

| | | | | | | |
|--------------|------------------|----------|----------------|----------------|------------------|------------------|
| CCCCCCCCCCCC | LLL | IIIIIIII | UUU | UUU | TTTTTTTTTTTTTTTT | LLL |
| CCCCCCCCCCCC | LLL | IIIIIIII | UUU | UUU | TTTTTTTTTTTTTTTT | LLL |
| CCCCCCCCCCCC | LLL | IIIIIIII | UUU | UUU | TTTTTTTTTTTTTTTT | LLL |
| CCC | LLL | III | UUU | UUU | TTT | LLL |
| CCC | LLL | III | UUU | UUU | TTT | LLL |
| CCC | LLL | III | UUU | UUU | TTT | LLL |
| CCC | LLL | III | UUU | UUU | TTT | LLL |
| CCC | LLL | III | UUU | UUU | TTT | LLL |
| CCC | LLL | III | UUU | UUU | TTT | LLL |
| CCC | LLL | III | UUU | UUU | TTT | LLL |
| CCC | LLL | III | UUU | UUU | TTT | LLL |
| CCC | LLL | III | UUU | UUU | TTT | LLL |
| CCC | LLL | III | UUU | UUU | TTT | LLL |
| CCC | LLL | III | UUU | UUU | TTT | LLL |
| CCC | LLL | III | UUU | UUU | TTT | LLL |
| CCC | LLL | III | UUU | UUU | TTT | LLL |
| CCC | LLL | III | UUU | UUU | TTT | LLL |
| CCC | LLL | III | UUU | UUU | TTT | LLL |
| CCC | LLL | III | UUU | UUU | TTT | LLL |
| CCCCCCCCCCCC | LLLLLLLLLLLLLLLL | IIIIIIII | UUUUUUUUUUUUUU | UUUUUUUUUUUUUU | TTTT | LLLLLLLLLLLLLLLL |
| CCCCCCCCCCCC | LLLLLLLLLLLLLLLL | IIIIIIII | UUUUUUUUUUUUUU | UUUUUUUUUUUUUU | TTTT | LLLLLLLLLLLLLLLL |
| CCCCCCCCCCCC | LLLLLLLLLLLLLLLL | IIIIIIII | UUUUUUUUUUUUUU | UUUUUUUUUUUUUU | TTTT | LLLLLLLLLLLLLLLL |

```

RRRRRRRR      UU      UU  NN      NN  DDDDDDDD  EEEEEEEEE  TTTTTTTTTT
RRRRRRRR      UU      UU  NN      NN  DDDDDDDD  EEEEEEEEE  TTTTTTTTTT
RR      RR      UU      UU  NN      NN  DD      DD  EE      TT
RR      RR      UU      UU  NN      NN  DD      DD  EE      TT
RR      RR      UU      UU  NN      NN  DD      DD  EE      TT
RR      RR      UU      UU  NNNN     NN  DD      DD  EE      TT
RRRRRRRR      UU      UU  NN      NN  DD      DD  EEEEEEE  TT
RRRRRRRR      UU      UU  NN      NN  DD      DD  EEEEEEE  TT
RR      RR      UU      UU  NN      NN  DD      DD  EE      TT
RR      RR      UU      UU  NN      NN  DD      DD  EE      TT
RR      RR      UU      UU  NN      NN  DD      DD  EE      TT
RR      RR      UU      UU  NN      NN  DD      DD  EE      TT
RR      RR      UU      UU  NN      NN  DD      DD  EE      TT
RR      RR      UU      UU  NN      NN  DD      DD  EE      TT
RR      RR      UUUUUUUUU  NN      NN  DDDDDDDD  EEEEEEEEE  TT
RR      RR      UUUUUUUUU  NN      NN  DDDDDDDD  EEEEEEEEE  TT

```

```

LL      IIIIII  SSSSSSSS
LL      IIIIII  SSSSSSSS
LL      II      SS
LL      II      SS
LL      II      SS
LL      II      SS
LL      II      SSSSSS
LL      II      SSSSSS
LL      II      SS
LL      II      SS
LL      II      SS
LL      II      SS
LL      IIIIII  SSSSSSSS
LL      IIIIII  SSSSSSSS

```

```

1 0001 0 MODULE rundet (XTITLE 'Run Detached Process -- CLI Utility Procedure'
2 0002 0 IDENT = 'V04-000',
3 0003 0 MAIN = run_detached) =
4 0004 1 BEGIN
5 0005 1
6 0006 1 *****
7 0007 1 *
8 0008 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
9 0009 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
10 0010 1 * ALL RIGHTS RESERVED.
11 0011 1 *
12 0012 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
13 0013 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
14 0014 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
15 0015 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
16 0016 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
17 0017 1 * TRANSFERRED.
18 0018 1 *
19 0019 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
20 0020 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
21 0021 1 * CORPORATION.
22 0022 1 *
23 0023 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
24 0024 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
25 0025 1 *
26 0026 1 *
27 0027 1 *****
28 0028 1
29 0029 1 **
30 0030 1
31 0031 1 Facility:
32 0032 1
33 0033 1 CLI Utility
34 0034 1
35 0035 1 Abstract:
36 0036 1
37 0037 1 This module contains the routines necessary to act as a CLI interface
38 0038 1 to the create process ($CREPRC) and schedule wakeup ($SCHDWK) system
39 0039 1 services.
40 0040 1
41 0041 1 Environment:
42 0042 1
43 0043 1 VAX/VMS User Mode, Non-Privileged
44 0044 1
45 0045 1 Author:
46 0046 1
47 0047 1 Michael T. Rhodes, Creation Date: March, 1983
48 0048 1
49 0049 1 Modified By:
50 0050 1
51 0051 1 V03-007 MCN0159 Maria del C. Nasr 28-Mar-1984
52 0052 1 Use LIB$TPARSE to parse uic instead of RUN_CVTUIC.
53 0053 1
54 0054 1 V03-006 RAS0274 Ron Schaefer 20-Mar-1984
55 0055 1 Make this work with searchlists by using LIB$FIND FILE
56 0056 1 to parse the image filespec. Wildcards are not allowed.
57 0057 1

```

| | | | | | | |
|----|------|---|---------|--|-------------------|-------------|
| 58 | 0058 | 1 | V03-005 | TMK0001 | Todd M. Katz | 13-Nov-1983 |
| 59 | 0059 | 1 | | Add the qualifier /JOB_TABLE_QUOTA. The use of this | | |
| 60 | 0060 | 1 | | qualifier allows the creator of a detached process to | | |
| 61 | 0061 | 1 | | specify its job-wide logical name table creation quota. | | |
| 62 | 0062 | 1 | | | | |
| 63 | 0063 | 1 | V03-004 | MTR0002 | Michael T. Rhodes | 22-Jul-1983 |
| 64 | 0064 | 1 | | Correct qualifier name /PROCESS to /PROCESS_NAME. | | |
| 65 | 0065 | 1 | | | | |
| 66 | 0066 | 1 | V03-003 | MTR0001 | Michael T. Rhodes | 29-Apr-1983 |
| 67 | 0067 | 1 | | Convert privilege processing to use common CLI utility | | |
| 68 | 0068 | 1 | | routine PRVSSETPRIV. Also change PID message to PROC_ID. | | |
| 69 | 0069 | 1 | | | | |
| 70 | 0070 | 1 | V03-002 | WMC0002 | Wayne Cardoza | 14-Apr-1983 |
| 71 | 0071 | 1 | | Qualifier is /DETACHED | | |
| 72 | 0072 | 1 | | | | |
| 73 | 0073 | 1 | V03-001 | WMC0001 | Wayne Cardoza | 11-Apr-1983 |
| 74 | 0074 | 1 | | Add /DETACH and /DUMP flags. | | |
| 75 | 0075 | 1 | | | | |
| 76 | 0076 | 1 | | | | |
| 77 | 0077 | 1 | | | | |

```

79 0078 1 %SBTTL 'Declarations'
80 0079 1
81 0080 1   Include Files:
82 0081 1
83 0082 1 LIBRARY 'SYSSLIBRARY:LIB';           ! Define VMS structures.
84 0083 1 LIBRARY 'SYSSLIBRARY:TPAMAC';       ! T$PARSE structures
85 0084 1
86 0085 1
87 0086 1   Table of Contents:
88 0087 1
89 0088 1 FORWARD ROUTINE
90 0089 1   run_detached      : NOVALUE,           ! Create a sub or detached process.
91 0090 1   init_arg_list      : NOVALUE,           ! Initialize the $CREPRC and $SCHDWK argument lists.
92 0091 1   parse_image_spec  : NOVALUE,           ! Obtain the expanded image file specification.
93 0092 1   get_wakeup_info    : NOVALUE,           ! Get info for the $SCHDWK .
94 0093 1   get_uic           : NOVALUE,           ! Get the UIC value.
95 0094 1   get_privileges   : NOVALUE,           ! Set up the privilege mask vector for the process.
96 0095 1   get_quotas      : NOVALUE,           ! Set up the quota list for the process.
97 0096 1   get_cpulm       : NOVALUE,           ! Special case for CPl time limit quota.
98 0097 1   get_value       : NOVALUE,           ! Get the value of the current command line entity.
99 0098 1   insert_quota    : NOVALUE,           ! Insert a quota list entry.
100 0099 1   get_stsflgs     : NOVALUE,           ! Set up the initial process status flag vector.
101 0100 1   schedule_process: NOVALUE;         ! Schedule the process to be executed.
102 0101 1
103 0102 1
104 0103 1   External references:
105 0104 1
106 0105 1 EXTERNAL ROUTINE
107 0106 1   CLISGET VALUE      : ADDRESSING_MODE (GENERAL), ! CLI call back routine to get command line entity.
108 0107 1   CLISPRESNT      : ADDRESSING_MODE (GENERAL), ! CLI call back routine to determine entity presence
109 0108 1   lib$cvd_dtime    : ADDRESSING_MODE (GENERAL), ! Private CLI routine to convert a string to a delta
110 0109 1   LIB$CVT_DX_DX     : ADDRESSING_MODE (GENERAL), ! General conversion routine.
111 0110 1   lib$cvd_time    : ADDRESSING_MODE (GENERAL), ! Private CLI routine to convert a string to an abso
112 0111 1   LIB$GET_VM       : ADDRESSING_MODE (GENERAL), ! Library routine to obtain virtual memory.
113 0112 1   LIB$FIND FILE  : ADDRESSING_MODE (GENERAL), ! Library routine to parse filespecs.
114 0113 1   LIB$TPARSE    : ADDRESSING_MODE (GENERAL), ! Table driven parser
115 0114 1   prv$setpriv   : ADDRESSING_MODE (GENERAL); ! Private CLI routine to process privileges and set
116 0115 1
117 0116 1
118 0117 1   Define message codes...
119 0118 1
120 P 0119 1 $SHR_MSGDEF (RUN,192,GLOBAL,           ! Define the shareable messages first.
121 P 0120 1 (INSVIRMEM,SEVERE),           ! Insufficient virtual memory.
122 P 0121 1 (INVQUAVAL,ERROR)           ! Invalid qualifier value.
123 P 0122 1 (PARSEFAIL,SEVERE),         ! Error parsing file spec.
124 0123 1 (SYNTAX,SEVERE));           ! Error parsing command entity.
125 0124 1
126 0125 1 EXTERNAL LITERAL
127 0126 1   cli$_negated,           ! Command line entity was explicitly negated.
128 0127 1   run$_creprc,         ! Create process failed.
129 0128 1   run$_proc_id,        ! Identification of created process.
130 0129 1   run$_getjpi,       ! Error obtaining job and process information.
131 0130 1   run$_cvterr,       ! Error converting entity value.
132 0131 1   run$_invuic,       ! Invalid UIC.
133 0132 1   run$_llval,        ! Illegal CPl time limit.
134 0133 1   run$_schdwk;       ! Failed to schedule the wakeup request.
135 0134 1

```

RUNDET
V04-000

Run Detached Process -- CLI Utility Procedure
Declarations

B 13
16-Sep-1984 00:27:00
14-Sep-1984 12:08:54

VAX-11 Bliss-32 V4.0-742
[CLIUTL.SRC]RUNDET.B32;1

Page 4
(2)

: 136

0135 1

```

138 0136 1 %SBTTL 'Declarations -- Private Storage'
139 0137 1
140 0138 1 : Command qualifiers, keywords, and keyword paths
141 0139 1
142 0140 1 BIND
143 0141 1     accounting = $descriptor ('ACCOUNTING'),      ! Log accounting records for created process
144 0142 1     authorize = $descriptor ('AUTHORIZE'),      ! Perform user authorization when image is L
145 0143 1     delay = $descriptor ('DELAY'),              ! Hibernate process and awaken after delta t
146 0144 1     detach = $descriptor ('DETACHED'),          ! Detached process
147 0145 1     dump = $descriptor ('DUMP'),                  ! Image dump requested
148 0146 1     error = $descriptor ('ERROR'),              ! Error device (SYSSERROR).
149 0147 1     input = $descriptor ('INPUT'),              ! Input device (SYSSINPUT).
150 0148 1     interval = $descriptor ('INTERVAL'),      ! Hibernate process and awaken at regularly
151 0149 1     mailbox = $descriptor ('MAILBOX'),          ! Unit number of termination mailbox for thi
152 0150 1     output = $descriptor ('OUTPUT'),           ! Output device (SYSSOUTPUT).
153 0151 1     pl = $descriptor ('P1'),                    ! Image file specification parameter.
154 0152 1     priority = $descriptor ('PRIORITY'),      ! Base priority at which the created process
155 0153 1     privileges = $descriptor ('PRIVILEGES'),    ! Defines the privileges for the created pro
156 0154 1     process = $descriptor ('PROCESS NAME'),      ! Specifies the process name.
157 0155 1     resource_wait = $descriptor ('RESOURCE_WAIT'), ! Enable/disables resource wait mode for the
158 0156 1     schedule = $descriptor ('SCHEDULE'),          ! Hibernate the process and awaken at absolu
159 0157 1     service_fail = $descriptor ('SERVICE_FAILURE'), ! Enable/disable system service failure exce
160 0158 1     swapping = $descriptor ('SWAPPING'),         ! Enable/disable process swapping.
161 0159 1     uic = $descriptor ('UIC');                  ! Detached process UIC.
162 0160 1
163 0161 1 LITERAL
164 0162 1     true = 1, false = 0,
165 0163 1     jpientries = 3,
166 0164 1     jpilistsize = jpientries * 12,
167 0165 1     list_k_entry_size = 5,
168 0166 1     priv_entries = 31;
169 0167 1
170 0168 1 MACRO
171 0169 1     list_entry (name, entity) = BYTE (name), LONG ($descriptor (entity)) %,
172 0170 1     list_b_name = 0,0,8,0 %;
173 0171 1     list_l_value = 1,0,32,0 %;
174 0172 1
175 0173 1 OWN
176 0174 1     run$a_error      : $bblock [dsc$c_s_bln],      ! Descriptor for error device specification.
177 0175 1     run$a_image     : $bblock [dsc$c_s_bln],      ! Descriptor for image file specification.
178 0176 1     run$a_input     : $bblock [dsc$c_s_bln],      ! Descriptor for input device specification.
179 0177 1     run$a_input_desc : $bblock [dsc$c_s_bln],      ! General purpose dynamic input descriptor.
180 0178 1     run$a_output    : $bblock [dsc$c_s_bln],      ! Descriptor for output device specification
181 0179 1     run$a_prcnam    : $bblock [dsc$c_s_bln],      ! Descriptor for process name.
182 0180 1     run$a_quota     : $bblock [dsc$c_s_bln],      ! Quota list head address.
183 0181 1     run$l_baspri    : $bblock [dsc$c_s_bln],      ! Base execution priority for the process.
184 0182 1     run$l_mbxunt    : $bblock [dsc$c_s_bln],      ! Termination mailbox unit number.
185 0183 1     run$l_pid       : $bblock [dsc$c_s_bln],      ! Created process' PID.
186 0184 1     run$l_status    : $bblock [dsc$c_s_bln],      ! Global Status vector.
187 0185 1     run$l_stsflg    : $bblock [4],                ! Initial process state status flags.
188 0186 1     run$l_uic       : $bblock [4],                ! Detached process' UIC.
189 0187 1     run$q_daytim    : VECTOR [2, LONG] INITIAL (-1,-1), ! Time at which the process is to be awakene
190 0188 1     run$q_interval  : VECTOR [2, LONG] INITIAL (0,0), ! Interval at which the wake up request is t
191 0189 1     run$q_privadr   : VECTOR [2, LONG],          ! Privilege vector.
192 0190 1     run$a_image_buf : $bblock [nam$c_maxrss],    ! Buffer to hold the expanded file specifica
193 0191 1     run$a_findfile : ref $bblock,                ! Context ptr for LIB$FIND_FILE.
194 0192 1

```

```
: 196 0193 1 %SBTTL 'Data Structures -- $GETJPI Item List'
: 197 0194 1 ++
: 198 0195 1
: 199 0196 1 Functional Description:
: 200 0197 1
: 201 0198 1 This structure is used to obtain default job and process information
: 202 0199 1 for the created process.
: 203 0200 1
: 204 0201 1 --
: 205 0202 1 run$a_getjpi : $bblock [jpilistsize] ! Item list for $GETJPI.
: 206 0203 1 INITIAL (
: 207 0204 1 WORD (4, jpi$_prib), LONG (run$a_baspri, 0), ! Process base priority.
: 208 0205 1 WORD (8, jpi$_procpriv), LONG (run$a_prvadr, 0), ! Process 'SAME' privileges.
: 209 0206 1 WORD (0, 0), LONG (0, 0), ! Item list terminator.
: 210 0207 1
```



```

212 0208 1 %SBTTL 'Data Structures -- Quota Table'
213 0209 1 :++
214 0210 1 :
215 0211 1 : Functional Description:
216 0212 1 :
217 0213 1 : This structure is used to parse and process the command line entries
218 0214 1 : which comprise the quota list values and to establish the initial quota
219 0215 1 : context for the $CREPRC system service.
220 0216 1 :
221 0217 1 :--
222 0218 1 run$a_quota_tbl : $bblock [(pql$_length - 1) * list_k_entry_size] : Quota table.
223 0219 1 INITIAL (
224 0220 1 list_entry (pql$_astlm, 'AST LIMIT'), : AST limit quota.
225 0221 1 list_entry (pql$_bytln, 'BUFFER LIMIT'), : Maximum number of bytes fo
226 0222 1 list_entry (pql$_enqlm, 'ENQUEUE LIMIT'), : Maximum # of outstanding l
227 0223 1 list_entry (pql$_wextent, 'EXTENT'), : Max size to which an image
228 0224 1 list_entry (pql$_fillm, 'FILL LIMIT'), : Maximum number of simultan
229 0225 1 list_entry (pql$_biolm, 'IO BUFFERED'), : Max number of concurrent b
230 0226 1 list_entry (pql$_diolm, 'IO DIRECT'), : Maximum number of concurre
231 0227 1 list_entry (pql$_wsquota, 'MAXIMUM WORKING_SET'), : Max size which process can
232 0228 1 list_entry (pql$_pgflquota, 'PAGE FICE'), : Max number of pages alloca
233 0229 1 list_entry (pql$_tqelm, 'QUEUE LIMIT'), : Maximum number of timer qu
234 0230 1 list_entry (pql$_prclm, 'SUBPROCESS LIMIT'), : Max number of subprocesses
235 0231 1 list_entry (pql$_cpulm, 'TIME LIMIT'), : Maximum CPU time allocated
236 0232 1 list_entry (pql$_wsdefault, 'WORKING SET'), : Default working set size f
237 0233 1 list_entry (pql$_jtquota, 'JOB_TABLE_QUOTA') : Job-wide logical name tabl
238 0234 1 );
239 0235 1 :
240 0236 1 : IPARSE state table to parse the uic value
241 0237 1 :
242 0238 1 :
243 0239 1 $INIT_STATE ( uic_states, uic_keys );
244 0240 1 :
245 P 0241 1 $STATE (
246 P 0242 1 (tpa$_ident,...,run$l_uic)
247 0243 1 );
248 0244 1 :
249 P 0245 1 $STATE (
250 P 0246 1 (tpa$_eos, tpa$_exit)
251 0247 1 );

```

```

253 0248 1 %SBTTL 'run_detached -- Create a sub or detached process'
254 0249 1 ROUTINE run_detached : NOVALUE =
255 0250 1 ++
256 0251 1
257 0252 1 Functional Description:
258 0253 1
259 0254 1 This routine is responsible for calling the initialization procedure
260 0255 1 which will build the argument lists used for performing the create
261 0256 1 process ($CREPRC) system service. If the process is to be scheduled
262 0257 1 to be awakened or re-executed at specific intervals, a call is made
263 0258 1 to perform that action.
264 0259 1
265 0260 1 Implicit Inputs:
266 0261 1
267 0262 1 The command line supplied by the CLI.
268 0263 1
269 0264 1 Implicit Outputs:
270 0265 1
271 0266 1 The process is created with the process identification (PID) of the
272 0267 1 created process SIGNALled to the caller as a success status message.
273 0268 1
274 0269 1 Routine Value:
275 0270 1
276 0271 1 SSS_NORMAL Process was created/scheduled without errors.
277 0272 1
278 0273 1 false Error status from failed routine.
279 0274 1
280 0275 1 Side Effects:
281 0276 1
282 0277 1 The process may have been hibernated with a scheduled wake up pending.
283 0278 1
284 0279 1 --
285 0280 2 BEGIN
286 0281 2
287 0282 2 init_arg_list (); ! Initialize the argument lists for the $CREPRC and
288 0283 2
289 P 0284 2 IF NOT (run$l_status = $CREPRC (PIDADR = run$l_pid, ! Process ID of created process.
290 P 0285 2 IMAGE = run$a_image, ! File specification of the image to execute.
291 P 0286 2 INPUT = run$a_input, ! Input device specification (assigned to SYSS$INPUT)
292 P 0287 2 OUTPUT = run$a_output, ! Output device specification (assigned to SYSS$OUTPUT)
293 P 0288 2 ERROR = run$a_error, ! Error device specification (assigned to SYSS$ERROR)
294 P 0289 2 PRVADR = run$g_privadr, ! Quadword privilege vector address.
295 P 0290 2 QUOTA = .run$a_quota, ! List head address for the quota list.
296 P 0291 2 PRCNAM = run$a_prcnam, ! Process name.
297 P 0292 2 BASPRI = .run$l_baspri, ! Process base execution priority.
298 P 0293 2 UIC = .run$l_uic, ! Process UIC.
299 P 0294 2 MBXUNT = .run$l_mbxunt, ! Termination mailbox unit number.
300 0295 2 STSFLG = .run$l_stsflg)) ! Initial process state status flags.
301 0296 2 THEN
302 0297 2 SIGNAL_STOP (run$creprc, 0, .run$l_status) ! Any errors, stop here and inform the user.
303 0298 2 ELSE
304 0299 2 SIGNAL (run$proc_id, 1, .run$l_pid); ! Process was created successfully, show it's PID to
305 0300 2
306 0301 2 IF .run$l_stsflg [prc$u_hiber] ! Should we leave a wake up call?
307 0302 2 THEN schedule_process ();
308 0303 2
309 0304 1 END; ! of ROUTINE run_detached

```

.TITLE RUNDET Run Detached Process -- CLI Utility Proc
edure

.IDENT \V04-000\

.PSECT _LIB\$STATES,NOWRT, SHR, PIC,1

00000 UIC_STATES::

.BLKB 0

45EC 00000 ;TPASTYPE

.WORD 17900

00000000* 00002 ;TPASADDR

.LONG <<RUNSL_UIC-U.3>-4>

15F7 00006 ;TPASTYPE

.WORD 5623

FFFF 00008 ;TPASTARGET

.WORD -1

.PSECT _LIB\$KEYOS,NOWRT, SHR, PIC,1

00000 UIC_KEYS::

.BLKB 0

00000 ;TPASKEYO

.BLKB 0

.PSECT \$SPLITS,NOWRT,NOEXE,2

| | | | | | | | | | | | | | | | |
|----|----|----|----|----|----|----|----|----|----|-----------|--------|--------|--------------|-------|---|
| 47 | 4E | 49 | 54 | 4E | 55 | 4F | 43 | 43 | 41 | 00000 | P.AAB: | .ASCII | \ACCOUNTING\ | : | |
| | | | | | | | | | | 0000A | | .BLKB | 2 | : | |
| | | | | | | | | | | 00000000A | 0000C | P.AAA: | .LONG | 10 | : |
| | | | | | | | | | | 000000000 | 00010 | | .ADDRESS | P.AAB | : |
| 45 | 5A | 49 | 52 | 4F | 48 | 54 | 55 | 41 | | 00014 | P.AAD: | .ASCII | \AUTHORIZE\ | : | |
| | | | | | | | | | | 0001D | | .BLKB | 3 | : | |
| | | | | | | | | | | 000000009 | 00020 | P.AAC: | .LONG | 9 | : |
| | | | | | | | | | | 000000000 | 00024 | | .ADDRESS | P.AAD | : |
| | | | | | 59 | 41 | 4C | 45 | 44 | 00028 | P.AAF: | .ASCII | \DELAY\ | : | |
| | | | | | | | | | | 0002D | | .BLKB | 3 | : | |
| | | | | | | | | | | 000000005 | 00030 | P.AAE: | .LONG | 5 | : |
| | | | | | | | | | | 000000000 | 00034 | | .ADDRESS | P.AAF | : |
| 44 | 45 | 48 | 43 | 41 | 54 | 45 | 44 | | | 00038 | P.AAH: | .ASCII | \DETACHED\ | : | |
| | | | | | | | | | | 000000008 | 00040 | P.AAG: | .LONG | 8 | : |
| | | | | | | | | | | 000000000 | 00044 | | .ADDRESS | P.AAH | : |
| | | | | | 50 | 4D | 55 | 44 | | 00048 | P.AAJ: | .ASCII | \DUMP\ | : | |
| | | | | | | | | | | 000000004 | 0004C | P.AAI: | .LONG | 4 | : |
| | | | | | | | | | | 000000000 | 00050 | | .ADDRESS | P.AAJ | : |
| | | | | | 52 | 4F | 52 | 52 | 45 | 00054 | P.AAL: | .ASCII | \ERROR\ | : | |
| | | | | | | | | | | 00059 | | .BLKB | 3 | : | |
| | | | | | | | | | | 000000005 | 0005C | P.AAK: | .LONG | 5 | : |
| | | | | | | | | | | 000000000 | 00060 | | .ADDRESS | P.AAL | : |
| | | | | | 54 | 55 | 50 | 4E | 49 | 00064 | P.AAN: | .ASCII | \INPUT\ | : | |
| | | | | | | | | | | 00069 | | .BLKB | 3 | : | |
| | | | | | | | | | | 000000005 | 0006C | P.AAM: | .LONG | 5 | : |
| | | | | | | | | | | 000000000 | 00070 | | .ADDRESS | P.AAN | : |
| 4C | 41 | 56 | 52 | 45 | 54 | 4E | 49 | | | 00074 | P.AAP: | .ASCII | \INTERVAL\ | : | |
| | | | | | | | | | | 000000008 | 0007C | P.AAO: | .LONG | 8 | : |
| | | | | | | | | | | 000000000 | 00080 | | .ADDRESS | P.AAP | : |

| | | | | | | | | | | | | | | | | | | | | | | | |
|----|----|----|----|----|----|----|----------|--------|----------|------------|------------|--------|--------------|--------|----------------|-----------------|--------|-------------------|-------------|--------------|--------|----------------|---|
| 58 | 4F | 42 | 4C | 49 | 41 | 4D | 00084 | P.AAR: | .ASCII | \MAILBOX\ | : | | | | | | | | | | | | |
| | | | | | | | 0008B | | .BLKB | 1 | : | | | | | | | | | | | | |
| | | | | | | | 00000007 | P.AAQ: | .LONG | 7 | : | | | | | | | | | | | | |
| | | | | | | | 00000000 | | .ADDRESS | P.AAR | : | | | | | | | | | | | | |
| 54 | 55 | 50 | 54 | 55 | 4F | | 00094 | P.AAT: | .ASCII | \OUTPUT\ | : | | | | | | | | | | | | |
| | | | | | | | 0009A | | .BLKB | 2 | : | | | | | | | | | | | | |
| | | | | | | | 00000006 | P.AAS: | .LONG | 6 | : | | | | | | | | | | | | |
| | | | | | | | 00000000 | | .ADDRESS | P.AAT | : | | | | | | | | | | | | |
| | | | | | | 31 | 50 | 000A4 | P.AAV: | .ASCII | \P1\ | : | | | | | | | | | | | |
| | | | | | | | 000A6 | | .BLKB | 2 | : | | | | | | | | | | | | |
| | | | | | | | 00000002 | P.AAU: | .LONG | 2 | : | | | | | | | | | | | | |
| | | | | | | | 000C0000 | | .ADDRESS | P.AAV | : | | | | | | | | | | | | |
| 59 | 54 | 49 | 52 | 4F | 49 | 52 | 50 | 000B0 | P.AAX: | .ASCII | \PRIORITY\ | : | | | | | | | | | | | |
| | | | | | | | 00000008 | P.AAW: | .LONG | 8 | : | | | | | | | | | | | | |
| | | | | | | | 00000000 | | .ADDRESS | P.AAX | : | | | | | | | | | | | | |
| 53 | 45 | 47 | 45 | 4C | 49 | 56 | 49 | 52 | 50 | 000C0 | P.AAZ: | .ASCII | \PRIVILEGES\ | : | | | | | | | | | |
| | | | | | | | | | | 000CA | | .BLKB | 2 | : | | | | | | | | | |
| | | | | | | | 0000000A | P.AAY: | .LONG | 10 | : | | | | | | | | | | | | |
| | | | | | | | 00000000 | | .ADDRESS | P.AAZ | : | | | | | | | | | | | | |
| 45 | 4D | 41 | 4E | 5F | 53 | 53 | 45 | 43 | 4F | 52 | 50 | 000D4 | P.ABB: | .ASCII | \PROCESS_NAME\ | : | | | | | | | |
| | | | | | | | 0000000C | P.ABA: | .LONG | 12 | : | | | | | | | | | | | | |
| | | | | | | | 00000000 | | .ADDRESS | P.ABB | : | | | | | | | | | | | | |
| 54 | 49 | 41 | 57 | 5F | 45 | 43 | 52 | 55 | 4F | 53 | 45 | 52 | 000E8 | P.ABD: | .ASCII | \RESOURCE_WAIT\ | : | | | | | | |
| | | | | | | | | | | 000F5 | | | 000F8 | P.ABC: | .LONG | 13 | : | | | | | | |
| | | | | | | | 0000000D | | .ADDRESS | P.ABD | : | | | | | | | | | | | | |
| | | | | | | | 00000000 | | .ASCII | \SCHEDULE\ | : | | | | | | | | | | | | |
| | | | | | | | 00000008 | P.ABE: | .LONG | 8 | : | | | | | | | | | | | | |
| | | | | | | | 00000000 | | .ADDRESS | P.ABF | : | | | | | | | | | | | | |
| 45 | 52 | 55 | 4C | 49 | 41 | 46 | 5F | 45 | 43 | 49 | 56 | 52 | 45 | 53 | 00110 | P.ABH: | .ASCII | \SERVICE_FAILURE\ | : | | | | |
| | | | | | | | | | | | | | 0011F | | .BLKB | 1 | : | | | | | | |
| | | | | | | | 0000000F | P.ABG: | .LONG | 15 | : | | | | | | | | | | | | |
| | | | | | | | 00000000 | | .ADDRESS | P.ABH | : | | | | | | | | | | | | |
| | | | | | | | 00000000 | | .ASCII | \SWAPPING\ | : | | | | | | | | | | | | |
| | | | | | | | 00000008 | P.ABJ: | .LONG | 8 | : | | | | | | | | | | | | |
| | | | | | | | 00000000 | | .ADDRESS | P.ABJ | : | | | | | | | | | | | | |
| | | | | | | | 43 | 49 | 55 | 00134 | P.ABI: | .LONG | 8 | : | | | | | | | | | |
| | | | | | | | | | | 00138 | P.ABL: | .ASCII | \UIC\ | : | | | | | | | | | |
| | | | | | | | | | | 0013B | | .BLKB | 1 | : | | | | | | | | | |
| | | | | | | | 00000003 | P.ABK: | .LONG | 3 | : | | | | | | | | | | | | |
| | | | | | | | 00000000 | | .ADDRESS | P.ABL | : | | | | | | | | | | | | |
| | | | | | | | 54 | 49 | 4D | 49 | 4C | 5F | 54 | 53 | 41 | 00144 | P.ABN: | .ASCII | \AST_LIMIT\ | : | | | |
| | | | | | | | | | | 0014D | | | | | | 0014D | | .BLKB | 3 | : | | | |
| | | | | | | | 00000009 | P.ABM: | .LONG | 9 | : | | | | | | | | | | | | |
| | | | | | | | 00000000 | | .ADDRESS | P.ABN | : | | | | | | | | | | | | |
| | | | | | | | 54 | 49 | 4D | 49 | 4C | 5F | 52 | 45 | 46 | 46 | 55 | 42 | 00158 | P.ABP: | .ASCII | \BUFFER_LIMIT\ | : |
| | | | | | | | 0000000C | P.ABO: | .LONG | 12 | : | | | | | | | | | | | | |
| | | | | | | | 00000000 | | .ADDRESS | P.ABP | : | | | | | | | | | | | | |
| 54 | 49 | 4D | 49 | 4C | 5F | 45 | 55 | 45 | 55 | 51 | 4E | 45 | 0016C | P.ABR: | .ASCII | \ENQUEUE_LIMIT\ | : | | | | | | |
| | | | | | | | | | | 00179 | | | | | | 00179 | | .BLKB | 3 | : | | | |
| | | | | | | | 0000000D | P.ABQ: | .LONG | 13 | : | | | | | | | | | | | | |
| | | | | | | | 00000000 | | .ADDRESS | P.ABR | : | | | | | | | | | | | | |
| | | | | | | | 54 | 4E | 45 | 54 | 58 | 45 | 00184 | P.ABT: | .ASCII | \EXTENT\ | : | | | | | | |
| | | | | | | | | | | 0018A | | | | | | 0018A | | .BLKB | 2 | : | | | |
| | | | | | | | 00000006 | P.ABS: | .LONG | 6 | : | | | | | | | | | | | | |
| | | | | | | | 00000000 | | .ADDRESS | P.ABT | : | | | | | | | | | | | | |
| | | | | | | | 54 | 49 | 4D | 49 | 4C | 5F | 45 | 4C | 49 | 46 | 00194 | P.ABV: | .ASCII | \FILE_LIMIT\ | : | | |
| | | | | | | | | | | 0019E | | | | | | 0019E | | .BLKB | 2 | : | | | |

```

0000000A 001A0 P.ABU: .LONG 10
00000000' 001A4 .ADDRESS P.ABV
44 45 52 45 46 46 55 42 5F 4F 49 001A8 P.ABX: .ASCII \IO_BUFFERED\
001B3 .BLKB 1
0000000B 001B4 P.ABW: .LONG 11
00000000' 001B8 .ADDRESS P.ABX
54 43 45 52 49 44 5F 4F 49 001BC P.ABZ: .ASCII \IO_DIRECT\
001C5 .BLKB 3
00000009 001C8 P.ABY: .LONG 9
00000000' 001CC .ADDRESS P.ABZ
47 4E 49 4B 52 4F 57 5F 4D 55 4D 49 58 41 4D 001D0 P.ACB: .ASCII \MAXIMUM_WORKING_SET\
54 45 53 5F 001DF .BLKB 1
00000013 001E4 P.ACA: .LONG 19
00000000' 001E8 .ADDRESS P.ACB
45 4C 49 46 5F 45 47 41 50 001EC P.ACD: .ASCII \PAGE_FILE\
001F5 .BLKB 3
00000009 001F8 P.ACC: .LONG 9
00000000' 001FC .ADDRESS P.ACD
54 49 4D 49 4C 5F 45 55 45 55 51 00200 P.ACF: .ASCII \QUEUE_LIMIT\
0020B .BLKB 1
0000000B 0020C P.ACE: .LONG 11
00000000' 00210 .ADDRESS P.ACF
49 4D 49 4C 5F 53 53 45 43 4F 52 50 42 55 53 00214 P.ACH: .ASCII \SUBPROCESS_LIMIT\
54 00223 .BLKB 2
00000010 00224 P.ACG: .LONG 16
00000000' 00228 .ADDRESS P.ACH
54 49 4D 49 4C 5F 45 4D 49 54 0022C P.ACJ: .ASCII \TIME_LIMIT\
00236 .BLKB 2
0000000A 00238 P.ACI: .LONG 10
00000000' 0023C .ADDRESS P.ACJ
54 45 53 5F 47 4E 49 4B 52 4F 57 00240 P.ACL: .ASCII \WORKING_SET\
0024B .BLKB 1
0000000B 0024C P.ACK: .LONG 11
00000000' 00250 .ADDRESS P.ACL
41 54 4F 55 51 5F 45 4C 42 41 54 5F 42 4F 4A 00254 P.ACN: .ASCII \JOB_TABLE_QUOTA\
00263 .BLKB 1
0000000F 00264 P.ACM: .LONG 15
00000000' 00268 .ADDRESS P.ACN

```

.PSECT \$OWNS,NOEXE,2

```

00000 RUNSA_ERROR:
.BLKB 8
00008 RUNSA_IMAGE:
.BLKB 8
00010 RUNSA_INPUT:
.BLKB 8
00018 RUNSA_INPUT_DESC:
.BLKB 8
00020 RUNSA_OUTPUT:
.BLKB 8
00028 RUNSA_PRCNAM:
.BLKB 8
00030 RUNSA_QUOTA:
.BLKB 4
00034 RUNSL_BASPRI:

```

.....

```
00038 RUNSL_MBXUNT: .BLKB 4
0003C RUNSL_PID: .BLKB 4
00040 RUNSL_STATUS: .BLKB 4
00044 RUNSL_STSFLG: .BLKB 4
00048 RUNSL_UIC: .BLKB 4
FFFFFFF FFFFFFFF 0004C RUNSQ_DAYTIM: .LONG -1, -1
00000000 00000000 00054 RUNSQ_INTERVAL: .LONG 0, 0
0005C RUNSQ_PRIVADR: .BLKB 8
00064 RUNSA_IMAGE_BUF: .BLKB 255
00163 .BLKB 1
00164 RUNSA_FINDFILE: .BLKB 4
0309 0004 00168 RUNSA_GETJPI: .WORD 4, 777
00000000' 0016C .ADDRESS RUNSL_BASPRI
00000000 00170 .LONG 0
0204 0008 00174 .WORD 8, 516
00000000' 00178 .ADDRESS RUNSQ_PRIVADR
00000000 0017C .LONG 0
0000 0000 00180 .WORD 0, 0
00000000 00184 .LONG 0, 0
01 0018C RUNSA_QUOTA_TBL: .BYTE 1
00000000' 0018D .ADDRESS P.ABM
03 00191 .BYTE 3
00000000' 00192 .ADDRESS P.ABO
0C 00196 .BYTE 12
00000000' 00197 .ADDRESS P.ABQ
0D 0019B .BYTE 13
00000000' 0019C .ADDRESS P.ABS
06 001A0 .BYTE 6
00000000' 001A1 .ADDRESS P.ABU
02 001A5 .BYTE 2
00000000' 001A6 .ADDRESS P.ABW
05 001AA .BYTE 5
00000000' 001AB .ADDRESS P.ABY
0A 001AF .BYTE 10
00000000' 001B0 .ADDRESS P.ACA
07 001B4 .BYTE 7
00000000' 001B5 .ADDRESS P.ACC
09 001B9 .BYTE 9
00000000' 001BA .ADDRESS P.ACE
08 001BE .BYTE 8
00000000' 001BF .ADDRESS P.ACG
04 001C3 .BYTE 4
00000000' 001C4 .ADDRESS P.ACI
0B 001C8 .BYTE 11
```

.....

00000000' 001C9 .ADDRESS P.ACK
OE 001CD .BYTE 14
00000000' 001CE .ADDRESS P.ACM

RUNS_INSVIRMEM== 12587764
RUNS_INVQUAVAL== 12587818
RUNS_PARSEFAIL== 12587596
RUNS_SYNTAX== 12587260
ACCOUNTING= P.AAA
AUTHORIZE= P.AAC
DELAY= P.AAE
DETACH= P.AAG
DUMP= P.AAI
ERROR= P.AAK
INPUT= P.AAM
INTERVAL= P.AAO
MAILBOX= P.AAQ
OUTPUT= P.AAS
P1= P.AAU
PRIORITY= P.AAW
PRIVILEGES= P.AAY
PROCESS= P.ABA
RESOURCE_WAIT= P.ABC
SCHEDULE= P.ABE
SERVICE_FAIL= P.ABG
SWAPPING= P.ABI
UIC= P.ABK
.EXTRN CLISGET_VALUE, CLISPRESENT
.EXTRN LIBSCVT_DTIME, LIBSCVT_DX_DX
.EXTRN LIBSCVT_TIME, LIBSGET_VM
.EXTRN LIBSFIND_FILE, LIBSTPARSE
.EXTRN PRVSSETPRIV, CLIS_NEGATED
.EXTRN RUNS_CREPRC, RUNS_PROC_ID
.EXTRN RUNS_GETJPI, RUNS_CVTERR
.EXTRN RUNS_INVUIC, RUNS_ILLVAL
.EXTRN RUNS_SCHDWK, SYSSCREPRC
.PSECT \$CODE\$,NOWRT,2

0004 00000 RUN_DETACHED:
0000V 52 0000' CF 9E 00002 .WORD Save R2
CF 00 FB 00007 MOVAB RUNSL_STSFLG, R2
7E D4 0000C CALLS #0, INIT_ARG_LIST
62 DD 0000E CLRL -(SP)
F4 A2 DD 00010 PUSHL RUNSL_STSFLG
04 A2 DD 00013 PUSHL RUNSL_MBXUNT
F0 A2 DD 00016 PUSHL RUNSL_UIC
E4 A2 9F 00019 PUSHL RUNSL_BASPRI
EC A2 DD 0001C PUSHAB RUNSA_PRCNAM
18 A2 9F 0001F PUSHL RUNSA_QUOTA
BC A2 9F 00022 PUSHAB RUNSQ_PRIVADR
DC A2 9F 00025 PUSHAB RUNSA_ERROR
CC A2 9F 00028 PUSHAB RUNSA_OUTPUT
C4 A2 9F 0002B PUSHAB RUNSA_INPUT
F8 A2 9F 0002E PUSHAB RUNSA_IMAGE
00000000G 00 OD FB 00031 PUSHL RUNSL_PID
CALLS #13, SYSSCREPRC

⋮

⋮ 0249
⋮ 0282
⋮ 0295
⋮

RUNDET
V04-000

Run Detached Process -- CLI Utility Procedure L 13
run_detached -- Create a sub or detached proces 14-Sep-1984 12:08:54

VAX-11 Bliss-32 V4.0-742
[CLIUTL.SRC]RUNDET.B32;1

Page 14
(6)

| | | | | | | | | | | | |
|----|-----------|----|----|----|-------|-------|------|-------|-----------------------|--|------|
| | FC | A2 | | 50 | D0 | 00038 | | MOVL | R0, RUNSL_STATUS | | |
| | | 14 | | 50 | E8 | 0003C | | BLBS | R0, 1\$ | | |
| | | | FC | A2 | DD | 0003F | | PUSHL | RUNSL_STATUS | | 0297 |
| | | | | 7E | D4 | 00042 | | CLRL | -(SP)- | | |
| | 00000000G | 00 | | 8F | DD | 00044 | | PUSHL | #RUNS_CREPRC | | |
| | | | | 03 | FB | 0004A | | CALLS | #3, LIB\$STOP | | |
| | | | F8 | A2 | DD | 00053 | 1\$: | BRB | 2\$ | | |
| | | | | 01 | DD | 00056 | | PUSHL | RUNSL_PID | | 0299 |
| | | | | 01 | DD | 00056 | | PUSHL | #1 | | |
| | 00000000G | 00 | | 8F | DD | 00058 | | PUSHL | #RUNS_PROC_ID | | |
| | | | | 03 | FB | 0005E | | CALLS | #3, LIB\$SIGNAL | | |
| 05 | | 62 | | 05 | E1 | 00065 | 2\$: | BBC | #5, RUNSL_STSFLG, 3\$ | | 0301 |
| | 0000V | CF | | 00 | FB | 00069 | | CALLS | #0, SCHEDULE_PROCESS | | 0302 |
| | | | | 04 | 0006E | 3\$: | | RET | | | 0304 |

: Routine Size: 111 bytes. Routine Base: \$CODE\$ + 0000

: 310 0305 1


```

312 0306 1 %SBTTL 'init_arg_list -- Initialize argument lists'
313 0307 1 ROUTINE init_arg_list : NOVALUE =
314 0308 1 ++
315 0309 1
316 0310 1 Functional Description:
317 0311 1
318 0312 1 This routine is responsible for calling the procedures to parse
319 0313 1 the command line and establish the arguments for creating and
320 0314 1 scheduling the requested process.
321 0315 1
322 0316 1 Implicit Inputs:
323 0317 1
324 0318 1 Global data for the argument lists, command line entities etc..
325 0319 1
326 0320 1 Implicit Outputs:
327 0321 1
328 0322 1 The command line has been parsed with the resultant information
329 0323 1 available for calls to the create process ($CREPRC) and the schedule
330 0324 1 wake up ($SCHDWK) system services.
331 0325 1
332 0326 1 Side Effects:
333 0327 1
334 0328 1 Errors encountered during initialization will be signalled by
335 0329 1 the routine which detected the problem.
336 0330 1
337 0331 1 --
338 0332 2 BEGIN
339 0333 2
340 0334 2 Initialize the various descriptors.
341 0335 2
342 0336 2 CH$FILL (0, dsc$c_s_bln, run$a_image); ! Image file specification descriptor.
343 0337 2 CH$FILL (0, dsc$c_s_bln, run$a_input); ! Input device specification descriptor.
344 0338 2 CH$FILL (0, dsc$c_s_bln, run$a_input_desc); ! Initialize the dynamic input descriptor.
345 0339 2 CH$FILL (0, dsc$c_s_bln, run$a_output); ! Output device specification descriptor.
346 0340 2 CH$FILL (0, dsc$c_s_bln, run$a_error); ! Error device specification descriptor.
347 0341 2 CH$FILL (0, dsc$c_s_bln, run$a_prcnam); ! Process name descriptor.
348 0342 2 run$a_input [dsc$b_class] = dsc$k_class_d; ! Input device.
349 0343 2 run$a_output [dsc$b_class] = dsc$k_class_d; ! Output device.
350 0344 2 run$a_error [dsc$b_class] = dsc$k_class_d; ! Error device.
351 0345 2 run$a_prcnam [dsc$b_class] = dsc$k_class_d; ! Process name.
352 0346 2 run$a_input_desc [dsc$b_dtype] = dsc$k_dtype_t; ! General purpose input descriptor (type req for con
353 0347 2 run$a_input_desc [dsc$b_class] = dsc$k_class_d; ! General purpose input descriptor.
354 0348 2
355 0349 3 IF NOT (run$l_status = $GETJPI (ITMLST = run$a_getjpi)) ! Obtain the necessary job and process info.
356 0350 2 THEN SIGNAL_STOP (run$_getjpi, 0, .run$_status); ! If we encounter an error, quit.
357 0351 2
358 0352 2 CLISGET_VALUE (input, run$a_input); ! Obtain the input device specification.
359 0353 2 CLISGET_VALUE (output, run$a_output); ! Obtain the output device specification.
360 0354 2 CLISGET_VALUE (error, run$a_error); ! Obtain the error device specification.
361 0355 2 CLISGET_VALUE (process, run$a_prcnam); ! Obtain the process name.
362 0356 2
363 0357 2 parse_image_spec (); ! Obtain the expanded image file specification.
364 0358 2 get_value (mailbox, run$_mbxunt); ! Get the mailbox unit number if supplied.
365 0359 2 get_value (priority, run$_baspri); ! Get the process base priority if supplied.
366 0360 2 get_wakeup_info (); ! See if we should leave a wake up call.
367 0361 2 get_uic (); ! Get the user identification code (UIC).
368 0362 2 get_privileges (); ! Set up the process privilege vector.

```

: 369
: 370
: 371
: 372

0363 2 get_quotas ();
0364 2 get_stsflgs ();
0365 2
0366 1 END; ! of ROUTINE init_arg_list

! Set up the default process quotas.
! Set up the initial process context flags.

| | | | | .EXTRN SYSSGETJPI | | | |
|----|-----------|--------------------------|-----------|-------------------|----------|--|------|
| | | 01FC 0000 INIT_ARG_LIST: | | | | | |
| | | 58 | 0000' | CF | 9E 00002 | WORD Save R2,R3,R4,R5,R6,R7,R8 | 0307 |
| | | 57 | 00000000G | 00 | 9E 00007 | MOVAB INPUT, R8 | |
| | | 56 | 0000' | CF | 9E 0000E | MOVAB CLISGET VALUE, R7 | |
| 08 | 00 | 6E | | 00 | 2C 00013 | MOVAB RUNSA_INPUT, R6 | |
| | | | F8 | A6 | 00018 | MOVCS #0, (SP), #0, #8, RUNSA_IMAGE | 0336 |
| 08 | 00 | 6E | | 00 | 2C 0001A | MOVCS #0, (SP), #0, #8, RUNSA_INPUT | 0337 |
| | | | | 66 | 0001F | | |
| 08 | 00 | 6E | | 00 | 2C 00020 | MOVCS #0, (SP), #0, #8, RUNSA_INPUT_DESC | 0338 |
| | | | 08 | A6 | 00025 | | |
| 08 | 00 | 6E | | 00 | 2C 00027 | MOVCS #0, (SP), #0, #8, RUNSA_OUTPUT | 0339 |
| | | | 10 | A6 | 0002C | | |
| 08 | 00 | 6E | | 00 | 2C 0002E | MOVCS #0, (SP), #0, #8, RUNSA_ERROR | 0340 |
| | | | F0 | A6 | 00033 | | |
| 08 | 00 | 6E | | 00 | 2C 00035 | MOVCS #0, (SP), #0, #8, RUNSA_PRCNAM | 0341 |
| | | | 18 | A6 | 0003A | | |
| | 03 | A6 | | 02 | 90 0003C | MOVB #2, RUNSA_INPUT+3 | 0342 |
| | 13 | A6 | | 02 | 90 00040 | MOVB #2, RUNSA_OUTPUT+3 | 0343 |
| | F3 | A6 | | 02 | 90 00044 | MOVB #2, RUNSA_ERROR+3 | 0344 |
| | 1B | A6 | | 02 | 90 00048 | MOVB #2, RUNSA_PRCNAM+3 | 0345 |
| | 0A | A6 | 020E | 8F | B0 0004C | MOVW #526, RUNSA_INPUT_DESC+2 | 0346 |
| | | | | 7E | 7C 00052 | CLRQ -(SP) | 0349 |
| | | | | 7E | D4 00054 | CLRL -(SP) | |
| | | | 0158 | C6 | 9F 00056 | PUSHAB RUNSA_GETJPI | |
| | | | | 7E | 7C 0005A | CLRQ -(SP) | |
| | | | | 7E | D4 0005C | CLRL -(SP) | |
| | 00000000G | 00 | | 07 | FB 0005E | CALLS #7, SYSSGETJPI | |
| | 30 | A6 | | 50 | D0 00065 | MOVL R0, RUNSL_STATUS | |
| | | 12 | | 50 | E8 00069 | BLBS R0, 1\$ | |
| | | | 30 | A6 | DD 0006C | PUSHL RUNSL_STATUS | 0350 |
| | | | | 7E | D4 0006F | CLRL -(SP) | |
| | 00000000G | 00 | 00000000G | 8F | DD 00071 | PUSHL #RUNS_GETJPI | |
| | | | | 03 | FB 00077 | CALLS #3, LIBSSTOP | |
| | | | | 56 | DD 0007E | PUSHL R6 | 0352 |
| | | | | 58 | DD 00080 | PUSHL R8 | |
| | | 67 | | 02 | FB 00082 | CALLS #2, CLISGET VALUE | |
| | | | 10 | A6 | 9F 00085 | PUSHAB RUNSA_OUTPUT | 0353 |
| | | | 30 | A8 | 9F 00088 | PUSHAB OUTPUT | |
| | | 67 | | 02 | FB 0008B | CALLS #2, CLISGET VALUE | |
| | | | F0 | A6 | 9F 0008E | PUSHAB RUNSA_ERROR | 0354 |
| | | | F0 | A8 | 9F 00091 | PUSHAB ERROR | |
| | | 67 | | 02 | FB 00094 | CALLS #2, CLISGET VALUE | |
| | | | 18 | A6 | 9F 00097 | PUSHAB RUNSA_PRCNAM | 0355 |
| | | | 74 | A8 | 9F 0009A | PUSHAB PROCESS | |
| | | 67 | | 02 | FB 0009D | CALLS #2, CLISGET VALUE | |
| | 0000V | CF | | 00 | FB 000A0 | CALLS #0, PARSE_IMAGE_SPEC | 0357 |
| | | | 28 | A6 | 9F 000A5 | PUSHAB RUNSL_MBXUNT | 0358 |

RUNDET
V04-000

Run Detached Process -- CLI Utility Procedure
init_arg_list -- Initialize argument lists

B 14
16-Sep-1984 00:27:00
14-Sep-1984 12:08:54

VAX-11 Bliss-32 V4.0-742
[CLIUTL.SRC]RUNDET.832;1

Page 17
(7)

| | | | | | | | | |
|-------|----|----|----|-------|-------|--------|---------------------|--------|
| 0000V | CF | 20 | A8 | 9F | 000A8 | PUSHAB | MAILBOX | : |
| | | | 02 | FB | 000AB | CALLS | #2, GET_VALUE | : |
| | | 24 | A6 | 9F | 000B0 | PUSHAB | RUN\$B\$P\$P\$R\$I | : 0359 |
| | | 4C | A8 | 9F | 000B3 | PUSHAB | PRIORITY | : |
| 0000V | CF | | 02 | FB | 000B6 | CALLS | #2, GET_VALUE | : |
| 0000V | CF | | 00 | FB | 000BB | CALLS | #0, GET_WAKEUP_INFO | : 0360 |
| 0000V | CF | | 00 | FB | 000C0 | CALLS | #0, GET_UIC | : 0361 |
| 0000V | CF | | 00 | FB | 000C5 | CALLS | #0, GET_PRIVILEGES | : 0362 |
| 0000V | CF | | 00 | FB | 000CA | CALLS | #0, GET_QUOTAS | : 0363 |
| 0000V | CF | | 00 | FB | 000CF | CALLS | #0, GET_STSFLGS | : 0364 |
| | | | 04 | 000D4 | RET | | | : 0366 |

: Routine Size: 213 bytes, Routine Base: \$CODE\$ + 006F

: 373 0367 1

```

375 0368 1 %SBTTL 'parse_image_spec -- Parse image file specification'
376 0369 1 ROUTINE parse_image_spec : NOVALUE =
377 0370 1 ++
378 0371 1
379 0372 1 Functional Description:
380 0373 1
381 0374 1 This routine obtains the image file specification from the command
382 0375 1 line and parses it, producing a resultant file specification with
383 0376 1 defaults applied.
384 0377 1
385 0378 1 Implicit Inputs:
386 0379 1
387 0380 1 none
388 0381 1
389 0382 1 Implicit Outputs:
390 0383 1
391 0384 1 The image file has been parsed with the resultant file specification in
392 0385 1 run$a_image_buf, and the appropriate fields of the descriptor run$a_image
393 0386 1 initialized.
394 0387 1
395 0388 1 Side Effects:
396 0389 1
397 0390 1 Parse errors from RMS or errors from LIB$FIND_FILE will result
398 0391 1 with a FATAL error signalled.
399 0392 1
400 0393 1 --
401 0394 2 BEGIN
402 0395 2
403 0396 2 bind
404 0397 2 default_name = uplit (%charcount('.EXE'),uplit byte ('.EXE'));
405 0398 2
406 0399 2 local
407 0400 2 status,
408 0401 2 findfilenam : ref block [ ,byte ];
409 0402 2
410 0403 2 CLISGET_VALUE (p1, run$a_input_desc); ! Obtain the image file specification.
411 0404 2
412 0405 2 run$a_image [DSC$B_CLASS] = DSC$K_CLASS_D;
413 0406 2 run$a_image [DSC$B_DTYPE] = DSC$K_DTYPE_T;
414 0407 2
415 0408 2 ! Get the next file name to search for, no wildcards permitted.
416 0409 2
417 0410 2 status = LIB$FIND_FILE(
418 0411 2 run$a_input_desc, run$a_image,
419 0412 2 run$a_findfile,
420 0413 2 default_name, 0, 0, %REF(1));
421 0414 2
422 0415 2 ! If the filename has wildcards in it it's an error
423 0416 2
424 0417 2 if (.status and sts$m_msg_no) eql shr$_nowild
425 0418 2 then
426 0419 2 run$a_findfile [fab$l_sts] = .status;
427 0420 2
428 0421 2 ! Report miscellaneous errors from LIB$FIND_FILE
429 0422 2
430 0423 2 if not .status
431 0424 2 then
  
```

```

: 432 0425 2 SIGNAL_STOP (run$_parsefail, 1
: 433 0426 ~~~~~ run$a_input_desc, .run$a_findfile [fab$l_sts], .run$a_findfile [fab$l_stv]);
: 434 0427 ~~~~~
: 435 0428 ~~~~~ findfilenam = .run$a_findfile [fab$l_nam];
: 436 0429 ~~~~~
: 437 0430 ~~~~~ ! If an explicit version number was not specified, remove the version number.
: 438 0431 ~~~~~
: 439 0432 ~~~~~ IF NOT .findfilenam [nam$b_exp_ver]
: 440 0433 ~~~~~ THEN
: 441 0434 ~~~~~ run$a_image [dsc$b_length] = .run$a_image [dsc$b_length] - .findfilenam [nam$b_ver];
: 442 0435 ~~~~~
: 443 0436 1 END; ! of ROUTINE parse_image_spec
  
```

```

.PSECT SPLITS, NOWRT, NOEXE, 2
45 58 45 2E 0026C P.ACP: .ASCII \.EXE\
00000004 00270 P.ACO: .LONG 4
00000000' 00274 .ADDRESS P.ACP
  
```

DEFAULT_NAME= P.ACO

.PSECT \$CODE\$, NOWRT, 2

```

0004 00000 PARSE_IMAGE_SPEC:
52 0000' CF 9E 00002 .WORD Save R2
5E 04 C2 00007 MOVAB RUN$a_FINDFILE, R2
FEB4 C2 9F 0000A SUBL2 #4, SP
0000' CF 9F 0000E PUSHAB RUN$a_INPUT_DESC
00000000G 00 02 FB 00012 PUSHAB P1
FEA6 C2 020E 8F B0 00019 CALLS #2, CLISGET_VALUE
6E 01 D0 00020 MOVW #526, RUN$a_IMAGE+2
5E DD 00023 MOVL #1, (SP)
7E 7C 00025 PUSHL SP
0000' CF 9F 00027 CLRQ -(SP)
52 DD 0002B PUSHAB DEFAULT_NAME
FEA4 C2 9F 0002D PUSHL R2
FEB4 C2 9F 00031 PUSHAB RUN$a_IMAGE
00000000G 00 07 FB 00035 PUSHAB RUN$a_INPUT_DESC
50 51 FFFF0007 50 D0 0003C CALLS #7, LIB$FIND_FILE
00001128 8F 8F CB 0003F MOVL R0, STATUS
50 D1 00047 BICL3 #-65529, STATUS, R0
07 12 0004E CMPL R0, #4592
BNEQ 1$
08 50 62 D0 00050 MOVL RUN$a_FINDFILE, R0
A0 51 D0 00053 MOVL STATUS, 8(R0)
1A 51 E8 00057 1$: BLBS STATUS, 2$
50 62 D0 0005A MOVL RUN$a_FINDFILE, R0
7E 08 A0 7D 0005D MOVQ 8(R0), -(SP)
FEB4 C2 9F 00061 PUSHAB RUN$a_INPUT_DESC
01 DD 00065 PUSHL #1
00000000G 00 08C0124C 8F DD 00067 PUSHL #12587596
50 05 FB 0006D CALLS #5, LIB$STOP
50 62 D0 00074 2$: MOVL RUN$a_FINDFILE, R0
50 28 A0 D0 00077 MOVL 40(R0), FINDFILENAM
  
```

RUNDET
V04-000

Run Detached Process -- CLI Utility Procedure ^{E 14} 16-Sep-1984 00:27:00 VAX-11 Bliss-32 V4.0-742
parse_image_spec -- Parse image file specificat 14-Sep-1984 12:08:54 [CLIUTL.SRC]RUNDET.B32;1

Page 20
(8)

| | | | | | | | | |
|------|----|----|----|-------|-------|--------|---------------------|--|
| | 09 | 34 | A0 | E8 | 0007B | BLBS | 52(FINDFILENAM), 38 | |
| | 51 | 3D | A0 | 9A | 0007F | MOVZBL | 61(FINDFILENAM); R1 | |
| FEA4 | C2 | | 51 | A2 | 00083 | SUBW2 | R1, RUNSA_IMAGE | |
| | | | 04 | 00088 | 38: | RET | | |

: 0432
: 0434
: 0436

; Routine Size: 137 bytes, Routine Base: \$CODE\$ + 0144

; 444 0437 1

```

: 446 0438 1 %SBTTL 'get_wakeup_info -- Process the $$SCHDWK time values'
: 447 0439 1 ROUTINE get_wakeup_info : NOVALUE =
: 448 0440 1 ++
: 449 0441 1
: 450 0442 1 Functional Description:
: 451 0443 1
: 452 0444 1 This routine is responsible for obtaining and converting the time values
: 453 0445 1 used to schedule wake up requests for the created process.
: 454 0446 1
: 455 0447 1 Implicit Outputs:
: 456 0448 1
: 457 0449 1 runSq_interval delta Reschedule the process to execute at
: 458 0450 1 this interval.
: 459 0451 1
: 460 0452 1 runSq_daytim absolute or Schedule the process to execute at
: 461 0453 1 delta this time.
: 462 0454 1
: 463 0455 1 Side Effects:
: 464 0456 1
: 465 0457 1 The time quantities are obtained as .ASCII strings from the CLI. To
: 466 0458 1 convert them we call the appropriate library routine. Any errors
: 467 0459 1 encountered during the conversion will be signalled, and execution
: 468 0460 1 of this image terminated.
: 469 0461 1
: 470 0462 1 --
: 471 0463 2 BEGIN
: 472 0464 2
: 473 0465 2 IF CLISGET_VALUE (delay, run$a_input_desc) ! Was /DELAY specified?
: 474 0466 2 THEN
: 475 0467 3 IF NOT (run$l_status = LIB$CVT_DTIME (run$a_input_desc, run$q_daytim))
: 476 0468 2 THEN
: 477 0469 2 SIGNAL_STOP (run$cvterr, 2, delay, run$a_input_desc, .run$l_status);
: 478 0470 2
: 479 0471 2 IF CLISGET_VALUE (interval, run$a_input_desc) ! Was /INTERVAL specified?
: 480 0472 2 THEN
: 481 0473 3 IF NOT (run$l_status = LIB$CVT_DTIME (run$a_input_desc, run$q_interval))
: 482 0474 2 THEN
: 483 0475 2 SIGNAL_STOP (run$cvterr, 2, interval, run$a_input_desc, .run$l_status);
: 484 0476 2
: 485 0477 2 IF CLISGET_VALUE (schedule, run$a_input_desc) ! How about /SCHEDULE?
: 486 0478 2 THEN
: 487 0479 3 IF NOT (run$l_status = LIB$CVT_TIME (run$a_input_desc, run$q_daytim))
: 488 0480 2 THEN
: 489 0481 2 SIGNAL_STOP (run$cvterr, 2, schedule, run$a_input_desc, .run$l_status);
: 490 0482 2
: 491 0483 1 END; ! of ROUTINE get_wakeup_info

```

00FC 00000 GET_WAKEUP_INFO:

```

57 00000000G 00 9E 00002 .WORD Save R2,R3,R4,R5,R6,R7
56 00000000G 00 9E 00009 MOVAB LIB$CVT_DTIME, R7
55 00000000G 8F D0 00010 MOVAB LIB$STOP, R6
54 00000000G 00 9E 00017 MOVL #RUN$ CVERR, R5
MOVAB CLISGET_VALUE, R4

```

: 0439
:
:
:
:

| | | | | | | | |
|-----------|-------|----|-------|-------|--------|----------------------|------|
| 53 | 0000' | CF | 9E | 0001E | MOVAB | DELAY, R3 | |
| 52 | 0000' | CF | 9E | 00023 | MOVAB | RUNSA_INPUT_DESC, R2 | |
| | | S2 | DD | 00028 | PUSHL | R2 | 0465 |
| | | S3 | DD | 0002A | PUSHL | R3 | |
| 64 | | 02 | FB | 0002C | CALLS | #2, CLISGET_VALUE | |
| 1D | | 50 | E9 | 0002F | BLBC | R0, 1\$ | |
| | 34 | A2 | 9F | 00032 | PUSHAB | RUNSQ_DAYTIM | 0467 |
| | | S2 | DD | 00035 | PUSHL | R2 | |
| 67 | | 02 | FB | 00037 | CALLS | #2, LIBSCVT_DTIME | |
| 28 | A2 | 50 | DD | 0003A | MOVL | R0, RUNSL_STATUS | |
| | 0E | 50 | EB | 0003E | BLBS | R0, 1\$ | |
| | | A2 | DD | 00041 | PUSHL | RUNSL_STATUS | 0469 |
| | | S2 | DD | 00044 | PUSHL | R2 | |
| | | S3 | DD | 00046 | PUSHL | R3 | |
| | | 02 | DD | 00048 | PUSHL | #2 | |
| | | S5 | DD | 0004A | PUSHL | R5 | |
| 66 | | 05 | FB | 0004C | CALLS | #5, LIB\$STOP | |
| | | S2 | DD | 0004F | PUSHL | R2 | 0471 |
| | 4C | A3 | 9F | 00051 | PUSHAB | INTERVAL | |
| 64 | | 02 | FB | 00054 | CALLS | #2, CLISGET_VALUE | |
| 1E | | 50 | E9 | 00057 | BLBC | R0, 2\$ | |
| | 3C | A2 | 9F | 0005A | PUSHAB | RUNSQ_INTERVAL | 0473 |
| | | S2 | DD | 0005D | PUSHL | R2 | |
| | | 02 | FB | 0005F | CALLS | #2, LIBSCVT_DTIME | |
| 28 | A2 | 50 | DD | 00062 | MOVL | R0, RUNSL_STATUS | |
| | 0F | 50 | EB | 00066 | BLBS | R0, 2\$ | |
| | | A2 | DD | 00069 | PUSHL | RUNSL_STATUS | 0475 |
| | | S2 | DD | 0006C | PUSHL | R2 | |
| | 4C | A3 | 9F | 0006E | PUSHAB | INTERVAL | |
| | | 02 | DD | 00071 | PUSHL | #2 | |
| | | S5 | DD | 00073 | PUSHL | R5 | |
| 66 | | 05 | FB | 00075 | CALLS | #5, LIB\$STOP | |
| | | S2 | DD | 00078 | PUSHL | R2 | 0477 |
| | 00D8 | C3 | 9F | 0007A | PUSHAB | SCHEDULE | |
| 64 | | 02 | FB | 0007E | CALLS | #2, CLISGET_VALUE | |
| 23 | | 50 | E9 | 00081 | BLBC | R0, 3\$ | |
| | 34 | A2 | 9F | 00084 | PUSHAB | RUNSQ_DAYTIM | 0479 |
| | | S2 | DD | 00087 | PUSHL | R2 | |
| 00000000G | 00 | 02 | FB | 00089 | CALLS | #2, LIBSCVT_TIME | |
| | 28 | 50 | DD | 00090 | MOVL | R0, RUNSL_STATUS | |
| | | 50 | EB | 00094 | BLBS | R0, 3\$ | |
| | | A2 | DD | 00097 | PUSHL | RUNSL_STATUS | 0481 |
| | | S2 | DD | 0009A | PUSHL | R2 | |
| | 00D8 | C3 | 9F | 0009C | PUSHAB | SCHEDULE | |
| | | 02 | DD | 000A0 | PUSHL | #2 | |
| | | S5 | DD | 000A2 | PUSHL | R5 | |
| 66 | | 05 | FB | 000A4 | CALLS | #5, LIB\$STOP | |
| | | 04 | 000A7 | 3\$: | RET | | 0483 |

; Routine Size: 168 bytes, Routine Base: \$CODE\$ + 01CD

; 492 0484 1


```

494 0485 1 %SBTTL 'get_uic -- Process the UIC, converting it to a longword value'
495 0486 1 ROUTINE get_uic : NOVALUE =
496 0487 1 ++
497 0488 1
498 0489 1 Functional Description:
499 0490 1
500 0491 1 This routine is responsible for obtaining and converting a UIC string
501 0492 1 of the form [group, member] to a longword value.
502 0493 1
503 0494 1 Implicit Inputs:
504 0495 1
505 0496 1 run$a_input_desc adr Address of a general purpose dynamic
506 0497 1 string descriptor.
507 0498 1
508 0499 1 Side Effects:
509 0500 1
510 0501 1 If the UIC could not be converted, we will inform the user with
511 0502 1 an invalid UIC diagnostic and exit.
512 0503 1
513 0504 1 --
514 0505 2 BEGIN
515 0506 2
516 0507 2 LOCAL
517 0508 2 TPA_PARAM : $BLOCK [TPASK_LENGTH]
518 0509 2 INITIAL (REP TPASK_LENGTH OF BYTE (0));
519 0510 2
520 0511 2 IF CLISGET_VALUE (uic, run$a_input_desc) ! If the user supplied a UIC
521 0512 2 THEN ! convert it to a longword value.
522 0513 2 BEGIN
523 0514 2 tpa_param [tpa$l_count] = tpa$k_count0;
524 0515 2 tpa_param [tpa$l_stringcnt] = .run$a_input_desc [dsc$w_length];
525 0516 2 tpa_param [tpa$l_stringptr] = .run$a_input_desc [dsc$a_pointer];
526 0517 2
527 0518 2 IF NOT LIB$TPARSE (tpa_param, uic_states, uic_keys)
528 0519 2 THEN
529 0520 2 SIGNAL_STOP (run$_invaic, 1, run$a_input_desc); ! converting the UIC, inform the user.
530 0521 2 END;
531 0522 2
532 0523 1 END; ! of ROUTINE get_uic

```

.PSECT \$SPLITS,NOWRT,NOEXE,2

00# 00278 P.ACQ: .BYTE 0[36]

.PSECT \$CODES,NOWRT,2

| | | | | | | |
|----|----------|-------|------------|----------------|-----------------------|--------|
| | | | 007C 0000 | GET_UIC: .WORD | Save R2,R3,R4,R5,R6 | : 0486 |
| | 56 | 0000' | CF 9E 0002 | MOVAB | RUN\$a_INPUT_DESC, R6 | : |
| | 5E | | 24 C2 0007 | SUBL2 | #36, SP | : |
| 6E | 0000' | CF | 24 28 000A | MOVCS | #36, P.ACQ, TPA_PARAM | : 0509 |
| | | | 56 DD 0010 | PUSHL | R6 | : 0511 |
| | | | CF 9F 0012 | PUSHAB | UIC | : |
| | 0000000G | 00 | 02 FB 0016 | CALLS | #2, CLISGET_VALUE | : |

RUNDET
V04-000

Run Detached Process -- CLI Utility Procedure 16-Sep-1984 00:27:00 VAX-11 Bliss-32 V4.0-742
get_uic -- Process the UIC, converting it to a 14-Sep-1984 12:08:54 [CLIUTL.SRC]RUNDET.B32;1

| | | | | | | | | |
|-----------|----|-----------|----|-------|-------|--------|----------------------------------|------|
| | 32 | | 50 | E9 | 0001D | BLBC | R0, 1\$ | |
| | 6E | | 08 | D0 | 00020 | MOVL | #8, TPA_PARAM | 0514 |
| 08 | AE | | 66 | 3C | 00023 | MOVZWL | RUNSA_INPUT_DESC, TPA_PARAM+8 | 0515 |
| 0C | AE | 04 | A6 | D0 | 00027 | MOVL | RUNSA_INPUT_DESC+4, TPA_PARAM+12 | 0516 |
| | | 0000' | CF | 9F | 0002C | PUSHAB | UIC_KEYS | 0518 |
| | | 0000' | CF | 9F | 00030 | PUSHAB | UIC_STATES | |
| | | 08 | AE | 9F | 00034 | PUSHAB | TPA_PARAM | |
| 00000000G | 00 | | 03 | FB | 00037 | CALLS | #3, LIB\$TPARSE | |
| | 11 | | 50 | E8 | 0003E | BLBS | R0, 1\$ | |
| | | | 56 | DD | 00041 | PUSHL | R6 | 0520 |
| | | | 01 | DD | 00043 | PUSHL | #1 | |
| | | 00000000G | 8F | DD | 00045 | PUSHL | #RUNS_INVUIC | |
| 00000000G | 00 | | 03 | FB | 0004B | CALLS | #3, LIB\$STOP | |
| | | | 04 | 00052 | 1\$: | RET | | 0523 |

: Routine Size: 83 bytes, Routine Base: \$CODE\$ + 0275

: 533 0524 1

```

535 0525 1 %SBTTL 'get_privileges -- Obtain the process privileges
536 0526 1 ROUTINE get_privileges : NOVALUE =
537 0527 1 **
538 0528 1
539 0529 1 Functional Description:
540 0530 1
541 0531 1 This routine iteratively calls the routine CLISGET_VALUE to obtain the
542 0532 1 privileges specified by the user. The private CLI routine PRV$SETPRIV
543 0533 1 is then called to convert the ascii string name into a bit number and
544 0534 1 set/clear the appropriate bit in the privilege mask.
545 0535 1
546 0536 1 Implicit Inputs:
547 0537 1
548 0538 1     prv$a_input_desc   adr   Address of a general purpose dynamic
549 0539 1     string descriptor.
550 0540 1     prv$q_prvadr       adr   Address of the privilege mask quadword.
551 0541 1
552 0542 1 Implicit Outputs:
553 0543 1
554 0544 1 The privilege vector has been established. The privileges are set/cleared
555 0545 1 according to whether they were explicitly specified, explicitly negated
556 0546 1 or set as a result of the 'SAME' privilege.
557 0547 1
558 0548 1 Side Effects:
559 0549 1
560 0550 1 The 'SAME' privilege is special cased...
561 0551 1
562 0552 1 If an invalid privilege keyword is detected we will signal a fatal error.
563 0553 1
564 0554 1 --
565 0555 2 BEGIN
566 0556 2
567 0557 2 BIND
568 0558 2     same = $descriptor ('SAME'): $block,
569 0559 2     nosame = $descriptor ('NOSAME'): $block;
570 0560 2
571 0561 3 IF (CLISPRESENT ($descriptor ('PRIVILEGES.SAME'))) EQL cli$_negated)           ! Default action is to use t
572 0562 2 THEN     run$q_prvadr [1] = run$q_prvadr [0] = 0;                             ! privileges as the creating
573 0563 2
574 0564 2 WHILE CLISGET_VALUE (privileges, run$a_input_desc)                             ! Preclude an invalid keywor
575 0565 2 DO                                                                                   ! screening out the '[NO]SAM
576 0566 2     IF CH$NEQ (.run$a_input_desc [dsc$_length], .run$a_input_desc [dsc$a_pointer], .same [dsc$_length], .s
577 0567 2     AND CH$NEQ (.run$a_input_desc [dsc$_length], .run$a_input_desc [dsc$a_pointer], .nosame [dsc$_length],
578 0568 2     THEN
579 0569 2         IF NOT prv$setpriv (run$a_input_desc, run$q_prvadr)
580 0570 2         THEN     SIGNAL_STOP (run$_invquaval, 2, run$a_input_desc, privileges);
581 0571 2
582 0572 1 END:     ! of ROUTINE get_privileges

```

.PSECT SPLITS,NOWRT,NOEXE,2

```

45 4D 41 53 0029C P.ACS: .ASCII \SAME\
00000004 002A0 P.ACR: .LONG 4
00000000' 002A4 .ADDRESS P.ACS
45 4D 41 53 4F 4E 002A8 P.ACU: .ASCII \NOSAME\

```

```

              002AE
              00000006 002B0 P.ACT: .BLKB 2
              00000000 002B4 .LONG 6
53 45 47 45 4C 49 56 49 52 50 002B8 P.ACW: .ADDRESS P.ACU
              002C7 P.ACW: .ASCII \PRIVILEGES.SAME\
              0000000F 002C8 P.ACV: .BLKB 1
              00000000 002CC P.ACV: .LONG 15
              .ADDRESS P.ACW

```

```

SAME=
NOSAME= P.ACR
          P.ACT

```

```
.PSECT $CODE$,NOWRT,2
```

```
001C 0000 GET_PRIVILEGES:
```

| | | | | | | | | | |
|-------|-----------|----|----------|----|-------|----------|------------|--|------|
| | | 54 | 0000' | CF | 9E | 00002 | .WORD | Save R2,R3,R4 | 0526 |
| | | | 0000' | CF | 9F | 00007 | MOVAB | RUN\$A_INPUT_DESC, R4 | |
| | 00000000G | 00 | | 01 | FB | 0000B | PUSHAB | P.ACV | 0561 |
| | 00000000G | 8F | | 50 | D1 | 00012 | CALLS | #1, CLIS\$PRESENT | |
| | | | | 03 | 12 | 00019 | CML | R0, #CLIS\$_NEGATED | |
| | | 44 | | A4 | 7C | 0001B | BNEQ | 1\$ | 0562 |
| | | | | 54 | DD | 0001E | CLRQ | RUN\$Q_PRVADR | 0564 |
| | | | 0000' | CF | 9F | 00020 | 1\$: PUSHL | R4 | |
| | 00000000G | 00 | | 02 | FB | 00024 | PUSHAB | PRIVILEGES | |
| | | 40 | | 50 | E9 | 0002B | CALLS | #2, CLIS\$GET_VALUE | |
| 0000' | CF | | 00 | 04 | B4 | 0002E | BLBC | R0, 2\$ | |
| | | | 0000' | DF | | 00036 | CMPCS | RUN\$A_INPUT_DESC, @RUN\$A_INPUT_DESC+4, #0, - | 0566 |
| | | | | E3 | 13 | 00039 | BEQL | 1\$ | |
| 0000' | CF | | 00 | 04 | B4 | 0003B | CMPCS | RUN\$A_INPUT_DESC, @RUN\$A_INPUT_DESC+4, #0, - | 0567 |
| | | | 0000' | DF | | 00043 | BEQL | NOSAME, @NOSAME+4 | |
| | | 44 | | A4 | 9F | 00048 | 1\$ PUSHAB | RUN\$Q_PRVADR | 0569 |
| | | | | 54 | DD | 0004B | PUSHL | R4 | |
| | 00000000G | 00 | | 02 | FB | 0004D | CALLS | #2, PRV\$SETPRIV | |
| | | C7 | | 50 | E8 | 00054 | BLBS | R0, 1\$ | |
| | | | 0000' | CF | 9F | 00057 | PUSHAB | PRIVILEGES | 0570 |
| | | | | 54 | DD | 0005B | PUSHL | R4 | |
| | | | | 02 | DD | 0005D | PUSHL | #2 | |
| | 00000000G | 00 | 00C0132A | 8F | DD | 0005F | PUSHL | #12587812 | |
| | | | | 04 | FB | 00065 | CALLS | #4, LIB\$STOP | |
| | | | | B0 | 11 | 0006C | BRB | 1\$ | 0566 |
| | | | | 04 | 0006E | 2\$: RET | | | 0572 |

```
; Routine Size: 111 bytes, Routine Base: $CODE$ + 02C8
```

```
; 583 0573 1
```

```

585 0574 1 %SBTTL 'get_quotas -- Obtain the process quota values'
586 0575 1 ROUTINE get_quotas : NOVALUE =
587 0576 1 !++
588 0577 1
589 0578 1 Functional Description:
590 0579 1
591 0580 1 This routine is responsible for establishing the quota list.
592 0581 1
593 0582 1 NOTE: The ending address of the quota table computation in the INCR
594 0583 1 loop below subtracts 2 from the pql$_length value. This is the
595 0584 1 correction value for the unused pql$_listend entry and the true
596 0585 1 ending address of the quota table.
597 0586 1
598 0587 1 Implicit Inputs:
599 0588 1
600 0589 1 The quota list table.
601 0590 1
602 0591 1 Quota table format:
603 0592 1
604 0593 1 31          7          0
605 0594 1          +-----+
606 0595 1          | name |
607 0596 1          +-----+
608 0597 1 ! cmd line ent desc !
609 0598 1          +-----+
610 0599 1
611 0600 1 Where the name field contains the PQL$_xxx value
612 0601 1 and the cmd line ent desc contains the address
613 0602 1 of the quota name descriptor.
614 0603 1
615 0604 1 Implicit Outputs:
616 0605 1
617 0606 1 The quota list is established.
618 0607 1
619 0608 1 --
620 0609 2 BEGIN
621 0610 2
622 0611 2 LOCAL
623 0612 2 value : volatile; ! Command line entity's value.
624 0613 2
625 0614 2 INCR entry FROM run$a_quota_tbl ! Search through the quota list looking for
626 0615 3 TO run$a_quota_tbl + ((pql$_length - 2) * list_k_entry_size) ! quota info specified in the comman
627 0616 2 BY list_k_entry_size
628 0617 2 DO
629 0618 3 BEGIN ! If we find some, put it into the $CREPRC q
630 0619 3 BIND quota_entry = .entry : $block; ! Change reference structure.
631 0620 3 IF CL$PRESENT (.quota_entry [list_l_value]) ! Was this entity in the command line?
632 0621 3 THEN ! Yes...
633 0622 4 BEGIN
634 0623 4 IF .quota_entry [list_b_name] EQL pql$ cpulm ! Is this quota the CPU time limit?
635 0624 4 THEN get_cpulm (.quota_entry [list_l_value], value) ! Yes, special case for CPU time limit...
636 0625 4 ELSE get_value (.quota_entry [list_l_value], value); ! Normal case, just get the value.
637 0626 4 insert_quota (.quota_entry [list_b_name], .value); ! Insert this quota into the quota list.
638 0627 3 END;
639 0628 2 END; ! of DO statement.
640 0629 2
641 0630 2 IF .run$a_quota NEQ 0 ! If any quotas have been processed, be sure

```

RUNDET
V04-000

Rur Detached Process -- CLI Utility Procedure
get_quotas -- Obtain the process quota values

M 14
16-Sep-1984 00:27:00
14-Sep-1984 12:08:54

VAX-11 Bliss-32 V4.0-742
[CLIUTL.SRC]RUNDET.B32;1

Page 28
(12)

: 642
: 643
: 644

0631 2 THEN insert_quota (pql\$_listend, 0);
0632 2
0633 1 END; ! of ROUTINE get_quotas

! terminate the list.

```
000C 0000 GET_QUOTAS:
      52 0000' CF 9E 00002 .WORD Save R2,R3
      53 0000' CF 9E 00007 MOVAB RUNSA_QUOTA_TBL, R2
      35 10 0000C MOVAB RUNSA_QUOTA_TBL+65, R3
      01 A2 DD 0000E 1$: BSBB 5$
00000000G 00 01 01 FB 00011 PUSHL 1(ENTRY)
      25 50 E9 00018 CALLS #1, CLISPRESNT
      04 62 91 0001B BLBC R0, 4$
      0C 12 0001E CMPB (ENTRY), #4
      5E DD 00020 BNEQ 2$
      01 A2 DD 00022 PUSHL SP
0000V CF 01 02 FB 00025 PUSHL 1(ENTRY)
      0A 11 0002A CALLS #2, GET_CPULM
      5E DD 0002C 2$: BRB 3$
0000V CF 01 A2 DD 0002E PUSHL SP
      02 FB 00031 CALLS #2, GET_VALUE
      6E DD 00036 3$: PUSHL VALUE
0000V 7E 62 9A 00038 MOVZBL (ENTRY), -(SP)
      CF 02 FB 0003B CALLS #2, INSERT_QUOTA
      52 05 C0 00040 4$: ADDL2 #5, ENTRY
      53 52 D1 00043 5$: CMPL ENTRY, R3
      0000' C6 15 00046 BLEQ 1$
      07 D5 00048 TSTL RUNSA_QUOTA
      7E 7C 0004E BEQL 6$
0000V CF 02 FB 00050 CLRQ -(SP)
      04 00055 6$: CALLS #2, INSERT_QUOTA
      RET
: 0575
: 0614
: 0615
: 0614
: 0620
: 0623
: 0624
: 0625
: 0626
: 0614
: 0630
: 0631
: 0633
```

: Routine Size: 86 bytes, Routine Base: \$CODE\$ + 0337

: 645 0634 1

```

647 0635 1 %SBTTL 'get_cpulm -- Special case the CPU time limit quota'
648 0636 1 ROUTINE get_cpulm (entry, value) : NOVALUE =
649 0637 1 ++
650 0638 1
651 0639 1 Functional Description:
652 0640 1
653 0641 1 This routine handles the special case of the CPU time limit quota,
654 0642 1 since a simple string to numeric conversion is not the case.
655 0643 1
656 0644 1 We perform the CLI call back locally to get the cpu time limit string,
657 0645 1 then we call the time conversion routine followed by reducing the time
658 0646 1 value to a single longword.
659 0647 1
660 0648 1 Inputs:
661 0649 1
662 0650 1 entry adr Address of the string descriptor for the
663 0651 1 cpu time limit quota command qualifier.
664 0652 1
665 0653 1 Outputs:
666 0654 1
667 0655 1 value adr The address of the resultant time value.
668 0656 1
669 0657 1 Side Effects:
670 0658 1
671 0659 1 If any errors are encountered the will be signalled.
672 0660 1
673 0661 1 --
674 0662 2 BEGIN
675 0663 2
676 0664 2 BUILTIN
677 0665 2 EDIV;
678 0666 2
679 0667 2 LOCAL
680 0668 2 delta : VECTOR [2, LONG], ! Delta time quadword.
681 0669 2 psl : $bblock [4]; ! Copy of the processor status longword.
682 0670 2
683 0671 2 MAP
684 0672 2 entry : REF $bblock,
685 0673 2 value : REF $bblock;
686 0674 2
687 0675 2 CLISGET VALUE (.entry, run$a_input_desc); ! First we must get the time string and
688 0676 3 IF (run$l_status = LIB$CVT_DTIME (run$a_input_desc, delta)) ! convert the .ASCII string to a delta time.
689 0677 2 THEN ! If conversion was ok then condense the del
690 0678 2 BEGIN ! time to a single longword value.
691 0679 2 psl = EDIV (%REF(-200000), delta [0], delta [0], delta [1]); ! Convert the delta time to a CPU time limit
692 0680 2
693 0681 2 IF .psl [psl$v_v] ! Check for overflow.
694 0682 2 THEN SIGNAL_STOP (run$_illval, 1, run$a_input_desc); ! CPU time limit value was too big.
695 0683 2
696 0684 2 IF .delta [1] NEQ 0 ! No overflow...should we round the
697 0685 2 THEN delta [1] = 1; ! CPU time limit?
698 0686 2
699 0687 2 .value = (.delta [0] * 2) + .delta [1]; ! Set the CPU time limit.
700 0688 2 END
701 0689 2 ELSE
702 0690 2 SIGNAL_STOP (run$_syntax, 1, run$a_input_desc, .run$l_status);
703 0691 2

```

: 704 0692 1 END; ! of ROUTINE get_cpulm

000C 00000 GET_CPULM:

| | | | | | | | | | | | |
|----|----|-----------|----|-----------|------|-------|-------|--------|---------------------------------|---|------|
| | | | 53 | 00000000G | 00 | 9E | 00002 | .WORD | Save R2,R3 | : | 0636 |
| | | | 52 | 0000' | CF | 9E | 00009 | MOVAB | LIB\$STOP, R3 | : | |
| | | | 5E | | 08 | C2 | 0000E | MOVAB | RUN\$A_INPUT_DESC, R2 | : | |
| | | | | | 52 | DD | 00011 | SUBL2 | #8, SP | : | |
| | | | | 04 | AC | DD | 00013 | PUSHL | R2 | : | 0675 |
| | | 00000000G | 00 | | 02 | FB | 00016 | PUSHL | ENTRY | : | |
| | | | | 4004 | 8F | BB | 0001D | CALLS | #2, CLISGET_VALUE | : | |
| | | 00000000G | 00 | | 02 | FB | 00021 | PUSHR | #^M<R2,SP> | : | 0676 |
| | | | 28 | A2 | 50 | D0 | 00028 | CALLS | #2, LIB\$CVT_DTIME | : | |
| | | | | 30 | 50 | E9 | 0002C | MOVL | R0, RUN\$SL_STATUS | : | |
| 04 | AE | | 6E | FFFCF2C0 | 8F | 7B | 0002F | BLBC | R0, 3\$ | : | |
| | | | | | 50 | DC | 00039 | EDIV | #-200000, DELTA, DELTA, DELTA+4 | : | 0679 |
| | | | OD | | 50 | 01 | E1 | MOVPSL | R0 | : | |
| | | | | | 01 | E1 | 0003B | BBC | #1, PSL, 1\$ | : | 0681 |
| | | | | | 52 | DD | 0003F | PUSHL | R2 | : | 0682 |
| | | | | | 01 | DD | 00041 | PUSHL | #1 | : | |
| | | | 63 | 00000000G | 8F | DD | 00043 | PUSHL | #RUN\$ ILLVAL | : | |
| | | | | 04 | 03 | FB | 00049 | CALLS | #3, LIB\$STOP | : | |
| | | | | | AE | D5 | 0004C | CALLS | #3, LIB\$STOP | : | |
| | | | | | 04 | 13 | 0004F | TSTL | DELTA+4 | : | 0684 |
| | | | | | 04 | 13 | 0004F | BEQL | 2\$ | : | |
| | 04 | AE | | | 01 | D0 | 00051 | MOVL | #1, DELTA+4 | : | 0685 |
| | | | | | 6E | D0 | 00055 | MOVL | DELTA, R0 | : | 0687 |
| | 08 | BC | | 04 | BE40 | 3E | 00058 | MOVAV | @DELTA+4[R0], @VALUE | : | |
| | | | | | 04 | 0005E | | RET | | : | 0676 |
| | | | | 28 | A2 | DD | 0005F | PUSHL | RUN\$SL_STATUS | : | 0690 |
| | | | | | 52 | DD | 00062 | PUSHL | R2 | : | |
| | | | | | 01 | DD | 00064 | PUSHL | #1 | : | |
| | | | | | 8F | DD | 00066 | PUSHL | #12587260 | : | |
| | | | 63 | 00C010FC | 04 | FB | 0006C | CALLS | #4, LIB\$STOP | : | |
| | | | | | 04 | 0006F | | RET | | : | 0692 |

: Routine Size: 112 bytes, Routine Base: \$CODE\$ + 038D

: 705 0693 1


```

: 707 0694 1 %SBTTL 'get_value -- Obtain and convert a command line entity'
: 708 0695 1 ROUTINE get_value (entry, ret_val) : NOVALUE =
: 709 0696 1 ++
: 710 0697 1
: 711 0698 1 Functional Description:
: 712 0699 1
: 713 0700 1 This routine will obtain a command line entity and convert
: 714 0701 1 the text representation of the value into a numeric quantity.
: 715 0702 1
: 716 0703 1 Inputs:
: 717 0704 1
: 718 0705 1 entry adr The address of the current command line quota entity
: 719 0706 1 in the quota table.
: 720 0707 1
: 721 0708 1 Outputs:
: 722 0709 1
: 723 0710 1 ret_val adr The address of a longword to receive the converted value
: 724 0711 1 of the command line entity.
: 725 0712 1
: 726 0713 1 Side Effects:
: 727 0714 1
: 728 0715 1 Errors encountered during conversion will be signalled.
: 729 0716 1
: 730 0717 1 --
: 731 0718 2 BEGIN
: 732 0719 2
: 733 0720 2 LOCAL
: 734 0721 2 value_desc : $bblock [dsc$c_s_bln], ! Descriptor for conversion to a numeric val
: 735 0722 2 value; ! Resultant value from conversion.
: 736 0723 2
: 737 0724 2 MAP
: 738 0725 2 entry : REF $bblock,
: 739 0726 2 ret_val : REF $bblock;
: 740 0727 2
: 741 0728 2 value_desc [dsc$w_length] = 0; ! Length is left upto the conversion routine
: 742 0729 2 value_desc [dsc$b_dtype] = dsc$k_dtype_lu; ! Type is unsigned longword.
: 743 0730 2 value_desc [dsc$b_class] = dsc$k_class_s; ! Scalar.
: 744 0731 2 value_desc [dsc$a_pointer] = value; ! Address to store result of conversion.
: 745 0732 2
: 746 0733 2 IF CLISGET_VALUE (.entry, run$a_input_desc) ! If the entity is present in the command
: 747 0734 2 THEN ! line and has a value associated with it,
: 748 0735 3 IF NOT (run$l_status = ! perform the conversion according to the in
: 749 0736 3 LIB$CVT_DX_DX (run$a_input_desc, value_desc) ! supplied by the descriptors.
: 750 0737 2 THEN SIGNAL_STOP (run$cvtterr, 2, .entry, run$a_input_desc, .run$l_status)
: 751 0738 2 ELSE .ret_val = .value;
: 752 0739 2
: 753 0740 1 END; ! of ROUTINE get_value
: INFO#250 LI:0738
: Referenced LOCAL symbol VALUE is probably not initialized

```

0004 0000 GET_VALUE:

52 0000' CF 9E 00002 .WORD Save R2
MOVAB RUN\$a_INPUT_DESC, R2

: 0695
:

| | | | | | | | | |
|-----------|----|-----------|----|-----------|-----------|--------|-----------------------|------|
| 04 | SE | | 0C | C2 | 00007 | SUBL2 | #12, SP | |
| 08 | AE | 01040000 | 8F | D0 | 0000A | MOVL | #17039360, VALUE_DESC | 0728 |
| | | | 6E | 9E | 00012 | MOVAB | VALUE, VALUE_DESC+4 | 0731 |
| | | | 52 | DD | 00016 | PUSHL | R2 | 0733 |
| | | 04 | AC | DD | 00018 | PUSHL | ENTRY | |
| 00000000G | 00 | | 02 | FB | 0001B | CALLS | #2, CLISGET_VALUE | |
| | 2F | | 50 | E9 | 00022 | BLBC | R0, 28 | |
| | | 04 | AE | 9F | 00025 | PUSHAB | VALUE_DESC | 0736 |
| | | | 52 | DD | 00028 | PUSHL | R2 | |
| 00000000G | 00 | | 02 | FB | 0002A | CALLS | #2, LIBSCVT_DX_DX | |
| | 28 | | 50 | D0 | 00031 | MOVL | R0, RUNSL_STATUS | |
| | | | 50 | E8 | 00035 | BLBS | R0, 18 | |
| | | 28 | A2 | DD | 00038 | PUSHL | RUNSL_STATUS | 0737 |
| | | | 52 | DD | 0003B | PUSHL | R2 | |
| | | 04 | AC | DD | 0003D | PUSHL | ENTRY | |
| | | | 02 | DD | 00040 | PUSHL | #2 | |
| 00000000G | 00 | 00000000G | 8F | DD | 00042 | PUSHL | #RUNS CVTERR | |
| | | | 05 | FB | 00048 | CALLS | #5, LIBSSTOP | |
| | | | | 04 | 0004F | RET | | |
| | 08 | BC | 6E | D0 | 00050 18: | MOVL | VALUE, @RET_VAL | 0738 |
| | | | 04 | 00054 28: | RET | | | 0740 |

: Routine Size: 85 bytes. Routine Base: \$CODE\$ + 03FD

: 754 0741 1

```

: 756 0742 1 %SBTTL 'insert_quota -- Insert a quota into the quota list'
: 757 0743 1 ROUTINE insert_quota (name, value) : NOVALUE =
: 758 0744 1 :++
: 759 0745 1 :
: 760 0746 1 : Functional Description:
: 761 0747 1 :
: 762 0748 1 : This routine is responsible for entering the specified
: 763 0749 1 : quota list entry into the quota list.
: 764 0750 1 :
: 765 0751 1 : Inputs:
: 766 0752 1 :
: 767 0753 1 : name val The name of the quota (pql$_xxx)
: 768 0754 1 :
: 769 0755 1 : value val The value of the quota.
: 770 0756 1 :
: 771 0757 1 : Implicit Outputs:
: 772 0758 1 :
: 773 0759 1 : The quota entry has been added to the quota list and
: 774 0760 1 : the list pointer has been updated to point to the next
: 775 0761 1 : entry slot.
: 776 0762 1 :
: 777 0763 1 : Side Effects:
: 778 0764 1 :
: 779 0765 1 : If this is the first call, we will allocate a heap of
: 780 0766 1 : memory to store the quota list in. This will be
: 781 0767 1 : deallocated at image rundown.
: 782 0768 1 :
: 783 0769 1 :--
: 784 0770 2 BEGIN
: 785 0771 2
: 786 0772 2 OWN
: 787 0773 2 quota_ptr : REF $bblock [4];
: 788 0774 2
: 789 0775 2 IF .run$a_quota EQL 0 : Has a quota list been allocated?
: 790 0776 2 THEN : No, allocate one...
: 791 0777 3 BEGIN : Call library routine to get virtual memory.
: 792 0778 4 IF NOT (run$l_status = : The number of bytes required for the quota list
: 793 0779 4 LIB$GET_VM (%REF(list_k_entry_size * pql$_length), : ! is computed as the size of a quota list entry
: 794 0780 4 run$a_quota)) : times the number of entries (including the termina
: 795 0781 3 THEN :
: 796 0782 3 SIGNAL_STOP (run$_insvirmem, 0, .run$l_status); : As usual, errors will be sent to the caller.
: 797 0783 3
: 798 0784 3 quota_ptr = .run$a_quota; : Point to the first entry in the list.
: 799 0785 2 END;
: 800 0786 2
: 801 0787 2 quota_ptr [list_b_name] = .name; : Set the quota name (pql$_xxxx).
: 802 0788 2 quota_ptr [list_l_value] = .value; : Set the initial value.
: 803 0789 2 quota_ptr = .quota_ptr + list_k_entry_size; : Point to the next quota list entry.
: 804 0790 2
: 805 0791 1 END; ! of ROUTINE insert_quota

```

.PSECT \$OWNS,NOEXE,2

001D2 .BLKB 2
001D4 QUOTA_PTR:

.BLKB 4

.PSECT \$CODE\$,NOWRT,2

| | | | | | | | |
|-----------|----|----------|------|-------|---------------|--------|------------------------|
| | | | 0004 | 0000 | INSERT_QUOTA: | | |
| | S2 | 0000' | CF | 9E | 00002 | .WORD | Save R2 |
| | SE | | 04 | C2 | 00007 | MOVAB | RUNSA_QUOTA, R2 |
| | | | 62 | D5 | 0000A | SUBL2 | #4, SP |
| | | | 2F | 12 | 0000C | TSTL | RUNSA_QUOTA |
| | | | 52 | DD | 0000E | BNEQ | 2\$ |
| 04 | AE | 4B | 8F | 9A | 00010 | PUSHL | R2 |
| | | 04 | AE | 9F | 00015 | MOVZBL | #75, 4(SP) |
| 00000000G | 00 | | 02 | FB | 00018 | PUSHAB | 4(SP) |
| 10 | A2 | | 50 | D0 | 0001F | CALLS | #2, LIB\$GET_VM |
| | 12 | | 50 | E8 | 00023 | MOVL | R0, RUNSL_STATUS |
| | | 10 | A2 | DD | 00026 | BLBS | R0, 1\$ |
| | | | 7E | D4 | 00029 | PUSHL | RUNSL_STATUS |
| | | 00C012F4 | 8F | DD | 0002B | CLRL | -(SP) |
| 00000000G | 00 | | 03 | FB | 00031 | PUSHL | #12587764 |
| 01A4 | C2 | | 62 | D0 | 00038 | CALLS | #3, LIB\$STOP |
| | 50 | 01A4 | C2 | D0 | 0003D | MOVL | RUNSA_QUOTA, QUOTA_PTR |
| | 60 | 04 | AC | 90 | 00042 | MOVL | QUOTA_PTR, R0 |
| | A0 | 08 | AC | D0 | 00046 | MOVB | NAME, -(R0) |
| 01 | A0 | | AC | D0 | 00046 | MOVL | VALUE, 1(R0) |
| 01A4 | C2 | | 05 | C0 | 0004B | ADDL2 | #5, QUOTA_PTR |
| | | | 04 | 00050 | RET | | |

: Routine Size: 81 bytes, Routine Base: \$CODE\$ + 0452

: 806 0792 1

```

808 0793 1 %SBTTL 'get_stsflgs -- Set up initial process state flags'
809 0794 1 ROUTINE get_stsflgs : NOVALUE =
810 0795 1 **
811 0796 1
812 0797 1 Functional Description:
813 0798 1
814 0799 1 This routine performs call backs to the CLI to obtain the initial
815 0800 1 process state flag settings.
816 0801 1
817 0802 1 Implicit Inputs:
818 0803 1
819 0804 1 run$l_stsflg adr The address of the status flags vector.
820 0805 1
821 0806 1 Implicit Outputs:
822 0807 1
823 0808 1 run$l_stsflg adr The various state flags have been set.
824 0809 1
825 0810 1 Side Effects:
826 0811 1
827 0812 1 Several state flags have been defaulted to false. They are:
828 0813 1 BATCH, INTER, DISAWS and NETWRK.
829 0814 1
830 0815 1 --
831 0816 2 BEGIN
832 0817 2
833 0818 2 run$l_stsflg [prc$V_ssrwait] = NOT (CLISPRESNT (resource_wait)); ! Resource wait mode.
834 0819 2 run$l_stsflg [prc$V_ssfexcu] = CLISPRESNT (service_fail); ! System service failure exception mode.
835 0820 2 run$l_stsflg [prc$V_pswapm] = NOT (CLISPRESNT (swapping)); ! Process swap mode.
836 0821 2 run$l_stsflg [prc$V_noacct] = NOT (CLISPRESNT (accounting)); ! Process accounting.
837 0822 2 run$l_stsflg [prc$V_batch] = false; ! Not a batch process.
838 0823 2 run$l_stsflg [prc$V_inter] = false; ! Not an interactive process.
839 0824 2 run$l_stsflg [prc$V_hiber] = (CLISPRESNT (delay) OR ! Process hibernation state can be set
840 0825 2 CLISPRESNT (interval) OR ! by any combination of the /DELAY,
841 0826 2 CLISPRESNT (schedule)); ! /INTERVAL or /SCHEDULE command qualifiers.
842 0827 2 run$l_stsflg [prc$V_login] = NOT (CLISPRESNT (authorize)); ! Process authorization if image is LOGINOUT
843 0828 2 run$l_stsflg [prc$V_netwrk] = false; ! Not a network connect object.
844 0829 2 run$l_stsflg [prc$V_disaws] = false; ! Automatic working set adjustment state.
845 0830 2 run$l_stsflg [prc$V_imgdmp] = CLISPRESNT (dump); ! Dump requested
846 0831 2 run$l_stsflg [prc$V_detach] = CLISPRESNT (detach); ! Detached process
847 0832 2
848 0833 1 END; ! of ROUTINE get_stsflgs

```

65

01

007C 0000 GET_STSFLGS:

| | | | | | | | | |
|----|-----------|----|----|-------|--------|---------------------------|--|------|
| | | | | | WORD | Save R2,R3,R4,R5,R6 | | 0794 |
| 56 | 0000' | CF | 9E | 00002 | MOVAB | RESOURCE WAIT, R6 | | |
| 55 | 0000' | CF | 9E | 00007 | MOVAB | RUN\$L_STSFLG, R5 | | |
| 54 | 00000000G | 00 | 9E | 0000C | MOVAB | CLISPRESNT, R4 | | |
| | | 56 | DD | 00013 | PUSHL | R6 | | 0818 |
| 64 | | 01 | FB | 00015 | CALLS | #1, CLISPRESNT | | |
| 51 | | 50 | D2 | 00018 | MCOML | R0, R1 | | |
| 00 | | 51 | F0 | 0001B | INSV | R1, #0, #1, RUN\$L_STSFLG | | |
| | 28 | A6 | 9F | 00020 | PUSHAB | SERVICE FAIL | | 0819 |
| 64 | | 01 | FB | 00023 | CALLS | #1, CLISPRESNT | | |

RUNDET
V04-000

Run Detached Process -- CLI Utility Procedure M 15
get_stsflgs -- Set up initial process state fla 14-Sep-1984 00:27:00
16-Sep-1984 12:08:54 VAX-11 Bliss-32 V4.0-742
[CLIUTL.SRC]RUNDET.B32;1

Page 36
(16)

| | | | | | | | | | |
|----|----|----|------|----|-------|-------|--------|----------------------------|------|
| 65 | 01 | 01 | | 50 | F0 | 00026 | INSV | R0, #1, #1, RUNSL_STSFLG | |
| | | | 38 | A6 | 9F | 0002B | PUSHAB | SWAPPING | 0820 |
| | | 64 | | 01 | FB | 0002E | CALLS | #1, CLISPRESNT | |
| | | 51 | | 50 | D2 | 00031 | MCOML | R0, R1 | |
| 65 | 01 | 02 | | 51 | F0 | 00034 | INSV | R1, #2, #1, RUNSL_STSFLG | |
| | | | FF14 | C6 | 9F | 00039 | PUSHAB | ACCOUNTING | 0821 |
| | | 64 | | 01 | FB | 0003D | CALLS | #1, CLISPRESNT | |
| | | 51 | | 50 | D2 | 00040 | MCOML | R0, R1 | |
| 65 | 01 | 03 | | 51 | F0 | 00043 | INSV | R1, #3, #1, RUNSL_STSFLG | |
| | | 65 | 0410 | 8F | AA | 00048 | BICW2 | #1040, RUNSL_STSFLG | 0823 |
| | | | FF38 | C6 | 9F | 0004D | PUSHAB | DELAY | 0824 |
| | | 64 | | 01 | FB | 00051 | CALLS | #1, CLISPRESNT | |
| | | 53 | | 50 | D0 | 00054 | MOVL | R0, R3 | |
| | | | 84 | A6 | 9F | 00057 | PUSHAB | INTERVAL | 0825 |
| | | 64 | | 01 | FB | 0005A | CALLS | #1, CLISPRESNT | |
| | | 52 | | 50 | D0 | 0005D | MOVL | R0, R2 | |
| | | 52 | | 53 | CB | 00060 | BISL2 | R3, R2 | |
| | | | 10 | A6 | 9F | 00063 | PUSHAB | SCHEDULE | 0826 |
| | | 64 | | 01 | FB | 00066 | CALLS | #1, CLISPRESNT | |
| | 53 | 50 | | 52 | 89 | 00069 | BISB3 | R2, R0, R3 | |
| 65 | 01 | 05 | | 53 | F0 | 0006D | INSV | R3, #5, #1, RUNSL_STSFLG | |
| | | | FF28 | C6 | 9F | 00072 | PUSHAB | AUTHORIZE | 0827 |
| | | 64 | | 01 | FB | 00076 | CALLS | #1, CLISPRESNT | |
| | | 51 | | 50 | D2 | 00079 | MCOML | R0, R1 | |
| 65 | 01 | 06 | | 51 | F0 | 0007C | INSV | R1, #6, #1, RUNSL_STSFLG | |
| | | 65 | 0180 | 8F | AA | 00081 | BICW2 | #384, RUNSL_STSFLG | 0829 |
| | | | FF54 | C6 | 9F | 00086 | PUSHAB | DUMP | 0830 |
| | | 64 | | 01 | FB | 0008A | CALLS | #1, CLISPRESNT | |
| 01 | A5 | 01 | | 50 | F0 | 0008D | INSV | R0, #3, #1, RUNSL_STSFLG+1 | |
| | | | FF48 | C6 | 9F | 00093 | PUSHAB | DETACH | 0831 |
| | | 64 | | 01 | FB | 00097 | CALLS | #1, CLISPRESNT | |
| 01 | A5 | 01 | | 50 | F0 | 0009A | INSV | R0, #1, #1, RUNSL_STSFLG+1 | |
| | | | | 04 | 000A0 | | RET | | 0833 |

: Routine Size: 161 bytes, Routine Base: \$CODE\$ + 04A3

: 849 0834 1

```

: 851 0835 1 %SBTTL 'schedule_process -- Schedule the process for execution'
: 852 0836 1 ROUTINE schedule_process : NOVALUE =
: 853 0837 1 **
: 854 0838 1
: 855 0839 1 Functional Description:
: 856 0840 1
: 857 0841 1 This routine is responsible for scheduling wake up request(s) for the
: 858 0842 1 created process.
: 859 0843 1
: 860 0844 1 Implicit Inputs:
: 861 0845 1
: 862 0846 1 run$l_pid adr Address of the created process' PID.
: 863 0847 1 run$q_daytim adr Address of the wake up time quadword.
: 864 0848 1 run$q_interval adr Address of the repeat time quadword.
: 865 0849 1
: 866 0850 1 Implicit Outputs:
: 867 0851 1
: 868 0852 1 The wake up request has been scheduled.
: 869 0853 1
: 870 0854 1 Side Effects:
: 871 0855 1
: 872 0856 1 If any error is encountered while attempting to schedule the wake up
: 873 0857 1 request(s), we will signal a warning diagnostic to the user and exit.
: 874 0858 1
: 875 0859 1 --
: 876 0860 2 BEGIN
: 877 0861 2
: 878 P 0862 3 IF NOT (run$l_status = $SCHDWK (PIDADR = run$l_pid, : Schedule a wake up for the created process
: 879 P 0863 3 DAYTIM = run$q_daytim, : At the specified time of day.
: 880 0864 3 REPTIM = run$q_interval)) : Repeat the scheduled wake up at this inter
: 881 0865 2 THEN
: 882 0866 2 SIGNAL (run$_schdwk, 1, .run$l_pid, .run$l_status); : Report any problems scheduling the wake up
: 883 0867 2
: 884 0868 1 END; ! of ROUTINE schedule_process
    
```

.EXTRN SYSSCHDWK

```

0004 0000 SCHEDULE_PROCESS:
: 0836
: 0864
: 0866
: 0868
    
```

| | | | | | | | | |
|-----------|-------|-----------|----|-------|--------|------------------|---|------|
| 52 | 0000' | CF | 9E | 00002 | WORD | Save R2 | : | 0836 |
| | 18 | A2 | 9F | 00007 | MOVAB | RUNSL_PID, R2 | : | |
| | 10 | A2 | 9F | 0000A | PUSHAB | RUNSQ_INTERVAL | : | 0864 |
| | | 7E | D4 | 0000D | PUSHAB | RUNSQ_DAYTIM | : | |
| | | | | | CLRL | -(SP) | : | |
| | | 52 | DD | 0000F | PUSHL | R2 | : | |
| 00000000G | 00 | 04 | FB | 00011 | CALLS | #4, SYSSCHDWK | : | |
| | 04 | A2 | 50 | D0 | MOVL | R0, RUNSL_STATUS | : | |
| | | 12 | 50 | E8 | BLBS | R0, 1\$ | : | |
| | | 7E | 62 | 7D | MOVQ | RUNSL_PID, -(SP) | : | 0866 |
| | | | 01 | DD | PUSHL | #1 | : | |
| | | 00000000G | 8F | DD | PUSHL | #RUNS_SCHDWK | : | |
| 00000000G | 00 | 04 | FB | 0002A | CALLS | #4, LIBSIGNAL | : | |
| | | | 04 | 00031 | RET | | : | 0868 |

: Routine Size: 50 bytes, Routine Base: \$CODE\$ + 0544

RUNDET
V04-000

Run Detached Process -- CLI Utility Procedure J 15
schedule_process -- Schedule the process for ex 14-Sep-1984 12:08:54

VAX-11 Bliss-32 V4.0-742
[CLIUTL.SRC]RUNDET.B32;1

Page 38
(17)

: 885 0869 1
: 886 0870 1 END ! of MODULE rundet
: 887 0871 0 ELUDOM

.EXTRN LIB\$SIGNAL, LIB\$STOP

PSECT SUMMARY

| Name | Bytes | Attributes |
|--------------|-------|---|
| SPLITS | 720 | NOVEC,NOWRT, RD ,NOEXE,NOSHR, LCL, REL, CON,NOPIC,ALIGN(2) |
| SOWNS | 472 | NOVEC, WRT, RD ,NOEXE,NOSHR, LCL, REL, CON,NOPIC,ALIGN(2) |
| _LIB\$KEYOS | 0 | NOVEC,NOWRT, RD , EXE, SHR, LCL, REL, CON, PIC,ALIGN(1) |
| _LIB\$STATES | 10 | NOVEC,NOWRT, RD , EXE, SHR, LCL, REL, CON, PIC,ALIGN(1) |
| \$CODE\$ | 1398 | NOVEC,NOWRT, RD , EXE,NOSHR, LCL, REL, CON,NOPIC,ALIGN(2) |
| . ABS . | 0 | NOVEC,NOWRT,NORD ,NOEXE,NOSHR, LCL, ABS, CON,NOPIC,ALIGN(0) |

Library Statistics

| File | Total | Symbols Loaded | Percent | Pages Mapped | Processing Time |
|------------------------------------|-------|----------------|---------|--------------|-----------------|
| _\$255\$DUA28:[SYSLIB]LIB.L32:1 | 18619 | 69 | 0 | 1000 | 00:01.8 |
| _\$255\$DUA28:[SYSLIB]TPAMAC.L32:1 | 42 | 19 | 45 | 14 | 00:00.2 |

: Information: 1
: Warnings: 0
: Errors: 0

COMMAND QUALIFIERS

: BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LISS:RUNDET/OBJ=OBJ\$:RUNDET MSRC\$:RUNDET/UPDATE=(ENH\$:RUNDET)

: Size: 1398 code + 1202 data bytes
: Run Time: 00:26.6
: Elapsed Time: 01:36.1
: Lines/CPU Min: 1966
: Lexemes/CPU-Min: 20041
: Memory Used: 169 pages
: Compilation Complete

