

PC

PARTNER-N64



PARTNER-N64

PARTNER-N64



N64

NINTENDO⁶⁴



Partner-N64PC

– User's Guide –

D.C.N. NUS-06-0091-001 Rev. A

This document contains confidential and proprietary information and is also protected under the copyright laws of the United States and foreign countries. No part of this document may be released, distributed, transmitted or reproduced in a any form or by any electronic or mechanical means, including information storage and retrieval systems, without the written permission of Nintendo.

© 1997 Nintendo of America Inc.

TM® and the "N" logo are trademarks of Nintendo of America Inc.

Partner-N64PC is a trademark of Kyoto Micro Computer Co., Ltd.

Pentium is a trademark of Intel Corporation.

Windows 95 is a trademark of Microsoft Corporation.

IBM is a trademark of International Business Machines Corporation.

Table of Contents

CHAPTER 1.....	9
Product Introduction and Overview.....	9
Partner-N64PC Features	9
Operating Environment	12
Hardware Requirements	12
Software Requirements	14
About this Manual	14
On-line Help	14
Partner-N64PC Specifications	14
Partner-N64PC Components.....	15
CHAPTER 2.....	17
Hardware and Software Installation	17
Equipment Configuration.....	17
Connecting the ROM Emulator Cartridge to the Modified N64 Control Deck...18	
ROM Emulator Cartridge Components.....	19
Installing the Dedicated Interface Board.....	20
Host PC I/O Addresses.....	20
Interface Board Switch Settings	21
Fuseless Breaker	24
Installing the Interface Board.....	24
Power On, Off Sequences	27
Software Installation	28
Game Pak Connector	29
CHAPTER 3.....	31
Partner-N64PC Basics	31
Basic Partner-N64PC Operation	31

CHAPTER 437

Startup	37
Configuration of Partner-N64PC.....	37
Configuration Items.....	37
Setting Command Line Startup Options	42
Related Files	45
Files Needed to Start Partner-N64PC.....	45
Files Created when Partner-N64PC is Exited	47
Starting Partner-N64PC.....	48
When Partner-N64 will not Start	49

CHAPTER 555

Window Commands.....	55
Window Configuration	55
Structural Elements	55
Screen Display	58
Menus.....	59
File Menu	60
Edit Menu.....	61
Search Menu	62
View Menu	63
Run Menu.....	64
Local Menu Windows.....	66
Window Menu	71
Settings Menu	73
Help Menu.....	74
Shortcut Keys	75
Shortcut Keys Common to Multiple Windows	75
Shortcut Keys Unique to Certain Windows	77
Mouse Operation	78
Mouse Operations Common to Multiple Windows	78
Mouse Operations Unique to a Window	80
Code Windows.....	81
Code Window Shortcut Keys.....	82
Code Window Local Menu	85
Code Window Mouse Operations	87

Command Window	88
Command Window Shortcut Keys	89
Command Window Local Menu.....	91
Memory Window	92
Memory Window Shortcut Keys.....	93
Memory Window Local Menu.....	94
Memory Window Mouse Operations	96
Register Window	96
Register Window Shortcut Keys.....	99
Register Window Local Menu.....	100
Register Window Mouse Operations	102
Stack Window.....	102
Stack Window Shortcut Keys	102
Stack Window Local Menu	103
Local Window	103
Local Window Shortcut Keys.....	104
Local Window Local Menus	105
Local Window Mouse Operations.....	106
Backtrace Window	106
Backtrace Window Shortcut Keys.....	107
Backtrace Window Local Menu.....	107
Backtrace Window Mouse Operations	108
Watch Window	108
Watch Window Shortcut Keys.....	109
Watch Window Local Menu	110
Watch Window Mouse Operations	111
Break Window	111
Break Window Shortcut Keys	112
Break Window Local Menus	112
Break Window Mouse Operations.....	114
Inspect Window.....	114
Inspect Window Shortcut Keys	114
Inspect Window Local Menu	115
Inspect Window Mouse Operations.....	116
Memo Window	117
Memo Window Shortcut Keys	118
Memo Window Local Menu	118

Toolbar	119
Toolbar Settings	120
Function of Each Button	120
Window Bar	124
Status Bar	126
Dialog Boxes	126
Open File Dialog Box	127
Specify Font Dialog Box	127
Toolbar Setup Dialog Box	128
Specify Colors Dialog Box	128
Search Character String Dialog Box	129
Inspect Setup Dialog Box	129
Watch Set Dialog Box	129
Module Dialog Box	130
Line Number Specification Dialog Box	131
Display Address Specification Dialog Box (Code)	131
Command History Dialog Box	132
Symbol Extension Dialog Box	132
Display Address Specification Dialog Box (Memory)	133
Data Set Dialog Box	133
Register Dialog Box	134
Software Breakpoint Set Dialog Box	134
Hardware Breakpoint Set Dialog Box	135

CHAPTER 6..... 137

Dialog commands	137
Data Expressions	137
Symbols Handled by Partner.....	137
Global Symbols.....	137
Local Symbols.....	139
Special Symbols.....	139
Numeric Values Used by Partner	140
Addresses.....	140
Line Numbers.....	141
Character Strings.....	142
Register Names	143
Operation Expressions.....	144
Data Expressions at the C-Language Level.....	146
C Expressions	147
C Variables	148
C Variable Scope	149
Constants.....	149
Operators	150
Expressions with Side Effects	152

CHAPTER 7..... 152

Command Reference.....	153
Command Conventions.....	153
Commands by Function	153

CHAPTER 8..... 167

SPECIAL COMMANDS AND THEIR USE.....	167
On-the-Fly Function	167
System Calls.....	168
Single Character Output	168
Single Character Key Input.....	169

Hardware Breakpoints..... 171
Using the Hardware Breakpoint 172
Controlling Emulation ROM..... 173
Transfer Command 175
Profile Function 177
Game Pak Connector..... 177

INDEX179

Chapter 1

Product Introduction and Overview

Partner-N64PC Features

The Partner-N64PC is a system for use in developing Nintendo N64 game programs. It provides a personal-computer-based development environment, using an IBM/PC compatible machine as a host.

Partner-N64PC, the Windows 95 version of the Partner system, allows efficient debugging. Designed to handle large amounts of debugging information, it employs Windows features such as MDI windows, menus, and scroll bars. In addition, the system's interface with the host computer is a dedicated parallel interface, for fast, smooth debugging.

Support for High-Memory-Capacity Emulation

Emulation ROM size: 32 MB (Model 11)

Software Break Function

Up to 15 breakpoints can be set in the program code area.

Hardware Break Function

In addition to breakpoints in the program code area, breakpoints can be set when accessing I/O and data areas.

On-the-Fly Function

Commands can be accepted while the user program is running. This allows areas such as memory, I/O, and registers to be freely referenced and changed. This is also true for commands related to real-time trace memory.

Real-Time Count Function

The time from start of execution of the user program until it is stopped can be measured in 1 μ s units up to 4,294 seconds.

Break on Illegal ROM Area Access

The user program can be made to break with memory writes to a ROM area or with access (read/write) to a AD16 bus space other than a ROM area.

Source Level Debugging (when using exeGCC(N64))

Compiling with exeGCC(N64) allows debugging to be performed entirely at the assembler source level. With a single key, a breakpoint can be specified or a program can be executed up to the cursor position while comments are viewed, eliminating the need for burdensome symbolic debugging. Symbolic debugging can also be performed for loaded objects that have been created using methods such as IS-ASM.

Inspect Function

With a single key, any variable can be referenced and changed in a format that conforms to the data structure of the variable. This can be accomplished by simply moving the cursor to the assembler source variable that you wish to reference or change.

Macro Function

A powerful macro function (language) with a C-like control structure (if, for, while, do, break, etc.) is provided. This allows efficient debugging by means such as combining multiple commands to define new commands or by combining commands with breakpoints.

Shell Function

This features a history function that records 1,500 characters of entered commands. These are saved at the end of debugging session and automatically loaded at restart. The shell function also includes tools that are useful during command entry, such as line edit, command search, and global symbol search. When the same commands are entered repeatedly, previous commands can easily be recalled.

Improved Help Function

This is provided in help menu format. The help display for entry errors can be called up with a single mouse click.

System Call Function

Substitute input/output functions for single character input/output make it possible to load PC keyboard data or output text to the PC screen from the user program.

Profile Function

In the execution of a user program, the execution time for each subroutine can be totaled and displayed.

Compatibility with Partner for Work Stations

Keyboard operation of the Code and Command windows (including function keys) is virtually the same as that for Partner NW, allowing seamless transition from this version of Partner.

Strongest Data Reference/Change and Link Functions

The functionality of Partner's functions for inspecting and modifying data has been vastly improved. For example, the Inspect window can be opened and constant data can be easily viewed and modified simply by double clicking on a constant viewed in the Source window. Constant data can be similarly viewed or modified in the Watch and Local windows. In addition, the mouse can be used to modify data in the Register window and Memory windows.

Tool Bars

Commonly used functions * such as loading programs to be debugged, setting breakpoints, running programs, and inspecting constants * can be registered in a tool bar and easily executed simply by clicking on a tool bar button.

User Customization Function

Features including the tool bars, display fonts, display colors, and window layout can be freely customized by the user. In addition, up to three different window layouts can be registered.

Operating Environment

The following describes the operating environment required to begin the Partner-N64PC setup procedure. Before starting the setup program, please refer to this section to confirm that you have the appropriate hardware and software.

Hardware Requirements

Personal Computer and Memory

Operating system:	Windows 95
CPU:	Pentium class
RAM:	32 MB minimum
Expansion bus:	16-bit ISA slot

Display

Windows 95-compatible
800 x 600 graphics resolution

Hard Disk

At least 200 MB of available hard disk space required for setup.

Mouse

Windows 95-compatible mouse (required for full use of Partner functions).

CD-ROM Drive

Partner-ET Dedicated Interface Board

The interface board is required to connect the ROM Emulator to the personal computer. See **Hardware Setup** (pg. 17) or the **N64 Development Systems Guide** regarding setup and other procedures.

Parallel Interface Cable

Required to connect the Partner-N64PC Dedicated Interface Board to the ROM Emulator.

Software Requirements

Windows 95

Before starting the Partner-N64PC Setup program, ensure that Windows 95 is running in the auto-start configuration.

About this Manual

The following typographical conventions are used throughout this manual.

[Menu]	Menu names are shown inside square brackets ([]).
[Menu]-[Command]	Command names are shown inside square brackets ([]) following the name of the menu in which the command is included.
[Dialog]	Dialog box names are shown inside square brackets ([]).
<Button>	The names of the various buttons are shown inside angle brackets (< >).

On-line Help

The on-line help available in Partner-N64PC explains system functions and provides operating instructions on the screen. To display on-line help, press the HELP key (the End key on IBM/PC compatible keyboards), execute the HELP or [HELP]-[Index] command, or press the <Help> button.

Partner-N64PC Specifications

Host Interface

Dedicated parallel interface

IBM/PC compatible machines: ISA bus

Power Supply

Power is supplied to the Partner unit by the host PC.

Current Consumption:

Dedicated parallel interface: 200mA

ROM Emulator Cartridge: 400mA

Service Environment

Ambient Temperature: 5°C to 35°C

Ambient Humidity: 35% to 85% RH (without condensation)

Breaks

Software breakpoints: 15

Hardware breakpoints: 1

A user program can be halted by pressing CTRL+ALT on IBM/PC compatible machines

Real-Time Count

The speed of user program execution can be measured in 1 μ s units up to 4,296 seconds.

Partner-N64PC Components

- Partner-N64PC User's Manual 1
- N64 Development Systems Installation Manual 1
- Modified N64 Control Deck 1
- N64 Controller 1
- Dedicated Interface Board 1
- 40-pin Flat Cable 1
- N64 ROM Emulator Cartridge 1
- N64 AC Adapter 1
- AV cable 1
- CD-ROM (N64 environment, Partner-N64PC debugger software) 2

Chapter 2

Hardware and Software Installation

Equipment Configuration

The Partner-N64PC System is shown in Figure 1. The ROM Emulator Cartridge is inserted into the Game Pak slot in the modified N64 Control Deck, and is connected to the personal computer by a dedicated interface board (parallel interface). In addition, a backup memory pack or Game Pak can be inserted into the connector on the Partner-N64 unit.



Figure 1. Partner-N64PC System

There is no backup memory built into the ROM Emulator Cartridge. When developing games with backup memory, insert a Game Pak or memory board with the appropriate memory configuration into the Game Pak connector on the ROM Emulator Cartridge.

There are several specifications for backup memory and backup memory cartridges, which vary in capacity.

Connecting the ROM Emulator Cartridge to the Modified N64 Control Deck

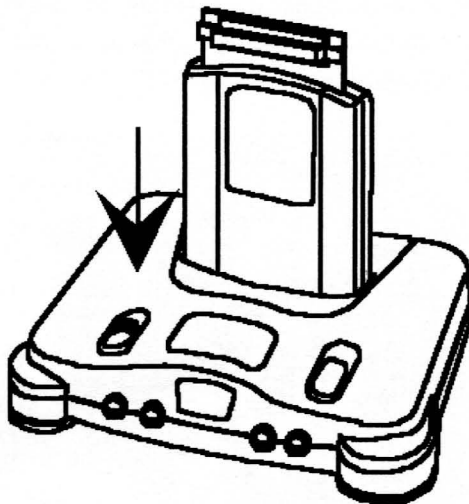


Figure 2: Connecting the ROM Emulator Cartridge to the Modified N64 Control Deck

The ROM Emulator Cartridge is connected to the modified N64 Control Deck as shown in Figure 2. First confirm that power to the modified N64 Control Deck is turned off. Then fully insert the ROM Emulator Cartridge into the Game Pak slot.

Do not insert or remove the ROM Emulator Cartridge while power to the modified N64 Control Deck is on.

Only a modified N64 Control Deck can be used with this product. DO NOT connect this product to a retail N64 Control Deck.

ROM Emulator Cartridge Components

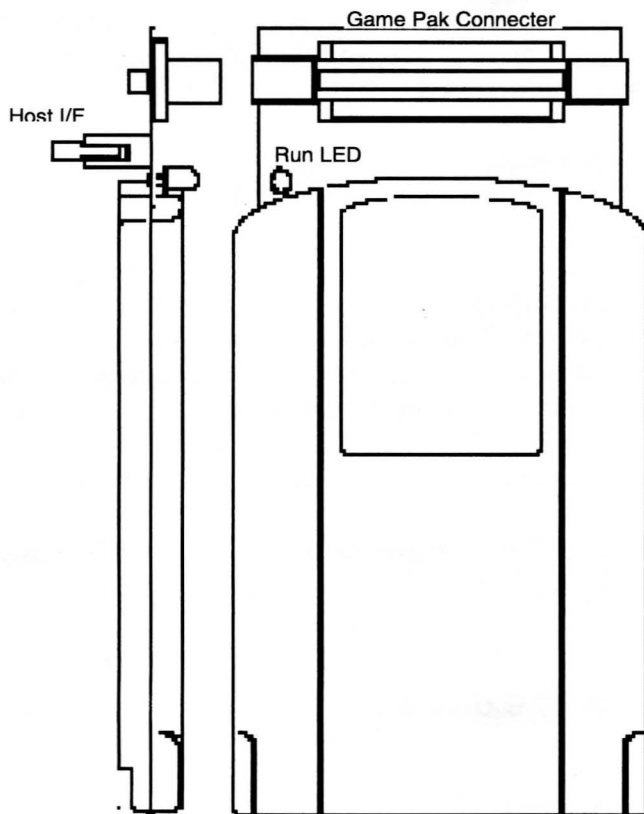


Figure 3: ROM Emulator Cartridge Top View

The Partner-N64 has an LED and connectors, as shown in Figure 3 above. These parts are described below.

HOST I/F Connector

This is used to connect the unit to the dedicated interface board installed in the host PC, by means of a 40-pin flat cable.

Do not connect the unit to any board other than the dedicated interface board included with the Partner-N64PC.

RUN Indication LED

This indicates that the user program is being executed. This LED is lit while the user program is running and extinguished when execution is stopped.

Game Pak Connector

This connector enables a backup memory Pak or Game Pak to be connected.

There is no backup memory built into the ROM Emulator Cartridge. When developing games with backup memory, insert a Game Pak or memory board with the appropriate memory configuration into the Game Pak connector on the ROM Emulator Cartridge. There are several types of backup memory and backup memory cartridges, which vary in capacity.

A normal Game Pak may also be inserted into this connector for data transfer. Refer to the Z command or TR command in the debugger section of this manual.

Installing the Dedicated Interface Board

Host PC I/O Addresses

Before installing the interface board, use System Setup in the host PC's Control Panel to determine which I/O port addresses are available. Eight addresses are required for system

installation. The default settings on the interface board at the time of shipment are addresses 320-327.

Once the addresses have been determined, turn off power to the host PC. Ensure that the modified N64 Control Deck's power is also off.

Interface Board Switch Settings

There are three DIP switches on the interface board that can be set by the user: SW1, SW2 and SW3. Set these switches to match the configuration of the system being used. Their locations are shown in the figure below.

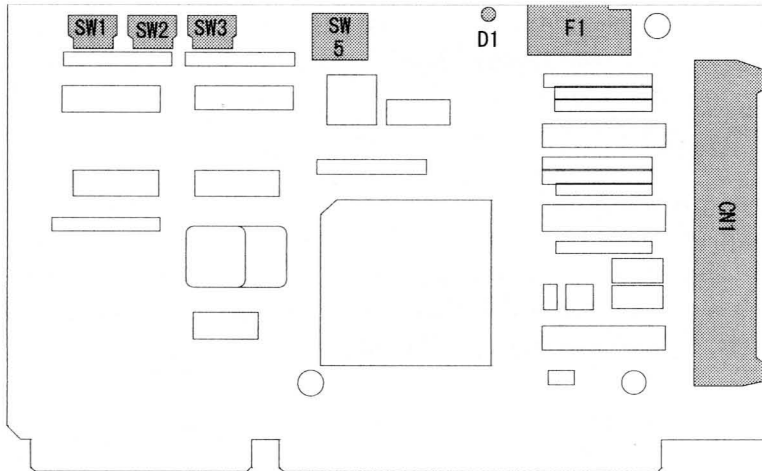


Figure 4: Dedicated parallel interface board

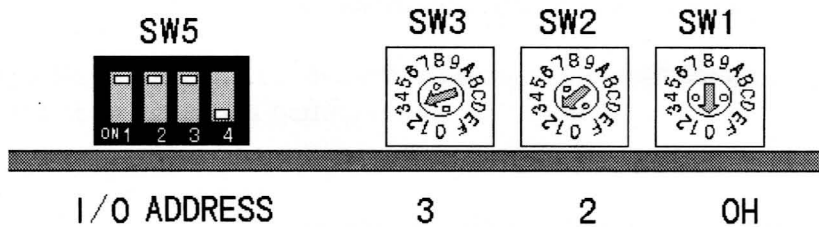


Figure 5: Default switch settings

SW1, SW2, SW3

These switches set the I/O address used for communication between the Partner-N64 and the personal computer. Specify bits A3 through A11 of an empty address on the personal computer being used. The 8 bytes from the specified I/O address are used by the board (i.e. 320-327 for the default setting of 320).

SW1 refers to A3, SW2 refers to A4 through A7 and SW3 refers to A8 through A11. The value that can be set for SW1 is 0 or 8. Any other value are rounded to 0 or 8. (Bits A0 through A2 are ignored by SW1.) In addition, the upper 4 bits of the address (A12-A15) are fixed at 0 (full 16-bit decoding).

The I/O address setting at the time the product is shipped is 320H.

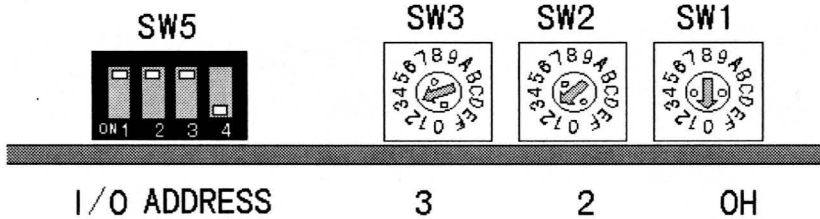


Figure 6: Setting the I/O address

Switch Name	Address	Range	Default Setting
SW3	A9-A8	0-F	3
SW2	A7-A4	0-F	2
SW1	A3-A0	0 or 8	0

Table 1: Switch Settings

The I/O address settings that can normally be set by the user in an IBM-PC compatible machine are listed below. Set the I/O address to an address that will not conflict with another board in this range. Specify this I/O address in the PORT ADDRESS field of the environment file (EPTN64.CFG).

SW5

Use this switch in its default state. Do not change its settings. When shipped, sw5-1 through 3 are OFF and sw5-4 is ON.

Address
100h-1EFH
220h-26FH
280h-2AFH
300h-35FH
3E0h-3EFH

Table 2: Usable I/O Map

Fuseless Breaker

When not connected to an external power supply, power is supplied to the Partner-N64PC from the personal computer through a fuseless breaker. In the event of an overcurrent condition, the white button on the breaker pops out and the current is interrupted. Once the cause has been eliminated, press the white button back in. The LED on the board is on when the breaker is in the normal state and off when an overcurrent has occurred.

Installing the Interface Board

Once the interface board switches have been set, install the board in an available 16-bit ISA expansion slot in the personal computer according to the following procedure. Appropriate ESD precautions must be observed while performing these steps.

- 1) Check the switch settings on the interface board. If they are not set to match the personal computer being set up, check **Interface Board Switch Settings** (pg. 21) and change the settings.
- 2) Turn off the power to the computer, and remove the chassis cover. (Refer to the manual for the personal computer for a description of how to remove the chassis cover.) Then remove the expansion slot cover.

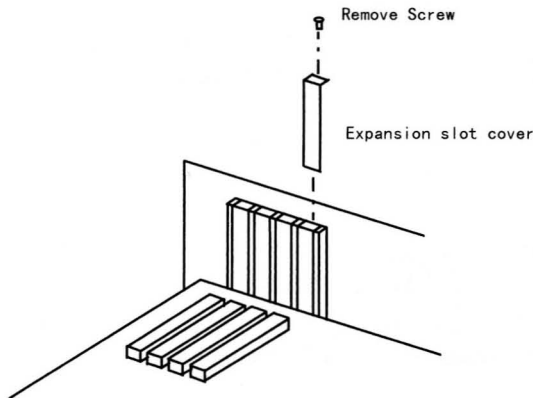


Figure 7: Removing the expansion slot cover

- 3) Install the interface board. Be sure to secure the board with a screw, or the board may be pulled loose when the interface cable is removed. Use the screw provided with the host PC for this purpose.

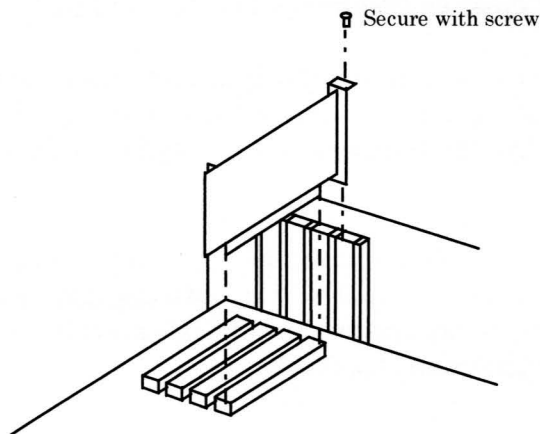


Figure 8: Installing the interface board

- 4) Connect one end of the 40-pin flat cable to the 40-pin connector on the interface board. Connect the other end of the cable to the 40-pin connector on the back of the ROM Emulator Cartridge.
- 5) Plug the ROM Emulator Cartridge into the Game Pak slot of the modified N64 Control Deck. When the ROM Emulator Cartridge is properly inserted, the 40-pin cable connector will face the rear of the modified N64 Control Deck.
- 6) Connect the included N64 AC Adapter to the modified N64 Control Deck, and connect the included standard N64 Controller to socket no. 1 of the Control Deck.

- 7) Turn on power first to the PC, then to the N64 Control Deck. Check whether the personal computer boots up. If it doesn't, check the DIP switch settings, and confirm that the board has been properly installed. If no installation abnormalities can be found, contact the Nintendo support group at (206) 861-2715.
- 8) After confirming that the personal computer boots up normally, replace the chassis cover.
- 9) After installing the Partner-N64PC software (see Software Installation, pg. 28), start Partner-N64PC. If the debugger starts, everything is normal.

If the following message is displayed, either the power to the modified N64 Control Deck is not turned on, or there is a faulty connection between the ISA interface and the modified N64.

Waiting N64 Power Up

If the following message is displayed, check the DIP switches and connections.

Hardware abnormality. Check DIP switches, cables, etc.

If the DIP switch settings have been changed, the environment (CFG) file must be also changed before starting EPTN64. Refer to the manual section that describes environment (CFG) file settings.

Power On, Off Sequences

The sequence for turning power on is as follows:

- 1) Personal Computer
- 2) Modified N64 Control Deck

The power-off sequence is the opposite of the power-on sequence:

- 1) Modified N64 Control Deck
- 2) Personal Computer

Not following these sequences may damage the interface and/or the modified N64 Control Deck. In some cases, the personal computer cannot be started if power to the N64 is turned on first. If this happens, turn off power to all units and start over.

Software Installation

Start Windows 95 in the default (auto-play) configuration. Place either of the two CDs in the host computer's CD-ROM drive. After several seconds, installation will begin automatically. The installation program will walk the user step-by-step through the software installation. When the installation is complete, eject the CD. Repeat the above procedure with the second CD, then store the CDs in a safe place.

Game Pak Connector

A Game Pak or backup RAM board can be inserted into the Game Pak connector on the ROM Emulator Cartridge.

When inserting or removing a Game Pak or other device, be sure that power to the personal computer and modified N64 Control Deck is off. If a device is inadvertently inserted while the power is on, initialize the debugger with the `INIT` command.

This connector cannot be accessed (read) from the CPU while the emulation ROM in the debugger unit is enabled (i.e., when the ROM size in the `ZROM` command is non-zero). To access data on the inserted cartridge, specify 0 as the emulation ROM size with the `Z` command. In this case, the ROM access speed must be set to the same speed as the access speed of the game program. For details on access speeds, refer to **Environment File Settings (ROM SPEED/MONITOR SPEED)**.

Game programs can also be run on this connector, but software breakpoints cannot be set. Software breakpoints can be set if the program in the Game Pak is executed after it has been transferred to the emulation memory inside the debugger by the method described below.

In addition, data can be transferred between the Game Pak ROM, the emulation memory installed in the debugger, and a file. For details on data transfer, refer to the TR command in the on-line help.

Chapter 3

Partner-N64PC Basics

Basic Partner-N64PC Operation

The basic operation sequence for Partner-N64PC is shown below.

1) Startup

Start Partner-N64PC (Figure 9).

The screenshot shows the Partner-N64PC startup screen. The main window displays assembly code with addresses, hex values, mnemonics, and registers. Below the code, there is a status bar with emulator information.

Address	Hex	Mnemonic	Registers
80000000	03600008	JR	r27
80000004	00000000	NOP	
80000008	00000000	NOP	
8000000C	00000000	NOP	
80000010	00000000	NOP	
80000014	01244824	AND	r9, r9, r10
80000018	AC290000	SW	r9, 0(r1)
8000001C	3C08A460	LUI	r8, A460
80000020	8D080010	LW	r8, 10(r8)
80000024	31080002	ANDI	r8, r8, 2
80000028	550FFFD	ENEL	r8, r0, 80000020
8000002C	3C08A460	LUI	r8, A460
80000030	24081000	ADDIU	r8, r0, 1000
80000034	010E4020	ADD	r8, r8, r11
80000038	010A4024	AND	r8, r8, r10
8000003C	3C01A460	LUI	r1, A460

Below the code, the status bar displays the following information:

```

PARTNER-N64/W32 Ver 1.02 Copyright (c) 1996 Kyoto Micro Compu
Monitor Code address=0xBFF00000 size=0x2000(8192)
Emulation ROM size= 16 MB
  
```

Figure 9: Partner-N64PC Startup Screen

2) Load program

Load the program to be debugged (Figure 10).

See also:



Button Functions (pg. 121), **Load Program** (pg. 154)

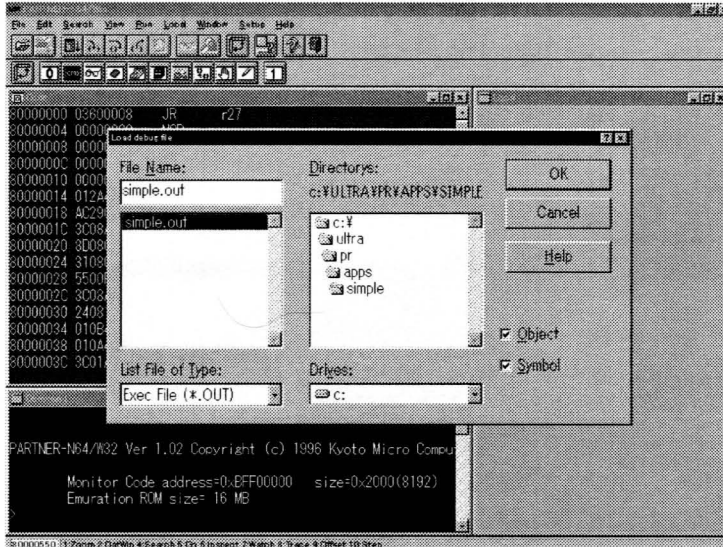


Figure 10: Load program to be debugged

3) Set breakpoints

Click the mouse in the line-number area or address area of the Code window to set breakpoints (Figure 11).

See also:

BP command, (p. 87), (p. 87), [Break]-[Set Break], **Break Window Local Menu** (p.112)

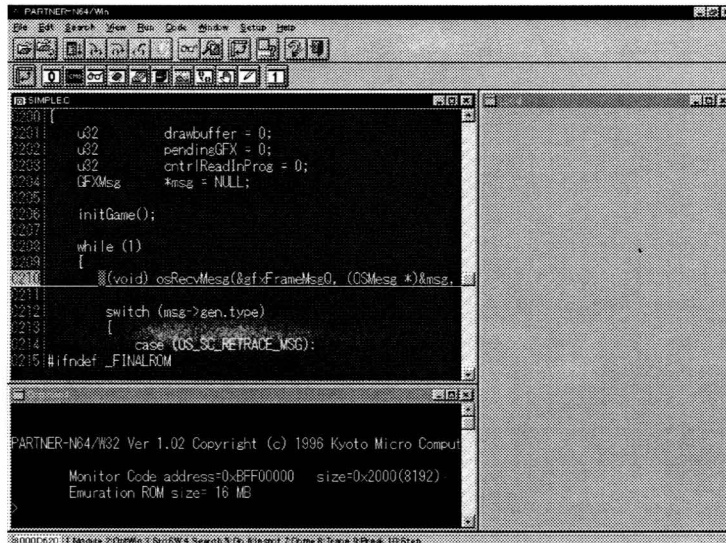


Figure 11: Set Breakpoints

4) Run program

Start execution of the program. The program will stop at the breakpoints set in step 3) or can be forcibly stopped with ESC.

See also:



buttons (pg. 121), G command, ESC command (pg. 64)

5) Inspect variables

Display the variable to be referenced (or changed) in the Inspect window by double clicking on it. (Figure 12)

See also:



button (pg. 121), Code Window Shortcut Keys (pg. 82), INS command (pg. 87)

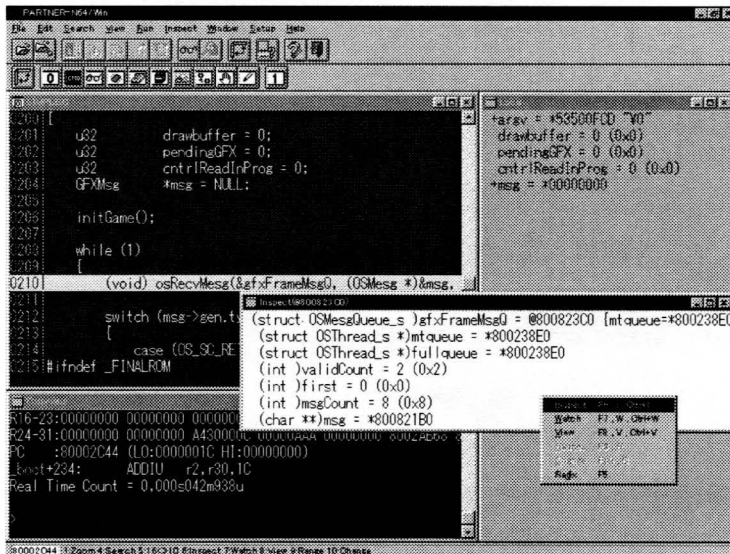


Figure 12: Inspect Display of IntLoc3

6) Set variable in Watch

Set the variables that you wish to monitor in the Watch window (Figure 13).

See also:



button (pg. 121), Code Window Shortcut Keys (pg. 82), w command (pg. 82),

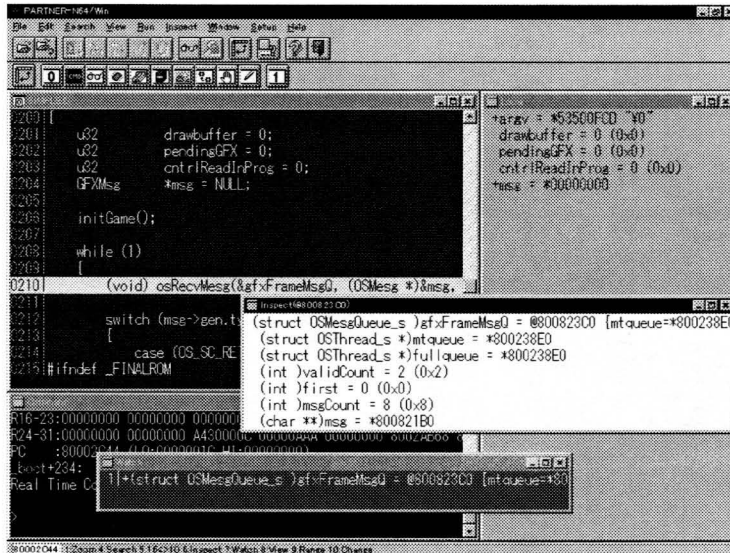


Figure13: Register IntLoc3 in the Watch Window

7) Exit Partner-N64PC

Exit Partner-N64PC (Figure 14).

See also:



button (pg. 121), [File]-[Exit] (pg. 60), Q/Exit Command

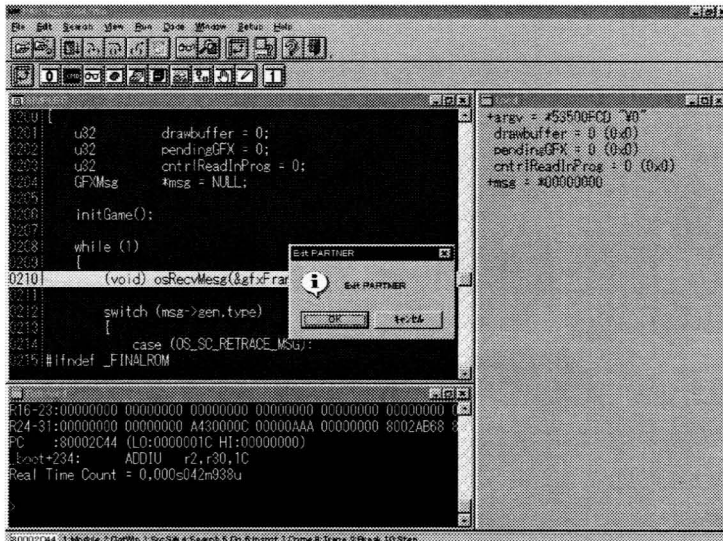


Figure14: Confirmation Box to Exit Partner-N64PC

Chapter 4

Startup

When installation of the Partner-N64PC was completed, a Partner-N64PC group was created in the Program Manager or Start menu.

To start Partner-N64PC, the Partner-N64PC environment file must be set up.

Configuration of Partner-N64PC

Configuration Items

The following items can be set in the Partner-N64PC environment (CFG) file. Key words are noted inside ().

- Port address of interface board (PORT ADDRESS)
- Select vector used by monitor (BREAK VECTOR)
- Specify memory area accessible by Partner commands (MAP)
- Set ROM access speed (ROM SPEED, MONITOR SPEED)
- Set monitor usage area (MONITOR AREA)
- Set debugging argument area (DEBUG PARAMETER)

Port Address of Interface Board (PORT ADDRESS)

The interface board uses 8 bytes of I/O space as an input/output port. This port address can be changed with switches 1, 2 and 3. Refer to **Installing the Dedicated Interface Board** (pg. 20). If the port address has been changed, be sure to change this setting in the CFG file. (**Setting at the time of shipping is 320H**)

Example for I/O address of 320H:

PORT ADDRESS 320

Select Vector used by Monitor (BREAK VECTOR)

Specify the vector to be used by the monitor. Always set this item to USER0 for Partner-N64PC.

Specify memory area accessible by Partner commands (MAP)

In hexadecimal notation, specify the memory areas accessible by Partner-N64PC commands or internal processing (up to 20 areas) .

Syntax MAP *start address, end address*

Memory space from the *start address* to the *end address* can be accessed by Partner-N64PC commands. Up to 20 areas can be specified in the MAP range. When specifying multiple ranges, enter these values on multiple lines. (By default, this is set to 0~0xBFFFFFFF)

In some systems, continuous operation is prevented by generation of a CPU exception when an illegal memory area is accessed. Therefore, specify an memory area accessible to Partner-N64PC.

If memory not been specified by the MAP field is accessed with a Partner command, a message like the following is displayed, and execution of the command is interrupted.

Bus/address/map error

Access to areas other than those specified by MAP is restricted only for Partner commands; such access is unrestricted for the user program.

Set ROM Access Speed (ROM SPEED, MONITOR SPEED)

Set the ROM access speed with this field. Use ROM SPEED to set the ROM access speed during game program execution. The speed is ultimately determined by the performance of the mask ROM. This field is normally set to 80371240. Do not change this value unless instructed to do so by Nintendo.

Use MONITOR SPEED to set the ROM access speed during monitor execution. This value is the ROM access speed during a break in the user program and is set to a faster value than that of ROM SPEED. MONITOR speed is normally set to 80370902. If you wish the ROM speed during monitor execution to match the speed during game program execution, set MONITOR SPEED to the same value as ROM SPEED.

Set Monitor Usage Area (MONITOR AREA)

This field specifies the area used by the monitor program. The default ROM setting assigns the monitor program to a ROM area (PI area: 0xBFF00000-0xBFF01FFF). Because this memory area is not normally used by the game program, this imposes no memory restrictions on game program development. Moreover, the monitor area cannot be destroyed by the game program. When the ROM area is specified by the user, however, Partner-N64PC commands may be slowed, because the ROM areas may not be cached and ROM access may be slow. This is particularly true of commands that access a large amount of memory, such as the RD and WR commands.

Specifying RAM assigns the monitor program to the RAM area (0x803FE00-0x803FFFFFF). This memory area corresponds with the last 8 bytes of RAM space for N64. Consequently, when RAM is specified, the game program cannot use this RAM space. When RAM is specified, Partner-N64PC commands may be executed at high speed, because high-speed RAM space may be cached. This is particularly true of commands that access a large amount of memory, such as the RD and WR commands.

Set Debugging Argument Area (DEBUG PARAMETER)

Set the argument area for debugging in this field.

Syntax DEBUG PARAMETER <ON / OFF>{, *argument address*}

When set to OFF, the debug argument function is disabled.

When set to ON, the debugging argument function is enabled. In addition, a destination address can be specified to indicate the memory area where the debugging arguments are stored (ROM area only; default: 0xB0FFB000).

When this field is set to ON, load the debugging program with the Partner-N64PC L command as follows:

```
>L onetri -T<return>
```

The debug arguments are preceded by a dash (-). In the above example,

-T is a debug argument. These arguments are automatically stored in the memory area specified in the ARGUMENT ADDRESS field. Referring to this area with the game program enables optional processing by the game program.

Example of EPTN64.CFG:

Port address

```
PORT ADDRESS          320
```

Select vector used by monitor

```
BREAK VECTOR          USER0
```

Memory area accessible by Partner commands (up to 20 areas)

```
EX.                   MAP          10000000, 1000FFFF
                       MAP          80000000, 8FFFFFFF
                       MAP          00000000, 8FFFFFFF
```

Set ROM access speed during game program execution

```
ROM SPEED             80371240
```

Set ROM access speed during monitor execution

```
MONITOR SPEED        80370902
```

Set monitor usage area (ROM, RAM)

```
MONITOR AREA          ROM
```

Set debug argument constants

Memory area accessible by Partner commands (up to 20 areas)

EX.	DEBUG PARAMETER	OFF
	DEBUG PARAMETER	ON, B0FFB000
	DEBUG PARAMETER	ON, B0FFB000

Setting Command Line Startup Options

The startup options for Partner-N64PC are shown below.

-B Option

Specify the sizes of the debug information and macro areas. A large amount of information is stored in the debug information area, including global and local symbols, line number information, and function and variable type information.

`-B [size D][,size M]`

size D

Size (in 16 KB units) of the area in which debug information is registered. The default size is 512 KB.

size M

Size (in 1 KB units) of the area in which macro definition information is stored. The default size is 2 KB.

-D Option

Specify the current Partner-N64PC directory. The config file (EPTN64.CFG), monitor file (MON_N64.EPT) and Windows initial settings file (EPTN64.INI) must all exist in the current directory. See **Files Needed to Start Partner-N64PC**.

`-D directory`

directory is the current directory when Partner-N64PC is started. If this option is omitted, the directory is that specified in [Working Directory] of the [Program Item Properties] dialog box or in the directory containing the executable program specified in [Command Line], in that order.

-E Option

Specify the default extension for source files to be displayed/referenced in the Code window.

-E extension

extension is the default extension of the source files. Specify the extension portion of the filename, without the period (.). If this option is omitted, the default extension *file.C* or *file.ASM* is used.

-SD Option

Specify the directory that contains the source file to be referenced in the Code window. Separate multiple directory specifications with a semicolon (;).

-SD directory

directory is the directory containing the source files. If this option is omitted, the directory is that specified with the *-D* option.

-TAB Option

Specify the size of the tabs used when displaying source code in the Code window. This function makes it easy to display files in which the tab size has been changed.

-TAB tab size

Tab size is the number specified by *tab size*. If this option is omitted, a tab size of 8 is used.

-! Option

Starts Partner-N64PC regardless of monitor program operation. Use this to forcibly start Partner-N64PC when display of a message box prevents startup. When Partner-N64PC is forcibly started, only commands that display or change the contents of emulation ROM can be used. If other memory space or I/O space is displayed, only zeroes (0) are displayed. Commands that result in execution, such as the G, T, and P commands, cannot be used in this case.

This function is used to reference the monitor program with the U command when abnormal operation of the monitor program prevents Partner-N64PC startup. It is also used to write and execute a simple test program in emulation ROM. To execute a program written to emulation ROM, reset the target system. To do this, press the reset switch on the target, or use the RESET command.

Related Files

Files used by Partner-N64PC are shown below.

Files Needed to Start Partner-N64PC

The files needed when Partner-N64PC starts are the config file (.CFG), the monitor program (.EPT) and the Windows initial settings file (.INI). The INIT.MCR file will also be loaded if it exists. This file is discussed below in **Files Created when Partner-N64PC is Exited**.

The current directory and the directory containing the WPTN64.EXE file are searched in that order, and each file is loaded from the directory where it resides.

Config File (EPTN64.CFG)

The config file specifies the resources used by Partner-N64. Partner-N64PC uses a file that is identical to the config file used in the network version of Partner (Partner-N64NW). For a detailed description of the config file contents, refer to **Configuration of Partner-N64PC** (pg. 37).

Monitor Program (MON_N64.EPT)

The monitor program resides in the RAM area of the ROM Emulator Cartridge. Partner-N64PC uses a file that is identical to that used in the network version of Partner (Partner-N64NW).

Hardware Definition File (N64DEB1.SEQ)

This file defines the Partner-N64 hardware. Partner-N64PC uses a file that is identical to that used in the network version of Partner (Partner-N64NW).

Windows Initial Settings File (PTXX.INI)

This file sets the initial state of Partner-N64PC. It stores the window arrangement, font, color specifications, tool bar settings, breakpoint settings, and startup options.

Autoexec File (INIT.MCR)

The autoexec file automatically accesses and executes files when Partner-N64PC is started. It is nearly equivalent to the MS-DOS AUTOEXEC.BAT file. If pre-processing necessary for debugging (e.g., loading of user programs) is entered in this file, this processing can be automatically executed when Partner-N64PC starts.

Files Created when Partner-N64PC is Exited

Partner-N64PC creates several files when it is terminated. These files are stored in the directory from which the program was loaded during startup.

Contents of Memory Window

The contents of the Memory window are stored in the memo.pt file.

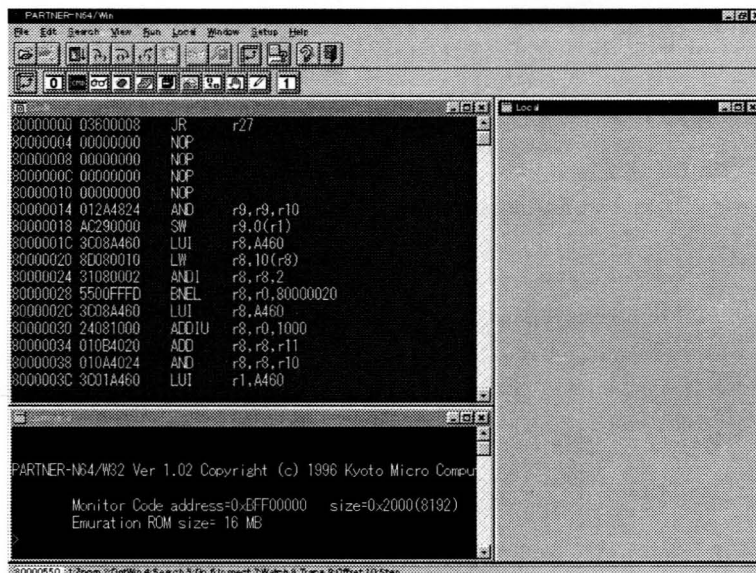
Command History

The contents of the Command History are stored in the PTN64.dat file.

Starting Partner-N64PC

Partner-N64PC can be started once the CFG file has been edited. To start, double click on the Partner-N64 icon. Command line entry is also possible.

When Partner-N64PC starts, the screen shown in Figure 15 is displayed.



```

00000000 03600008 JR      r27
00000004 00000000 NOP
00000008 00000000 NOP
0000000C 00000000 NOP
00000010 00000000 NOP
00000014 012A4824 AND    r9,r9,r10
00000018 AC290000 SW     r9,0(r1)
0000001C 3C08A460 LUI   r8,A460
00000020 8D080010 LW    r8,10(r8)
00000024 31080002 ANDI  r8,r8,2
00000028 5500FFD0 ENEL  r8,r0,80000020
0000002C 3C08A460 LUI   r8,A460
00000030 24081000 ADDIU  r8,r0,1000
00000034 01084020 ADD   r8,r8,r11
00000038 010A4024 AND   r8,r8,r10
0000003C 3C01A460 LUI   r1,A460
  
```

PARTNER-N64/W32 Ver 1.02 Copyright (c) 1996 Kyoto Micro Comput
 Monitor Code address=0xEFF00000 size=0x2000(8192)
 Emulation ROM size= 16 MB

Figure 15: Partner-N64PC Startup Screen

When Partner-N64 will not Start

If any of the message boxes shown in figures 16 through 22 is displayed during startup, refer to **Trouble Shooting**, **Customizing the Monitor Program**, **Configuring Partner-ET**, or **Error Messages at Startup** in the **Installation** portion of this manual.

Error Messages Related to the Environment (CFG) File

Edit the CFG file using an editor for the user's target system. The CFG file is loaded when the debugger software (WPTN64.EXE) is started. If there is an error or a conflict in the specified CFG file, one of the following messages is displayed. Re-edit the CFG file.

Wrong configuration file (EPTN64.CFG) format

This message is displayed with the line number of the field in the CFG file that contains an error . Correct the specified field.

Not found Configuration file (WEPTN64.CFG)

The CFG file cannot be found in the current directory or in the directory containing WPTN64.EXE. Create EPTN64.CFG in the current directory or in the directory containing WEPTN64.EXE.

Error Messages at Startup

The following messages is displayed when there is an error during startup. Retry as instructed by the corresponding message.

Please turn ON Nintendo 64 power switch

Power is not turned on to the modified Nintendo 64 Control Deck. Turn on power to the Control Deck.

Hardware trouble. Please check DIPSW, cable, connector and etc.

There is an abnormality in the connection between the interface board and the ROM Emulator Cartridge. This message is displayed when the PC is unable to recognize the ROM Emulator Cartridge. Check the following items.

- 1) The switch settings and/or I/O port address settings on the interface board or the port address setting (PORT ADDRESS) in the environment (CFG) file. This message is displayed if PORT ADDRESS does not match the I/O address setting on the interface board.
- 2) The I/O port address used on the interface board. It may conflict with another board in an expansion slot in the PC.
- 3) The dedicated interface board. It may not be properly connected to the ROM Emulator Cartridge by the 40-pin flat cable.
- 4) The fuseless breaker on the interface board. If power is supplied from the PC, and if the breaker button has popped out, an overcurrent surge may have occurred. Check for possible causes of such an overcurrent, then depress the white button again.

Monitor program (MON_N64.EPT) not found!

The monitor program file (MON_N64.EPT) cannot be found in the current directory or in the directory containing WPTN64.EXE. Confirm that WEPTN64.EPT exists. First the current directory and then the directory containing the WPTN64.EXE file are searched for MON_N64.EPT when WEPTN64 is started.

Not found Hardware Seq. Data (n64deb1.seq)

The hardware definition file, N64DEB1.SEQ, must be in the current directory or the directory containing the WPTN64.EXE file in order to start the debugger. Confirm that N64DEB1.SEQ exists. First the current directory and then the directory containing the WPTN64.EXE file are searched for N64DEB1.SEQ when WEPTN64 is started.

Wait Monitor (CPU) RUN!

The monitor program cannot start normally. When this message is displayed and processing will not proceed, confirm that the Partner-N64PC is properly connected. Also confirm that the N64 is a modified unit. Partner-N64PC will not start properly with a retail N64 Control Deck.

This message may also be displayed while the game program is loading. This happens when there is an error in the game program itself or when an incorrect method was used in creating the program.

Hardware (FPGA) initialize error

Initialization of the debugger hardware has failed. Confirm that the Partner-N64PC is properly connected. Also confirm that the I/O address has been properly set.

Abnormal environment file (EPTN64.CFG) format

Environment file (EPTN64.CFG) not found

Error in initial stack setting

Abnormal monitor file

These messages concern errors in the environment (CFG) file. Refer to **Configuration of Partner-N64PC** (pg. 37) and correct the CFG file.

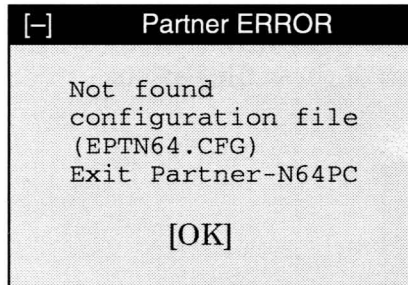


Figure 16

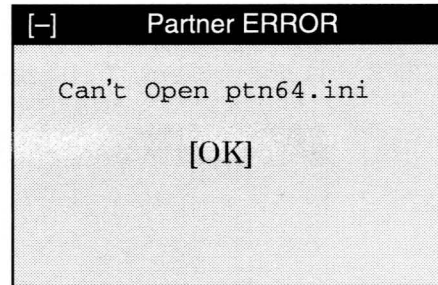


Figure 17



Figure 18

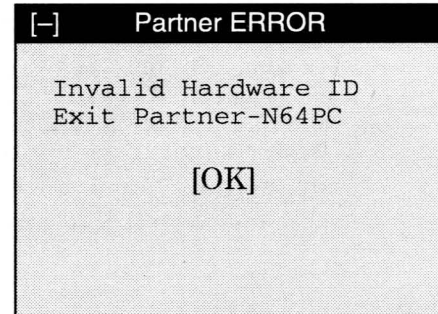


Figure 19

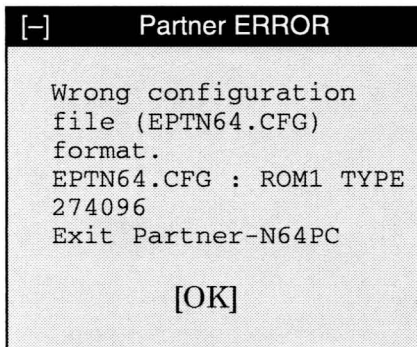


Figure 20

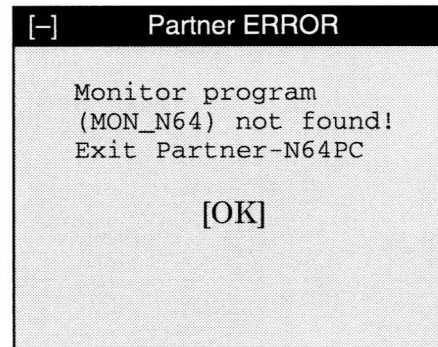


Figure 21

The message in Figure 21 is displayed when the monitor file (MON_N64), initial setting file (.INI), or environment file (.CFG) is not present. Confirm that each of these files exists.

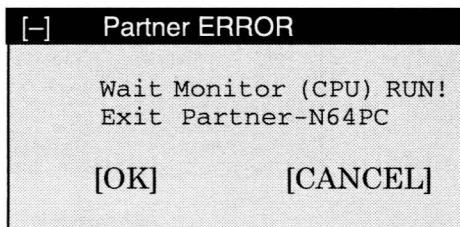


Figure 22

If the message in Figure 22 is displayed, the monitor program could not be properly executed . If processing will not proceed with this message box displayed, refer to Troubleshooting in the installation portion of this manual. Click on the <OK> button in this message box to terminate Partner-N64PC startup.

Chapter 5

Window Commands

The Partner-N64PC debugger uses MDI (Multi Document Interface) to display necessary information in various slave windows during debugging, providing a greater volume of information than the DOS version of Partner. In addition, a fast operating environment is provided through the use of shortcut keys, menus, tool bars, status bars, and dialog boxes.

Window Configuration

This section describes the main structural elements of the Partner-N64PC screens, such as menus, tool bars, status bars, and various debug windows.

Structural Elements

The window elements are shown in Figure 23.

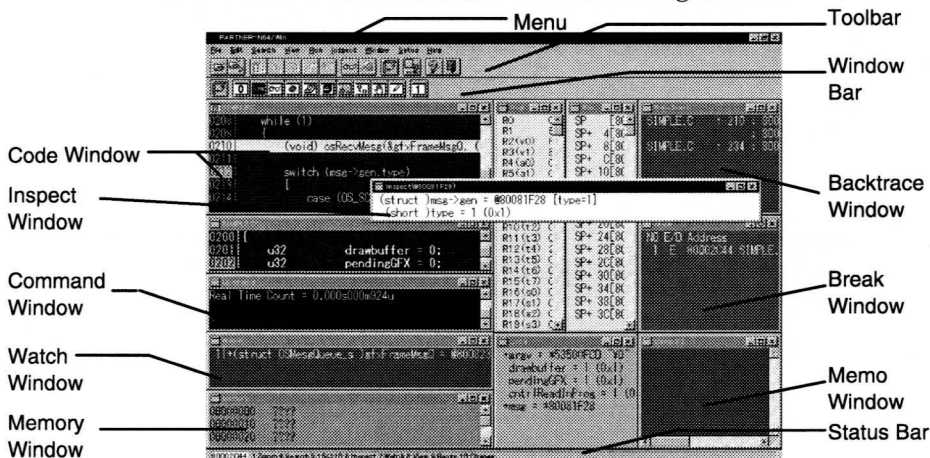


Figure 23: Screen Elements

WINDOW COMMANDS

Main Window

This is the main window for Partner-N64PC. It contains the menu bar, tool bar, and status bar.

Code Window



This window displays source and reverse assembly. The program counter is highlighted, and software breakpoints are underlined. There are two of these windows.

Command Window



This window is for entering dialog commands and displaying execution results.

Memory Window



This window is for the dump display of memory space. It is updated with new data when the CPU stops.

Register Window



This window displays the CPU registers. It is updated with new data when the CPU stops.

Stack Window



This displays the stack contents. It is updated with new data when a break occurs in the CPU.

Local Window



This displays the contents of the local variables in the C functions that currently have a program counter. It is updated with new data when a break occurs in the CPU.

Watch Window



This displays the stored watch data. It is updated with new data when the CPU stops.

Backtrace Window



This displays a backtrace of C functions. It is updated with new data when the CPU stops.

Break Window



This displays the breakpoints that are currently set.

Memo Window



This is a simple editor for use during debugging. Its contents are saved when the user exits Partner-N64PC.

Inspect Window

This inspects and displays the contents of variables.

Screen Display

The windows generally cannot be scrolled horizontally to display data in the off-screen portion of a window. To view such data, the size of the window or the font size must be changed. Windows that can be scrolled horizontally are the Code, Memory, and Memo windows.

In addition, when a portion of the results of a character string search is off-screen, this portion is indicated by three (3) rightward angle brackets (>) on the corresponding line. To display search results (Figure 24), change the window size or font size, and select [Find Next] in the [Search] menu.

```

Register
R0 00000000 (
R1 80030000 (-2147287C
R2(v0) BFF00008 (-10747903
R3(v1) BFF00008 (-10747903
R4(a0) 4E4D4300 ( 1263: >>>
R5(a1) 00000000 (
R6(a2) 00000000 (
R7(a3) 00000000 (
R8(t0) 80000100 (-21474832
R9(t1) 80000184 (-21474832
R10(t2) 00000000 (
R11(t3) 00004000 ( 163
R12(t4) 03400008 ( 545259
R13(t5) 00000000 (
R14(t6) 1F703636 ( 527447E
R15(t7) 5A2EEC3D ( 15130245
R16(s0) 00000400 ( 1C
R17(s1) A3F08000 (-1544519E
R18(s2) 00180000 ( 1572E
R19(s3) 00000000 (
  
```

Figure 24: Display Screen with Search Character String Off

Menus

Partner-N64PC commands are grouped in the menu. Commands include those that immediately execute processing, and those that display a dialog box allowing the user to select additional options.

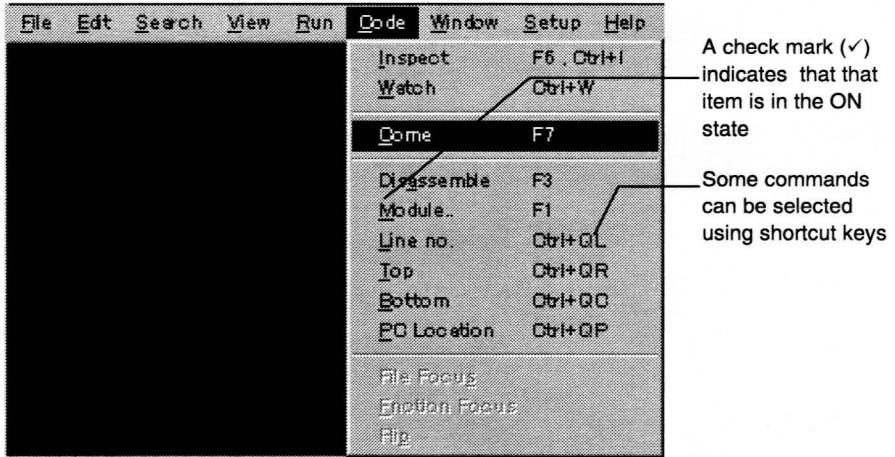


Figure 25. Menu

File Menu

File (F)
Load (L)
Reload (R)
Module (M)
Exit (X)

The [File] menu provides file-related commands. [Module] can be executed only when the Code window or the Command window is active.

Menu Item	Function
Load (<u>L</u>)	Displays the dialog box for loading the program to be debugged. See Open File Dialog Box (pg. 127).
Reload (<u>R</u>)	Reloads the program to be debugged
Module (<u>M</u>)	Displays the dialog box for selecting a module contained in the loaded debug program. See Module Dialog Box (pg. 130)
Exit (<u>X</u>)	Exits Partner-N64PC

Edit Menu

Edit (E)
Paste (P) Shift+Ins
Copy Page (C) Ctrl+Ins

The [Edit] menu provides commands for moving character strings within and between windows via the clipboard.

Edit (E)	
Undo (U)	Alt+BkSp
Cut (C)	Shift+Del
Copy (O)	Ctrl+Ins
Paste (P)	Shift+Ins
Delete (D)	Del
Clear All (L)	Ctrl+Del

The [Edit] menu changes to that shown above when the Memo window is active.

Menu Item	Function
Paste (<u>P</u>)	Pastes the contents of the clipboard
Copy Page (<u>C</u>)	Copies the information displayed in the window to the clipboard
Undo (<u>U</u>)	Cancels previous editing
Cut (<u>C</u>)	Moves the selected character string to the clipboard
Copy (<u>O</u>)	Copies the selected character string to the clipboard
Delete (<u>D</u>)	Deletes the selected character string
Clear All (<u>L</u>)	Deletes the entire contents of the Memo window

Search Menu

Search (S)	
Search for String.. (F)	F4, Ctrl+QF
Find Next (N)	Ctrl+F4, Ctrl+L

The [Search] menu provides commands for searching for character strings in a window.

Menu Item	Function
Search for String.. (F)	Displays a dialog box for specifying the character string to be found. See color in each window. Search Character String Dialog Box (pg. 128).
Find Next (N)	Searches for the specified character string in the specified direction

View Menu

View (V)
✓ Status Bar (S)
✓ Toolbars (T)
✓ Window Bar (W)
Hint (F)

The [View] menu provides commands to display or not display the toolbars and other bars.

Menu Item	Function
Status Bar (S)	Choose whether to the status bar. See Status Bar (pg. 126).
Toolbars (T)	Choose whether to display the toolbars. See Tool Bars (pg. 12).
Window Bar (W)	Choose whether to display the window bar. See Window Bar (pg. 124).
Hint (F)	Choose whether to display hints related to the buttons.

Run Menu

Run (R)	
Run Program (G)	F5
Come (C)	F7
Trace (T)	F8
Step (S)	F10
Return Run (U)	Ctrl+F5
Software Break (S)	F9
Forced Break (B)	ESC

The [Run] menu provides commands related to running and interrupting user programs.

[Come] and [Software Break] can be executed only when the Code window is active.

Menu Item	Function
Run Program (<u>G</u>)	Executes user program from current PC.
Come (<u>C</u>)	Runs user program to the line where cursor is located.
Trace (<u>T</u>)	Executes a trace run in units of source code lines units or machine language commands.
Step (<u>S</u>)	Executes a step run in units of source code lines or machine language commands.
Return Run (<u>U</u>)	Runs a program until execution of the current function is completed (Return) and program control returns to the calling function.
Software Break (<u>S</u>)	Sets and cancels software breakpoints.
Forced Break (<u>B</u>)	Forcibly interrupts the user program and returns control to Partner-N64PC (interrupts linked execution during linked execution of commands or macros).

Local Menu Windows

Local menus contain commands that can be executed in the corresponding window. The local menus for the active window are registered on the menu bar. The various local menus can be displayed and selected by clicking the right mouse button.

Code Menu

Code (C)	
Inspect (I)	F6, Ctrl+I
Watch (W)	Ctrl+W
Come (C)	F7
Assembler (A)	F3
Module (M)..	F1
Line No. (L)	Ctrl+QL
Top (T)	Ctrl+QR
Bottom (B)	Ctrl+QC
PC Location (P)	Ctrl+QP
Set File (S)	
Set Function (F)	
Flip (P)	

The [Code] menu provides commands for controlling the Code window.

See **Code Window Local Menu** (pg. 85) for descriptions of commands in this menu.

Command Menu

Command (C)	
Load (L)..	
Reload (R)	
Paste (P)	Shift+Ins
History (H)..	Shift+F2
Expand Symbol (S)..	Shift+F6

The [Command] menu provides commands for controlling the Command window.

See **Code Window Local Menu** (pg. 85) for descriptions of commands in this menu.

Memory Menu

Memory (M)	
✓ View Byte (B)	B
View Word (W)	W
View Double Word (D)	D
Short Float (S)	S, F
Long Float (L)	L
Temp. View Float (T)	T
View ASCII (C)	C
✓ View Hexdecimal (H)	6, H
View Decimal (D)	D
Set Address (A)..	A, F7
Edit Data (E)..	E, F6

The [Memory] menu provides commands for controlling the Memory window.

See **Memory Window Local Menu** (pg. 94) for descriptions of commands in this menu.

Register Menu

Register (R)	
Initialize (Z)	F3
Increment (+)	
Decrement (-)	
Change Value (C)	Enter
View Decimal (D)	F9
View Symbol (S)	F6
View Memory (M)	F7

The [Register] menu provides commands for controlling the Register window.

See **Register Window Local Menu** (pg. 100) for descriptions of commands in this menu.

Stack Menu

Stack (S)	
View Symbols (S)	F6
View Address (A)	F7

The [Stack] menu provides commands for controlling the Stack window.

See **Stack Window Local Menu** (pg. 103) for descriptions of commands in this menu.

Local Menu

Local (L)	
Inspect (I)	F6, Ctrl+I
Watch (W)	F7, Ctrl+W
View Element (+)	Enter
Offset (O)	F9

The [Local] menu provides commands for controlling the Local window.

See **Local Window Local Menu** (pg. 105) for descriptions of commands in this menu.

Watch Menu

Watch (W)	
Inspect (I)	F6, Ctrl+I
Watch (W)	Ctrl+W
View Elements (+)	Enter
Add Watch (A)	Ins
Clear (C)	Del
Clear All (I)	F3

The [Watch] menu provides commands for controlling the Watch window.

See **Watch Window Local Menu** (pg. 110) for descriptions of commands in this menu.

Backtrace Menu

Backtrace (B)	
Inspect (I)	F6, Ctrl+I
Source (S)	F9
✓ Address (A)	F7

The [Backtrace] menu provides commands for controlling the Backtrace window.

See **Backtrace Window Local Menu** (pg. 107) for descriptions of commands in this menu.

Break Menu

Break (B)	
Set Break (A)..	Ins
Forbid (D)	F9, Enter
Clear (C)	Del
Clear All (I)	C
Save Settings (S)	F4
Load Settings (L)	F6
Hardware Break (H)..	F7

The [Break] menu provides commands for controlling the Break window.

See **Backtrace Window Local Menu** (pg. 107) for descriptions of commands in this menu.

Memo Menu

Edit (E)	
Undo (U)	Alt+BkSp
Cut (C)	Shift+Del
Copy (O)	Ctrl+Ins
Paste (P)	Shift+Ins
Delete (D)	Del
Clear All (L)	Ctrl+Del

The [Memo] menu provides commands for controlling the Memo window.

See **Memo Window Local Menu** (pg. 118) for descriptions of commands in this menu.

Inspect Menu

Inspect (I)	
Inspect (I)	F6, I, Ctrl+I
Watch (W)	F7, W, Ctrl+W
View (V)	F8, V, Ctrl+V
Range (R)..	F9, R
Change (C)..	F10, C
Change Base (D)	F5

The [Inspect] menu provides commands for controlling the Inspect window.

See **Inspect Window Local Menu** (pg. 115) for descriptions of commands in this menu.

Window Menu

Window (W)	
Cascade (C)	
Tile (T)	
Arrange Icons (I)	
User Setup 1 (1)	Ctrl+1
User Setup 2 (2)	Ctrl+2
User Setup 3 (3)	Ctrl+3
Save Window Setup (S)	
User Setup 1 (1)	
User Setup 2 (2)	
User Setup 3 (3)	
1 Memo.PT	
2 _exit-1	
3 Register	
4 Local	
5 Memory	
6 Backtrace	
7 Stack	
8 Watch	
✓ 9 DRY.C	
More Windows (M)...	

The [Window] menu provides commands for controlling the windows as a whole.

Menu Item	Function
Cascade (<u>C</u>)	Displays open windows in an overlapping pattern.
Tile (<u>T</u>)	Displays open windows side by side (special Partner-N64PC arrangement).
Arrange Icons (<u>I</u>)	Arranges window icons in rows.
User Setup 1 (<u>1</u>)	Retrieves the window arrangement saved with [Save Window Setup]-[User Setup 1].
User Setup 2 (<u>2</u>)	Retrieves the window arrangement saved with [Save Window Setup]-[User Setup 2].
User Setup 3 (<u>3</u>)	Retrieves the window arrangement saved with [Save Window Setup]-[User Setup 2].
Save Window Setup (<u>S</u>)	Saves current window arrangement.

Settings Menu

Settings (S)
Select Colors (S)..
Set Toolbars (T)..
Set Font (F)..
Set Options (O)..
✓ Resize Window (R)

The [Settings] menu provides commands for setting up Partner-N64PC.

Menu Item	Function
Select Colors (S)	Displays a dialog box for selecting the window and text colors. See Specify Colors Dialog Box (pg. 128).
Set Toolbars (T)	Displays a dialog box for registering the toolbar buttons. See Toolbar Setup Dialog Box (pg. 128).
Set Font (F)	Displays a dialog box for selecting the text size. See Specify Font Dialog Box (pg. 127).
Set Options (O)	Displays a dialog box for specifying the various Partner-N64PC settings
Resize Window (R)	Changes the size tiled windows to fit the size of the main window

Help Menu

Help (H)
Contents (C)
Search for Help on (S)...
How to Use Help (H)
About Partner-N64 (A)...

The [Help] menu provides commands concerning the Help feature.

Menu Item	Function
Contents (C)	Displays the table of contents for Partner-N64PC Help.
Search for Help on (S)	Searches Partner-N64PC Help for a keyword.
How to Use Help (H)	Displays instructions for using Windows95 Help.
About Partner-N64 (A)	Displays information about Partner-N64PC.

Shortcut Keys

Shortcut keys are key combinations defined to execute frequently used Partner-N64PC commands more smoothly and easily.

Shortcut Keys Common to Multiple Windows

The following key operations are shortcut keys that are common to various windows.

Key Operation	Function
F1	Toggle the active window between [Fill Screen] and [Return to Original Size]. This key is defined for different commands in the Code window and Command window.
F2 CTRL+O	Moves the active window to the bottom level, making the next window active.
CTRL+B	Makes the previously active window active (Reverse of F2, CTRL+O).
F4 CTRL+QF	Opens the Search Character String dialog box to specify search string and search direction. See color in each window.° Search Character String Dialog Box (pg. 128).
F5	Runs user program from current PC.
F8	Trace run of user program.
F10	Step run of user program.
CTRL+C Page Down	Displays one screen down.
CTRL+D →	Moves cursor one position to the right.
CTRL+E ↑	Moves cursor one line up.
CTRL+L	Repeats search from cursor position for character string specified with F4 .
CTRL+R Page Up	Displays one screen up.
CTRL+S ←	Moves cursor one position to the left.
CTRL+X <return>	Moves cursor one line down.
CTRL+INS	Copies one screen of information from the active window to the clipboard
CTRL+Alt+F1 CTRL+Alt+O	Toggles Code window 0 between open and minimized.
CTRL+Alt+F2 CTRL+Alt+C	Toggles Command window between open and minimized.
CTRL+Alt+F3 CTRL+Alt+W	Toggles Watch window between open and minimized.

Key Operation	Function
CTRL+Alt+F5 CTRL+Alt+R	Toggles Register window between open and minimized.
CTRL+Alt+F6 CTRL+Alt+S	Toggles Stack window between open and minimized.
CTRL+Alt+F7 CTRL+Alt+L	Toggles Local window between open and minimized.
CTRL+Alt+F8 CTRL+Alt+T	Toggles Backtrace window between open and minimized.
CTRL+Alt+F9 CTRL+Alt+B	Toggles Break window between open and minimized.
CTRL+Alt+F10 CTRL+Alt+E	Toggles Memo window between open and minimized.
CTRL+1	Load user setup 1.
CTRL+2	Load user setup 2.
CTRL+3	Load user setup 3.
Alt+F10	Display local menu for active window.

Shortcut Keys Unique to Certain Windows

Unlike the previously described shortcut keys, which are used in multiple windows, these shortcut keys are defined for individual windows. See the page listed for the shortcut keys for the corresponding window.

Window Name	Reference	Page
Code Window	Code Window Shortcut Keys	82
Command Window	Command Window Shortcut Keys	82
Memory Window	Memory Window Shortcut Keys	93
Register Window	Register Window Shortcut Keys	99
Stack Window	Stack Window Shortcut Keys	102
Local Window	Local Window Shortcut Keys	104
Backtrace Window	Backtrace Window Shortcut Keys	107
Watch Window	Watch Window Shortcut Keys	109
Break Window	Break Window Shortcut Keys	107
Inspect Window	Inspect Window Shortcut Keys	114
Memo Window	Memo Window Shortcut Keys	118

Mouse Operation

Mouse operations in the windows are divided into those that are common to multiple windows and those that are unique to specific windows.

Mouse Operations Common to Multiple Windows

Click Right Button

Clicking the right mouse button in the active window displays the local menu for that window, from which you can select a command. See **Local Menu Window** (pg. 66).

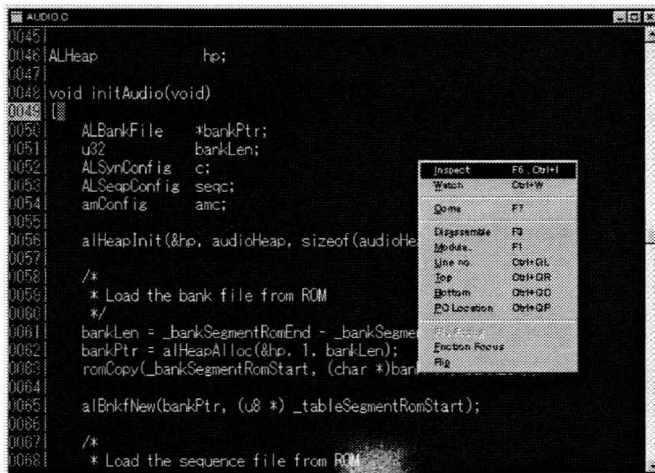



Figure 26: Code Window Local Menu

Drag Left Button

A character string in a window can be highlighted and copied to the clipboard by dragging the mouse across the character string while depressing the left mouse button, then releasing the button.



```
0045 |
0046 | ALHeap      hp;
0047 |
0048 | void initAudio(void)
0049 | {
0050 |     ALBankFile  *bankPtr;
0051 |     u32         bankLen;
0052 |     ALSynConfig c;
0053 |     ALSeapConfig seac;
0054 |     amConfig    amc;
0055 |
0056 |     alHeapInit(&hp, audioHeap, sizeof(audioHeap));
0057 |
0058 |     /*
0059 |      * Load the bank file from ROM
0060 |      */
0061 |     bankLen = _bankSegmentRomEnd - _bankSegmentRomStart;
0062 |     bankPtr = alHeapAlloc(&hp, 1, bankLen);
0063 |     romCopy(_bankSegmentRomStart, (char *)bankPtr, bankLen);
0064 |
0065 |     alBnkfNew(bankPtr, (u8 *) _tableSegmentRomStart);
0066 |
0067 |     /*
0068 |      * Load the sequence file from ROM
```

Figure 27: Selection of Copy Range

Dragging the mouse means to move the mouse while holding down a mouse button.

Mouse Operations Unique to a Window

Unlike **Mouse Operations Common to Multiple Windows** these mouse operations are independently defined for each window. Refer to the pages listed below for the mouse operations in each window.

Window Name	Reference	Page
Code Window	Code Window Mouse Operations	87
Memory Window	Memory Window Mouse Operations	96
Register Window	Register Window Mouse Operations	102
Local Window	Local Window Mouse Operations	106
Backtrace Window	Backtrace Window Mouse Operations	108
Watch Window	Watch Window Mouse Operations	111
Break Window	Break Window Mouse Operations	108
Inspect Window	Inspect Window Mouse Operations	116

Code Windows

The Code windows display either source code or a mixed display of assembler and source codes. There are two Code windows, enabling various display modes to be specified.

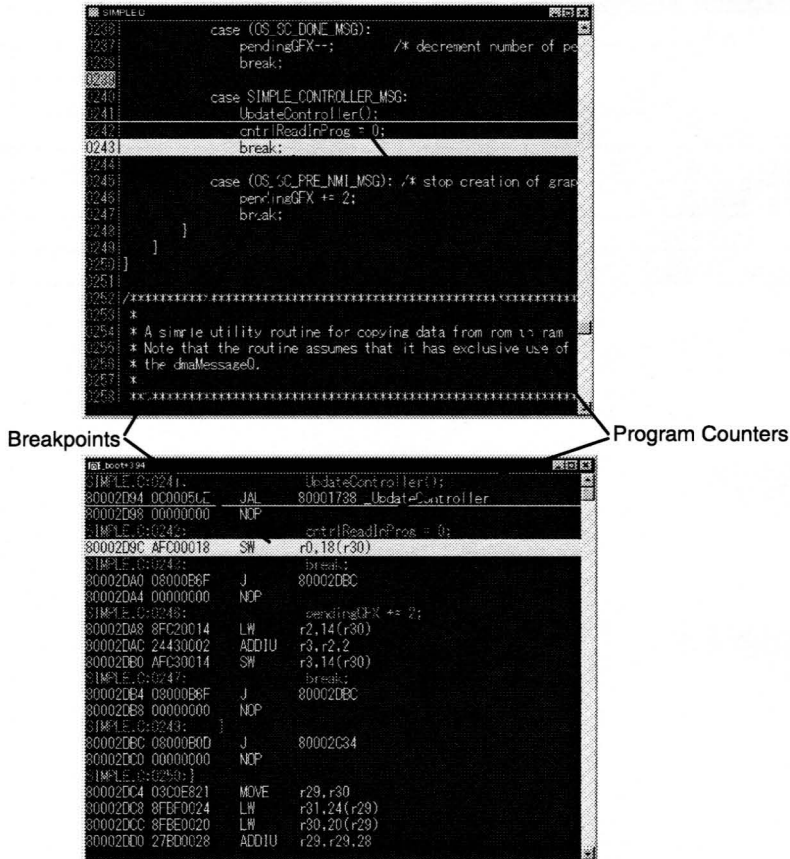


Figure 28: Code Windows

Code Window Shortcut Keys

Commands can be executed in the Code window using the following shortcut keys. These shortcut keys are valid when the Code window is active.

Key Operation	Function	Dialog Command
F1	Displays a dialog box for selecting the module. See Module Dialog Box (pg. 130).	
F3	Toggles the Code window between source code and reverse assembler display. Switches so that the line of the current PC is displayed.	V command U command
F6 CTRL+F6 CTRL+I	Inspects the variable at the cursor position.	INS command
F7	Runs user program for the current PC up to the cursor position.	
F9	Sets or clears the software breakpoint at the cursor position.	BP command
CTRL+F1	Toggles the Code window between [Fill Screen] and [Return to Original Size].	
CTRL+F2	Makes next window active.	
CTRL+F3	Toggles the Code window between source code and reverse assembler display. Switches so that the line of the cursor is displayed.	V command U command

Key Operation	Function	Dialog Command
CTRL+F4 CTRL+L	From the cursor position, repeats search for the same character string and in the same direction as specified in the previous character string search.	
CTRL+F5	Ends execution of current function (Return) and executes user program to point where control returns to calling function.	G command
CTRL+F7 CTRL+W	Registers variable at cursor position in the Watch window.	W command
CTRL+F8 CTRL+V	Displays variable at cursor position in the Command window.	VAL command ? command
CTRL+F9 CTRL+G	Places character string at cursor position at the cursor position in the Command window.	
CTRL+F10	Copies the character string at the cursor position to the clipboard.	
CTRL+QL	Displays a dialog box for specifying the line no. and address to be displayed.	V command U command
CTRL+QR	Displays top of source code.	
CTRL+QC	Displays end of source code.	
CTRL+QJ	Returns to initial window arrangement.	
CTRL+QP	Displays line of current PC.	

Key Operation	Function	Dialog Command
CTRL+F	Moves the cursor to the next character string when source code is displayed. When reverse assembler is displayed, moves the cursor sequentially in the order: address→code→nemonic.	
CTRL+A	Moves the cursor to the previous character string when source code is displayed.	

SHIFT+??? is described in **Command Window Shortcut Keys** (pg. 89).

Code Window Local Menu

Code (C)	
Inspect (I)	F6, Ctrl+I
Watch (W)	Ctrl+W
Come (C)	F7
Assembler (A)	F3
Module (M)..	F1
Line No. (L)	Ctrl+QL
Top (T)	Ctrl+QR
Bottom (B)	Ctrl+QC
PC Location (P)	Ctrl+QP
Set File (S)	
Set Function (F)	
Flip (P)	

The [Code] menu provides commands for controlling the Code window.

Menu Item	Function
Inspect (I)	Opens the Inspect window for the variable at the cursor position. If the character string at the cursor cannot be inspected, the Inspect Setup dialog box is displayed. See Inspect Setup Dialog Box (pg. 129)
Watch (W)	Registers the variable at the cursor position in the Watch window. If the character string at the cursor position cannot be registered for Watch, the Watch Set dialog box is displayed. See Watch Set Dialog Box (pg. 129).
Assembler (A) / Source (S)	Toggles source and reverse assembler display.
Module (M)	Displays the dialog box for selecting a module contained in the loaded debug program. See Module Dialog Box (pg. 130).

Menu Item	Function
Line No. (<u>L</u>) / Address (<u>A</u>)	Displays the dialog box for entering the line number or address to be displayed. See Line Number Specification Dialog Box (pg. 131), Display Address Specification Dialog Box (Code) (pg. 131).
Top (<u>T</u>)	Displays top of file being displayed.
Bottom (<u>B</u>)	Displays end of file being displayed.
PC Location (<u>P</u>)	Displays current PC location.
Set File (<u>S</u>)	If a file in the PC is currently being checked and the PC moves out of this file, another Code window is activated. In addition, if the PC moves to the file that was checked, the checked Code window is activated.
Set Function (<u>F</u>)	If a function in the PC is being checked and the PC moves out of this function, another Code window is activated. In addition, if the PC moves to the checked function, the checked Code window is activated.
Flip (<u>F</u>)	Another Code window is activated each time the function on the PC changes.

Set File, Set Function and **Flip** are valid only when two Code windows are displayed.

Code Window Mouse Operations

Commands frequently used in the Code window are assigned to mouse operations.

Function	Mouse Operation	Dialog Command
Inspect	Double click left button on variable.	INS command
Software Break Point	Click left button on line no./address.	BP command
Trace run	SHIFT+left click.	T command
Step run	SHIFT+right click.	P command
Select [Run] menu (See pg. 64)	CTRL+right click.	

Command Window

The Command window is used for entering dialog commands and outputting execution results.

Refer to on-line help regarding the dialog commands that are entered in the Command window.

```

Command
R24-31:00000000 00000000 A430000C 80002BE8 00000000 8002CB70 8002CB70 80002BE8
PC :80002BE8 (LO:00000000 HI:00000000)
_boot+1D8: J 80002BE8
Real Time Count = 0,007s741m071u

R0 / R8 R1 / R9 R2 /R10 R3 /R11 R4 /R12 R5 /R13 R6 /R14 R7 /R15
R16/R24 R17/R25 R18/R26 R19/R27 R20/R28 R21/R29 R22/R30 R23/R31
R 0- 7:00000000 80000000 00000005 00000005 00000001 00000008 00000001 00000001
R 8-15:2000FF01 2000FF00 00000008 00002000 2000FF01 00000000 00000000 00000000
R16-23:00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
R24-31:00000000 00000000 A430000C 00000AAA 00000000 8002AB68 8002AB68 80002C60
PC :80002D94 (LO:00000000 HI:00000005)
_boot+384: JAL 80001738 _updateController
Real Time Count = 0,000s003m832u
  
```

Figure 29: Command Window

Command Window Shortcut Keys

Commands can be executed in the Command window using the following shortcut keys. These shortcut keys are valid when the Command window is active.

Key Operation	Function
SHIFT+F1	Copies one character of previously entered command (C1).
SHIFT+F2	Displays dialog box showing command history entered to that point. See Command History (pg. 48)
SHIFT+F3	Copies the character string of the previous command after cursor position (CA).
SHIFT F4 CTRL+A	Moves the cursor to the head of the line (Ln Top).
SHIFT F5 CTRL+F	Moves the cursor to the end of the line (Ln Bot).
SHIFT+F6	Displays a dialog box listing global symbols, starting from the last character string of the command entry line, and expands the command. See Symbol Extension Dialog Box (pg. 132).
SHIFT+F7 CTRL+U	Deletes all characters in the line currently being edited.
SHIFT+F8	Deletes all characters in the line currently being edited and clears the entire contents of the command history.
BS CTRL+H	Deletes one character to the left of the cursor.
← SHIFT+← CTRL+S	Moves the cursor to the left.
→ SHIFT+→ CTRL+D	Moves the cursor to the right. When the cursor is at the end of a line, has the same effect as SHIFT+F3.
DEL CTRL+G	Deletes the character at the cursor position

Key Operation	Function
INS CTRL+V	Toggles between insert mode and overwrite mode
SHIFT+↑ CTRL+W	Displays the last command history. When a character string is entered in the command input line, commands starting with that character string, beginning with the oldest command, are searched for and displayed. Using this key combination again causes the next command history to be searched.
SHIFT+<return>	Displays the last command history. When a character string is entered in the command input line, commands starting with that character, beginning with the oldest command, are searched for and displayed. Using this key combination again causes the next command history to be searched.
SHIFT+INS	Enters the character string on the clipboard as a command.

The function keys and CTRL+function keys are the same as for the Code window shortcut keys. See **Code Window Shortcut Keys** (pg. 82). However, the commands related to cursor position cannot be used.

Command Window Local Menu

Command (C)	
Load (L)..	
Reload (R)	
Paste (P)	Shift+Ins
History (H)..	Shift+F2
Expand Symbol (S)..	Shift+F6

The [Command] menu provides commands for controlling the Command window.

Menu Item	Function
Load (<u>L</u>)	Displays the dialog box for loading a debug program.
Reload (<u>R</u>)	Reloads the debug program.
Paste (<u>P</u>)	Pastes the contents of the clipboard.
History (<u>H</u>)	Displays the Command History dialog box. See Command History Dialog Box (pg. 132).
Expand Symbol (<u>S</u>)	Displays a dialog box listing the global symbols, starting from the last character string on the command input line. See Symbol Extension Dialog Box (pg. 132).

Memory Window

The Memory window displays the contents of memory in various display formats.

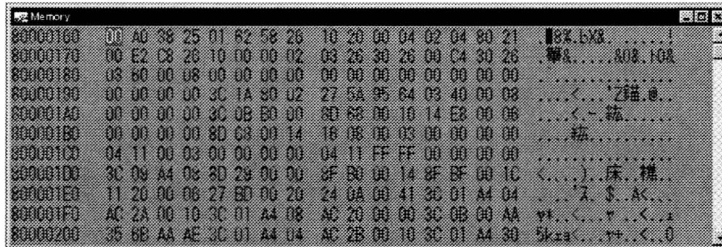


Figure 30: Memory Window

Memory Window Shortcut Keys

Commands can be executed in the Memory window using the following shortcut keys. These shortcut keys are valid when the Memory window is active.

Key Operation	Function
F7 A	Displays the dialog box for specifying the display start address. See Display Address Specification Dialog Box (Code) (pg. 131).
F6 E	Displays the dialog box for changing data. See Data Set Dialog Box (pg. 133)
B	Specifies byte display of memory data.
W	Specifies word display of memory data.
D	Specifies double word display of memory data.
S, F	Specifies short float display of memory data.
L	Specifies long float display of memory data.
T	Specifies temporary float display of memory data.
C	Specifies ASCII display of memory data.
6, H	Specifies hexadecimal (base 16) display of memory data.
1	Specifies decimal (base 10) display of memory data.

Memory Window Local Menu

Memory (M)	
✓ View Byte (B)	B
View Word (W)	W
View Double Word (D)	D
Short Float (S)	S, F
Long Float (L)	L
Temp. View Float (T)	T
View ASCII (C)	C
✓ View Hexdecimal (H)	6, H
View Decimal (D)	D
Set Address (A)..	A, F7
Edit Data (E)..	E, F6

The [Memory] menu provides commands for controlling the Memory window.

Menu Item	Function
View Byte (B)	Specifies byte display of memory data.
View Word (W)	Specifies word display of memory data.
View Double Word (D)	Specifies double word display of memory data.
Short Float (S)	Specifies short float display of memory data.
Long Float (L)	Specifies long float display of memory data.
Temp. View Float (T)	Specifies temporary float display of memory data.
View ASCII (C)	Specifies ASCII display of memory data.
View Hexdecimal (H)	Specifies hexadecimal (base 16) display of memory data.
View Decimal (D)	Specifies decimal (base 10) display of memory data.

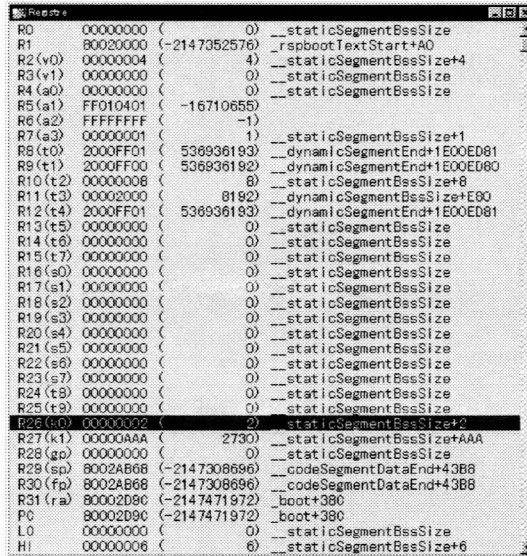
Menu Item	Function
Set Address(<u>A</u>)	Displays the dialog box for specifying the display start address. See Display Address Specification Dialog Box (Code) (pg. 131).
Edit Data (<u>E</u>)	Displays the dialog box for changing data at the cursor. See Data Set Dialog Box (pg. 133).

Memory Window Mouse Operations

Double click the left mouse button in the data area to display the dialog box for changing data. See **Data Set Dialog Box** (pg. 133).

Register Window

The following registers are displayed in the Register window: the r0-r3, pc, hi and lo registers (Figure 31); the CP0 register (Figure 32); the TLB register (Figure 33); and the FPU register (Figure 34).



Register	Value	Comment
R0	00000000	(0) __staticSegmentBssSize
R1	80020000	(-2147352576) _rsbootTextStart+A0
R2 (v0)	00000004	(4) __staticSegmentBssSize+4
R3 (v1)	00000000	(0) __staticSegmentBssSize
R4 (a0)	00000000	(0) __staticSegmentBssSize
R5 (a1)	FF010401	(-16710655) __staticSegmentBssSize
R6 (a2)	FFFFFFF	(-1) __staticSegmentBssSize+1
R7 (a3)	00000001	(1) __staticSegmentBssSize+1
R8 (t0)	2000FF01	(536936183) __dynamicSegmentEnd+1E00ED81
R9 (t1)	2000FF00	(536936182) __dynamicSegmentEnd+1E00ED80
R10 (t2)	00000008	(8) __staticSegmentBssSize+8
R11 (t3)	00002000	(8192) __dynamicSegmentBssSize+E90
R12 (t4)	2000FF01	(536936183) __dynamicSegmentEnd+1E00ED81
R13 (t5)	00000000	(0) __staticSegmentBssSize
R14 (t6)	00000000	(0) __staticSegmentBssSize
R15 (t7)	00000000	(0) __staticSegmentBssSize
R16 (s0)	00000000	(0) __staticSegmentBssSize
R17 (s1)	00000000	(0) __staticSegmentBssSize
R18 (s2)	00000000	(0) __staticSegmentBssSize
R19 (s3)	00000000	(0) __staticSegmentBssSize
R20 (s4)	00000000	(0) __staticSegmentBssSize
R21 (s5)	00000000	(0) __staticSegmentBssSize
R22 (s6)	00000000	(0) __staticSegmentBssSize
R23 (s7)	00000000	(0) __staticSegmentBssSize
R24 (t8)	00000000	(0) __staticSegmentBssSize
R25 (t9)	00000000	(0) __staticSegmentBssSize
R26 (t0)	00000002	(2) __staticSegmentBssSize+2
R27 (k1)	00000AAA	(2730) __staticSegmentBssSize+AAA
R28 (gp)	00000000	(0) __staticSegmentBssSize
R29 (sp)	8002A868	(-2147308696) __codeSegmentDataEnd+43B8
R30 (fp)	8002A868	(-2147308696) __codeSegmentDataEnd+43B8
R31 (ra)	80002B9C	(-2147471972) _boot+39C
PC	80002B9C	(-2147471972) _boot+39C
LO	00000000	(0) __staticSegmentBssSize
HI	00000006	(6) __staticSegmentBssSize+6

Figure 31: CPU Register

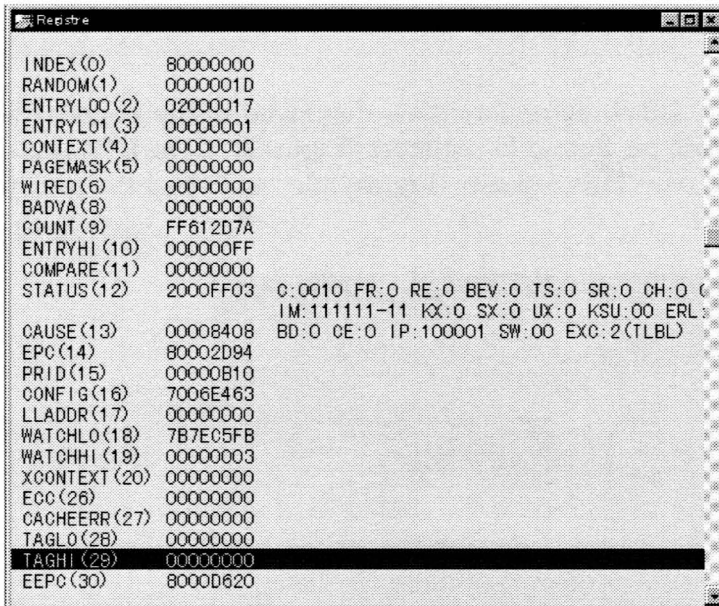


Figure 32: CP0 Register

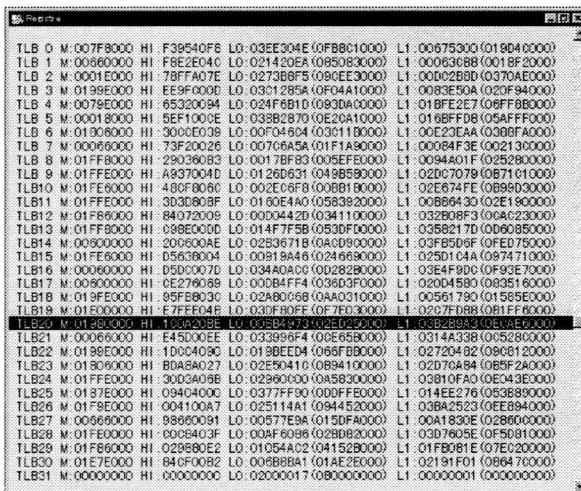
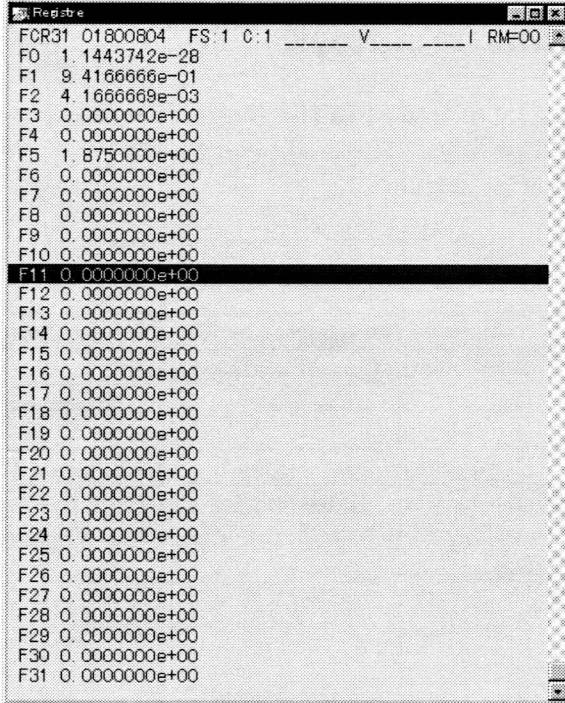


Figure 33: TLB Register



```
Registers
FCR31 01800804 FS:1 C:1 V I RM=00
F0 1.1443742e-28
F1 9.4166666e-01
F2 4.1666669e-03
F3 0.0000000e+00
F4 0.0000000e+00
F5 1.8750000e+00
F6 0.0000000e+00
F7 0.0000000e+00
F8 0.0000000e+00
F9 0.0000000e+00
F10 0.0000000e+00
F11 0.0000000e+00
F12 0.0000000e+00
F13 0.0000000e+00
F14 0.0000000e+00
F15 0.0000000e+00
F16 0.0000000e+00
F17 0.0000000e+00
F18 0.0000000e+00
F19 0.0000000e+00
F20 0.0000000e+00
F21 0.0000000e+00
F22 0.0000000e+00
F23 0.0000000e+00
F24 0.0000000e+00
F25 0.0000000e+00
F26 0.0000000e+00
F27 0.0000000e+00
F28 0.0000000e+00
F29 0.0000000e+00
F30 0.0000000e+00
F31 0.0000000e+00
```

Figure 34: FPU Register

Register Window Shortcut Keys

Commands can be executed in the Register window using the following shortcut keys. These shortcut keys are valid when the Register window is active.

Key Operation	Function
F3	Initializes the register or flag at the cursor position.
F6	Symbolic display of register value.
F7	Displays the contents of memory indicated by the register value.
F9	Displays register values in decimal (base 10) format.
.	Displays the dialog box for changing the register value or flag value at the cursor position. See Register Dialog Box (pg. 134).

Register Window Local Menu

Register (R)	
Initialize (Z)	F3
Increment (+)	
Decrement (-)	
Change Value (C)	Enter
View Decimal (D)	F9
View Symbol (S)	F6
View Memory (M)	F7

The [Register] menu provides commands for controlling the Register window.

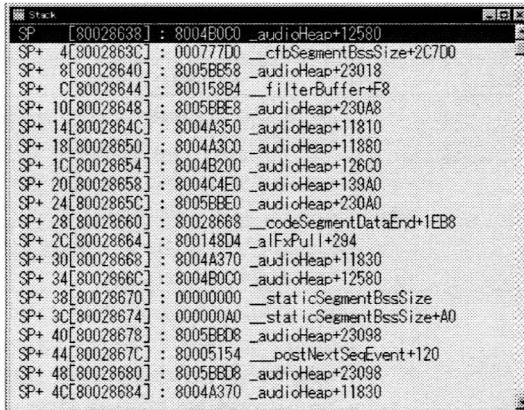
Menu Item	Function
Initialize (<u>Z</u>)	Sets the register value at the cursor position to 0 (zero).
Increment (<u>+</u>)	Increments the register value at the cursor position.
Decrement (<u>-</u>)	Decrements the register value at the cursor position.
Change Value (<u>C</u>)	Displays the dialog box for changing the register value or flag value at the cursor position. See Register Dialog Box (pg. 134).
View Decimal (<u>D</u>)	Toggles between displaying and not displaying register values in decimal (base 10).
View Symbol (<u>S</u>)	Toggles between displaying and not displaying symbols for register values.
View Memory (<u>M</u>)	Toggles between displaying and not displaying the contents of memory indicated by register values.

Register Window Mouse Operations

Double click the left mouse button in the register or flag area to display the dialog box for changing values. See **Register Dialog Box** (pg. 134).

Stack Window

The contents of the current stack memory are displayed in the Stack window.



```

Stack
SP [80028638] : 8004B0C0 _audioHeap+12580
SP+ 4[8002863C] : 000777D0 __cfbSegmentBssSize+2C7D0
SP+ 8[80028640] : 8005B858 _audioHeap+23018
SP+ C[80028644] : 800158B4 _filterBuffer+F8
SP+ 10[80028648] : 8005B8E8 _audioHeap+230A8
SP+ 14[8002864C] : 8004A350 _audioHeap+11810
SP+ 18[80028650] : 8004A3C0 _audioHeap+11880
SP+ 1C[80028654] : 8004B200 _audioHeap+126C0
SP+ 20[80028658] : 8004C4E0 _audioHeap+139A0
SP+ 24[8002865C] : 8005B8E0 _audioHeap+230A0
SP+ 28[80028660] : 80028668 __codeSegmentDataEnd+1EB8
SP+ 2C[80028664] : 800148D4 _aIFxPulI+294
SP+ 30[80028668] : 8004A370 _audioHeap+11830
SP+ 34[8002866C] : 8004B0C0 _audioHeap+12580
SP+ 38[80028670] : 00000000 __staticSegmentBssSize
SP+ 3C[80028674] : 000000A0 __staticSegmentBssSize+A0
SP+ 40[80028678] : 8005B8D8 _audioHeap+23098
SP+ 44[8002867C] : 80005154 __postNextSecEvent+120
SP+ 48[80028680] : 8005B8D8 _audioHeap+23098
SP+ 4C[80028684] : 8004A370 _audioHeap+11830
  
```

Figure 35: Stack Window

Stack Window Shortcut Keys

Commands can be executed in the Stack window using the following shortcut keys. These shortcut keys are valid when the Stack window is active.

Key Operation	Function
F6	Symbolic display of stack memory contents.
F7	Display address of stack memory.

Stack Window Local Menu

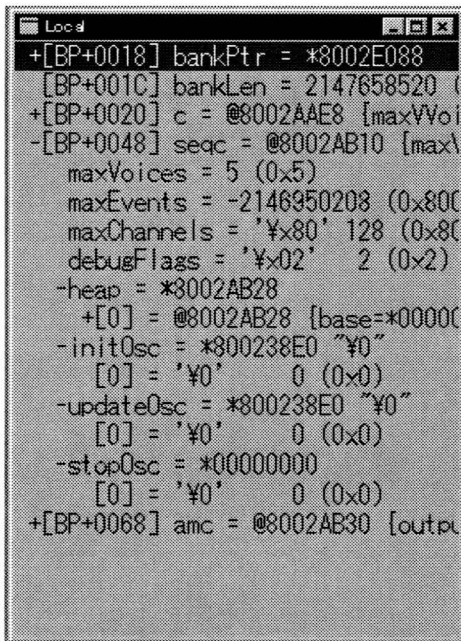
Stack (S)	
View Symbols (S)	F6
View Address (A)	F7

The [Stack] menu provides commands for controlling the Stack window.

Menu Item	Function
View Symbols (<u>S</u>)	Symbolic display of stack memory contents.
View Address (<u>A</u>)	Display actual memory address of stack

Local Window

The Local window displays the contents of local variables of functions. Variables with a "+" at the left of the local variable name contain a variable element that can be displayed. When the element is displayed, a "-" appears to the left of the local variable name.



```
Local
+[BP+0018] bankPtr = *8002E088
[BP+001C] bankLen = 2147658520 (
+[BP+0020] c = @8002AAE8 [maxVvoi
-[BP+0048] seqc = @8002AB10 [max\
maxVoices = 5 (0x5)
maxEvents = -2146950208 (0x80C
maxChannels = '¥x80' 128 (0x8C
debugFlags = '¥x02' 2 (0x2)
-heap = *8002AB28
+[0] = @8002AB28 [base=*0000C
-initOsc = *800238E0 "¥0"
[0] = '¥0' 0 (0x0)
-updateOsc = *800238E0 "¥0"
[0] = '¥0' 0 (0x0)
-stopOsc = *00000000
[0] = '¥0' 0 (0x0)
+[BP+0068] amc = @8002AB30 [outpc
```

Figure 36: Local Window

Local Window Shortcut Keys

Commands can be executed in the Local window using the following shortcut keys. These shortcut keys are valid when the Local window is active.

Key Operation	Function
F6 CTRL+I	Inspect the local variable at the cursor position.
F7 CTRL+W	Register the local variable at the cursor position in the Watch window.
F9	Display the local variable offset.
.	Toggle between displaying and not displaying the local variables with a variable element (variables with a + or - to the left of the variable name).

Local Window Local Menus

Local (L)	
Inspect (I)	F6, Ctrl+I
Watch (W)	F7, Ctrl+W
View Element (+)	Enter
Offset (O)	F9

The [Local] menu provides commands for controlling the Local window.

Menu Item	Function
Inspect (<u>I</u>)	Display the Inspect window for the local variable(s) selected with the cursor.
Watch (<u>W</u>)	Register the local variable(s) selected with the cursor in the Watch window.
View Element (<u>±</u>)	Toggle between displaying and not displaying the contents of local variables with elements such as arrays and structures.
Offset (<u>O</u>)	Toggle between displaying and not displaying the offset values of local variables.

Local Window Mouse Operations

Double click the left mouse button on a local variable with a variable element to toggle between displaying and not displaying variable elements.

Backtrace Window

The processes called up by the current function from the main() function are displayed in the Backtrace window.

```

BackTrace
AUDIO.C : 114 : 800006F4 __codeSegmentStart+2A4()
SIMPLE.C : 309 : 80002F5C initGame+EC()
SIMPLE.C : 208 : 80002C34 gameproc+2C(argv=#00000020 "¥0")
           : 8001A2FC __osCleanupThread()
AUDIO.C : 114 : 800006F4 __codeSegmentStart+2A4()
  
```

Figure 37: Backtrace Window

Backtrace Window Shortcut Keys

Commands can be executed in the Backtrace window using the following shortcut keys. These shortcut keys are valid when the Backtrace window is active.

Key Operation	Function
F6 CTRL+I	Display the source code in the Code window for the line at the cursor position.
F7	Toggle between displaying and not displaying the address for the symbol displayed in the window
F9	Toggle between displaying and not displaying the source name and line number for the symbol displayed in the window

Backtrace Window Local Menu

Backtrace (B)	
Inspect (I)	F6, Ctrl+I
Source (S)	F9
✓ Address (A)	F7

The [Backtrace] menu provides commands for controlling the Backtrace window.

Menu Item	Function
Inspect (<u>I</u>)	Display in the Code window the backtrace address selected by the cursor.
Source (<u>S</u>)	Toggle between displaying and not displaying the source name and line number.
Address (<u>A</u>)	Toggle between displaying and not displaying the address.

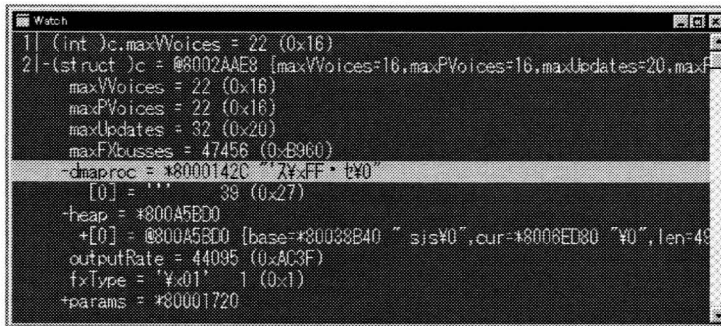
Backtrace Window Mouse Operations

Double click the left mouse button in the backtrace area to display the source code for the selected line in the Code window .

Watch Window

The variables registered for watching are displayed in the Watch window.

Variables with a + at the left of the variable name contain a variable element that can be displayed. When the variable element is displayed, a - will appear to its left.



```

Watch
1 | (int)c.maxVoices = 22 (0x16)
2 | -(struct)c = @8002AAE8 [maxVoices=16,maxPVoices=16,maxUpdates=20,maxF
    maxVoices = 22 (0x16)
    maxPVoices = 22 (0x16)
    maxUpdates = 32 (0x20)
    maxF/busses = 47456 (0xB960)
    -dmaproc = *8000142C "AxFF * t%0"
    [0] = " " 39 (0x27)
    -heap = *800A5B00
    +[0] = @800A5B00 [base=*80038B40 "sis%0",cur=*8006ED80 "%0",len=49
    outputRate = 44095 (0xAC3F)
    fxType = 'Yx01' 1 (0x1)
    +params = *80001720
  
```

Figure 38: Watch Window

Watch Window Shortcut Keys

Commands can be executed in the Watch window using the following shortcut keys. These shortcut keys are valid when the Watch window is active.

Key Operation	Function
F3	Clear all registered variables.
F6 CTRL+I	Inspect the selected variable.
F7 CTRL+W	Register the selected variable in the Watch window.
.	Toggle between displaying and not displaying variables with variable elements (variables with a + or - to the left of the variable name).
INS	Display the dialog box for adding watch registrations.
DEL	Clear the selected variable.

Watch Window Local Menu

Watch (W)	
Inspect (I)	F6, Ctrl+I
Watch (W)	Ctrl+W
View Elements (+)	Enter
Add Watch (A)	Ins
Clear (C)	Del
Clear All (I)	F3

The [Watch] menu provides commands for controlling the Watch window.

Menu Item	Function
Inspect (I)	Display the selected variable in the Inspect window.
Watch (W)	Register the selected variable in the Watch window.
View Elements (±)	Toggle between displaying and not displaying the contents of the local variables with elements such as arrays and structures.
Add Watch (A)	Display the dialog box for entering variable names to be registered in a new Watch window
Clear (C)	Clear the selected variable(s) from the Watch window
Clear All (L)	Clear vars registered in the Watch window.

Watch Window Mouse Operations

Double click the left mouse button on a variable with variable elements to toggle between displaying and not displaying the variable element.

Break Window

Currently set breakpoints are displayed in the Break window.

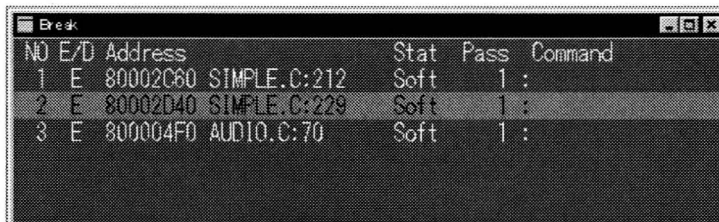


Figure 39: Break Window

Break Window Shortcut Keys

Commands can be executed in the Break window using the following shortcut keys. These shortcut keys are valid when the Break window is active.

Key Operation	Function
F3	Clear all breakpoints.
F4	Save the current breakpoint settings.
F6	Retrieve the breakpoint settings saved with F4.
F7	Display the dialog box for setting the hardware breakpoint. See Hardware Breakpoint Set Dialog Box (pg. 135).
F9	Toggle between enabling and disabling the currently selected breakpoint.
INS	Display the dialog box for setting software breakpoints. See Software Breakpoint Set Dialog Box (pg. 134).
DEL	Clear the selected breakpoint(s).

Break Window Local Menu

Break (B)	
Set Break (A)..	Ins
Disable (D)	F9, Enter
Clear (C)	Del
Clear All (I)	C
Save Settings (S)	F4
Load Settings (L)	F6
Hardware Break (H)..	F7

The [Break] menu provides commands for controlling the Break window.

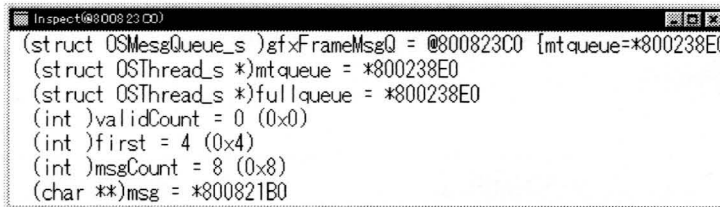
Menu Item	Function
Set Break (<u>A</u>)	Display the dialog box for setting new software breakpoints. See Software Breakpoint Set Dialog Box (pg. 134).
Disable (<u>D</u>)/Enable (<u>E</u>)	Toggle between enabling and disabling the breakpoint at the cursor position.
Clear (<u>C</u>)	Clear the breakpoint at the cursor position.
Clear All (<u>I</u>)	Clears all the breakpoints.
Save Settings (<u>S</u>)	Save the current breakpoint settings.
Load Settings (<u>L</u>)	Load previously saved breakpoint settings.
Hardware Break (<u>H</u>)	Display the dialog box for setting the hardware breakpoint. See Hardware Breakpoint Set Dialog Box (pg. 135).

Break Window Mouse Operations

Double click the left mouse button on a breakpoint to enable and disable the breakpoint.

Inspect Window

The Inspect window displays specified variables in a format corresponding to their data structure.

A screenshot of a debugger's 'Inspect' window. The window title is 'Inspect (@800823C0)'. The content shows a list of variables and their values in a structured format:

```
(struct OSMsgQueue_s)gfxFrameMsgQ = @800823C0 {mtqueue=*800238E0
(struct OSThread_s *)mtqueue = *800238E0
(struct OSThread_s *)fullqueue = *800238E0
(int)validCount = 0 (0x0)
(int)first = 4 (0x4)
(int)msgCount = 8 (0x8)
(char **)msg = *800821B0
```

Figure 40: Inspect Window

Inspect Window Shortcut Keys

Commands can be executed in the Inspect window using the following shortcut keys. These shortcut keys are valid when the Inspect window is active.

Key Operation	Function
F5	Toggle between decimal/hexadecimal data display of array variables.
F6 I CTRL+I	Display the selected variable in the Inspect window.
F7 W CTRL+W	Register the selected variable in the Watch window.
F8 V CTRL+V	Display the selected variable in the Command window.
F9 R	Display the dialog box for specifying the display range for the selected variable.
F10 C	Display the dialog box for changing the value of the selected variable.

Inspect Window Local Menu

Inspect (I)	
Inspect (I)	F6, I, Ctrl+I
Watch (W)	F7, W, Ctrl+W
View (V)	F8, V, Ctrl+V
Range (R)..	F9, R
Change (C)..	F10, C
Change Base (D)	F5

Commands related to controlling the Inspect window are handled in the [Inspect] menu.

Menu Item	Function
Inspect (I)	Open an Inspect window for the selected variable.
Watch (W)	Register in the Watch window for the selected variable.
View (V)	Display the selected variable in the Command window.
Range (R)	Display the dialog box to specify the range of the variable element displayed in the Inspect window
Change (C)	Display the dialog box for changing the value of the selected variable.
Change Base (D)	Change the base of the data displayed in the array

Inspect Window Mouse Operations

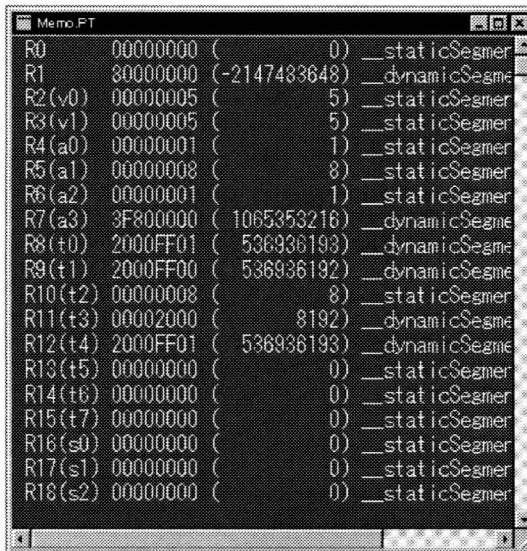
Double click the left mouse button on a variable to display the variable in an Inspect window.

Memo Window

The Memo window is a simple editor. This window supports the clipboard, allowing data to be moved between the various windows.

For example, register values can be copied from the Register window so that the values before a user program is executed can be compared with those after a break. In addition, Partner macro commands can first be entered in the Memo window, then in the Command window.

The contents of this window are saved in a file (MEMO.PT) when Partner-N64PC is exited and loaded when it is started.



```

Memo PT
R0 00000000 ( 0) __staticSegmer
R1 80000000 (-2147483648) __dynamicSegme
R2(v0) 00000005 ( 5) __staticSegmer
R3(v1) 00000005 ( 5) __staticSegmer
R4(a0) 00000001 ( 1) __staticSegmer
R5(a1) 00000008 ( 8) __staticSegmer
R6(a2) 00000001 ( 1) __staticSegmer
R7(a3) 3F800000 ( 1065353216) __dynamicSegme
R8(t0) 2000FF01 ( 536936193) __dynamicSegme
R9(t1) 2000FF00 ( 536936192) __dynamicSegme
R10(t2) 00000008 ( 8) __staticSegmer
R11(t3) 00002000 ( 8192) __dynamicSegme
R12(t4) 2000FF01 ( 536936193) __dynamicSegme
R13(t5) 00000000 ( 0) __staticSegmer
R14(t6) 00000000 ( 0) __staticSegmer
R15(t7) 00000000 ( 0) __staticSegmer
R16(e0) 00000000 ( 0) __staticSegmer
R17(s1) 00000000 ( 0) __staticSegmer
R18(s2) 00000000 ( 0) __staticSegmer

```

Figure 41: Memo Window

Memo Window Shortcut Keys

Commands can be executed in the Memo window using the following shortcut keys. These shortcut keys are valid when the Memo window is active.

Key Operation	Function
CTRL+INS	Copy the selected character string to the clipboard.
CTRL+DEL	Delete the contents of the Memo window.
SHIFT+INS	Paste a character string from the clipboard to the cursor position.
SHIFT+DEL	Copy the selected character string to the clipboard, then delete it.
GRPH+BS	Undo the preceding editing.

Memo Window Local Menu

Edit (E)	
Undo (U)	Alt+BkSp
Cut (C)	Shift+Del
Copy (O)	Ctrl+Ins
Paste (P)	Shift+Ins
Delete (D)	Del
Clear All (L)	Ctrl+Del

The [Memo] menu provides commands for controlling the Memo window.

Menu Item	Function
Paste (P)	Paste the contents of the clipboard
Undo (U)	Undo the preceding edit
Cut (C)	Move the selected character string to the clipboard
Copy (O)	Copy the selected character string to the clipboard
Delete (D)	Delete the selected character string
Clear All (L)	Delete the entire contents of the Memo window

Toolbar

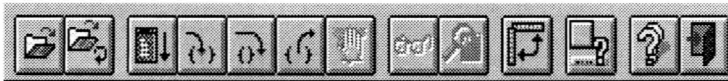


Figure 42: Example Toolbar

The toolbar is used to store Partner-N64PC commands as buttons. These commands can then be called up simply by clicking on the corresponding button.

The method of storing buttons and the functions of the various buttons are explained below. The toolbar can be enabled and disabled with the [View]-[Toolbar] command (See **View Menu** (pg. 63)).

Toolbar Settings

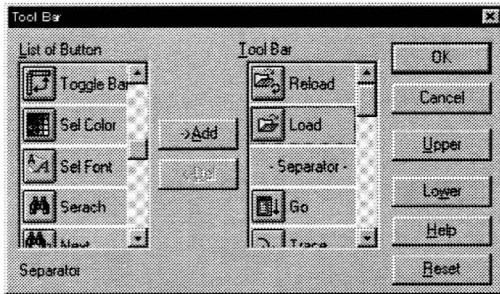


Figure 43: Toolbar Setup Dialog Box

Select the [Settings]-[Toolbar Setup] menu to display the Toolbar Setup dialog box. Select the button to store in the toolbar from the [Button List], and add it to [Toolbar Definition]. For details, see **Toolbar Setup Dialog Box** (pg. 128).

Function of Each Button

The buttons that can be stored in the toolbar are indicated below, along with their functions.

Button Functions



Load debug program (See **Open File Dialog Box** (pg. 127)).



Reload previously loaded program.



Run user program from current PC.



Trace run in units of source lines or machine language commands.



Step run in units of source lines or machine language commands.



Run until execution of current function is ended (Return) and program control returns to calling function.



Forcibly interrupt user program and return control to Partner-N64PC (interrupt linked execution of commands or macros).



Insert the variable at the cursor position in the Watch window. If there isn't a variable that can be registered in Watch at the cursor position, the Watch registration dialog box is displayed. See **Watch Set Dialog Box** (pg. 129).



Display the variable at the cursor position in the Inspect window n. If there isn't a variable that can be displayed in Inspect at the cursor position, the Inspect dialog box is displayed. See **Inspect Setup Dialog Box** (pg. 129).



Toggle Hint mode. In Hint mode, a description of a toolbar button is displayed in the Status window if the mouse cursor is left on the button.



Toggle the toolbar display position.



Display the dialog box for setting the display colors of the active window. See **Specify Colors Dialog Box** (pg. 128).



Display the dialog box for setting the display font of the active window. See **Specify Font Dialog Box** (pg. 127).



Display the dialog box for specifying a character string to search for in the active window. See **Search Character String Dialog Box** (pg. 128).



Search for the previously specified character string from the cursor position in the active window



Display the dialog box for selecting the module(s) contained in the loaded debug program. See **Module Dialog Box** (pg. 130).



Retrieve window setup 1.



Retrieve window setup 2.



Retrieve window setup 3.



Display open windows in an overlapping pattern.



Displays open windows side by side. (Partner-N64PC default arrangement).



Paste character string from the clipboard into the active window.



Copy active window display to the clipboard.



Display the Toolbar Setup dialog box. See **Toolbar Setup Dialog Box** (pg. 128)



Sets the size of each tiled window to match the size of the main window.



Toggle the Code window between source code and reverse assembler display.



Display Partner-N64PC Help.



Exit Partner-N64PC.

Window Bar

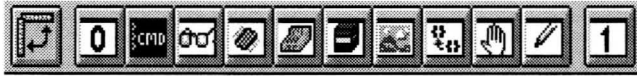


Figure 44: Window Bar

The window bar allows each of the Partner-N64PC windows to be quickly opened. It can be toggled on and off by the [View]-[Window Bar] command. See **View Menu** (pg. 63).

The various buttons and their corresponding windows are shown below.

Button	Window
	Code window
	Command window
	Watch window
	Memory window
	Register window
	Stack window
	Local window
	Backtrace window
	Break window
	Memo window
	Code window 1

The button setup for the window bar cannot be changed.

Status Bar

Items displayed on status bar include the real-time trace address; hints for using shortcut keys, buttons, and menus; error messages; and the Partner-N64PC status display. The status bar can be toggled on and off with the [View]-[Status Bar] command. (See **View Menu** (pg. 63)).

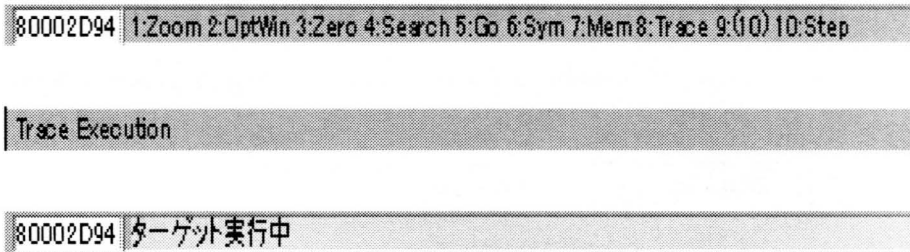


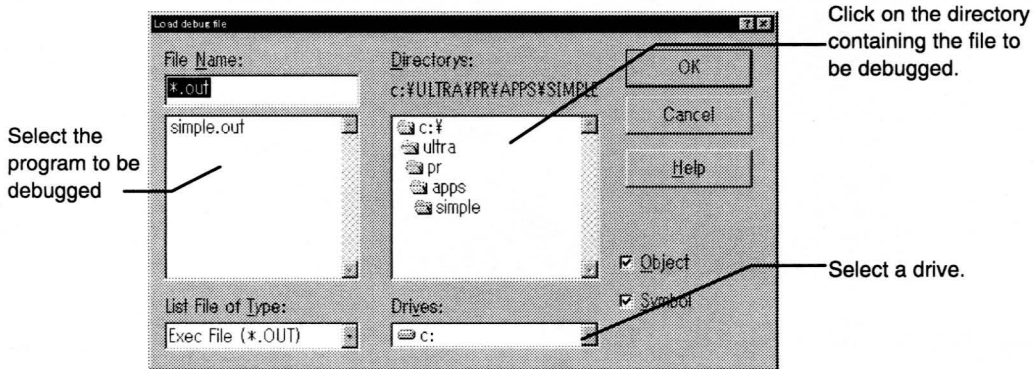
Figure 45: Status Bar Examples

Dialog Boxes

Open File Dialog Box

[File]-[Load] 

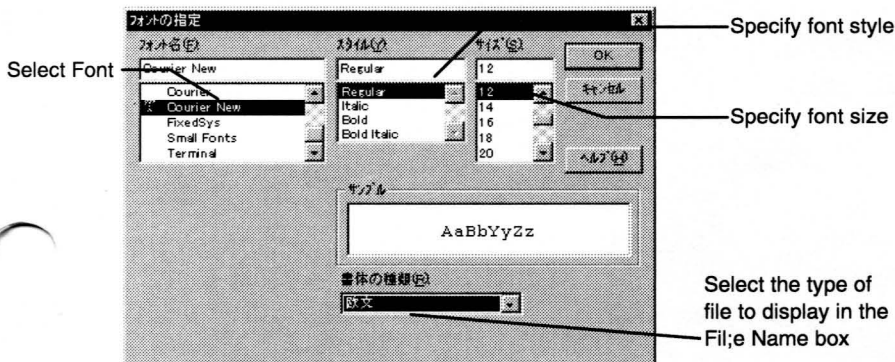
Use the [Open File] dialog box to select and load the program to be debugged.



Specify Font Dialog Box

[Settings]-[Set Font] 

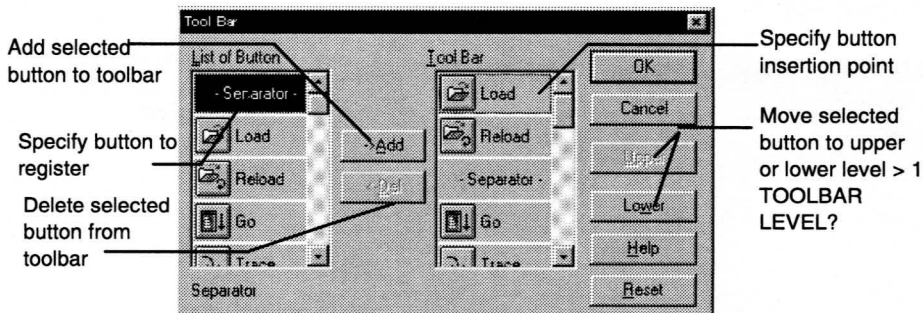
Use the [Specify Font] dialog box to specify the font for the currently selected window.




Toolbar Setup Dialog Box

[Settings]-[Toolbar Setup] 

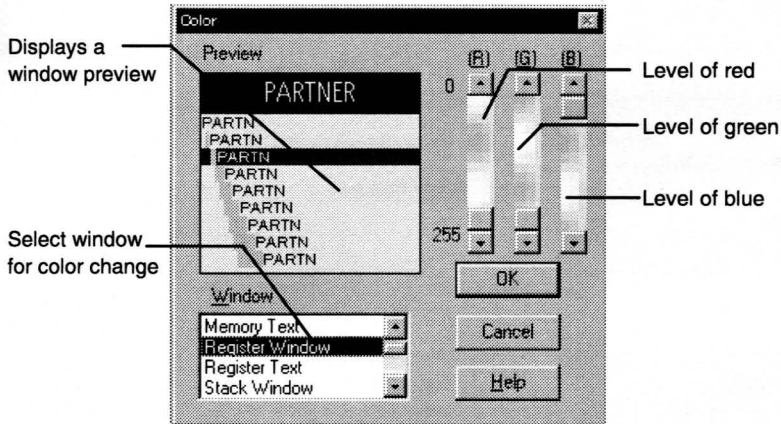
Use the [Toolbar Setup] dialog box to specify the buttons to be shown on the toolbar. Press the DEL key in the [Toolbar Definition] list box to delete a selected button.




Specify Colors Dialog Box

[Setting]-[Specify Colors] 

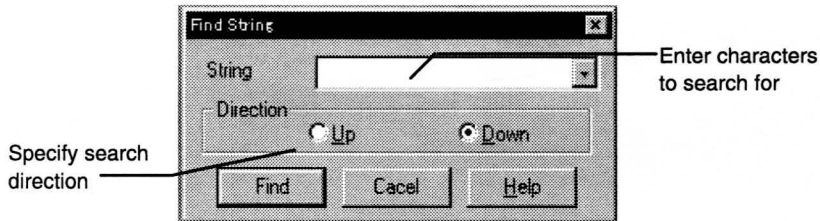
Use the [Specify Colors] dialog box to specify the background color and text color in each window.



Search Character String Dialog Box

[Search]-[Search Character String] 

Use the [Search Character String] dialog box to specify the character string and direction for searching in the active window.

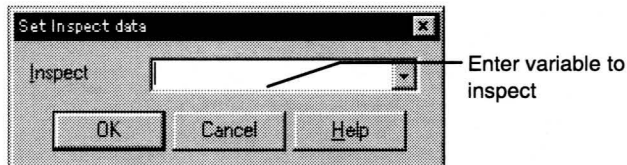


The default search direction is [Up] in the Command window.
The default search direction in all other windows is [Down].


Inspect Setup Dialog Box

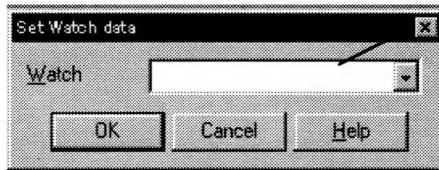
[Code]-[Inspect], [Local/Watch/Inspect]-[Inspect] 

Use this dialog box to specify the variable name to be inspected.



Watch Set Dialog Box

[Code]-[Watch], [Inspect/Local/Watch]-[Watch] 
Use this dialog box to specify the name of the variable to be registered for a watch.



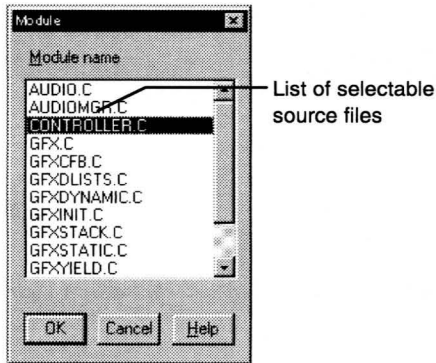
Enter variable to register for Watch

C variables can be stored for watches with this dialog box. Use the W command to directly store memory contents. See on-line help regarding the w command.

Module Dialog Box

[File]-[Module], [Code]-[Module] 

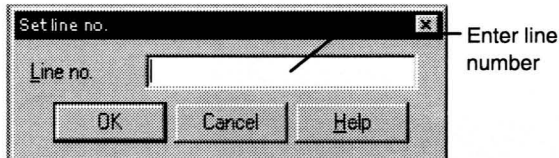
Specify the source file to be displayed in the Code window. The [Module Name] list displays the source filenames defined in the debugging information for the loaded debug files and the filenames referenced by the v command.



Line Number Specification Dialog Box

[Code]-[Line No.]

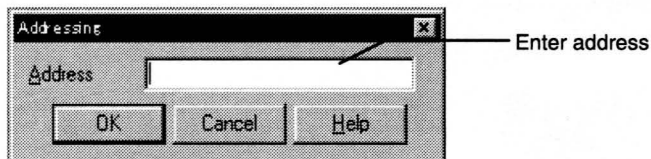
Specify the line number from which to begin display of the source file in the Code window.



Display Address Specification Dialog Box (Code)

[Code]-[Address]

Specify the address from which to begin display of the reverse assembler list file in the Code window.

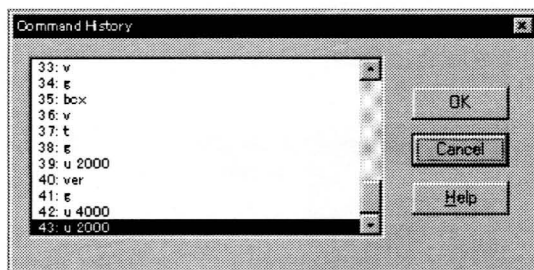


Command History Dialog Box

[Command]-[History]

Display and select the command history entered in the Command window.

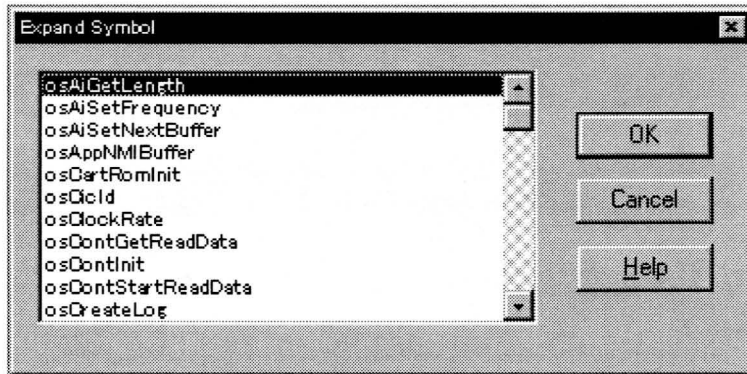
This box displays a list of the character strings that have been entered at the command input line (current prompt) in the Command window, beginning with the most recently entered string.



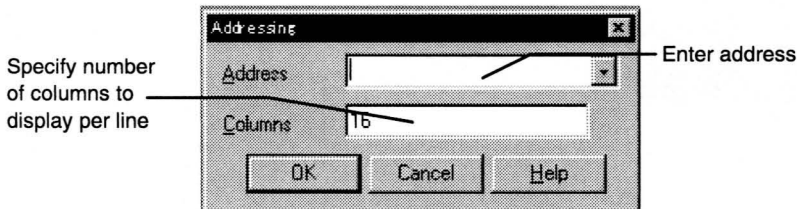
Symbol Extension Dialog Box

[Command]-[Expand Symbol]

This is a list box showing the expansions of the global symbol, starting from the last character string on the command input line. (In the example below, "ins T" has been entered on the command input line.)



Display Address Specification Dialog Box (Memory)



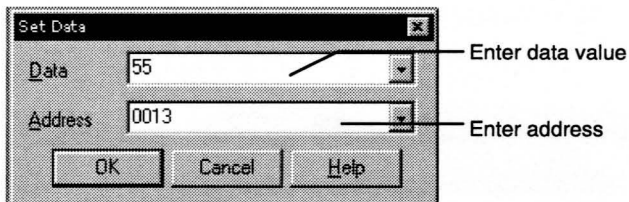
[Memory]-[Specify Address]

Specify the address from which to begin display of the dump list in the Memory window.

Data Set Dialog Box

[Memory]-[Change Data]

Use this dialog box to change the contents of memory.

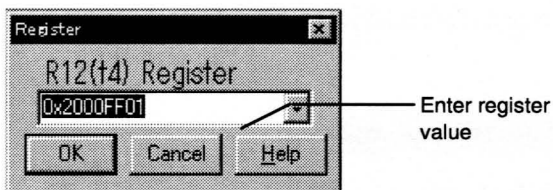


Click on the [OK] button to change the data at the next address. Click on the <Cancel> button to close this dialog box.

Register Dialog Box

[Register]-[Change Value]

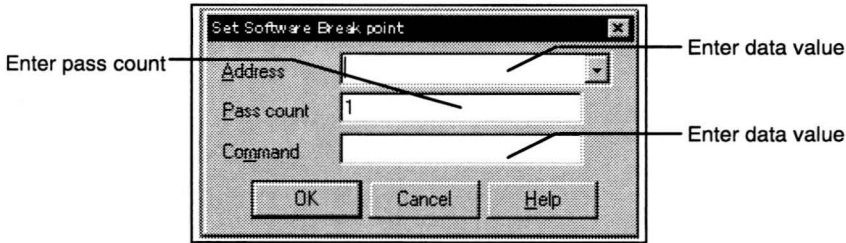
Use this dialog box to change register values.



Software Breakpoint Set Dialog Box

[Break]-[Set Break]

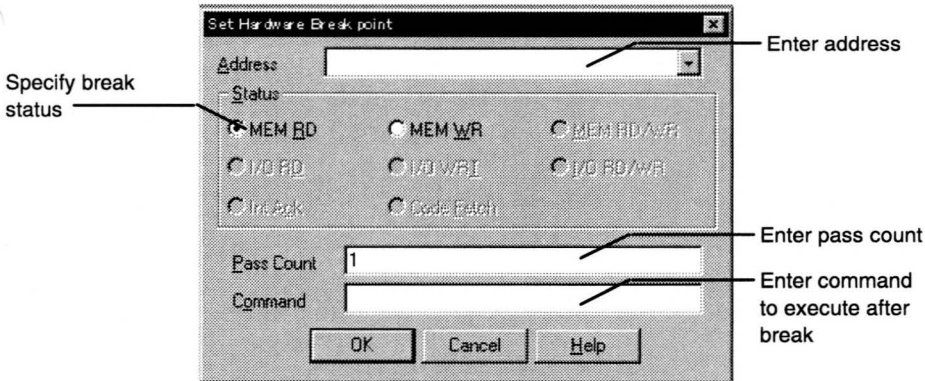
Use this dialog box to set software breakpoints.



Hardware Breakpoint Set Dialog Box

[Break]-[Hardware Break]

Use this dialog box to set the hardware breakpoint.



WINDOW COMMANDS

Chapter 6

Dialog commands

In Partner-N64PC, debugging can be performed using dialog boxes and the toolbar (WINDOWS commands) or by entering dialog commands in the Command window.

Data Expressions

The data expressions that Partner-N64PC handles as addresses and data values are diverse, ranging from symbols to operations.

Symbols Handled by Partner

Partner-N64PC handles two types of symbols. One is global symbols, which are valid in all program areas. The other is local symbols, which are valid only in C functions and represent local or static variables.

Global Symbols

Global symbols can be used in place of address values when entering labels and addresses in reverse assembler. The C `extern` variable and function names are registered by these global symbols. In C, these generally become symbol names with an underscore bar (`_`) either before or after the name of the variable or function. However, entering an underscore bar for each global symbol is inconvenient. With Partner-N64PC, global symbols can be specified without an underscore bar. In addition, Partner-N64PC can be set to discriminate or not discriminate

between upper- and lower-case text. See **OPTION** Command, [Settings]-[Set Options].

If a symbol is given the same name as a CPU register name, the register name takes precedence. Consequently, such symbols cannot be referenced.

Using a global symbol to specify an address

```
>u _main<return>          /*Display reverse assembler from
                             _main symbol value*/
>u main<return>           /* Same as above */
```

Specify Address

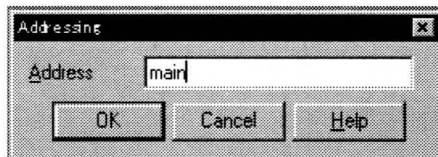


Figure 46: Using the Display Address Dialog Box to enter the Value of Symbol `_main`

Using a global symbol in a macro

```
>i=5<return>                /* 1 */
Symbol has been registered
>while{ i!=0<return>        /* 2 */
?T<return>                 /* 3 */
?i=i-1<return>            /* 4 */
?}<return>                 /* 5 */
>
```

- 1) Assign the value 5 to the symbol `i` (if `i` is an undefined symbol name, register `i` as a symbol).
- 2) The value of symbol `i` is compared with 0.
- 3) If `i` is not equal to 0, a trace run is executed by the `T` command.

- 4) Assign the value $i - 1$ to i .
- 5) This signifies the end of the while { command. When this line is executed, condition 2) of the while { command is evaluated again.

When the above commands are executed, the T command is executed five times before the while{ loop is exited.

In the above example, the symbol i is treated the same as a variable in a higher-level language such as C or BASIC. The symbol name used at this time must not duplicate a previously registered global or local symbol name.

Local Symbols

Local symbols are symbols (statements) for automatic C variables or real function variables that are valid only inside functions and for variables that are declared static. In general, local symbols are automatically registered when the debugging information is loaded.

Local symbols, besides an address, contain information such as the scope (valid range) and type (e.g., int, char, double) of the local symbol.

Special Symbols

`-- _ERR_ --`

The special symbol `-- _ERR_ --` has the value 1 when an error is generated by a previously executed command and 0 when a command is executed normally.

`-- _ERR_ --` can be used for error processing inside a macro command.

`_ _RUN_ _`

The special symbol `_ _RUN_ _` has the value 1 while a user program is run, and 0 during a break. This symbol can be used in a macro when waiting for a break in a user program.

Numeric Values Used by Partner

Partner-N64PC can handle numbers expressed in base 2, 8, 10, or 16. These are distinguished by a symbol placed in front of the value to indicate the base. The treatment of values for which the base symbol is omitted depends the base specified by the N command.

Symbol	Base
<code>@number</code>	2 (binary)
<code>\number</code>	8 (octal)
<code>_number</code>	10 (decimal)
<code>\$number</code>	16 (hexadecimal)
<code>0×number</code>	16 (hexadecimal)
<code>number</code>	according to base specification (16 or 10)

For example, `@11001010`, `\312`, `_202`, `$CA` and `0×CA` all represent the same numeric value.

In addition, there are commands (DS, DL, DT, SS, SL, ST) for handling 4-, 8-, and 10-byte real numbers (IEEE format).

Addresses

A symbol name and/or line number can be entered as an address parameter various commands.

Line Numbers

Debugging work is performed by Partner-N64PC at the source level using line numbers in the source file.

The line numbers are used to specify a target line in the source file. A line number is valid only when the source file line information is included in the debugging information that has been loaded. The formats for three line number input formats are shown below.

Syntax 1 .*[filename:]line number*
Syntax 2 .*±line number*
Syntax 3 .*symbol±line number*

The line number specifies a specific source line in the debug program in a form that combines a decimal (base 10) number and a filename or symbol name.

Syntax 1 indicates an absolute line number. When a filename is entered, the n^{th} line in the specified file is specified. If the filename is omitted, the n^{th} line in the current file (file displayed in the code window) is indicated.

```
>u. kmc:120<return>     /* Specify the 120th line in kmc.c */
>u. 100<return>         /* Specify the 100th line in the
                          currently selected source code */
>u. kmc:120<return>     /* Specify the 120th line in kmc.c */
```

Syntax 2 specifies a position relative to the source line shown by the current program counter. *+line number* specifies a line *line number* of lines after the current source line, and *-line number* specifies a line *line number* of lines before the current source line. However, if there is no line corresponding to the current PC in a Syntax 2 specification, an input error is generated.

```
>v. +10<return>          /* Specify the 10th line from the
                           source indicated by the PC */
```

Syntax 3 specifies a position relative to the source line corresponding to the value (address) of the specified symbol. *+line number* specifies a line *line number* of lines after the corresponding source line, and *-line number* specifies a line *line number* of lines before the corresponding source line. However, if there is no source line corresponding to the specified symbol, an input error is generated.

```
>bp.main+10<return>      /* Specify the 10th line from the
                           symbol
                           main */
```

Character Strings

Character strings (ASCII code strings) can be used in Partner-N64PC instead of numeric values. When used in this way, character strings are enclosed in single quotation marks (').

```
'A' = $41
'AB' = $4142
'ABCD' = $41424344
```

Character strings of up to 16 characters can be specified at one time when entering E/EB commands.

When a single quotation mark (') is contained in a character string, only the text up to the single quotation mark is entered; subsequent text is ignored. To specify single quotation marks ('), enter the numeric value (\$27).

Register Names

The contents of registers can be handled by Partner-N64PC as numeric values.

The register names that can be used are listed below.

Register Names

_R0, _R1, _R2, _R3, _R4, _R5, _R6, _R7, _R8, _R9, _R10, _R11, _R12, _R13, _R14, _R15, _R16, _R17, _R18, _R19, _R20, _R21, _R22, _R23, _R24, _R25, _R26, _R27, _R28, _R29, _R30, _R31, _pc, _hi, _lo

Register Aliases

In addition to the above register names, the register aliases listed below, have been defined by MIPS, can also be used.

_v0, _v1, _a0, _a1, _a2, _a3, _t0, _t1, _t2, _t3, _t4, _t5, _t6, _t7, _t8, _t9, _s0, _s1, _s2, _s3, _s4, _s5, _s6, _s7, _k0, _k1, _gp, _sp, _fp, _ra

```
>while { R0!= R1<return>      /* Compare contents of R0
                               register and R1 register */
? T<return>                  /* Trace run command */
?}<return>                    /* End macro */
>
```

In the above example, the T (trace) command is run until the register values in R0 and R1 are the same. If there is a symbol with the same name as a register, the register name takes precedence.

Operation Expressions

An operation expression consists of numeric values, symbols, and registers combined by operators, and it has a single value. Partner-N64PC uses C-like numeric and logical operators. Operation expressions can be used wherever where values (data, addresses) are specified by commands. The monadic operations and dyadic operators that can be used in operation expressions are listed below.

Monadic Operators		
*		32-bit data at specified address
+		Monadic plus
-		Monadic minus
~		NOT (complement of 1)
!		Logical NOT
Dyadic Operators		
1	*	Multiply
1	/	Divide
1	%	Modular division (remainder)
2	+	Add
2	-	Subtract
3	>>	Right shift
3	<<	Left shift
4	>=	Relational operator (1 when left side is greater than or equal to right side; otherwise 0)
4	<=	Relational operator (1 when left side is less than or equal to right side; otherwise 0)
4	>	Relational operator (1 when left side is greater than right side; otherwise 0)
4	<	Relational operator (1 when left side is less than right side; otherwise 0)
5	==	Relational operator (1 when left side is equal to right side; otherwise 0)
5	!=	Relational operator (1 when left side is not equal to right side; otherwise 0)

Dyadic Operators		
6	&	AND
7	^	XOR
8		OR
9	&&	Logical AND
10		Logical OR
System Functions		
VAL (C expression)	Evaluate contents of parentheses as a C expression See C Expressions (pg. 147)	

The number in the column to the left of each operator indicates the precedence of the operator. When the precedence of adjacent operators is the same, the expressions are evaluated from left to right. However, the precedence of an expression can be changed through the use of parentheses.

In addition, relational operators and the logical AND and logical OR operators are available for conditional processing in macros (e.g., for{, while{ commands) and for conditional processing commands (e.g. if{ commands).

```
>h -(1+2*3) <return>
    oct      dec      hex      asc      float
37777777771  -7      FFFFFFF9  '....'  -6.805644e++38
>
```

Data Expressions at the C-Language Level

Expressions that process (compute) global symbols and local symbols or line numbers as simple address values were explained in **Data Expressions**. Most of these expressions are valid for use by dialog commands. However, the expressions in the debug program are, of course, entered according to C protocol. The preceding expression processing is not adequate to handle these expressions. Therefore, Partner-N64PC now provides dialog and window commands that can handle C expressions. Specifically, C expressions can be handled in their original C syntax by Inspect-related commands, Watch Registration commands, the VAL command, and the ? command.

C Expressions

The evaluation of expressions at the C-language level differs from that of the expressions described in **Data Expressions**, even though the expressions may be identically entered. These differences are explained using the example of the C global variable abc.

```
>d abc<return> /* Display memory from address of variable abc */
00001000 00 01 02 03 . . . . .
>d abc+10<return> /* Display memory from address of variable abc
+10 */
00001010 AA BB CC DD . . . . .
>? abc<return> /* Display value of variable abc (evaluate as C
expression) */
(int) 1 (0x1)
>? abc+10<return> /* Display value of var abc +10 (evaluate as C
expression) */
(int) 11 (0xB)
>
```

As can be seen from this example, even though they are entered in exactly the same way, the expressions `abc` and `abc+10` have different meanings as a normal expression (the `D` command in the upper two examples) and as a `C` expression (the `?` command in the lower two examples). `abc` is evaluated as a C-language-level variable in `Inspect` and `Watch` and by the `VAL` and the `?` commands. It is evaluated as the address of the variable `abc` in all other commands.

C Variables

The variables or functions that can be used in a `C` expression are limited to those that have been declared in a source file compiled with detailed debugging information specified as an option. In addition, registers can be used as quasi-variables. All register quasi-variables are of the type `unsigned int`.

Register Quasi-variables	Registers
<code>_r0..._r31</code>	R0 register...R31 register
<code>_pc</code>	PC register
<code>_lo</code>	LO register
<code>_hi</code>	HI register
<code>_v0, _v1</code>	Alias for R2, R3 registers
<code>_a0..._a3</code>	Alias for R4...R7 registers
<code>_t0..._t7</code>	Alias for R8...R15 registers
<code>_s0..._s7</code>	Alias for R16... R23 registers
<code>_t8, _t9</code>	Alias for R24, R25 registers
<code>_k0, _k1</code>	Alias for R26, R27 registers
<code>_gp</code>	Alias for R28 register
<code>_sp</code>	Alias for R29 register
<code>_fp</code>	Alias for R30 register
<code>_ra</code>	Alias for R31 register

C Variable Scope

In writing and debugging C programs, the scope (usable range) of variables must be considered. For example, a variable declared `extern` is valid in all program areas; its scope is the entire program area. On the other hand, an automatic variable declared inside a function is valid only while that function is processed; its scope is within that function. If a variable declared `extern` and an automatic variable declared in a function have the same name, only the automatic variable is valid. The `extern` variable is inaccessible. Because an automatic variable in a function not being used does not exist in memory, it cannot be viewed no matter how often it is referenced.

This kind of processing is automatically performed by Partner-N64PC according to the scope information obtained from the debugging information.

Constants

These have the same specifications as for C syntax. In addition, the default base is always base 10 (decimal), regardless of the N command (change-base command) setting.

Format	Base
numeral	Base-10 number
0x numeral	Base-16 number
0X numeral	Base-16 number
0 numeral	Base-8 number

For example, 4096 (base-10), 0x1000 (base-16) and 010000 (base-8) all express the same value. In addition, C escape sequences are supported for text constants.

C text	Number	Meaning
'\a'	0x7	Bell
'\b'	0x8	Backspace
'\f'	0xC	Form feed
'\n'	0xA	Line feed
'\r'	0xD	Return
'\t'	0x9	Horizontal tab
'\v'	0xB	Vertical tab
'\¥'	0x5C	¥ (yen) symbol
'\nnn'	nnn	Base-8 (8 bit)
'\xnn'	nn	Base-16 (8 bit)

Operators

Operators are also supported, using exactly the same operators and the same grammar as C. However, operators other than the = operator (substitution operator) cannot be used with floating point values in Partner-N64PC.

The precedence of the operators is shown below.

Precedence	Operator
1	Function(n) Array(n) n.n n->n n++ n--
2	&n *n -n ~n !n ++n --n size of n #n
3	(Cast)n
4	n%n n/n n*n
5	n+n n-n
6	n<<n n>>n
7	n>n n<n n>=n n<=n
8	n==n n!=n
9	n&n
10	n^n
11	n n
12	n&&n
13	n n
14	nn?nn:nn
15	n=n n*=n n/=n n%=n n+=n n-=n n<<=n n>>=n n&=n n^=n n =n
16	n,n

When the precedence of adjacent operators is the same, the expression is evaluated from left to right. However, the substitution operator (precedence 16) is evaluated from right to left. The precedence of an expression can be changed through the use of parentheses.

Expressions with Side Effects

Assignment operators, such as ++, --, =, and function calls have side effects that change the content of variables of the program being debugged while the operation is processed. There are of course cases in which using an assignment operator to change the value of a variable is desirable. However, in almost all cases, data is relatively rarely changed during debugging simply by referencing data.

Thus, to prevent improper data changes during evaluation of an expression, the use of operators with side effects is prohibited by Partner-N64PC in Watch and Inspect and with the ? command. The exception is that use of operators with side effects is permitted with the VAL command. Therefore, avoid using the VAL command when only referencing data; instead use the ? command or Inspect. The val command should be used only when using operators with side effects for operations such as changing data.

Function calls pose bigger problems. Changes to global variables and static variables, or changes to other data areas by pointers, may also be caused during the processing of the function. Resumption of execution may be prevented if function calls are used without consideration of these potential problems. Particular caution is urged when a function call is used with the VAL command.

```
>? abc=1234<return>
Operator with byproduct cannot be used
>val abc=1234<return>          /* Substitute 1234 for
abc */
(int) 1234 (0x4D2)
>val fnc(1,2,3)<return>       /* Call fnc function */
(int) 10 (0xA)
```

Chapter 7

Command Reference

This section describes the commands that can be entered in the Partner-N64PC Command window. Refer to the on-line help for detailed descriptions including input examples.

Command Conventions

Partner commands consist of a command name and a parameter list. In some cases, parameters are optional. Optional parameters are shown in square brackets ([...]). If there are two or more selectable elements, the element contents are separated by curly braces ({...}) and |. If the parameters are omitted, the Partner default value or the value carried over from the previously executed command is used.

Commands by Function

Initialize Hardware

`INIT`

After initializing the Partner-N64PC hardware and reloading the monitor program, this command resets the target system. If the monitor program crashes or the Partner-N64PC should hang for any reason, the system hangup can be stopped by pressing `Ctrl+Shift+Alt`. Then execute the `INIT` command to initialize the system, and reload the user program using the `LP` command. These commands can also be used to reset the target system.

Reset CPU

RESET

When the target CPU is running, reset outputs a pulse to the reset probe connected to the target system. When the target CPU is idle, it performs the same operation as the INIT command.

Load Program

L *filename*

Loads the debug program and debugging information.

LP *filename*

Loads the debug program.

LS *filename*

Loads the debugging information only.

Read/Write File

RD *filename, address*

Reads the specified file to the specified address.

WR *filename, range*

Writes the memory contents in the specified range to the specified file.

ROM Control

ZROM [ROM size] [out-of-range access break] [ROM write break]

Specifies the size of the emulation ROM.

Enables or disables breaks for ROM out-of-range access or for writing to the ROM area.

TR CAS, ROM, [*size MB*]

Transmits data from the cartridge (cassette) to emulation ROM.

TR CAS, *filename*, [*size MB*]

Transmits data from cartridge (cassette) to a file.

TR ROM, *filename*, [*size MB*]

Transmits data from emulation ROM to a file.

TR *filename*, ROM, [*size MB*]

Transmits data from a file to emulation ROM.

Run Program

T [*iterations*]

Trace-runs the debug program (F8).

P [*iterations*]

Step-runs the debug program (F10).

G [=run address] [*break address*] [, /C] [, /W]

G@ [, /C] [, /W]

Runs program (F5, F7).

/C: continue real-time count; /W: inhibit on-the-fly

ESC

Forcibly breaks the user program (ESC).

Break Point

BP [*address* [, *iterations*] [, *command*]]

Sets a software breakpoint at the specified address (F9).

BC *list*

Cancels the software breakpoints specified in the list.

BD *list*

Disables the software breakpoints specified in the list.

BE *list*

Enables the software breakpoints specified in the list.

Hardware Break Point

BH *address* [, *CPU status*] [, *user status*] [, /*iterations*]
[, /*C command*]

Sets a hardware breakpoint. When a command is specified, executes the command when the breakpoint is reached.

CPU status: MEM, MR, MW

BH

Displays the current hardware breakpoint status.

BHE

Enables the hardware breakpoint.

BHD

Disables the hardware breakpoint.

BHC

Cancels the hardware breakpoint.

Display/Change Register

R

Displays the register and flag values.

register=expression

Changes the specified register to the value of the expression.

R *register*

Changes the register value.

Display/Change CP0 Register

CP0

Displays the CP0 register.

CP0 CP0 *register=data*

Changes the specified register to data.

<CP0 register>

INDEX, RANDOM, ENTRYLO0, ENTRYLO1, CONTEXT, PAGEMASK,
WIRED, BADVA, COUNT, ENTRYHI, COMPARE, STATUS, CAUSE, EPC,
PRID, CONFIG. LLADDR, WATCHLO, WATCHHI, XCONTEXT, ECC,
CACHEERR, TAGLO, TAGHI, EEP**Display/Change FPU Register**

FPU[D]

Displays the FPU register.

FPU[D]

FPU *register=data*

Changes the specified register to data.

<FPU register>

FCR31, F0, F1, ..., F7

Display/Change TLB Register

TLB [TLB register No.] [, TLB register No. E]

Displays the contents of TLB for the specified range.

TLB TLB register No., mask, entry Hi, entry Lo0, entry Lo1

Changes the specified TLB contents.

Display/Change Memory

D [*format*] *range* [, *iterations*] [, *base*]

Displays the contents of memory inside the *range*, in the specified format and according to the base specification.

E [*format*] *address*

Displays the contents of memory in the specified format, starting from the specified *address*.

F [*format*] *range*, *list*

Fills the specified *range* with the listed values in the specified format.

S [*format*] *range*, *list*

Searches the specified for the listed memory pattern in the specified *format*.

C *range*, *address*

Compares the specified *range* with the specified *address*.

M *range*, *address*

Block moves the specified *range* to the specified *address*.

Display Expression

H *expression*

Displays the value of the *expression* in base 8, 10, 16, ASCII, and real numbers.

H *expression 1*, *expression 2*

Displays the sum and the difference of *expression 1* and *expression 2*.

PRINTF form [, *parameter*]

PF form [, *parameter*]

Displays in the same format as the C function `printf ()`.

Display/Set Symbol

X [*symbol name*]

Displays the *symbol name* (all symbols if not specified).

[.] *name=address*

Registers (changes) the named symbol to the specified *address*.

Reference/Change C Data

INS C *expression* [, *function*]

Evaluates the C *expression* and displays it in the Inspect window (F6, Ctrl+I).

W? C *expression*

Registers the C *expression* in the Watch window (Shift+F7, Ctrl+W).

W [*format*] *address* [, *range*] [, *base*]

Registers the contents of memory specified by the *address* and *range* in the Watch window.

Y *list*

Cancels the Watch lines specified in the list.

VAL C *expression* [, *function*]

? C *expression* [, *function*]

Evaluates and displays the C *expression*.

Define Character String (Evaluate C Equation)

DEF *character string 1 character string 2*
#DEFINE *character string 1 character string 2*
Defines a *character string* for the mini pre-processor.

DEF
Displays the currently registered *character string* definitions.

DEF *
Disables all of the currently registered *character string* definitions.

Display Code

V [.] [*filename:*] [*line*]
Displays the specified *line* of the specified file in the Code window.

V *function name*
Displays the source file for the specified function in the Code window.

U [*address*]
Performs reverse assembly display from the specified *address* in the Code window.

UPUSH [*address*]
PUSHes the currently displayed *address* to the address stack (eight-level internal stack) and displays reverse assembly from the specified *address*.

UPOP
Display reverse assembly from the last UPUSHed *address*, and POPs the *address* stack.

UEND
Displays reverse assembly from the last UPUSHed *address*.

Assemble

A *address*

Assembles from the specified *address* and directly expands into memory.

Backtrace

K

Performs backtrace display of a C function.

Real Time Counter

RTC [ON | OFF]

Sets automatic display of the real-time counter at a breakpoint to *ON* or *OFF*.

RTC

Displays the current real-time counter value.

Profile

PROF

Displays the profile results.

PROF ON

Profile ON

PROF OFF

Profile OFF

PROF CLR

Clears profile results.

The profile sample timing in the Windows version is fixed at 55 msec.

System Call

SYSC address

Turns ON the system call with the *address* specified by *address* as the entry point.

SYSC OFF

Turns OFF the system call function.

SYSC

Displays the status of the system call function.

System Control

EXIT

Exits Partner.

Q

Quits Partner.

HELP

Displays Help.

VER

Displays the Partner version.

!! Displays the history.

! character string

Searches the history for the character string.

Change Base

N base

Sets the input radix to either base 10 (decimal) or base 16 (hexadecimal).

Logging/Batch

> *filename*

Outputs the display/input process in the Command window to a file.

>> *filename*

Performs APPEND logging output to the specified file.

>

Interrupts logging (closes log file).

< *filename*

Inputs to Command window from a file. The batch process can be interrupted with the ESC key.

I/O Port Input/Output

PI [*format*] *address* [, /C]

Displays the contents of the specified memory *address* in the specified *format*.

PO [*format*] *address*, *data* [, /C]

Outputs the specified memory *address*, *data* in the specified *format*.

Set Options

OPTION [*EXE*] [, *CASE*]

Sets the various options. (SHIFT+F10)

<Option>	<Parameter>	<Content>
<i>EXE</i>	(0/1/2)	Run mode
<i>CASE</i>	(ON/OFF)	Discriminate between
		lower/upper <i>case</i>

Screen Control

CLS

Clears the Command window.

HOME

Moves the cursor in the Command window to the home position.

LOCATE *X coordinate, Y coordinate*

Moves the cursor in the Command window to the specified position.

LALL

Displays macro output.

SALL

Suppresses macro display output.

LIST

Displays Command window output.

NLIST

Suppresses Command window display output.

BEL

Sounds bell.

TIME

Displays current time (HH:MM:SS).

WAIT

Temporarily suspends operation.

PROMPT *text*

Changes the prompt to the specified *text*.

*

Specifies a comment line.

Macro Command

{ *macro name*

Registers the body of the macro as the *macro name*.

DO { } WHILE *expression*

Macro command identical to `do..while` statement in C.

FOR { }

Macro command identical to `for` statement in C.

WHILE *expression*

Macro command identical to `while` statement in C.

REPEAT { *parameter*

Repeat macro command.

BREAK

Escapes from macro.

KILL *macro name*

Cancels the macro.

MLIST [*macro name*]

Displays the macro.

MLIST >*filename*

Writes all registered macros to the specified file.

< *filename*

Loads macro from the specified macro file.

IF { *expression*

Conditional control command identical to `if`, `elseif`, `else` statements in C.

Chapter 8

SPECIAL COMMANDS AND THEIR USE

On-the-Fly Function

The on-the-fly function allows memory and I/O to be viewed and modified while the user program is running. Consequently, important commands such as the `D` or `E` commands can be run even during user program execution. However, the contents of Watch windows other than the Real-Time Watch and Register windows are not updated until a breakpoint is reached.

Partner's on-the-fly function is provided by temporarily breaking the user program, executing the specified command, then resuming program execution. Therefore, please note that the real-time aspect of the user program is lost when the user inserts a command. When the on-the-fly function is used, the message "Warning: CPU temporarily stopped" is displayed at the top of the screen.

Execution of some commands is prohibited while the user program is running. If execution of such a command is attempted, the message "Target running. Cannot use" is displayed. Execute the command again after a breakpoint in the user program.

The on-the-fly function will not work when a system call is in use, that is, when a system-call entry address has been specified by the `SYSC` command.

System Calls

Two system calls are provided for the user program, one for keyboard input, the other for screen display. These functions can be used to display the status of the user program during the debugging or when keyboard input is needed.

To execute a system call, specify a system-call entry point with the `SYSC` command. This sets a software breakpoint for the system call at the specified *address*. The *address* specified as the entry point must be an NOP instruction. When this address is executed, the user program temporarily breaks, and the specified function is executed. The user program then resumes execution.

When a system call is used, key operations other than the forced-break key command are disabled during user program execution. Commands are not accepted until the program reaches a breakpoint or until a break is forced using `CTRL+ALT`. In other words, the on-the-fly function cannot be used.

Example

>sysc syscall<return> Specifies the address `SYSCALL` as the entry point.

>sysc off<return> Stops use of the system call.

Single Character Output

A single-character-output call displays a single specified character in the Command window of debug screen.

CALL

B register (upper 8 bits of ACC) = \$0

A register (lower 8 bits of ACC) = ASCII code

Call the system-call entry point

RETURN

None

Example

```
rep      #$20
```

```
lda      #$0041      * ASCII code 'A'
```

```
jsr      syscall     * Output one 'A' character
```

```
      .
```

```
syscall: nop
```

```
      rts
```

Single Character Key Input

A single-character-key input call checks the status of the keyboard and, when a key is pressed, returns the ASCII code for that character. If no key is pressed, it returns a 0 [*zero*].

CALL

B register (upper 8 bits of ACC) = \$1

Call the system call entry point

RETURN

A register (lower 8 bits of ACC) = ASCII code

When 0 [*zero*], no input

Example

```
                                rep      #$20
loop:                            sta      #$0100
                                jsr      syscall      * Key input
                                beq      loop        *Loop if return
code is 0
syscall:                          nop
                                rts
```

Technical Supplement

A software breakpoint is placed at the entry *address* specified by the `SYSC` command. When the program is suspended by this breakpoint, the B register (upper 8 bits of ACC) is interpreted as the function number of the system call.

B register=&l

Key input

The key code is set in the A register and the user program resumes. If there is no keyboard input, a 0 [zero] is returned.

B register=\$0

Single Character Output

The ASCII character code stored in the A register is displayed in the Command window.

The system call function and profile function cannot be used together.

Hardware Breakpoints

Partner-N64PC breakpoints comprise hardware breakpoints and software breakpoints. A software breakpoint is also called an execution pre-break: the user program is stopped immediately before executing the instruction at a set address.

Software breakpoints can be set only in the first opcode section of the code area in a user program that has been loaded into emulation ROM or RAM. When performing a software breakpoint, a break instruction is inserted immediately before user program execution.

Hardware breakpoints, also called access breakpoints, suspend the user program when the memory or I/O has been accessed by the CPU at a specified *address* and *status*. A hardware breakpoint can be set at any address in the memory space. A bus status, such as read or write, can also be specified.

Using the Hardware Breakpoint

A hardware breakpoint breaks the program when the CPU accesses a specified *address*. Unlike a software breakpoint, a hardware breakpoint occurs when a memory or I/O access run cycle is detected in the hardware. Parameters that can be specified as break conditions are *address*, *data*, and CPU *status* (MR< MW, MEM, CF).

Only one hardware break can be specified.

Use the BH command to set the hardware breakpoint. Hardware breakpoints can also be enabled or disabled using the BHE and BHD commands.

The format of the BH command is shown below.

Syntax BH *address* [, *status*] [, *IM*] [*data*] [, /*bus count*] [, /*C command*]

address

Enter the *address* at which to set the breakpoint.

status

Any of the 4 status types shown below can be specified.

Status	Meaning
MEM	Memory read/write cycle
MR	Memory read cycle
MW	Memory write cycle
CF	Code fetch cycle

Table 1: Status

If the status is omitted, the default status MEM is used.

IM

Access to the image space can also be a hardware break condition. Supported image spaces are I/O space, internal working RAM space, and backup RAM space. For example, if the *address* \$04201 is specified as the break *address*, access to the image space generated in BANK 0 through BANK 3F and BANK 80 through BANK BF becomes a break condition. If IM is not specified, access to the image space is not a break condition.

data

Data can be specified as a break condition. If omitted, data is not a break condition. A mask (dont care) can also be specified in bit units.

/bus count

Specifies the bus count. A break is generated when the break condition reaches the specified bus count.

For details regarding the commands, refer to the sections that discuss the BH, BHE, BHD commands.

Controlling Emulation ROM

The Partner-N64PC has built-in memory for emulating Game Pak ROM. The maximum memory capacity is 32 MB. However, this capacity can be set to 0 , 4 , 8 , 16, or 32 MB.

Some emulation ROM controls generate a break when an erroneous write is executed to the ROM area. A break can also be generated when the CPU accesses the Game Pak connector (AD 16 bus) outside the ROM area.

These controls are implemented by the Z command.

Syntax ZROM [*ROM capacity*][[, [*ROM out-of-range access break*][[, *ROM write break*]]]

The first parameter of the ZROM command is the ROM capacity. The second parameter is the ROM out-of-range access break. The third parameter is the ROM write break.

ROM capacity

This specifies the ROM capacity being emulated in MBs. The capacities that can be specified are 0, 4, 8, 16 or 32.

If 0 is specified, emulation memory is disabled. In this case, access to the ROM space is provided by the Partner-N64PC Game Pak connector.

When the debugger is started, ROM capacity is set to the maximum value.

ROM out-of-range access break

This specifies whether to generate a break when an AD16 bus (Game Pak connector) outside the ROM range is accessed.

Specify ON to enable an out-of-range access break and OFF to disable an out-of-range access break. The default setting is ON.

ROM area write break

This specifies whether or not to generate a break when a write is executed to the emulation ROM area.

Specify ON to enable a ROM area write break and OFF to disable a ROM area write break. The default setting is ON.

If all parameters are omitted, the current status is displayed when the ZROM command is executed.

Example

In the example below, emulation ROM capacity is set to 8 MB, the ROM out-of-range access break is disabled, and the ROM area write break is enabled.

>ZROM 8, OFF, ON<return>

With the following parameters, the ROM area write break is disabled, and the other parameters are left unchanged.

>ZROM, , OFF<return>

Transfer Command

The RD and WR commands transfer programs and data between emulation ROM and files. The TR command transfers programs and data between the ROM on the cartridge inserted into the Partner-N64PC Game Pak connector, emulation memory, and files. For details regarding each command, refer to the on-line help.

Syntax 1 TR CAS, {ROM | *filename*} [, {*size*} |
{/*address S* [, *address E*]}]

Syntax 2 TR ROM, {CAS | *filename*} [, {*size*} |
{/*address S* [, *address E*]}]

Function

Transfer command

Description

Transfers data between 3 locations: the Game Pak connector, emulation memory, and a file.

Syntax 1 transfers data from the Game Pak inserted into the Game Pak connector to emulation ROM and to a file.

Syntax 2 transfers data from emulation ROM to a RAM cartridge inserted into the Game Pak connector and to a file. Data cannot be transferred to a ROM cartridge.

size is specified in megabits.

To specify the *address range*, place a forward slash (/) in front of the start address *address S*, as shown in the example below. If the end address *address E* is omitted, data are transferred until the end of the target memory space is reached.

If the *size* and *address range* are omitted, data the size of the emulation ROM are transferred.

TR CAS, ROM	Transfers ROM on ROM cartridge to emulation ROM.
TR CAS, File	Transfers ROM on ROM cartridge to a file.
TR ROM, CAS	Transfers emulation ROM to ROM on ROM cartridge.
TR ROM, File	Transfers emulation ROM to a file.

Example

```
>tr rom, m:test.dat<return>  
  Transfer from E-ROM to File  
>tr rom, m:test.datm /B0000000, B000FFFF<return>  
  Transfer from E-ROM to File  
>tr rom, m:test.dat, 4<return>  
  Transfer from E-ROM to File  
>tr cas, rom, 8<return>  
  Transfer from CASSETTE (CARTRIDGE) to E-ROM  
>
```

Profile Function

Profile calculates the time required to run each function (subroutine) during execution of the user program. By analyzing how much CPU time each function consumes, it is possible to determine which portions of a program particularly require optimization and how much overall performance can be improved.

To use the profile function, refer to the section on the PROF command.

Game Pak Connector

The Game Pak or a backup RAM cartridge can be inserted into the Game Pak connector.

When inserting or removing a Game Pak or device from the Game Pak connector, be sure to turn the power off to the personal computer and modified N64 Control Deck. If a device is inserted or removed with the power on, initialize the debugger with the INIT command.

The Game Pak connector cannot be accessed from the CPU when emulation ROM in the debugger unit is enabled, that is, when the ROM *size* in the ZROM command is non-zero. To access data on an inserted cartridge, specify the size of the emulation ROM as 0 with the Z command. In this case, the access speed during monitoring, the ROM access speed, must be set to the same value as the game program access speed. Refer to **Configuration of Partner-N64PC** (ROM SPEED/MONITOR SPEED) regarding access speeds.

A game program can also be run using this connector, but software breakpoints cannot be set. Software breakpoints can be set if the program on the Game Pak is run after being transferred to the emulation memory in the debugger by the method described above (See **Transfer Command**, pg. 175).

In addition, data can be transferred between memory in the inserted cartridge, emulation memory built into the debugger, and a file. See the `TR` command description in the on-line help for details on data transfer commands.

Index

- .dat
 - Files created on exiting, 40
- .INI File
 - Files Needed for Startup, 39
- Address, 20, 21, 32, 58, 59, 73, 79, 88, 89, 107, 109, 113
- Assemble, 133
- Backtrace, 60, 62, 87, 88, 133
- Backtrace window, 87, 88
- Backtrace Window, 87
 - Local Menu, 88
 - Mouse Operations, 89
 - Screen Configuration, 51
 - Shortcut Keys, 88
- Base change, 94
- Batch, 134
- BH command, 129, 142
- BHD command, 129, 142, 143
- BHE command, 129
- Break point, hardware, 128
- Break Points, software, 128
- Break window, 91
- Break Window
 - Local Menu, 92
 - Mouse Operations, 93
 - Screen Configuration, 51
 - Shortcut Keys, 91
- breaker, 42
- Breaker, 21
- Button functions, 98
- C data, reference/change, 131
- C expressions, 120
- C variable scope, 122
- C variables, 106, 114, 121
- Change Base, 134
- Character string definition, 132
- Character strings, 117
- Code window, 28, 37, 38, 51, 53, 57, 65, 68, 69, 70, 71, 73, 74, 101, 107, 132
- Code Window
 - Local Menu, 72
 - Mouse Operations, 74
 - Screen Configuration, 50
 - Shortcut Keys, 70
- Command window, 74, 75
- Command Window
 - Local Menu, 77
 - Screen Configuration, 50
 - Shortcut Keys, 75
- Configuration, 15, 32, 39, 43, 148
- Constants, 122
- Device address, 32
- Dialog boxes, 103
- Dialog Boxes
 - Command History, 108
 - Expand Symbol, 108
 - Inspect Setup, 73, 99, 106
 - Module Dialog Box, 106
 - Open File, 103
 - Search Character String, 105
- Data Set, 109
- Hardware Break Point Set, 110
- Software Breakpoint Set, 110
- Set Various Conditions, 110
- Register, 109
- Display Address Specification, 107
- Specify Colors, 104
- Specify Font, 103
- Line Number Specification, 107
- Toolbar Setup, 104

- Watch Set, 106
- Dialog commands, 112
 - Data Expressions, 112
 - Data Expressions at the C Language Level, 120
- DIP switches, 19, 24, 42, 44
- Display Code, 132
- Display Expression, 131
- Display/Change CPO Register, 129
- Display/Change FPU Register, 129
- Display/Change Memory, 130
- Display/Change Register, 129
- Display/Change TLB Register, 130
- Display/Set Symbol, 131
- DSW1, 20
- DSW2, 20
- DSW3, 20
- DSW7, 21
- Error Messages at Startup, 41
- Exiting, Files Created On, 40
- expansion slot, 22, 42
- Expressions with Side Effects, 124
- external power supply, 21
- File Read/Write, 127
- Files Created on exiting, 40
- Files Needed for Startup, 39
- game cartridge connector, 25, 144, 145, 146, 147
- Global Symbols, 112
- Hardware breaks, 13, 142
- Hardware Initialization, 126
- Hardware Installation, 15
- Help Menu, 64
- I/O address, 20, 21, 33, 42, 43
- I/O port, 42
- I/O Port Input/Output, 135
- INIT.MCR, 39
- Initialize Hardware, 126
- Inspect window, 10, 87, 90, 93, 94, 95, 99, 131
- Inspect Window, 93
 - Local Menu, 94
 - Mouse Operations, 95
 - Screen Configuration, 51
 - Shortcut Keys, 93
- Installing the Interface Board, 22
- interface board, 12, 15, 17, 19, 22, 23, 32, 42
- Interface Board, 18
- Line Numbers, 115
- Load Program, 127
- Local Menus in Each Window
 - Backtrace Menu, 60
 - Break Menu, 60
 - Code Menu, 57
 - Command Menu, 57
 - Inspect Menu, 61
 - Local Menu, 59
 - Memo Menu, 60
 - Memory Menu, 58
 - Register Menu, 58
 - Stack Menu, 59
 - Watch Menu, 59
- Local symbols, 114
- Local Symbols, 114
- Local window, 10, 86
- Local Window, 85
 - Local Menu, 87
 - Mouse Operations, 87
 - Screen Configuration, 50
 - Shortcut Keys, 86
- Logging, 134
- Macro Command, 136
- MAP, 32, 33, 34, 36
- Memo window, 51, 54, 96
- Memo Window, 95
 - Local Menu, 97

- Screen Configuration, 51
- Shortcut Keys, 96
- Memory window, 10, 40, 58, 59, 60, 61, 78, 79, 109
- Memory Window, 77
 - Local Menu, 79
 - Mouse Operations, 80
 - Screen Configuration, 50
 - Shortcut Keys, 78
- Menus, 28, 52
 - Edit Menu, 53
 - File Menu, 53
 - Local Menus in Each Window, 56
 - Run Menu, 55
 - Search Menu, 54
 - View Menu, 55
- Module, 53, 57, 72, 73, 106, 107
- MONITOR INIT. STACK, 32, 41
- Monitor Program (MON_N64.EPT)
 - Files Need for Startup, 39
- Mouse Operation, 67
- Mouse Operations
 - Backtrace Window, 89
 - Break Window, 93
 - Click Right Button, 67
 - Drag Left Button, 68
 - Inspect Window, 95
 - Local Window, 87
 - Memory Window, 80
 - Mouse Operations, 74
 - Register Window, 84
 - Watch Window, 91
- On-the-Fly Function, 138
- Operation Expressions, 118
- Operators, 118, 123
- Options Settings, 135
- Partner N64 PC, Basic Operation
 - Procedure, 27
- Partner N64 PC, Features, 8
 - Partner N64 PC, Window
 - Configuration, 48
 - PORT ADDRESS, 21, 32, 33, 35, 42
 - Power ON, 24
 - Profile, 10, 133, 147
 - Program, Load, 127
 - Program, Run, 128
 - Real Time Counter, 133
 - Register Names, 117
 - Register window, 10, 80, 95, 138
 - Register Window, 80
 - Local Menu, 83
 - Mouse Operations, 84
 - Shortcut Keys, 82
 - Related Files, 38
 - Reset CPU, 127
 - ROM Control, 127
 - Run Program, 128
 - Screen Configuration, 48
 - Screen Control/Etc, 135
 - Set Hardware Break Point, 110
 - Set Options, 135
 - Settings Menu, 63
 - Setup, 11, 12, 62
 - Hardware Requirements, 11
 - Software Requirements, 12
 - Shortcut keys, 64
 - Shortcut Keys, 64
 - Backtrace Window, 88, 91
 - Code Window, 70
 - Command Window, 72, 75
 - Software Installation, 25
 - Stack window, 84
 - Stack Window, 84
 - Local Menu, 85
 - Screen Configuration, 50
 - Shortcut Keys, 84
 - Stack, Default Value, 41
 - Startup, 41
 - Startup Options, 36

- !, 38
- B, 36
- Command Line, 36
- D, 37
- E, 37
- SD, 37
- TAB, 38
- Status Bar, 102
- Switch Settings, 19
- Symbols
 - Display/Set, 131
 - Global, 112
 - Local, 114
 - Special, 114
- System Call, 133
- System Control, 134
- Toolbar, 97, 98
 - Settings, 98
- Watch window, 30, 71, 86, 87, 89, 90,
91, 94, 131, 138
- Watch Window, 89
 - Local Menu, 90
 - Mouse Operations, 91
 - Screen Configuration, 51
 - Shortcut Keys, 90
- WEPTN64.CFG, 35, 41
- Window Bar, 101
- Window Menu, 62

Nintendo®

PARTNER-N64

PC