Predefined Debugger Variables

$fpasingle If set, Pdbx prints floating-point registers as single-precision numbers.

$fpustackall If set, Pdbx prints nonzero floating-point stack entries even if in empty state (80387 FPU only).

$hexchars If set, Pdbx prints characters in hexadecimal.

$hexfloats If set, Pdbx prints floating-point stack registers in hexadecimal.

$hexin If set, Pdbx interprets integers in command input as hexadecimal.

$hexints If set, Pdbx prints integers in hexadecimal.

$hexoffsets If set, Pdbx prints offsets from registers in hexadecimal.

$hexstrings If set, Pdbx prints character arrays in hexadecimal.

$listwindow Default number of lines to display with the **list** command. If not set, Pdbx prints 10 lines. When used with **list *proc*** for listing a procedure, Pdbx lists *$listwindow* lines around the beginning of the procedure.

$noframe If set, Pdbx does not follow the chain of call frames on the stack. This allows access to global variables and register contents in code that does not have conventional stack frames.

$nostrict If set, Pdbx relaxes type checking rules for **call** and **assign** commands.

$octin If set, Pdbx interprets integers in command input as octal. If *$hexin* is set, *$octin* is ignored.

$whichreg If set, Pdbx includes the register name with the information displayed by the **whatis** command for the specified variable. Your program must declare the storage class for the variable as type `register`. Valid for C programs only.

Printed in the United States of America.

1003-48548-00

Type Conversion

To coerce the value of expression *expr* to type *typename*, use the construct *expr*\typename. To coerce the value of expression *expr* to a pointer to type *typename*, use the construct *expr*\\&*typename*. If *typename* is a struct, precede it with a double dollar sign ($$).

If Pdbx prints the message `incompatible types` or `type mismatch` after you enter a command that includes type conversion, you can use the **set $nostrict** command to relax type-checking rules and then reenter the command.

Formats

The following table lists the formats you can use with the *address*/ and *address*= commands.

Format	Type[a]	Radix
i	machine instruction	n/a
d	short word	decimal
D	long word	decimal
o	short word	octal
O	long word	octal
x	short word	hexadecimal
X	long word	hexadecimal
b	byte	octal
c	character	n/a
s	null-terminated string	n/a
f	single-precision real	n/a
g	double-precision real	n/a

[a] A short word is 16 bits; a long word is 32 bits.

Identifiers Containing Special Characters

When entering an identifier that contains special characters (such as a period) or an identifier that is also a Pdbx reserved word, you should enclose the identifier in back quotes. For example, entering the following command instructs Pdbx to stop in the routine ._SQRT:

```
stop in `._SQRT`
```

Register Names

Register Type	Processor		
	80386	80387	1167
scratch	$eax $ecx $edx	n/a n/a n/a	$fp2–$fp7
user[a]	$ebx $edi $esi	n/a n/a n/a	$fp8–$fp31
stack	n/a	$st0–$st7	n/a
other	$eip $esp $eflags $ebp	n/a n/a n/a n/a	n/a n/a n/a n/a

[a] program variable declared with the register keyword

The following aliases are also supported: `pc` for `eip`, `sp` for `esp`, and `fp` for `ebp`.

Pdbx Quick Reference Card

1003-48548-00

Invoking Pdbx

Use one of the following syntax forms to invoke Pdbx. Use the first form when debugging an application that consists of a single program; use the second form when debugging applications that consist of multiple programs. Invoking Pdbx as **pdbx** is useful when debugging a parallel program; invoking Pdbx as **dbx** is useful when debugging a program that creates child processes and only the parent process is of interest.

```
{pdbx} [ –i ] [ –u ] [ –a ] [ –c file ] [ –I dir ] ...
{dbx }   [ execfile [ coredump]]

{pdbx} [ –i ] [ –u ] [ –a ] [ –c file ] [[ –I dir ] ...
{dbx }   –O execfile ] ... [ –Q execfile ] ... [ –C coredump ] ...
```

Exiting Pdbx

Enter the following command to exit Pdbx:

```
quit
```

Pdbx Command Summary

Creating and Executing Processes

%n	Change the current process to n
call procedure ([parameters])	Execute specified procedure
cont [%procnum all] [signal] [to sourceline] [&]	Resume process execution
create [execfile] [args] [< infile] [> outfile]	Create new process; use cont to execute the process
next [%procnum all] [&]	Execute next line or, if the next line calls a procedure, execute the entire procedure
ps [%procnum]	Print processes and their states; an asterisk (*) in the resultant display marks the current process
release %procnum	Release process from Pdbx control
rerun [execfile] [args] [< infile] [> outfile] [&]	Same as run except uses arguments specified with last run or create command when no arguments are specified
return [procedure]	Continue until current procedure returns or until return to the specified procedure
run [execfile] [args] [< infile] [> outfile] [&]	Create and execute a new process; terminate old processes
step [%procnum all] [&]	Execute next source line
terminate [%procnum all]	Terminate processes and remove from process list

Tracing, Breakpoints, and Signals

catch [event signal]	Stop process when specified signal or event occurs
delete cmdnumber ...	Cancel the breakpoint or tracepoint associated with cmdnumber
ignore [event signal]	Do not stop when specified signal or event occurs
signal {%procnum all} sig	Send the signal specified by sig to one or all processes

Tracing, Breakpoints, and Signals (cont.)

status [> filename]	Print active breakpoints and tracepoints
stop [procid] in procedure [if condition]	Stop execution on entry to procedure
stop [procid] at sourceline [if condition]	Stop before executing specified source line
stop [procid] variable [if condition]	Stop when value of variable changes
stop [procid] if condition	Stop when condition becomes true
stop all	Stop all running processes
trace [procid] [in procedure] [if condition]	Print trace information as process executes
trace [procid] sourceline [if condition]	Print specified source line each time it is encountered (before executing it)
trace [procid] procedure [in procedure] [if condition]	Print which procedure called procedure and value of each parameter passed to it
trace [procid] expression at sourceline [if condition]	Evaluate and print expression before executing code at the specified source line
trace [procid] variable [in procedure] [if condition]	Print value of variable each time it changes

Command Aliases and Debugger Variables

alias name cmdname	Define an abbreviation for a Pdbx command name
alias name "cmdstring"	Define an abbreviation for a Pdbx command string
alias name (parameter [,parameter] ...) "cmdstring"	Define an abbreviation for a Pdbx command string with replaceable parameters
alias [name]	Print current aliases
unalias name	Undefine name as an alias
set [name [=expression]]	Set a debugger variable
unset name	Delete the specified debugger variable

Examining and Altering Variables

assign variable=expression	Assign value of expression to variable
dump [procedure] [> filename]	Print names and values of variables in current procedure
print expression [, expression] ...	Print value of variable or expression
print fpustack	Print contents of 80387 FPU registers

Accessing Source Files

/pattern[/]	Search forward for specified pattern
?pattern[?]	Search backward for specified pattern
edit [sourcefile]	Edit current source file
edit procedure	Edit current source file and place cursor at beginning of specified procedure
file [[%execfile] sourcefile]	Use sourcefile as current source file
list [linenumber [, linenumber]]	Print source lines in the current source file
list procedure	List specified procedure
use [%execfile] [directory...]	Add directory to source file directory search list

Examining and Altering the Current Context

down [nlevels]	Change the current procedure to the procedure nlevels down the stack
func [procedure]	Change the current procedure and current source file designations to the specified procedure
up [nlevels]	Change the current procedure to the procedure nlevels up the stack
whatis identifier	Print declaration for identifier
where [%procnum]	Print traceback

Examining and Altering the Current Context (cont.)

whereis identifier	Print qualified names for all instances of identifier
which identifier	Print qualified name for identifier in the current context

Machine-Level Debugging

address, address / [format]	Print values in memory in specified address range
address/ [count] [format]	Print count values in memory starting at address
address= [format]	Print value of the number specified by address in the specified format
listi [linenumber [, linenumber]]	Disassemble and print instructions
listi procedure	Disassemble and print instructions for specified procedure
nexti [%procnum all] [&]	Execute next instruction or, if next instruction calls a procedure, execute entire procedure
stepi [%procnum all] [&]	Execute next instruction
stopi [procid] [at] address [if condition]	Stop before executing the instruction at address
tracei [procid] [in procedure] [if condition]	Trace program at machine level
tracei [procid] address [if condition]	Print instruction at specified address before it executes
tracei [procid] variable [at address] [if condition]	Print value of variable before execution of instruction at specified address

Miscellaneous Commands

help	Print synopsis of frequently used Pdbx commands
sh [commandline]	Create new shell and pass it the specified command line
source filename	Execute Pdbx commands in filename
suspend	Suspend Pdbx and return to shell; use csh's fg to resume