SECTION 14

Stack Groups

14.1 INTRODUCTION

This chapter explains stack groups, the building blocks of multiprocessing.

A stack group is the fundamental unit of computation in the Explorer. It is the main data structure behind the implementation of a process. Interrupt context-switching, co-routines, and generators are facilitated by the use of stack groups.

At all times, there is exactly one active stack group, which corresponds to the process currently being run on a time-sharing system. Although there is no time-sharing between users on the Explorer, it is still useful to support multiple processes. For example, when a message is received from the network, some other stack group is activated to handle it.

14.2 THE STACK GROUP DATA STRUCTURE

The term stack group refers to the fact that each process must have its own control stack for remembering function call/return data and arguments and local data, and each process must have a Linear Binding stack to save the values of special variables. A stack group is a pointer of datatype DTP-STACK-GROUP, which points to an array header word the same way a DTP-ARRAY would. The reason for using an additional datatype is so that any routine will always be able to distinguish a stack group array from all other arrays. The array also has its own array type, ART-STACK-GROUP-HEAD, for the same reason.

The stack group structure is an array with no body, only a leader. The array leader holds many relevant data including a pointer to the regular push down list, a pointer to another array holding the Linear Binding PDL for the stack group, the PDL pointers for both PDLs, and various saved microcode registers (the Linear Binding PDL is also called the Special PDL or SPECPDL; see description below).

A useful feature is that by binding appropriate special variables, such as the default cons area and debugging

information, an error handler can be made a function of which stack group is active; each may have its own. This is because each stack group has a separate Linear Binding PDL.

The array leader contains miscellaneous data related to running and maintaining a stack group in the system. This data is divided up into sections according to how the data is used. The static section contains data such as the stack group name, and size and limits on the PDLs. This data is set up when the stack group is created and doesn't normally change during the course of system operation. This information is loaded when the stack group is entered; but since it doesn't change, the data needn't be saved when the stack group is left. The debugging section has information that the error handler needs to determine what error has occured and how to restart the stack group. This information is read directly by the error handler as needed and will not be loaded or saved on stack group entry or exit. The next section of the stack group contains information used to determine which operations are valid on this stack group. A layout of the stack group data structure follows:

14.2.1 Static Section.

SG-NAME	Name of stack group for conversing with user.
SG-REGULAR-PDL	The array serving as the regular PDL of this stack group. It must be an ART-REG-PDL array.
SG-REGULAR-PDL-LIMIT	Max PDL pointer value before overflow.
SG-SPECIAL-PDL	Special PDL for stack group. It must be an ART-SPECIAL-PDL array.
SG-SPECIAL-PDL-LIMIT	Max special PDL pointer value before overflow.
SG-INITIAL-FUNCTION-INDEX	Position in regular PDL of the topmost function pointer cell (normally 3).

14.2.2 Debugging Section.

SG-TRAP-TAG	Symbolic tag corresponding to SG-TRAP-MICRO-PC. Determined by looking up the trap PC in the

	microcode error table. Properties of this symbol drive error reporting and recovery.
SG-RECOVERY-HISTORY	Available for complex SG debugging routines to leave tracks in for debugging purposes. (This may be done away with in the future.)
SG-FOOTHOLD-DATA	Structure which saves dynamic section of "real" SG when executing in the error handler foothold.

14.2.3 High Level Section.

SG-STATE	The stack-group state. This has fields describing the high-level state of this stack group. Stack group states are listed in Table 14-1.
SG-PREVIOUS-STACK-GROUP	SG which caused resumption of this one.
SG-CALLING-ARGS-POINTER	Pointer to argument-block which last called this SG.
SG-CALLING-ARGS-NUMBER	Number of args in last call to this SG.
SG-TRAP-AP-LEVEL	Used for stepping. When stepping compiled functions, will cause a STEP-BREAK trap when PDL pointer is below this virtual address.

14.2.4 Dynamic Section.

SG-REGULAR-PDL-POINTER	Saved PDL pointer, stored as fixnum index into SG-REGULAR-PDL.
SG-SPECIAL-PDL-POINTER	Saved special PDL pointer, stored as fixnum index

	into SG-SPECIAL-PDL.
SG-TRAP-MICRO-PC	Micro-address from which a trap was signalled. Used as a key to lookup the TRAP-TAG in the microcode error table.
SG-SAVED-M-FLAGS	Saved processor flags as fixnum. The flags are shown in Table 14-2.
SG-PDL-PHASE	The index into the hardware PDL buffer cache of the Q at the top of the PDL. This preserves index-phasing of the pdl buffer across SG switches.
SG-SAVED-VMA	VMA register in a locative. The data type is stored in SG-VMA-M1-M2-TAGS.
SG-VMA-M1-M2-TAGS	Tags of VMA, M-1, and M-2 packed into a fixnum.
SG-M3-M4-TAGS	Tags of M-3 and M-4 packed into a fixnum.
SG-AC-1	Pointer fields of "raw" accumulators packed into fixnums. Tags are in one of previous packed-tag regs.
SG-AC-2	
SG-AC-3	
SG-AC-4	
SG-AC-A	Accumulators ...
SG-AC-B	
SG-AC-C	
SG-AC-D	
SG-AC-E	
SG-AC-F	
SG-AC-G	

SG-AC-H	
SG-AC-I	
SG-AC-J	
SG-AC-K	
SG-AC-L	
SG-AC-Q	
SG-AC-R	
SG-AC-S	
SG-AC-ZR	
SG-AC-T	Result register, pseudo indicators.
SG-CATCH-POINTER	Typed pointer to catch block.

Table 14-1 Stack Group States

Value	Name	Meaning
0	SG-STATE-ERROR	Should never get this.
1	SG-STATE-ACTIVE	Actually executing on the machine.
2	SG-STATE-RESUMABLE	Reached by interrupt or error recovery completed. Just restore state and continue.
3	SG-STATE-AWAITING-RETURN	After doing a "legitimate" sg-call. To resume this, reload sg, then return a value.
4	SG-STATE-INVOKE-CALL-ON-RETURN	To resume this, reload sg, then simulate a store in destination-last. The error system can produce this state when it wants to activate the foothold or perform a retry.
5	SG-STATE-INTERRUPTED-DIRTY	Get this if forced to take an interrupt at an inopportune time.
6	SG-STATE-AWAITING-ERROR-RECOVERY	Immediatedly after error, before recovery.
7	SG-STATE-AWAITING-CALL	The SG is ready to be called.
8	SG-STATE-AWAITING-INITIAL-CALL	Initial state of the SG, before it has been called.
9	SG-STATE-EXHAUSTED	State when the SG has finished running.

Table 14-2 Processor Flags

Bit(s)	Name	Meaning
0	QBBFL	This flag is no longer used.
1-2	CAR-SYM-MODE	CAR of symbol mode: 0: error 1: error except (CAR NIL) is NIL 2: NIL 3: error
3-4	CAR-NUM-MODE	CAR of number mode: 0: error 1: NIL 2: error 3: error
5-6	CDR-SYM-MODE	CDR of symbol mode: 0: error 1: error except (CDR NIL) is NIL 2: NIL 3: property-list
7-8	CDR-NUM-MODE	CDR of number mode: 0: error 1: NIL 2: error 3: error
9	DONT-SWAP-IN	Flag for creating fresh pages.
10	TRAP-ENABLE	When set, enable error trapping.
11-12	MAR-MODE	1-bit = read-trap, 2-bit = write-trap
13	PGF-WRITE	Flag used by page fault routine.
14	INTERRUPT-FLAG	In a microcode interrupt.
15	SCAVENGE-FLAG	In scavenger.
16	TRANSPORT-FLAG	In transporter.
17	STACK-GROUP-SWITCH-FLAG	Switching stack groups.
18	DEFERRED-SEQUENCE-BREAK-FLAG	Sequence break pending but inhibited.
19	METER-ENABLE	Metering enabled for this stack group.
20	TRAP-ON-CALL	Trap on attempting to activate new frame.

14.3 SPECIAL PDL

The Special PDL, also known as the Linear Binding PDL, is used to hold saved bindings of special variables. The Explorer uses shallow binding, so the current value of any symbol is always found in the symbol's value cell. When a symbol is bound, its previous value is saved on the Special PDL, and the new value is placed in the value cell. When a stack group switch is performed, the current process's Special PDL is saved in the stack group.

The Special PDL also serves some other functions. When a micro-to-macro call is made, the processor micro-stack (UPCS) of the machine is stored there (this is needed because the hardware UPCS is of a small fixed size).

The Special PDL is block oriented. The blocks are delimited by setting the SPECPDL-BLOCK-START-FLAG in the first binding made in a block. The data type of the top word (last pushed) of a block determines what kind of block this is, as shown in Table 14-3.

SPECPDL-BLOCK-START-FLAG and SPECPDL-CLOSURE-BINDING are stored in the CDR-Code field of Q's on the Special PDL. The CDR-Code is not otherwise used on the Special PDL. SPECPDL-CLOSURE-BINDING (bit 31) indicates that this binding was made "before" entering the function (i.e., by closure binding, or by the binding of SELF for a method). SPECPDL-BLOCK-START-FLAG is bit 30.

A normal binding block is stored as a pair of Q's for each binding; the first Q is a locative pointer to the bound location, and the second is the saved contents of the location. Note that any location can be bound; usually these locations will be the value cells of symbols, but they can also be array elements, etc. (only of arrays of type ART-Q or ART-Q-LIST).

The processor micro-stack (UPCS) blocks are always pushed onto the Special PDL all at once, and so are never "open." However, the normal binding blocks are created one pair at a time. To keep track of this, when a macrocompiled function is running, the binding-block-pushed bit is set in the call-info-word if a binding block has been opened on the Special PDL. This assures that when a compiled function is done, the call-info-word will correctly reflect whether it has done any bindings that must be popped off the Special PDL. If the bit is set, all of the bindings of the top-most block of the Special PDL must be undone. If not set, it means that no pairs have yet been pushed.

Micro-to-micro calls can also cause bindings, and in order to keep that straight, a bit on the UPCS is set to indicate that a block was bound.

The Special PDL is pointed to by the location SG-SPECIAL-PDL-POINTER in the stack group and by A-QLBNDP when the stack group

is executing on processor. There is an area devoted to storing both Special PDLs and stack groups called the SG-AND-BIND-PDL-AREA.

Table 14-3 Special PDL Block Type

DATATYPE	USE
LOCATIVE FIXNUM	The block is a normal binding block. This is a block transferred from the processor micro-stack (UPCS). Each word in the block should be a fixnum containing the old contents of the UPCS. Only the active part of the stack stack is transferred.