the program counter display follows the line number information, arrow marks of the program pointer will indicate more than one line if multiple lines are assigned to the same address. However, this is not a problem.

In the same manner as other windows, a pop-up menu appears by clicking the right mouse button in the window while source codes are being read. At this time, you may select **[Add Breakpoint]** if a breakpoint can be put. If the breakpoints are already put, you may make a desired breakpoint enabled or disabled using **[Toggle Breakpoint]** or delete it using **[Delete Breakpoint]**.

10.4 Run step

Lines of the C-source code are run one-by-one in "Run step". If this line contains a function, the program step does not enter the function and stops immediately after the program step is returned from the function. Therefore, if it takes a long time to run a function or if a module has specific functions, the program is not stopped until such function is exited. Great care should be taken.

Additionally, pay special attention so that "Run step" at the source code level is not executed unless the window focus is within the source code window.

10.5 Run trace

Lines are run one-by-one in "Trace run". If the running line includes a function, this function is also run. However, if the function does not have relevant line number information (for example, standard library function), the program is not stopped in the function and is run to a next line included in the standard

library function after the program step returns from the function.

It is also absolutely necessary that the window focus is within the source window in the same manner as described in the above step, Run step.

10.6 Run area

In "Run area", clicking *[Here]* will run the program if a line, which is clicked when the pop-up menu appears, has relevant line information. At this time, SIM3022 stops running of the program when the program step reaches that line you have clicked.

APPENDIX A. Windows Virtual-Key Code List

The following table shows the symbolic constant names, hexadecimal values, and keyboard equivalents for the virtual-key codes used by the Microsoft Windows operating system. The codes are listed in numeric order.

Symbolic constant name	Value (hexadecimal)	Mouse or keyboard equivalent
VK_LBUTTON	01	Left mouse button
VK_RBUTTON	02	Right mouse button
VK_CANCEL	03	Control-break processing
VK_MBUTTON	04	Middle mouse button (three-button mouse)
-	05-07	Undefined
VK_BACK	08	BACKSPACE key
VK_TAB	09	TAB key
-	0A-0B	Undefined
VK_CLEAR	0C	CLEAR key
VK_RETURN	0D	ENTER key
-	0E-0F	Undefined
VK_SHIFT	10	SHIFT key
VK_CONTROL	11	CTRL key
VK_MENU	12	ALT key
VK_PAUSE	13	PAUSE key
VK_CAPITAL	14	CAPS LOCK key
-	15-19	Reserved for Kanji systems
-	1A	Undefined
VK_ESCAPE	1B	ESC key
-	1C-1F	Reserved for Kanji systems
VK_SPACE	20	SPACEBAR
VK_PRIOR	21	PAGE UP key
VK_NEXT	22	PAGE DOWN key
VK_END	23	END key
VK_HOME	24	HOME key
VK_LEFT	25	LEFT ARROW key
VK_UP	26	UP ARROW key
VK_RIGHT	27	RIGHT ARROW key
VK_DOWN	28	DOWN ARROW key
VK_SELECT	29	SELECT key
-	2A	Original equipment manufacturer (OEM) specific
VK_EXECUTE	2B	EXECUTE key
VK_SNAPSHOT	2C	PRINT SCREEN key for Windows 3.0 and later
VK_INSERT	2D	INS key
VK_DELETE	2E	DEL key
VK_HELP	2F	HELP key
VK_0	30	0 key
VK_1	31	1 key
VK_2	32	2 key
VK_3	33	3 key

VK_4	34	4 key
VK_5	35	5 key
VK_6	36	6 key
VK_7	37	7 key
VK_8	38	8 key
VK_9	39	9 key
-	3A-40	Undefined

Symbolic constant name	Value (hexadecimal)	Mouse or keyboard equivalent
VK_A	41	A key
VK_B	42	B key
VK_C	43	C key
VK_D	44	D key
VK_E	45	E key
VK_F	46	F key
VK_G	47	G key
VK_H	48	H key
VK_I	49	I key
VK_J	4A	J key
VK_K	4B	K key
VK_L	4C	L key
VK_M	4D	M key
VK_N	4E	N key
VK_O	4F	O key
VK_P	50	P key
VK_Q	51	Q key
VK_R	52	R key
VK_S	53	S key
VK_T	54	T key
VK_U	55	U key
VK_V	56	V key
VK_W	57	W key
VK_X	58	X key
VK_Y	59	Y key
VK_Z	5A	Z key
VK_LWIN	5B	Left Windows key (Microsoft Natural Keyboard)
VK_RWIN	5C	Right Windows key (Microsoft Natural Keyboard)
VK_APPS	5D	Applications key (Microsoft Natural Keyboard)
-	5E-5F	Undefined
VK_NUMPAD0	60	Numeric keypad 0 key
VK_NUMPAD1	61	Numeric keypad 1 key
VK_NUMPAD2	62	Numeric keypad 2 key
VK_NUMPAD3	63	Numeric keypad 3 key
VK_NUMPAD4	64	Numeric keypad 4 key
VK_NUMPAD5	65	Numeric keypad 5 key
VK_NUMPAD6	66	Numeric keypad 6 key

VK_NUMPAD7	67	Numeric keypad 7 key
VK_NUMPAD8	68	Numeric keypad 8 key
VK_NUMPAD9	69	Numeric keypad 9 key
VK_MULTIPLY	6A	Multiply key
VK_ADD	6B	Add key
VK_SEPARATOR	6C	Separator key
VK_SUBTRACT	6D	Subtract key
VK_DECIMAL	6E	Decimal key
VK_DIVIDE	6F	Divide key

Symbolic constant name	Value (hexadecimal)	Mouse or keyboard equivalent
VK_F1	70	F1 key
VK_F2	71	F2 key
VK_F3	72	F3 key
VK_F4	73	F4 key
VK_F5	74	F5 key
VK_F6	75	F6 key
VK_F7	76	F7 key
VK_F8	77	F8 key
VK_F9	78	F9 key
VK_F10	79	F10 key
VK_F11	7A	F11 key
VK_F12	7B	F12 key
VK_F13	7C	F13 key
VK_F14	7D	F14 key
VK_F15	7E	F15 key
VK_F16	7F	F16 key
VK_F17	80	F17 key
VK_F18	81	F18 key
VK_F19	82	F19 key
VK_F20	83	F20 key
VK_F21	84	F21 key
VK_F22	85	F22 key
VK_F23	86	F23 key
VK_F24	87	F24 key
-	88-8F	Unassigned
VK_NUMLOCK	90	NUM LOCK key
VK_SCROLL	91	SCROLL LOCK key
-	92-B9	Unassigned
-	BA-C0	OEM specific
-	C1-DA	Unassigned
-	DB-E4	OEM specific
-	E5	Unassigned
-	E6	OEM specific
-	E7-E8	Unassigned
-	E9-F5	OEM specific
VK_ATTN	F6	Attn key
VK_CRSEL	F7	CrSel key
VK_EXSEL	F8	ExSel key
VK_EREOF	F9	Erase EOF key
VK_PLAY	FA	Play key
VK_ZOOM	FB	Zoom key
VK_NONAME	FC	Reserved for future use.
VK_PA1	FD	PA1 key
VK_OEM_CLEAR	FE	Clear key