THE PROGRAM OF THE WIRED STORE ON DASK

J. Jensen, P. Mondrup, P. Naur

Contents

Regnecentralen, Borups Alle 45, Copenhagen N.

Introduction.

The present report contains the complete set of programs wired into DASK. With a few exceptions these programs were designed to be used as subroutines for running DASK ALGOL programs. For an explanation of the conventions of the programs reference should therefore be made to A MANUAL OF THE DASK ALGOL LANGUAGE, 2nd edition. Indeed the section references of the present report all refer to the MANUAL. For further explanation reference may also be made to J. Jensen and P. Naur: An Implementation of ALGOL 60 Procedures, BIT vol. 1 no. 1 pag. 38 (1961), and J. Jensen, P. Mondrup, P. Naur: A Storage Allocation Scheme for ALGOL 60, BIT vol. 1 no. 2 pag. 89 (1961).

The programs in the wired store were written by J. Jensen and P. Mondrup. They were checked and debugged by P. Naur.

Instructions and references.

The instructions are written serially starting with the contents of the wired store and followed by the first 64 instructions of the core store which are used by the wired programs. Following the address of each instruction the addresses from which reference is made to this instruction, if any, are listed. The instructions themselves are given as two digits, the operation, followed by the address, and possibly by the index mark, B, C, or D. In references and within instructions adresses referring to the wired store itself are marked in front with the letter s. This mark, which can only occur in instructions within the wired programs, is represented in the machine by means of an extra bit in each wired instruction.

The annotation of the wired programs.

Entries. It is convenient to distinguish between three different kinds of entries into the wired program:

1. Algol entries. These are used by running ALGOL programs in a manner explained in section 11 of A Manual of the DASK ALGOL Language.
2. Special entries. A few non - ALGOL programs are available. These are explained below.
3. Internal routines. These are subroutines which will normally only be called from the wired store itself. A means for calling them from the core store is, however, provided among the special entries. (see the program at s713).

Notes to single instructions. In order to facilitate the study of the programs notes to some of the orders have been added, generally as follows:

Conditional jumps, operations 11, 51, 12, 52, 33, 53, 73. The condition for the jump to be effective is written as note.

Subroutine jump, operation 16. The name and effect of the subroutine is given.

Execute, operation 37. The form of the order being executed is written.

Store, operations 08, 28, 18, 58, 06, 26. The value stored in the core location is written.

However, deviations from these rules will be found in many places.

The programs of the wired store.

Special routine: primitive binary input from 8-hole tape. This routine will read information from the tape and place it in locations B, B + 1, B + 2, ... Each half-word will be loaded from three rows from the tape. The first of these must be of one of the forms

, xxx.xxx, or ,00xxx.xxx,

The half-word will be assembled from the combinations in the three rows as follows:

1st row: bits 0 - 5
2nd - : - 6 - 12
3rd - : - 13 - 19

Exit from the wired store to location D + 2 will take place when, immediately following the three rows of one word, a row having one of the forms

,0 xxx.xxx, or , 0xxx.xxx,

is encountered.

Parity errors will cause exit to D + 1. A checksum of the input symbols is formed in location 1. Location 0 and IRC are used as working locations.

The program may be entered in three different ways:

1. B in IRB
   17 4 (= 4A17)
   Then D = IRD
2. B, an odd address in IRB
   17 1 (= 1A17)
   This makes D = IRB - 2 and sets two orders
   B - 1 = D + 1: 30 2,D
   B = D + 2: 17 1
3. 17 0
   Sets B:= 3, then as 2.

Note that on entries 2 and 3 will ignore exit combinations accurring at the beginning of the tape and will stop on parity errors.

```
 0          35 s   3·
 1          75 s2046,B
 2          40 s  18·
 3          08     1,D
 4          68     1
 5  s  17  68     0
 6          55 s  21
 7  s  13  79 s   0·
 8          51     1,D  Wrong parity
 9          0C s  13,C
10          12     2,D  Exit combination
11          26     0·
12          55 s2041,C
13          53 s   7·   Word not completed
14          28     0,B
15          26     1·
16          35 s   1,B
17          10 s   5·
18          30     2,D
19          17     1
```

```
20  s1179                   patch in læs and læst
          43    12
21        11 s1182          Number = 0
22        10 s1214
23 - 33                     Not used at present
34        74    4
35        37    1,D
```

Central alarm administration. This is entered with previous IRD in location 4.

```
36  s 46, s 48, s 50, s 52, s 54, s 56, s 58, s 60, s 62,
    s 64, s 66, s 68, s 70, s 72, s 74, s 76, s 78, s 80,
    s 82
          08    8
37        12 s  39          Overflow in AR
38        75 s  1,D
39  s 37  5C s  86          AR:= YR; YR:= 86
40        28    5           YR
41        1F    0           store cells 0 - 64
42        1C s  14
43        1D    0
44        10    0
45        74    4
46        16 s  36
47        74    4
48        16 s  36
49        74    4
50        16 s  36
51        74    4
52        16 s  36
53        74    4
54        16 s  36
55        74    4           reserved for start of ALGOL-translator
56        16 s  36
```

Stack alarm.

```
57  s 732                   block begin
    s 783                   Simple stack control
    s 796                   Complete stack control
    s 907                   Array too big
    s 940                     -    -    -
          74    4
58        16 s  36
```

Overflow.

```
59  s 299                   addition
    s 344                   dividend = 0
    s 483                   power or exp
    s 493                     -    -    -
    s2004                   ln, arg ≤ 0
    s2043                   exp
```

```
          74     4
60        16 s  36
61  s 449                (-)↑(not integer)
    s 524                0↑(-1)
    s 526                0↑(≤0)
          74     4
62        16 s  36
63  s1794                sqrt(-)
          74     4
64        16 s  36
65  s 578                entier(>2↑19)
          74     4
66        16 s  36
67  s 178                element outside array
    s 180                   -       -       -
    s 186                   -       -       -
    s 214                   -       -       -
    s 216                   -       -       -
          74     4
68        16 s  36
69  s 991                upper bound < lower
          74     4
70        16 s  36
71  s1148                input syntax error
    s1149                  -       -       -
    s1151                  -       -       -
    s1154                  -       -       -
    s1155                  -       -       -
    s1159                  -       -       -
    s1162                  -       -       -
    s1173                  -       -       -
    s1174                  -       -       -
    s1238                  -       -       -
          74     4
72        16 s  36
73  s1222                Input overflow
          74     4
74        16 s  36
75  s1243                Input of not used symbol
    s1268                Input parity wrong
          74     4
76        16 s  36
77  s1332                Wrong input check sum
    s1334                  -       -       -       -
          74     4
78        16 s  36
79  s 960                Drum array too big
          74     4
80        16 s  36
81  s 705                Wrong number of actual parameters
          74     4
82        16 s  36
```

Constants.

83		00 s 0
84	s1727	0C s 0
85	s 413, s 645, s 854, s1498,D, s1638	0A s 0
86	s1498,D	00 s 9
87	s1498,D	00 s 99
88	s1498,D	00 s 999
89	s1442, s1665	00 s2047
90	s 616, s 661	17 s2047
91	s 826	55 s2047
92	s1133,C, s1470	00 s1280
		10/16
93	s 274, s 429, s 476, s 501, s 605, s1135, s1230, s1457, s1528	
94	s1133,C, s1464	66 s1638
		0.8
95		66 s1638
96	s1118, s1477	4C s 204,B
		0.1 - epsilon
97		4C s1228,D
98	s1527, s1534	4C s 204,B
		0.1 + epsilon
99		4D s1228,D
100	s 487, s 824, s 829, s1177, s1785, s1787, s1904	01 s 0
101	s 243, s 257, s 264, s 324, s 330, s 340, s 361, s 368, s 405, s 556, s 568, s 573, s 577, s 594, s 656, s1229, s1415, s1419, s1765, s1780, s1788, s1793, s1827, s1832, s1890, s1908, s1994, s2023	6D s 0
102	s 371, s 856	02 s 0
103		6D s 0
104	s 238, s 458, s 516, s 624, s 745, s1029, s1057, s1075, s1175, s1340, s1942	00 s 0,C
105	s 300, s 494, s 938, s1055, s1402, s1493, s1517, s1628, s1635, s1654, s1701, s1782, s1925	00 s 1
106	s 231, s 235, s 933	00 s 64
107	s 159, s 650, s1015, s1046, s1065, s1205	38 s 0
108	s1136,C, s1279, s1282	00 s2044,C
109		55 s 0
110	s1136,C	00 s 3
111	s 335, s 386, s 393, s 446, s 480, s 492, s1422, s1804, s1848, s1868, s1915, s2001	00 s 0,B
112	s1737	1E s 0
113	s 348, s 421, s 454, s 472, s 498, s 503, s 892, s1110, s1112, s1146, s1157, s1484, s1521, s1543, s1808, s1852, s1901, s1921, s1940	00 s1024
114	s1150	01 s 1

115	s 290, s 382, s 403, s 425, s 992, s1026, s1163, s1318, s1333, s1349, s1368, s1387, s1455, s1564, s1630, s1639, s1642, s1743, s1748, s1758, s1761	01	s	0
116	s 222	1D	s	0
117	s 353, s1265, s1326, s1719, s1963	7F	s	0
118	s 283, s 312	00	s	52
119	s 587, s 589, s 873	00	s	0,C
120	s 395	14	s	0
121	s1623	3B	s	0
122	s 456	00	s	512
123	s1593	05	s	0
124	s1894	11	s	0,B
125	s1594	1B	s	0
126	s1014, s1045	38	s	1
127	s1641	10	s	0
128	s1661, s1966	00	s	11
129	s1714, s1729	0B	s	0
130	s2014	59	s	0
131	s1734	40	s	0
132	s2031	5A	s	0
133	s1316	2C	s	0
134	s1731	2C	s	0
135	s1747	3F	s	0
136	s 955, s1723, s1725	1C	s	0
137	s1716	3D	s	0
138	s 748, s 789, s1186,B	00	s	2
139	s 772, s 812, s1127, s1171, s1186,B, s1223, s1380	00	s	1
140	s 220	1F	s	0
141	s1199	0D	s	0
142		13	s	0
143	s 485, s2036	00	s	69
144	s 398, s 581, s 583	00	s	21
145	s1919	00	s	31
146	s1221	58	s	0

Exit from general expression as parameter of procedure statement or function designator (section 11.5.4.4.3).

```
147        55 s   0·
148        37     1,D  75 call address
149        10     1,D
```

Exit from subscripted variable as parameter of procedure statement or function designator (section 11.5.4.3.4). The location call address may contain 37 f (section 11.5.2) or 77 f (section 11.6.3.1.2).

```
150        55 s   0·
151        37     1,D  75 call address
152        08  2046,B  Array identifier
153        60     0,D  77 or 37 f
154        29    41
155        21    41
156        11 s 596    77 f in OD
157        50 s 161
```

Assign to subscripted variable (section 11.6.3.3.3).

```
158  s 703 08     6    Value
159        61 s 107·   38
160        35 s   2,B
161  s 157 00  2046,B  Array identifier
```

Fetch value of subscripted variable (section 11.5.3.3).

```
162        08     2    40/60/08/28 a0,C  55 a1 (section 11.7.2.3)
163        37     3    55 a1
164        69     3·
165        64     1,C
166        51 s 188·   Only drum subscripts
167        61     0,C
168        29     3·
169  s 174 2A     0,B  -
170        06     3·
171        55 s   1,C  | Sum core subscripts
172        35 s   2,B  |
173        64     1,C
174        11 s 169    - There are more core subscripts
175        0C s   1
176        51 s 185    There are drum subscripts
177        21     3
178        51 s  67    Element outside array
179  s 199
     s 203 64     3
180        51 s  67    Element outside array
181        37     3    55 (- a2 + Sum(c[q]×i[q]))
182        40     6    Value
183        37     2    40/60/08/28 a0,C
```

```
184          10    1,D
185  s 176   21    3
186          11 s  67·    Element outside array
187          37    2,C    55 k1
188  s 166   61    0,C
189          28    27·
190          64    1,C
191  s 196   2A    0,B    -
192          06    27·
193          55 s  1,C    | Sum drum subscripts
194          35 s  2,B    |
195          64    1,C
196          51 s 191     - There are more drum subscripts
197          26    27
198          63    27     Available track address - wanted track address
199          11 s 179     Drum subscript unchanged
200          74    42
201          16 s 204     Change core section of array
202          37    42     75 (cf. s 200)
203          10 s 179
```

Internal routine: Exchange core section of drum array.

```
204  s 201                Subscripted variable
     s1089                Læs
          74    41
205       16 s 220·       Transfer core section to drum
206       60    3,C
207       78    11·
208       60    2,C
209       78    10
210       61    27·
211       26    1,C
212       0D s  9
213       21    10
214       51 s  67        Element outside array
215       21    11
216       11 s  67        Element outside array
217       16 s 222        Transfer drum section to cores
218       37    41·       75 (cf. s 204)
219       10 s  1,D
```

Internal routine: Transfer core section to drum
Input: 1,c: 1C track address
2,c: SS first address
3,c: SS last address

```
220  s 205                Arrays
          60 s 140
221       10 s 223
```

Internal routine. Transfer drum section to cores. Input as above.

```
222  s 217              Arrays
     s 776              Blocks
          60 s 116
223  s 221 28    10-    1F or 1D
224        60     2,C
225        29    10-    1F or 1D first address
226        60     1,C
227        28    11     1C track address
228  s 236
     s 241 37    11     1C track address
229        37    10     1F or 1D
230        66    11     1C track address + 2×n
231        60 s 106
232        26    10-    Add 64
233        21     3,C
234        11 s   1,D   Transfer is finished
235        24 s 106
236        51 s 228     Transfer another complete track
237        4F s  29
238        6A s 104
239        0C s  29
240        26    10     1F or 1D overlapping core address
241        10 s 228
```

0 - v (section 11.5.7.3.1).

```
242        35 s2046,B
243        00 s 101-
244  s 253 18     0,B   Mantissa
245        01     0,B
246        01     0,B
247        12 s 250-    Mantissa = -1
248  s 251 08     0,B   - Mantissa
249        10 s 597
250  s 247 01 s2046
251        10 s 248
```

0 - a (sections 11.5.5.2.2, 11.5.5.2.3, 11.5.7.3.3).

```
252        40     0,B
253        10 s 244
```

s + a (section 11.5.7.3.5).

```
254        35 s   2,B
255        40  2046,B
256        10 s 269
```

a + v (section 11.5.7.3.4).

```
257     4             for real
          00 s 101
258       10 s 269
259  s 268 01 s2046
260       10 s 269
```

```
s - a (section 11.5.7.3.5).

261          35 s   2,B
262          40  2046,B
263          10 s 265

a - v (section 11.5.7.3.4).

264       8                  for real
             00 s 101
265   s 263  18  2046,B      Mantissa
266          01  2046,B
267          01  2046,B
268          12 s 259        Mantissa = -1
269   s 256
      s 258
      s 260  18  2046,B      x1
270          58    13       128 - x2
271          40     0,B
272          18     0,B      y1
273          58    15        128 - y2
274          64 s  93
275          61    15
276          26    13
277          51 s 305        x2 > y2
278          43     0,B
279          11 s 319        y1 = 0
280          60    13
281          0C s   8
282          29    45
283          21 s 118
284          11 s 302        y2 >> x2
285          40  2046,B
286          37    45        4D (y2 - x2)
287          00     0,B
288   s 317
      s 360  52 s 292        No overflow
289          18     0,B
290          61 s 115
291          10 s 298
292   s 288  4E    13
293          18     0,B
294          43     0,B
295          11 s 324        z1 = 0
296          60    13
297          0D s   8
298   s 291
      s 447  26    15
299          51 s  59        abs(z) > 2↑128
300          21 s 105
301          11 s 324        abs(z) < 2↑(-127)
302   s 284
      s 306  40    15
303          06     0,B
```

```
304          10     1,D
305  s 277   43  2046,B
306          11 s 302     x1 = 0
307          60    13
308          26    15
309          61    13
310          0C s   8
311          29    45
312          21 s 118
313          11 s 318     x2 >> y2
314          40     0,B
315          37    45     4D (x2 - y2)
316          00  2046,B
317          10 s 288
318  s 313   50 s 320
319  s 279   40    13
320  s 318   00    15
321          00  2046,B
322          08     0,B
323          10     1,D
```

Result:= 0.

```
324  s 295                addition
     s 301                addition
     s 495                power
     s 525                power
     s2042                exp
          40 s 101
325       08     0,B
326       10     1,D
```

sxa (section 11.5.7.3.5).

```
327       35 s   2,B
328       40  2046,B
329       10 s 331
```

axv (section 11.5.7.3.4).

```
330       00 s 101
331  s 329
     s 376                x/2
          18  2046,B
332       58    15
333       44  2046,B
334       60    15
335       21 s 111
336       10 s 354
```

s/a (section 11.5.7.3.5).

```
337       35 s   2,B
338       40  2046,B
339       10 s 341
```

a/v (section 11.5.7.3.4).

```
340          00 s 101
341  s 339   18  2046,B
342          58    15
343          43  2046,B
344          11 s  59      Dividend = 0
345          40  2046,B
346          0E    17
347          08  2046,B
348          60 s 113
349          0B  2046,B
350          61    17
351          0D s   8
352          21    15
353          20 s 117
354  s 336   28    15
355          40     0,B
356          18     0,B
357          01     0,B
358          06    15
359          4A     0,B
360          10 s 288
```

a > v (section 11.5.7.3.6).

```
361       5              for real
            41 s 101
```

AR:= stack[IRB] unconverted; IRB:= IRB + 2 (call by 362 A 57, section 11.5.8.2).

```
362          00     0,B
363          35 s   2,B
364          10     1,D
```

s/a (section 11.5.7.3.5).

```
365          35 s   2,B
366          40  2046,B
367          10 s 369
```

a/v (section 11.5.7.3.4).

```
368          00 s 101
369  s 367   18  2046,B
370          58    15
371          01 s 102
372          08    10
373          43    10
374          51 s 377      exponent ≠ 2 (as integer)
375          40     0,B
376          10 s 331
377  s 374   40     0,B
378          58    13
```

```
379          01    13
380          0E    17
381          11 s 383     y1 ≠ -1
382          00 s 115·    Avoid y1 = -1
383  s 381   08     0,B
384          60    17
385          0D s   8
386          21 s 111
387          26    13·
388          43     0,B
389          11 s 523·    y = 0
390          43  2046,B
391          11 s 487     y ≠ 0 ∧ x = 0
392          61    15
393          20 s 111
394          51 s 448     abs(x) < 1, i.e. x is not integral
395          21 s 120
396          11 s 448     abs(x) > 2↑19, i.e. not integral
397          0C s   8
398          22 s 144
399          29    23·
400          40  2046,B
401          37    23     0C (cf. s 399)
402          51 s 448     x is not integral
403          01 s 115
404          11 s 448     x is not integral
405          61 s 101
406          20    15
407          0C s   8
408          29    45·
409          40  2046,B
410          37    45     4D (cf. s 408)
411          08    10
412          62    10
413          21 s  85
414          51 s 419     x is integral ∧ abs(x) ≤ 9 ∧ x ≠ 0
415          0C s  19
416          11 s 450·    x is even ∧ abs(x) > 9 ∧ x ≠ 0
417          40     0,B
418          10 s 450
419  s 414   60    10
420          11 s 428     x > 0
421          60 s 113·    -
422          0B     0,B    |
423          07 s   0·
424          08     0,B    | y:= 1/y
425          61 s 115
426          21    13      |
427          28    13     -
428  s 420   62    10
429          24 s  93
430          0C s   8
431          29    39
432          37    39     55 abs(x)
433          2A    13
434          0D s   1
```

```
435          08    15·
436          44     0,B
437          10 s 440·
438  s 441  0A     0,B  -
439          08     0,B  | Multiplication cycle
440  s 437  55 s2047,C  -
441          53 s 438
442          0E    17·
443          18     0,B
444          60    17
445          0D s   8
446          20 s 111
447          10 s 298
```

x is not integral.

```
448  s 394
     s 396
     s 402
     s 404  40     0,B
449          51 s  61    y < 0
450  s 416
     s 418  29     12    log is calculated as in Lance: Num. Methods,
451          74     42   pag. 39, using, however only terms up to
452          75 s 477·   r = 3. This gives max. error = 0.26₁₀-9.
453          43     0,B
454  s2005                entry from ln.
             24 s 113
455          0A s 544
456          20 s 122
457          08    10
458          20 s 104·
459          02     0,B
460          0B    10
461          07 s   0
462          08    10    w = (y - sqrt(0.5))/(y + sqrt(0.5))/(sqrt(2) - 1)↓2
463          0A    10·
464          08     0,B  w↓2
465          55 s   6·
466          44 s 528,C
467  s 470  55 s2046,C
468          0A     0,B
469          04 s 528,C
470          53 s 467    Product sum not finished
471          0A    10
472          24 s 113
473          60    13
474          0D s  20
475          4E    17
476          24 s  93·
477  s 452  10 s   1,D  Return to wired store, s 478 or s 2006
478          37    42    75 (cf. s 451)
479          60    15
480          21 s 111
```

```
481          0C s   8
482          20    17
483          51 s  59    Overflow in power or exp
484          29    45
485          21 s 143
486          51 s 490    z ≠ 1

Result := 1.

487  s 391               Power
     s2037               exp
             40 s 100
488          08     0,B  Result:= 1
489          10     1,D
490  s 486   4A  2046,B
491  s2040               exp
             37    45    4D. ARMR:= -argument×2↑(-39)
492          00 s 111
493          51 s  59    Overflow in power or exp
494          01 s 105
495          11 s 324    Result = 0
496          58    15    Characteristic
497          07 s   0    -
498          25 s 113
499          08    10     | 2↑x is calculated as exp
500          0A    10
501          24 s  93     | in Lance: Numerical Methods
502          0A s 536
503          20 s 113     | pag. 32 ff
504          08     0,B
505          0A s 538     |
506          04 s 540
507          0A    10     |
508          08    10
509          06     0,B   |
510          01    10
511          05    10     |
512          0A s 542
513          0B     0,B   -
514          60    12
515          11 s 518    Result ≥ 0
516          2A s 104
517          10 s 519
518  s 515   07 s   0
519  s 517   18     0,B  Mantissa
520          40    15
521          06     0,B  Final result of power or exp
522          10     1,D
523  s 389           -   0↑x
             41  2046,B
524          12 s  61    x = -1
525          51 s 324    x > 0
526          10 s  61    x ≤ 0
527          13 s   0
```

```
528  s 466,C          log coefficients
     s 469,C
          - 0.495 054 671 002     A1
530       - 0.004 857 703 444     A3
532       - 0.000 085 724 849     A5
534       - 0.000 001 900 516     A7
536  s 502                        2/x coefficients
          + 0.024 016 585 21
538  s 505
          + 0.001 385 511 01
540  s 506
          + 0.173 286 794 9
542  s 512                        sqrt(0.5)
          + 0.707 106 781 18
544  s 455                        - (sqrt(2) - 1)/2
     s1939
          - 0.207 106 781 187
```

ARi:= stacki[IRB] + round(AR); IRB:= IRB + 2 (section 11.5.7.4.1).

```
546       55 s 602
547       10 s 570
```

ARi:= stacki[IRB] - round(AR); IRB:= IRB + 2 (section 11.5.7.4.1).

```
548       55 s 601
549       10 s 570
```

ARi:= stacki[IRB]×round(AR); IRB:= IRB + 2 (section 11.5.7.4.1).

```
550       55 s 606
551       10 s 570
```

IRB:= IRB - 2; stacki[IRB]:= round(AR) (sections 11.5.7.3.1, 11.5.7.4.1).

```
552       55 s 594
553       10 s 570
```

hel i ARi:= round(stack[IRB]); IRB:= IRB + 2 (section 11.5.7.4.1).

```
554       40    0,B
555       35 s  2,B
556       01 s 101
```

ARi:= round(AR) (sections 11.5.7.3.2, 11.5.7.4.1).

```
557       55 s 604
558       10 s 570
```

ARi:= -round(AR) (sections 11.5.7.3.2, 11.5.7.4.1).

```
559       55 s 562
560       10 s 570
```

```
ARi:= -ARi (section 11.5.7.4.2).

 561          28  2046,B
 562  s 559   61  2046,B
 563          10     1,D

Internal round-off ARi:= round(AR).

 564  s 676              Procedure entry
      s 859              Array declaration
      s1753              trykml, skrvml, tryktom
           55 s    1,D
 565       10 s  570

stacki[IRB]:= round(stack[IRB]) (sections 11.5.3.1, 11.5.7.4.1).

 566       40     0,B
 567       35 s   2,B
 568       01 s  101

IRB:= IRB - 2; stack[IRB]:= index(AR) (section 11.5.3.1).

 569       55 s  596

Central round-off program.

 570  s 547
      s 549
      s 551
      s 553
      s 558
      s 560
      s 565
      s1777              entier
           08  2046,B
 571       63  2047,B
 572       11 s  591     Number has integer form already
 573       40 s  101
 574       06  2046,B   Number on stack packed form
 575       58    13      128 - exponent
 576       60    13
 577       21 s  101
 578       51 s   65     Overflow, exponent > 19
 579       0C s    8
 580       29    45      4D (19-exponent)
 581       21 s  144
 582       51 s  585     exponent > -2
 583       60 s  144
 584       29    45      21 shifts
 585  s 582 40  2046,B
 586       37    45      4D
 587       00 s  119
 588       73 s  590     Not entier
 589       01 s  119     Remove round-off
 590  s 588 08  2046,B
 591  s 572 64  2046,B
 592       73 s    0,C   Not entier
 593       10 s  657
```

```
0 + v (sections 11.5.7.3.1, 11.5.7.4.2, 11.6.5.2).

 594  s 552 00 s 101

IRB:= IRB - 2; stack[IRB]:= AR unconverted (sections 11.5.3.1, 11.5.8.2,
11.6.3.1.3.3).

 595        08  2046,B

Internal stack control.

 596  s 156                 Subscripted variable as actual parameter
      s 569 35 s2046,B
 597  s 249                 0 - v and 0 - a
           37    29         55 (2048 - arrow),B
 598       53     1,D       IRB ≠ arrow
 599       10 s 781

ARi:= stacki[IRB] - ARi; IRB:= IRB + 2 (section 11.5.7.4.2).

 600       28  2046,B
 601  s 548 61  2046,B

ARi:= stacki[IRB] + AR; IRB:= IRB + 2 (section 11.5.7.4.2).

 602  s 546 24     0,B
 603       35 s    2,B
 604  s 557 10     1,D

ARi:= stacki[IRB]×ARi; IRB:= IRB + 2 (section 11.5.7.4.2).

 605       24 s   93
 606  s 550 2A     0,B
 607       0C s   19
 608       35 s    2,B
 609       10     1,D

Special entry: execute instruction in wired store.

 610       37 s    0,C
 611       10     1,D

Special entry: fetch word in wired store.

 612       40 s    0,C
 613       10     1,D

Internal routine: Take name of first parameter.

 614  s1198                 læs, læst
      s1694                 tryktekst
           60 s 653
 615       28    33    00 24
```

Internal routine: Take name or value of first parameter.
This important routine is used to find the name or value of the first or only actual parameter of procedure calls.
Input parameters:
Contents of location 33: name: 00 24
value: 10 27
IRC: address of actual parameter.
Output:
IRB:= IRB - 4
stack[IRB + 2] = 17 return address
stack[IRB + 3] = 55 (address of following actual parameter - 1)
If the parameter is called by name then the routine will produce the results given in the following table:

Actual parameter	Store[IRC]	Name in AR (pair of instructions)
Simple variable, real or string	40 r(,c)	40 r 08 r
Simple variable, boolean or integer	60 i(,c)	60 i 28 i
Array identifier	44 a(,c)	44 a 0C a
Switch or procedure identifier	16 p(,c)	16 p 17 81
Standard procedure identifier	17 p(,c)	17 p 17 81
Subscripted variable	s: 74 w . . . . 17 150 w: 75 (,c)	16 s 17 158
Compound expression	e: 74 w . . . . 17 147 w: 75 (,c)	16 s 17 81
Formal variable	37 f(,c)	The contents of location f (always one of the above combinations)

If the parameter is called by value the routine will return with the value in AR.

```
616  s 833        switch designator
     s 858        array declaration
     s1357        streng
     s1374        trykkopi
     s1406        tryk
     s1752        trykml, tryktom
     s1764        abs
     s1775        entier
```

```
     s1779                  sign
     s1792                  sqrt
     s1831                  sin
     s1889                  cos
     s1907                  arctan
     s1993                  ln
     s2021                  exp
          40 s  90
617  s 721                  Special entry
          35 s2044,B
618       08    2,B
619       75 s   1,D
620       74    2,B        Return address
621       16 s 786         Stack control
622  s 712                  Take name or value of following parameter
          60    0,C        16/17/37/40/44/60/74 expr/74 subscr q(,c)
623       11 s 625         No C-index mark
624       20 s 104
625  s 623 54    3,B
626       28   27
627       29   47
628       21   47
629       51 s 642         Operation ≠ 74
630       61 s 654
631       26   27          Operation 16
632       29    3,B        Address of following actual parameter -1
633       54   27          16 address of actual parameter
634       37   33          Name: 00 24, Value: 10 27
635       08   33          Set value: 10 27
636       37    3,B        55 (address of following actual parameter -1)
637       60 2047,C
638       00 s 702
639       11 s 651         Parameter represents subscripted variable
640  s 646 40 s 705
641       10 s 651
642  s 629                  Operation ≠ 74
          37   33          Name: 00 24, Value: 10 27
643       08   33          Set value: 10 27
644       11 s 649         Operation 60, 44, 40
645       20 s  85
646       51 s 640         Operation 16, 17
647       26   27
648       10   27
649  s 644 40   27
650       01 s 107
651  s 639
     s 641 20   27
652       10   28
653  s 614
     s 709
     s 718 00   24
654  s 630 5E s   0
```

Exit from standard functions.

```
655     21              exp
      s1773             abs
      s1790             sign
      s1826             sqrt
      s1829             sqrt
      s1838             sin
      s1870             sin, cos
      s1872             sin, cos
      s1965             arctan
      s1967             arctan
      s2016             ln
            40    0,B
656         01 s 101
657   s 593             Central round-off program
      s1371             streng
      s1762             trykml, tryktom
            35 s  4,B
658         37  2047,B  55 (address of following parameter - 1)
659         10    32    Return through the stack
```

Procedure entry (section 11.7.4.1)
Input parameters:
IRC = address of procedure call instruction (i.e. actual parameters are in IRC + 1, IRC + 2, etc.)
IRD - 1: if there are parameters 08 formal place
else 55,d
IRD + 1, IRD + 2, ...: value and type codes. During the action of procedure entry the stack is used as follows:
IRB:= IRB - 6
stack[IRB] = 08 formal place to be filled
stack[IRB + 1] = 55 address of current value and type code
stack[IRB + 2] = 17 676 (value integer) else 677
stack[IRB + 3] = 55 (address of following parameter - 1)
stack[IRB + 4] = current value and type code
When the entry has been completed:
IRB:= IRB + 2

```
660         35 s2042,B
661         40 s  90
662         08     2,B  17 - , 55 -
663         54     3,B  55 (address of actual parameter - 1)
664         08     1,B
665         55 s   0,D
666         16 s 786    stack control
667         60  2047,C
668         11 s 671    There are formal parameters
669         35 s   2,B
670         10    32
671   s 668 28     0,B  08 formal place
```

```
672  s 688 55 s   1,C
673        54     1,B  Address of value and type code
674        60     0,C
675        10 s 680
676  s 684 16 s 564    AR:= round(AR)
677  s 694 37     0,B  08 formal place to be filled
678        66     0,B
679        60     4,B
680  s 675 0C s   2
681        28     4,B  current value and type code
682        75 s 704
683        12 s 694    name ∨ value real
684        55 s 676
685        51 s 695    value integer
686        37     1,B  55 address of current value and type code
687        20     4,B
688        51 s 672    There is another value and type code
689        74     2,B  Exit:= alarm
690        16 s 706    Take next parameter, alarm if there are any
691  s 792 37  2045,B  55 (return address - 1)
692        35 s2046,B
693        10    32
694  s 683 55 s 677
695  s 685 54     2,B  Address for exit when next parameter is found
696        11 s 706    name
```

Internal routine: Take value of following parameter.

```
697  s1410               tryk
           37     3,B  55 (address of following parameter - 1)
698        60     0,C
699        11 s 711    There are more parameters
700  s 708 35 s   4,B
701        10 s   1,D
702  s 638 69 s1897,D  = - (17 150)
703        17 s 158
704  s 682 13 s   0
705  s 640 17 s  81
```

Internal routine: Take name of next parameter.

```
706  s 690
     s 696
     s1195               læs
     s1703               tryktekst
           37     3,B  55 (address of following parameter - 1)
707        60     0,C
708        51 s 700    No more parameters
709        60 s 653
710        28    33    Set name indication
711  s 699 55 s   1,C
712        10 s 622
```

Special routine: Enter internal routine from core store. This is used as follows:

```
17 713
17 n      where n is the first address of the internal routine.
```

```
713        75 s   1,D
714        74    31
715        37     0,D
716  s 720 16    31
717        55 s   0
```

Special routine: Take name of first parameter, called from core store.

```
718        60 s 653
719        28    33
```

Special routine: Take value of first parameter, called from core store.

```
720        40 s 716
721        10 s 617
```

begin core block (section 11.6.1)
Structure of block-parameters in stack. Any entry into a block, whether core or drum block, or procedure body, will cause a transfer of the current values of certain of the universal block parameters (cf. section 11.8.1) to the stack and assignment of new values to these parameters. The necessary algorithms are given in Jensen, Mondrup, Naur: A Storage Allocation Scheme for ALGOL 60, BIT 1961, no. 2. In DASK the 7 parameters described in this article are packed into two full words in the stack as follows:

```
SR:       ss×2↑(-11) + ba×2↑(-19)
SR + 1:   55 ud      or     55,d   (order form)
SR + 2:   sr×2↑(-11) + tb×2↑(-19)
SR + 3:   g×2↑(-11) + fk×2↑(-19)
```

The correspondence between this notation and that of the article in BIT is as follows:

BIT-notation stack[stack reference]	corresponds to	ss, i.e. the value of IRB in between statements
stack[stack reference + 1]	- -	ba = current block number
stack[stack reference + 2]	- -	memory[SR + 1], ud is return address
stack[stack reference + 3]	- -	sr = previous stack reference
stack[stack reference + 4]	- -	tb = available drum block in previous level
stack[stack reference + 5]	- -	g = current limit in previous level
stack[stack reference + 6]	- -	fk = last drum track used in previous level.

```
722          68    54
723          35 s2044,B
724          60 s 811
725          28     1,B  55 0,d
```

begin core procedure body (section 11.6.1).

```
726          55 s   0
727  s 761                  drum block
     s 763                  drum procedure block
             60     1,D
728          29    17       store limit
729          78     0,B     block number
730          34    16
731          21    16
732          11 s  57       store limit ≥ index
733          53 s 764       drum block
734          55 s   1,D
735  s 780   40    54
736          08     2,B     available drum block, last drum track
737          40    48
738          09     2,B     stack reference, current limit
739          60    17
740          21    49
741          51 s 748       store limit < current limit
742          26    49       store limit
743          20    29
744          11 s 748       arrow ≥ current limit
745          60 s 104
746          21    49
747          29    29       2048 - store limit
748  s 741
     s 744   61 s 138
749          26    52       drum procedure depth - 2
750          68    54       available drum block
751          34    48       index
752          34     0,B
753          16 s 786       STACK CONTROL
754          10    32
```

begin drum block (section 11.6.1).

```
755          35 s2044,B
756          60 s 762
757          28     1,B  55 1
758          55 s   4,D
759          54     1,B
760          55 s   2
761          10 s 727
```

begin drum procedure block (section 11.6.1).

```
762  s 756 55 s   1
763        10 s 727
764  s 733 21    52,C
765        78     6     block number - (if procedure then available drum
                        procedure else available drum block)
766        26    52,C
767        63     6
768        51 s 773     block numbers do not match
769        60    17
770        20    29
771        11 s 778     arrow ≥ store limit
772        61 s 139
773  s 768 29    51,C  if procedure then drum procedure depth:= - 1
774        74    41    cf. s 777
775        55 s   1,D
776        16 s 222     TRANSFER DRUM SECTION TO CORES
777        37    41    75 (cf. s 774)
778  s 771 37     3,D  55 first order
779        55 s2047,C
780        10 s 735
```

Internal routine: STACK CONTROL AFTER index - 2 = arrow.

```
781  s 599 66    29     (2048 - arrow) + 2
782        20    49
783        51 s  57     arrow ≤ current store limit
784        40     0,B
785        10     1,D
```

Internal routine: STACK CONTROL AFTER ARBITRARY COUNT OF index.

```
786  s 621              Fetch name or value of parameter
     s 666              Procedure entry
     s 753              block begin
     s 980              array declaration
           34    16     index
787        60    16
788        20    29
789        21 s 138
790        51 s   1,D   index > arrow
791        61    16
792        21 s 691
793        29    29     2046 - index
794        20    49
795        11 s   1,D   index > current store limit
796        10 s  57     STACK ALARM
```

end (section 11.6.2).

```
797          55 s   0
798          08     6      value (cf. s 807)
799          37    48      35 stack reference
800  s 822                 goto label
             40     2,B
801          09    48      stack reference, current store limit
802          58    54      available drum block, last drum track
803          66    52      drum procedure depth + 2
804          51 s 806      drum procedure depth < 0
805          48    52
806  s 804 53 s 816        goto
807          40     6      cf. s 798
808          37     1,B    55,d or 55 return - 1
809          35 s   4,B
810          10    32
```

go to label in AR (section 11.6.4.2.2).

```
811  s 724 55 s   0,D
812          21 s 139
813          51    32      undefined switch designator
814          29    31      la - 1
815          78     6      block number
816  s 806 37    48      35 stack reference
817          60     0,B
818          29    30
819          21     6
820          78     7
821          63     7
822          51 s 800
823          10    30
```

AR:= value of switch designator (section 11.7.3.2).

```
824          21 s 100
825          51 s 846      Subscript ≤ 0
826          00 s  91
827          08  2044,B   Subscript , 55
828          54  2045,B    55 call address
829  s 845 61 s 100
830          26  2044,B
831          11 s 837      Element not yet found
832          55 s   1,D
833          16 s 616      Take value of first element
834          37     1,B    55 call address
835          35 s   4,B
836          10    32
837  s 831 75 s   1,D
838          60     0,D
839          29    43
840          29    47
841          21    47
842          51 s 844      Element is not an expression
843          37    43      75 (address of following element - 1)
844  s 842 60     0,D
845          11 s 829      Not last element
846  s 825 50    32
```

Array declaration (section 11.7.2.1).

```
847          55 s   0,D
848  s 855
     s 863   55 s   1,C
849          60     0,C
850          28  2046,B
851          78  2047,B
852  s 965   60  2047,B
853          35 s2046,B
854          21 s  85
855          51 s 848     operation < OA
856          21 s 102
857          51 s 864     OA ≤ operation ≤ OC (end of declaration)
858          16 s 616     Take value of first expression
859          16 s 564     AR:= round(AR)
860          35 s   4,B
861          37  2047,B   55 (address of previous parameter)
862          08     0,B
863          10 s 848
864  s 857   54    39
865          37    48     35 stack reference
866          60 s 904
867          28     2
868          60  2046,B
869          00    39
870          08    12     08 a,55
871  s 871   60 s 871
872          28    10     60
873          60 s 119
874          28    14     IRC2:= -1
875          28    11
876          34    10     a0:= SR
877          35 s2044,B
878          10 s 92
879  s 895
     s 912   60 s   1
880  s 973   37    13     55 IRC1
881          63  2047,B
882          51 s 900     Drum only
883  s 886   35 s2044,B
884          55 s   1,C
885          63  2047,B
886          11 s 883     Not last upper bound
887          60     6
888          0C s   9     . . . . . . . . .
889          11 s 902     real ∨ integer
890          60    13
891          28    11     IRC2:= IR1
892          61 s 113
893          08    14     tind:= -1/2; m:= 0
894          60    10
895          21 s 879
896          28     3,C   FL2:= a0 - 1
897          55 s   4,C
898          34    30     IRB2:= IRB
```

```
899          10 s 922
900  s 882   16 s 903
901          55 s   2,C
902  s 889   16 s 982     Calculate coefficients and sum
903  s 900
     s 914   60     6
904  s 866   26    15     m:= m + n
905          61     6
906          26    10     a0:= a0 - n
907          51 s  57     a0 < 0, alarm
908          00  1159,D   AR:= AR + (if drum only then IRC2 else IRC1)
909          20     7
910          37    12     08
911          66    12
912          20 s 879
913          21    13
914          51 s 903     More arrays with these bounds
915          37 s2047,D   if drum only then 10 s 922 else 55 s 2,c
916          60    14
917          26  2047,C
918          60    11
919          51 s 922     Not drum
920          28     0,C
921  s 925   55 s   1,C
922  s 899
     s 919   54    12
923          60    12
924          0C s  11
925          51 s 921
926          35 s2046,B
927  s 878   60     0,B
928          0C s  19
929          11 s 935     start drum indices ∨ real ∨ end core array
930          12 s 966     integer
931          55 s   0
932          60    15
933          21 s 106
934          11 s 936     m ≥ 64
935  s 929   26    10
936  s 934   0C s  11
937          11 s 941     a0 is even
938          61 s 105
939          26    10
940          51 s  57     a0 < 0 alarm
941  s 937   53 s 965     Not finish drum
942          34    37
943          37    30     35 IRB2
944          37    11     55 IRC2
945          61    15
946          0D s  26     - c[q]:= entier(-m/64)
947          18     1,C
948          16 s 987     Calculate drum coefficients
949          60     6     AR:= - length of array
950          0D s   9
951          28     7
952          26    55     FK:= FK - length
```

```
 953          28      2,C
 954          0C s    9
 955          20 s 136
 956          28      1,C   1C k4
 957          37      2     1C k00
 958          60     55
 959          21     44
 960          51 s   79     Drum track number too low, alarm
 961          60     10
 962          29      2,C   First address
 963          61      7
 964          26      3,C   k5x2↑(-20)
 965  s 941   60 s 852
 966  s 930   28      7     mod:= if real then -1 - 2↑(-14) else -1
 967          40      0,B
 968          20     12
 969          29     13     IRC1:= a + 2xn
 970          0C s   28
 971          28      6     n:= type
 972          0C s    8
 973          11 s 880      Not finished
 974          37     48     35 SR
 975          60     10
 976          29      0,B   ss:= a0
 977          29     37
 978          37     37     35 a0
 979          37     39     55 exit - 1
 980          16 s 786      STACK CONTROL
 981          10     32     exit
 982  s 902   34     37
 983          37     13     55 IRC1
 984          60      6
 985          0D s    1
 986          28      1,C
 987  s 948   54      2
 988          68      0,C
 989  s1003   60      0,B
 990          21      2,B
 991          51 s   69     upper < lower
 992          24 s 115
 993          2A      1,C
 994          0C s   19
 995          28      2,C   C[q - p -1]:= (u - l + 1)xC[q - p]
 996          64      1,C
 997          2A      2,B
 998          0C s   20
 999          37      2     26 a1  Sum(C[i]xlower[i])
1000          35 s    4,B
1001          55 s    1,C
1002          63      1,B
1003          11 s 989
1004          60      1,C
1005          20      1,C
1006          28      6
1007          37     37     35 IRB1
1008          10 s    1,D
```

. . . . . .

for integer 1 (section 11.6.5.2).

```
1009          66     0,D
1010          10 s1021
```

. . . . . .

for integer 2.

```
1011          60     1,D
1012          0C s  15
1013          11 s1015     Not formal
1014          60 s 126
1015   s1013  21 s 107
1016          20     1,D
1017          28     7    37 formal + 1, or 28 i
1018          37     1,D  37 formal, or 60 i
1019          20     0,B
1020          37     7    37 formal + 1, or 28 i
1021   s1010  55 s   1,D
1022          37     1,D  37 formal, or 60 i
1023          35 s   2,B
1024          21  2047,B
1025          51 s1029     i < c
1026          21 s 115
1027          51    32     i = c
1028          50 s1030
1029   s1025  60 s 104
1030   s1028  20  2046,B
1031          51    32     i < c ≤ b > 0
1032          55 s   2,C
1033          10    32
```

. . . . . .

for real 1 (section 11.6.5.2).

```
1034          66     0,D
1035          29    36     Set ≠ 0
```

. . . . . .

for real 2.

```
1036          55 s   1,D
1037          37    36
1038          69    36
1039          60     0,C
1040          28     7
1041          73     7     First entry
1042          28     3
1043          0C s  15
1044          11 s1046     Operation 40
1045          60 s 126
1046   s1044  21 s 107
1047          20     0,C
1048          28     6    08 r
1049          40     2,B
1050          08  2046,B
1051          35  2046,B
1052          10     3
1053       9  11 s1055     r ≤ c
```

```
1054          50 s1058
1055  s1053   01 s 105
1056          51 s1059     r = 0
1057          60 s 104-
1058  s1054   20     2,B
1059  s1056   35 s   4,B
1060          51    32-    Repeat statement
1061          55 s   2,C
1062          10    32     Statement finished
1063          00 s   0
```

Læs to array.

```
1064  s1210   37     2
1065          21 s 107
1066          08     2
1067          37     3-     55 address of array coefficients
1068          60     1,C
1069          51 s1080-     Only drum array
1070  s1072   55 s   1,C
1071          60     1,C
1072          11 s1070      Not yet C[0]
1073          0C s   1
1074          29    16      Number of half words
1075          20 s 104
1076          51 s1098      Only core subscripts
1077          29    16-
1078          37     2,C    55 address of drum coefficients
1079          10 s1082
1080  s1069   68    16-
1081  s1083   55 s   1,C
1082  s1079   60     1,C
1083          51 s1081      Not yet C[0] for drum
1084          0F s   9-
1085          21     2,C
1086          0C s  12
1087          0F s  23
1088          18    27      Current track - start track
1089  s1191   16 s 204      Change core section
1090          61    11-     1C
1091          20     1,C
1092          28    27-     Number of tracks per subscript
1093          60     2,C
1094          20     3,C
1095          0C s  12
1096          0F s   3
1097          21    11      AR:= last track - last track used + 2
1098  s1076   14     3
1099          11 s1101      Only core ∨ last drum section
1100  s1208   55 s   0
1101  s1099   54    39
```

The following program performs the conversion of the input symbols to binary form. An ALGOL description of this program, which, however, omits the mechanism for reading dittos (cf. sections 9.4.3.3, 9.4.3.5) is as follows:

```
integer S, B, C, exp 2, AR; comment B = 1 ved læsning af integer, B = 0
     ved taldel, B = -1 ved exponent. C = 0 under læsning af heltalsdel,
     positiv exponent og pos. integer. C = 2 efter komma og under negativ
     integer og exponent;
real MR; boolean Fortegn er tilladt, Talslut er tilladt;
integer array tal[-1 : 1]; comment
     tal[-1] = -exponent, tal[0] = -taldel ved real, tal[1] = -integer
     værdi;
switch Q:= Q1, Q2, Q3;

indgang real læs: B:= 0; goto præludium;
indgang integer læs: B:= 1;
præludium: exp 2:= tal[-1]:= tal[0]:= tal[1]:= 0;
           MR:= -0.5;
           C:= 0; Fortegn er tilladt:= true;
           Talslut er tilladt:= true;
næste symbol: S:= læst symbol;
     if S = plus ∨ S = minus then goto fortegn;
     if S = 10 then goto ti;
     if S = . then goto decimal punkt;
     if S = slutsymbol then goto slut;
     goto ciffer;

ciffer: Fortegn er tilladt:= false; Talslut er tilladt:= true;
     if tal[B] < -2↑39/10 then goto E37;
     tal[B]:= 10 × tal[B] - S; comment Udføres ved additioner;
     if tal[B] > -2↑39/10 then goto L1;
     tal[B]:= tal[B]/2; exp 2:= exp 2 - 1;
L1:  goto if B ≠ 0 ∨ C = 0 then næste symbol else Multiplicer;
E37: if C ≠ 0 then goto næste symbol;
Multiplicer: MR:= MR × (if C = 0 then 10/16 else 0.8);
     exp 2:= exp 2 + (if C = 0 then -4 else 3);
     comment MR normaliseres tillige;
     goto if B ≠ 0 then E39 else næste symbol;

fortegn: if -, Fortegn er tilladt then goto fejl;
     if S = plus then goto E71;
     if B ≠ 0 then goto E72;
     MR:= 0.5; goto E71;

decimal punkt: if B ≠ 0 ∨ C ≠ 0 then goto fejl;
     comment . efter 10 eller integer eller . ;
E72: C:= 2;
E71: Fortegn er tilladt:= false; Talslut er tilladt:= false;
     goto næste symbol;
ti:  if B ≠ 0 then goto fejl; comment 10 efter 10 eller integer;
     if (-, Fortegn er tilladt) ∧ Talslut er tilladt then goto E74;
     if C ≠ 0 then goto fejl; comment . uden cifre;
     tal[0]:= -1; comment Ren exponent;
E74: B:= -1; C:= 0;
```

```
     fortegn er tilladt:= true; Talslut er tilladt:= false;
     goto næste symbol;

slut: if Talslut er tilladt ∧ (¬Fortegn er tilladt) then
     begin AR:= 0; goto Q[B+2] end;
     if Talslut er tilladt then goto næste symbol;
     goto fejl; comment 10 eller . uden cifre;
E39: AR:= 1;
Q1:  tal[-1]:= AR:= AR + tal[-1];
Q2:  if AR ≥ 0 then goto talpakning;
Q3:  if AR < 0 then goto Multiplikation;
     AR:= if C = 0 then -tal [1] else tal[1];
     goto finis;
talpakning: AR:= MR × tal[0];
     comment exp 2 indeholder 2-potens med omvendt fortegn;

finis:
```

The correspondence between the variables of the ALGOL description and the DASK machine program is as follows:

ALGOL		DASK
integer	S	location 7
integer	B	IRB
integer	C	IRC
integer	exp 2	location 10 (half cell)
integer	AR	AR (unit in position 39)
real	MR	MR
boolean	Fortegn er tilladt	location 14, position 0, binary 1 for true
boolean	Talslut er tilladt	location 14, positions 1, = 0 for true
integer	tal[-1]	location 11 (half cell)
integer	tal[0]	location 12 (full cell)
integer	tal[1]	location 13 (half cell)

The control of ditto reading uses the remaining positions of location 14.

```
1102          60     2
1103          0C s  14
1104          0F s  11
1105          29    37
1106          37    37    35 (if integer then 1 else 0)
1107  s1189
      s1237 68     15    symbol
1108          48     10   p2:= p10:= x:= 0
1109          48     12
1110          65 s 113    MR:= - 1/2
1111  s1166 55 s   0
1112          21 s 113
1113  s1147
      s1153
      s1158 28     14
1114          10 s1262
```

```
Ciffer:

1115  s1249                    Input symbol = 0
            48     7
1116  s1250                    Input symbol = digit ≠ 0
            69    14·
1117        42    12,B
1118        01 s  96
1119        11 s1132·          tal[B] > - 2↑39/10
1120        40    12,B
1121        0C s   3·
1122        00    12,B
1123        01     7·
1124        06    12,B         tal[B]:= 10×tal[B] - S
1125        52 s1129·          tal[B] > - 2↑39/10
1126        08    12,B
1127        61 s 139
1128        26    10           exp 2:= exp 2 - 1
1129  s1125 33 s1262           B ≠ 0
1130        53 s1133           C ≠ 0
1131        10 s1262
1132  s1119 53 s1262           We are after .

Multiplicer:

1133  s1130
      s1180 0A s  92,C
1134        0E    17
1135        24 s  93·          MR:= MR×(if C = 0 then 10/16 else 0.8)
1136        60 s 108,C
1137        20    17
1138        26    10           exp 2 := exp + (if C = 0 then -4 else 3)
1139        33 s1177           Final assembly
1140        10 s1262

Fortegn:

1141  s1244 60    14·
1142        11 s1149,B         Previous symbol: digit, or sign, or .
1143        61    58
1144        11 s1157           sign = +
1145        33 s1156           Sign belongs to exponent or integer
1146        64 s 113
1147       ·10 s1113
1148  s1142,B
           ·10 s  71           More than one sign after 10
1149  s1142,B                  Sign after .
           ·53 s  71
1150  s1142,B
            21 s 114
1151        51 s  71           Sign follows upon digit
1152        35 s   1
1153        10 s1113
```

```
decimal point.

1154  s1245 33 s  71    Ditto, or integer, or previous symbol 10
1155        53 s  71    Previous symbol .
1156  s1145 55 s   2
1157  s1144 60 s 113
1158        10 s1113

ti:

1159  s1246 33 s  71    Ditto, or integer, or previous symbol 10
1160        63    14
1161        11 s1165    Digits have already appeared
1162        53 s  71    Previous symbol .
1163        41 s 115
1164        08    12    tal[0]:= -1 (pure exponent)
1165  s1161 35 s2047
1166        50 s1111

slut:

1167  s1247
      s1251 63    14
1168        11 s1179,B  Digits have appeared
1169        61    14
1170        12 s1262    Only leading terminators have come
1171        20 s 139
1172       -51 s1174,B
1173  s1172,B
           10 s  71    Digits after ditto, 10 without digits
1174  s1172,B           . without digits
           53 s  71
1175  s1172,B
           62 s 104
1176        10 s1182
1177  s1139 60 s 100
1178       -26    11    tal[1]:= tal[1] + 1
1179  s1168,B
           11 s  20    tal[1] = 0
1180        51 s1133
1181        37 s1240,C  61 13 (C = 0), 60 13 (C = 2)
1182  s1176    s  21
      s1225
      s1231 37     3    læst: 17 1238, otherwise 55 relative addresses
1183        12 s1185    Ditto
1184        37     2    08 address,C
1185  s1183 37    37    35 (if integer then 1 else 0)
1186        60 s 138,B
1187        26     3
1188        21    16
1189        51 s1107    Reading not finished
1190        37    39    55 ( if not læst drum array section then ≠ 0 else 0)
1191        53 s1089    Core section of drum array finished
1192        37    30    35 index
1193        55 s1199
1194        54     2,B
1195        16 s 706    Take name of following parameter
1196        10    32
```

læs (section 11.5.4.1).

```
1197         55 s   1,D
1198         16 s 614     Take name of first parameter
1199  s1193  00 s 141
1200         08     2
1201         28    27
1202         0C s  17
1203  s1211                Entry from taking value of subscr. var.
             34    30
1204         12 s1209     Array or subscr. var. first time
1205         61 s 107
1206         26     2     Store instruction 08 or 28 address
1207         68    16
1208         10 s1100
1209  s1204  0C s   1
1210         12 s1064     Array
1211         55 s1203     Set return
1212         54     2,B
1213         10    27     Fetch value of subscripted var.
```

Pack real input.

```
1214  s  22 4A    12
1215         4E    17
1216         08     6     Store full accuracy number
1217         18    14
1218         60    10
1219         20    17
1220         0D s   8
1221         20 s 146
1222         51 s  73     abs(input) > 3.4_10 38
1223         21 s 139
1224         51 s1226     abs(input) > 2.94_10 -39
1225         50 s1182
1226  s1224  78    11
1227         40    14
1228         00    11
1229         01 s 101
1230         20 s  93     Remove possible overflow
1231         10 s1182
```

læst (section 11.5.4.1).

```
1232         34    30
1233         74    31
1234         35 s   0
1235         60 s1241
1236         28     3
1237         10 s1107
1238  s1241  12 s  71     Ditto
1239         10    30
1240  s1181,C
             61    13
1241  s1235 -17 s1238
1242  s1181,C
             60    13
```

Input symbol sorter.
The jump from s1293 is made with IRD = table (input symbol), where the function table is defined through the table in s1294 to s1311. The contents of AR is given as follows:

Input procedure	AR
læs, læst	0
læsstreng	-1 with overflow
trykkopi, skrvkopi	-2↑(-11)

```
1243  s1293,D
            10 s  75    Not used symbol (alarm)
1244        11 s1141    + -
1245        11 s1154    .
1246        11 s1159
1247        11 s1167    10
                        Other signs
1248        12 s1353
1249        11 s1115    0
1250        11 s1116    1, 2, ... 9
1251        11 s1167    letter
1252        12 s1343
1253        12 s1262    SPACE _ STOP
1254        11 s1262
1255        10 s1382
1256        10 s1312    PUNCH OFF, TAPE FEED
1257        10 s1320    CLEAR CODE, SUM CODE, STOP CODE
1258        60     7    UPPER and LOWER CASE
1259        0C s  18
1260        0F s  13
1261        28    58
```

Central input program.
Read, check, classify symbol. Universal input mechanisms (section 9.2).
This program uses one input parameter:

Location 15	
0	læs, læst
-1	læsstreng
2↑(-11)	trykkopi, skrvkopi

It will read one or more symbols from the input tape and if necessary perform the appropriate universal input mechanisms and skip blind symbols (section 9.3). It will return as follows: Current input case in location 58 (cf. section 11.8.2). Current input sum in location 57 (cf. section 11.8.2). Last symbol from tape in location 7. It will return to an address determined by the parameter in location 15 and the class of the last symbol read, as follows:

Loc. 15		Symbol	Return to
0	læs, læst	+ -	s1141
		.	s1154
		10 0	s1159
		1, 2, ... 9	s1115
		<læs terminator> (Section 9.4.3.3)	s1116
-1	læsstreng	<læsstreng information>	s1167
		<læsstreng terminator>	s1343
2↑(-11)	trykkopi		s1353
			s1382

On return IRB, IRC, and MR are unchanged.

```
1262  s1114                 læs, læst, first digit
      s1129                 læs, læst
      s1131                  -     -
      s1132                  -     -
      s1140                  -     -
      s1170                  -     -
      s1253                 blind in læsstreng
      s1254                   -     - læs, læst
      s1266                 ALL HOLES
      s1271                 TAPE FEED, BLANK TAPE
      s1324                 END CODE
      s1336                 CLEAR CODE, SUM CODE
      s1342                 læsstreng
      s1344                     -
      s1352                     -
      s1354                     -
      s1393                 trykkopi
      s1395                     -
      s1400                     -
            37   38         79
1263        08    7         symbol = 16xs1 + s2
1264        11 s1272         correct parity
1265        01 s 117
1266        51 s1262         Symbol = ALL HOLES
1267        61    7
1268        51 s 75          Wrong parity ∧ symbol ≠ blank tape
1269  s1314 41    7
1270        06   57         Checksum (correction for TAPE FEED)
1271        10 s1262
1272  s1264 06   57         Sum checksum
1273        60    7
1274        0C s  4
1275        29   59         s1
1276        21   59
1277        0C s  3
1278        51 s1282         s2 ≤ 1
1279        20 s 108
1280        11 s1283         s2 ≥ 10
1281        50 s1282
1282  s1278
      s1281 21 s 108
1283  s1280 00   58
1284        0C s  2
1285        09   22
1286        37   22         75 symbol table address
1287        60 s1294,D
1288        37   23         0C 4xs1
1289        0F s  8
1290        29   22
1291        37   22         75 table (input symbol)
1292        61   15
1293        10 s1243,D
```

Table for input symbol sorting.
Locations s1294 to s1311 give the complete table. The arguments are shown in the notes. Upper case corresponds to even address for the bit-word. Thus, i.g. table (;) is found by using: upper case, s1 = 0, s2 = 5. This points to the first sedecimal character at s1310, so table (;) = 4.

			s1 =	0	1	2	3	4	s2
1294	s1287,D	B	000F0				LOWERCASE		10
1295		B	000F0				LOWERCASE		
1296		B	A3840	STOPCODE	¨	ø	:		11
1297		B	A4820	STOPCODE	'	ø	.		
1298		B	EE4F0	ENDCODE	CLEARCODE	PUNCHON	UPPERCASE		12
1299		B	EE4F0	ENDCODE	CLEARCODE	PUNCHON	UPPERCASE		
1300		B	000E0				SUMCODE		13
1301		B	000E0				SUMCODE		
1302		B	44000	|	TAB				14
1303		B	A4000	-	TAB				
1304		B	0D0D0		PUNCHOFF		TAPEFEED		15
1305		B	0D0D0		PUNCHOFF		TAPEFEED		
1306		B	A4184	SPACE	∧	+	Æ	CARRET	0
1307		B	A6184	SPACE	0	-	æ	CARRET	
1308		B	44880	∨	>	J	A		1
1309		B	74880	1	<	j	a		
1310		B	48880	<tegn>	<LETTER>	<LETTER>	<LETTER>		2-9
1311		B	78880	<digit>	<letter>	<letter>	<letter>		

1312	s1256			PUNCHOFF or TAPEFEED
		60	7	
1313		0C	s 14	
1314	s1319	51	s1269	symbol = TAPE FEED ∨ symbol = PUNCH ON
1315	s1317	37	38	79
1316		01	s 133	
1317		51	s1315	symbol < PUNCH ON
1318		01	s 115	
1319		10	s1314	
1320	s1257			END CODE, CLEAR CODE, SUM CODE
		60	7	
1321		0C	s 15	
1322		12	s1335	CLEAR CODE
1323		51	s1325	SUM CODE
1324		30	s1262	
1325	s1323	40	57	
1326		0B	s 117	
1327		0C	s 1	
1328		58	7	
1329		37	38	79
1330		0C	s 21	
1331		21	7	
1332		51	s 77	Sum does not check
1333		21	s 115	
1334		11	s 77	Sum does not check
1335	s1322	68	57	Checksum := 0
1336		10	s1262	

læsstreng (section 11.5.4.1).

```
1337          74    31
1338          55 s  40
1339          48    50     streng:= 0
1340          60 s 104
1341  s1381  28    15
1342          10 s1262
1343  s1252                Central input program
              53 s1345     Less than 6 symbols have been read
1344          10 s1262
1345  s1343  55 s2040,C
1346          60    58     Case
1347          0F s  19
1348          00.    7
1349          00 s 115
1350          0C s   0,C
1351          06    50     Form string
1352          10 s1262
1353  s1248                Central input program
              43    50
1354          11 s1262     Only terminators have been read
1355          10    31     Exit
```

streng (section 11.5.4.1).

```
1356          55 s   1,D
1357          16 s 616   Take value of parameter
1358          08     0,B
1359          01    50
1360          08    12
1361          55 s2040 -
1362  s1367  55 s   8,C
1363          40     0,B   | Find the number of non-zero
1364          0F s   0,C     characters in parameter
1365          58    11     |
1366          63    11
1367          11 s1362     -  Symbol in streng = 0
1368          41 s 115
1369          0C s   0,C
1370          02    12     AR:= streng (  )
1371          10 s 657
```

skrvkopi (section 11.5.4.1).

```
1372          14    56
```

trykkopi (section 11.5.4.1).

```
1373          55 s   1,D
1374          16 s 616     Take value of parameter
1375          0F s  12
1376          08     0,B   Testsymbol 1
1377          0F s  12
1378          58     1,B   Testsymbol 2
1379          16 s1608     Select medium
```

```
1380          60 s 139
1381          10 s1341
1382  s1255                  Central input program
              60     7
1383          20    58      Case
1384          20    58
1385          21     0,B     Testsymbol 1 (internal form: section 11.4.8)
1386          11 s1394       Symbol ≠ Testsymbol 1
1387          20 s 115
1388          51 s1394       Symbol ≠ Testsymbol 1
1389          63     1,B
1390          11 s1761       Testsymbol 2 = 0, exit
1391          60     1,B
1392          08     0,B
1393          10 s1262
1394  s1386
      s1388 63     1,B
1395          51 s1262       Testsymbol 2 ≠ 0, skip
1396          61    58
1397          16 s1601       Set output case
1398          40     7
1399          16 s1595       Output one character
1400          10 s1262
```

tryk (section 11.5.4.1).

```
1401          50 s1403
```

skrv (section 11.5.4.1).

```
1402          61 s 105
1403  s1401   20 s1690
1404          28  2045,B     Medium: tryk = 55 1413, skrv = 55 1412
1405          55 s   1,D
1406          16 s 616       Take value of first parameter (layout)
1407          37     1,B
1408          54     2,B     Prepare exit from take value of next parameter
1409          08     0,B     Layout
1410  s1418
      s1680   16 s 697       Take value of next parameter
1411          10 s1743       Exit if no more parameters
1412          14    56       Medium:= skrv
1413  s1690   75 s   0       Set ALGOL return
1414          11 s1419       Value ≠ nonsense
1415          00 s 101
1416          51 s1420       Value ≠ nonsense
1417          34    56
1418          10 s1410       Skip printing (value = nonsense)
1419  s1414   00 s 101
1420  s1416   18     6       x1
1421          01     6
1422          01 s 111
1423          0C s  28
1424          29    34       x2
1425          40     0,B     layout
```

Non-ALGOL entry. Number in 6 and 34. Layout in AR. Medium in 56.

```
1426          08     2
1427          74    31     IRD = if ALGOL then 0 else return
1428          34    30
1429          16 s1608     Select medium
1430          48    14     Clear location
1431          60     2     Layout bhdfs (section 11.4.7)
1432          0F s   8
1433          29    37     b
1434          09    35     f1f2 ns
1435          0C s  12
1436          0F s   8
1437          29    39     h
1438          0C s  12
1439          0F s   8
1440          09    40     d and clearing
1441          60     3     Layout pqrst (section 11.4.7)
1442          00 s  89
1443  s1666   0F s  16
1444          08     2     0 0 0 0 p  q r s t 15
1445          60    35
1446          13 s   0
1447  s1660   0C s 133
1448          0F s 137
1449          29    42     s
1450          35 s1024     y2 := 0 (IRB = y2 + 1024)
1451          43     6
1452          11 s1525     x1 = 0
```

The program from s1453 - s1479 converts the number from the form
x = x1×2↑x2 (x1 in 6, -x2 in 34 adr)
into the form
abs(x) = y1×10↑(y2) (y1 in 10, y2 + 1024 in IRB) with $0.1 \leq y1 < 1$.

```
1453          42     6     y1 := abs(x1)
1454          11 s1456     x ≠ -1
1455          01 s 115
1456  s1454
      s1465
      s1471   0E    14     -
1457          24 s  93       | y1:= y1×2↑p
1458          60    14       | k:= k + p
1459          26    34     -
1460          11 s1466     k ≥ 0
1461          20 s1474
1462          29    34     k:= k + 3
1463          35 s   1,B   y2:= y2 + 1
1464          0A s  94     y1:= y1×0.8
1465          10 s1456
1466  s1460   21 s1474
1467          51 s1474     k < 4
1468          29    34     k:= k - 4
1469  s1479   35 s2047,B   y2:= y2 - 1
```

```
1470          2A s  92     y1:= y1x10/16
1471          73 s1456
1472          0C s   1     y1:= y1x2
1473          10 s1476
1474  s1461
      s1466
      s1467   07 s   3,B
1475          37     34    0D k, y1:= y1x2↑(-k)
1476  s1473
      s1535   08     10
1477          01 s  96
1478          75 s   0
1479          51 s1469     y1 < 0.1
1480          60     37
1481          29     43    75 b
1482          21     39
1483          09     14    00 b - h - 1, 00
1484          61 s 113
1485          34     15    00 y2 + 1024
1486          26     15    00 y2
1487          21     37
1488          20     40
1489          29     34    0D y2 - b + d
1490          61     14
1491          20     40
1492          29     16    00 h + d - b + 1
1493          60 s 105
1494          2B     16    MR:= 1/(h + d - b + 1)
1495          2A     34
1496          29     17    00 k, k = entier((y2 - b + d)/(h + d - b + 1))
                           The exponent to be printed is kx(h + d - b + 1)
1497  s1519   37     42    75 s
1498          60 s  85,D   (10↑s - 1)x2↑(-11)
1499          29     34    10↑s - 1 = s2 = the greatest exponent which may
                           be printed
1500          64     17
1501          2A     16
1502          0C s  11
1503          28      7    z2x2↑(-11). z2 is the exponent to be printed
1504  s1513   20     34
1505          11 s1514     z2 ≥ -s2
1506          60     15    y2
1507          20     14    y2 + b - h - 1
1508          21      7
1509          29     43    b1 = y2 - z2 + b - h - 1
1510          51 s1525     There are no significant digits
1511          60     16
1512          26      7    z2:= z2 + h + d - b + 1
1513          10 s1504
1514  s1505   60     34
1515          21      7
1516          11 s1520     z2 < s2, z2 < 10↑s
```

```
1517          60 s 105
1518          26    42     s:= s + 1 (i.e. alarm printing)
1519          10 s1497
1520  s1516   37    43     75 b1
1521          64 s 113
1522          73 s1527     b1 ≠ 0
1523          00    10
1524          12 s1533     y1 > 0.5
1525  s1452
      s1510   48     6     x = 0
1526          16 s1541
1527  s1522
      s1530   0A s  98     -
1528          24 s  93      | Form round-off constant
1529          75 s2047,D
1530          73 s1527     -
1531          04    10
1532          52 s1536     y1 + rounding < 1
1533  s1524   35 s   1,B   -
1534          40 s  98      | Overflow from round-off,
1535          10 s1476     -  repeat analysis
1536  s1532   60     7
1537          21    15
1538          20    39
1539          29    37     fb = z2 - y2 + h = number of initial spaces
1540          37    37     35 fb
1541  s1526   60    35
1542          11 s1558     f = 0 ∨ f = 1
1543          20 s 113
1544          73 s1556     x = 0
1545          51 s1558     f = 2
1546          0C s  11
1547          52 s1551     Not exponent
1548          16 s1601     Output upper case
1549          16 s1594        -
                                10
1550          69    35
1551  s1547   60     6
1552          16 s1601     Output case
1553          16 s1593        -   sign
1554          68     6     sign:= +
1555          10 s1558
1556  s1544   28    35     f1, f2:= 0
1557          16 s1595     Output space

1558  s1542
      s1545 1555
            37    39     55 h
1559  s1566   53 s1571     There are characters before point
1560          10 s1584
1561  s1580   16 s1595     Output space
1562  s1576
      s1577   35 s2047,B
1563  s1570   16 s1595     Output space
```

```
1564          61 s 115
1565          26    2
1566          11 s1559    Digit group is not finished
1567          40    3     -
1568          0C s  4      |
1569          08    2        Take next digit group
1570          50 s1563-   -
1571  s1559   55 s2047,C
1572          60   35
1573          0C s  4
1574          33 s1576    There are characters before first significant
1575          10 s1578
1576  s1574   53 s1562    Characters before point > 1
1577          11 s1562    n = 0
1578  s1575   0C s  7
1579          52 s1583    Do not print 10 now
1580          33 s1561    Exponent = 0
1581          16 s1601    Output upper case
1582          16 s1594-   Output 10
1583  s1579   55 s  1,C
1584  s1560   60    6
1585          51 s1591    Number < 0
1586          60   35
1587          0C s  1
1588          52 s1645    f = 0 ∨ f = 3
1589          11 s1644    f = 1
1590  s1590   60 s1590
1591  s1585   16 s1601    Output sign
1592          75 s1644
1593  s1553   40 s 123
1594  s1549
      s1582   00 s 125
```

Output one character, form check sum (section 11.8.3).

```
1595  s1399               trykkopi
      s1557               tryk
      s1561                 -
      s1563                 -
      s1607                 -
      s1624               case
      s1644
      s1667
      s1708
      s1715
      s1717
      s1720
      s1742
      s1760 37    61      Output instruction 7A or 5A
1596  s1599               Entry from special entry
            06    63-
1597          50 s  1,D
```

Special entry: return to wired store after exit through replacing output instruction in location 61 by a jump.

```
1598        37    22    75 return
1599        10 s1596
```

Set output case = lower

```
1600  s1600
      s1622
      s1633 61 s1600
```

Set output case (section 11.8.3)
Input parameter in AR:
AR ≥ 0: upper case
AR < 0: lower case

```
1601  s1397             trykkopi
      s1548             tryk
      s1552               -
      s1581               -
      s1591               -
      s1706             tryktekst
            37    62    No case: 13, upper: 11 20, lower 51 20
1602        11 s1605    New case = upper
1603        40 s1682
1604        10 s1606
1605  s1602 40 s1684
1606  s1604 28    62
1607        10 s1595
```

Output medium selector (section 11.8.3).
Input parameter: location 56.
= 0 for skrv
≠ 0 - tryk
Location 56 is set and reset only in positions 1 - 11. By placing suitable contents in positions 0 and 12 - 19 the medium may be forced (section 11.8.3).

```
1608  s1379             trykkopi, skrvkopi
      s1429             tryk
      s1693             tryktekst
      s1713             tryksum
      s1740             Small output procedures
      s1756               -     -       -
            61    56
1609        74    56    Medium parameter:= (≠ 0)
1610        37    60    tryk: 51 20, skrv: 11 20
1611        11 s1614    skrv is wanted
```

```
1612          40 s1686
1613          10 s1615
1614  s1611   40 s1688
1615  s1613   08   60     Set new medium and output order
1616          41   62     -
1617          06   18
1618          06   62       | Interchange case and sum
1619          01   18
1620          08   18-     -
1621          10 s  1,D
1622  s1626
      s1629   16 s1600     Output lower case
1623          40 s 121
1624          16 s1595     Output point
1625          10 s1668
1626  s1650   33 s1622     More characters before first significant
1627          60   43
1628          21 s 105
1629          11 s1622     There are more significant digits
1630  s1646   61 s 115
1631          26    2
1632          51 s1667     No more characters in digit group
1633          16 s1600     Output lower case
1634          33 s1672     More characters before first significant
1635          61 s 105
1636          26   43
1637          51 s1674     No more significant digits
1638          4A s  85     AR:= next digit
1639  s1673
      s1676   01 s 115
1640          11 s1642     Digit ≠ 0
1641          00 s 127
1642  s1640   00 s 115-
1643  s1675   55 s2047,C
1644  s1589
      s1592   16 s1595     Output digit or zero
1645  s1588
      s1671   37   41     75
1646          53 s1630     More characters on this side of point
1647          73 s1651     After point
1648          66   41
1649          37   40     55 number of digits after point
1650          53 s1626     There are digits after point
1651  s1647   60   42     -
1652          29   37       | b:= h:= s
1653          29   39     -
1654          21 s 105
1655          51 s1677     s = 0, exit
```

Prepare printing of exponent.

```
1656          49    40    d:= 0
1657          40    35
1658          0D s  16
1659          0C s  18
1660          00 s1447
1661          21 s 128
1662          09    34    s:= 0, f1:= f2, n:= 1, mark:= exponent
1663          60     7
1664          08     6
1665          60 s  89
1666          10 s1443
1667  s1632   16 s1595    Output space
1668  s1625   40     3    -
1669          0C s   4    | Next digit group
1670          08     2    |
1671          10 s1645    -
1672  s1634   35 s2047,B
1673          50 s1639
1674  s1637   37    41    75 (if after point then ≠ 0 else 0)
1675          73 s1643    After point: print spaces
1676          50 s1639    Before - : - zeroes
1677  s1655   37    30    Restore IRB
1678          37    31    55 (if ALGOL then 0 else ≠ 0)
1679          53    32    Special exit
1680          10 s1410
1681          13 s   0
1682  s1603   51    20
1683          3A s   0
1684  s1605   11    20
1685          3C s   0
1686  s1612   51    20
1687          7A s   0
1688  s1614   11    20
1689          5A s   0
1690  s1403   55 s1413
```

skrvtekst (section 11.5.4.1).

```
1691  s1749   14    56
```

tryktekst (section 11.5.4.1).

```
1692          55 s   1,D
1693          16 s1608    Set medium
1694          16 s 614    Take name of first parameter
1695          28    27    40 (address of string)
1696  s1711   55 s2008
1697  s1709   55 s   8,C
1698          37    27    40 address of string
1699          0C s  32,C
```

```
1700          0F s   4
1701          25 s 105
1702          11 s1705     Not last symbol
1703          16 s 706     Take name of following parameter
1704          10 s1743     Exit if no more parameters
1705  s1702   0C s   4
1706          16 s1601     Output case
1707          4C s  11
1708          16 s1595     Output character
1709          53 s1697     More characters in word
1710          66    27     Address of string
1711          10 s1696
```

tryksum (section 11.5.4.1).

```
1712          55 s   0,D
1713          16 s1608     Set medium = tryk
1714          40 s 129
1715          16 s1595     Output STOP
1716          40 s 137
1717          16 s1595     Output SUM CODE
1718          40    63
1719          0B s 117
1720          16 s1595     Output sumcharacter
1721          68    63
1722          10 s1743
```

trykklar (section 11.5.4.1).

```
1723          61 s 136
1724          28    63     sum:= - CLEAR CODE
1725          60 s 136
1726          10 s1738
```

trykende (section 11.5.4.1).

```
1727          60 s  84
1728          10 s1738
```

trykstop (section 11.5.4.1).

```
1729          60 s 129
1730          10 s1738
```

trykslut (section 11.5.4.1).

```
1731          60 s 134
1732          10 s1738
```

skrvvr (section 11.5.4.1).

```
1733          14    56
```

trykvr (section 11.5.4.1).

```
1734          60 s 131
1735          10 s1738
```

```
skrvtab (section 11.5.4.1).

1736        14    56

tryktab (section 11.5.4.1).

1737        60 s 112
1738  s1726              trykklar
      s1728              trykende
      s1730              trykstop
      s1732              trykslut
      s1735              trykvr
            55 s   0,D
1739        28     7     symbol
1740        16 s1608     Set medium
1741        40     7
1742        16 s1595     Output symbol

Exit from output procedures.

1743  s1411              tryk
      s1704              tryktekst
      s1722              tryksum
            41 s 115     AR:= nonsense
1744        10    32

trykml (section 11.5.4.1).

1745        50 s1748

skrvml (section 11.5.4.1).

1746        50 s1749

tryktom (section 11.5.4.1).

1747        40 s 135
1748  s1745 21 s 115
1749  s1746 20 s1691
1750        08  2044↑B  13/14 56, symbol
1751        55 s   1,D
1752        16 s 616     Take value of parameter
1753        16 s 564     ARi:= round(AR)
1754        37     0↑B   Set medium parameter
1755        28     0,B
1756        16 s1608     Select medium
1757        16 s1758     IRD:= 1757
1758  s1757 61 s 115
1759        06     0,B   n, output symbol
1760        11 s1595
1761  s1390 41 s 115
1762        10 s 657
```

```
abs (section 11.5.4.1).

1763          55 s   1,D
1764          16 s 616     Take value of parameter
1765          00 s 101
1766          11 s1772     x > 0
1767          18     0,B
1768          01     0,B
1769          01     0,B
1770          11 s1772     x ≠ -1x2↑n
1771          01 s2046
1772    s1766
        s1770 08     0,B
1773          10 s 655

entier (section 11.5.4.1).

1774          55 s   1,D
1775          16 s 616     Take value of parameter
1776          75 s   0
1777          10 s 570

sign (section 11.5.4.1).

1778          55 s   1,D
1779          16 s 616     Take value of parameter
1780          00 s 101
1781          51 s1787     x < 0
1782          01 s 105
1783          11 s1785     x > 0
1784          50 s1788     x = 0
1785    s1783 60 s 100
1786          10 s1788
1787    s1781 61 s 100
1788    s1784
        s1786 00 s 101
1789          08     0,B
1790          10 s 655

sqrt (section 11.5.4.1).

1791          55 s   1,D
1792          16 s 616     Take value of parameter
1793          00 s 101
1794          51 s  63     x < 0, alarm
1795          18     0,B   x1
1796          58    15     128 - x2
1797          43     0,B
1798          11 s1827     x = 0
1799          40     0,B
1800          0E    17
1801          08     0,B   xx2↑p
1802          40    17
1803          0D s   8
```

```
1804          00 s 111
1805          00    15
1806          4D s   1
1807          08    15     128 - y2 = entier(128 + (p - x2)/2)
1808          60 s 113
1809          08    10     y1:= 0.5
1810          4C s  29
1811          29    45-    2×((p - x2)/2 - entier((p - x2)/2) = s
1812          40     0,B
1813          37    45-    x1:= x1×2↑(-s)
1814          08     0,B
1815  s1821   0D s   1     AR:= d/2
1816          06    10-    y1:= y1 + d/2
1817          40     0,B
1818          0B    10
1819          07 s   0
1820          01    10     d:= - (y1 - x1/y1)
1821          51 s1815     d < 0
1822          40    10-
1823          18     0,B
1824          40    15-
1825          06     0,B
1826          10 s 655     exit
1827  s1798                x = 0
              40 s 101-
1828          08     0,B
1829          10 s 655

sin (section 11.5.4.1).

1830          55 s   1,D
1831          16 s 616     Take value of parameter
1832          00 s 101-
1833          08     0,B   x1
1834          58    15     128 - x2
1835          05    15     MR:= x1
1836          61    15
1837          20 s1860
1838          51 s 655     x2 < -10, sin(x) = x, exit
1839          0C s   8
1840          29    46     x2 + 10
1841          4A s1886
1842          4D s  10
1843          37    46     4C (x2 + 10), AR:= 2×x/pi - entier(2×x/pi)
```

sin and cos are calculated by the same method as the one used in DASK - BIBLIOTEKSSEKVENS TF 1. However, since only 9 decimals are required the polynomial has been cut down to 6 terms by the usual economisation process.

```
1844  s1903                 cos
            18     0,B      t:= 2y
1845        52 s1850        abs(y) < 0.5 (1st and 4th quadrant)
1846        41     0,B
1847        08     0,B      t:= -2y - 2
1848        01 s 111
1849        12 s1871        t = 1
1850  s1845 44     0,B
1851        0A     0,B
1852        21 s 113
1853        08    10        w:= t/2 - 0.5
1854        55 s  10        -
1855        44 s1874,C      |
1856  s1859 55 s2046,C      | Polynomial evaluation
1857        0A    10        |
1858        04 s1874,C      |
1859        53 s1856        -
1860  s1837 0A     0,B
1861        00     0,B
1862        0E    17
1863        18     0,B
1864        43     0,B
1865        11 s1868        Result = 0
1866        40    17
1867        0D s   8
1868  s1865
      s2010                 ln(x) = 0
            00 s 111
1869        06     0,B
1870        10 s 655        Exit
1871  s1849
      s1905                 cos
            08     0,B      Result:= 0.999999999x2/0
1872        10 s 655        Exit
1873        13 s   0
```

Coefficients for sin and cos.

```
1874  s1855,C
      s1858,C
            + .267 162 131 344
1876        - .569 703 680 149
1878        + .072 906 209 920
1880        - .004 369 731 387
1882        + .000 151 656 333
1884        - .000 003 418 229
1886  s1841
      s1898
            + 636 619 772 367   2/pi
```

cos (section 11.5.4.1).

```
1888          55 s   1,D
1889          16 s 616     Take value of parameter
1890          00 s 101
1891          58    15     128 - x2
1892          05    15
1893          61    15
1894          20 s 124
1895          51 s1904     x2 < -17
1896          0C s   8
1897  s 702,D
              29    46     x2 + 17
1898          4A s1886
1899          4D s  18
1900          37    46     4C x2 + 17
1901          20 s 113
1902          0C s   1
1903          10 s1844
1904  s1895   40 s 100
1905          10 s1871
```

arctan (section 11.5.4.1).

```
1906          55 s   1,D
1907          16 s 616     Take value of parameter
1908          00 s 101
1909          08     0,B   x1
1910          58    15     128 - x2
1911          01    15
1912          0E    17
1913          08    10     x1:= x1×2↑p
1914          60    15
1915          21 s 111
1916          0C s   8
1917          26    17     p:= p - x2
1918          11 s1966     abs(x) < 1
1919          20 s 145
1920          51 s1972     p < -31
1921          61 s 113     -
1922          0B    10      |
1923          07 s   0        x:= 1/x
1924          08    10      |
1925          60 s 105
1926          26    17     -
1927          41 s1990
1928  s1968   00 s1988
1929          08     6
1930          40    10
1931          11 s1933     x > 0
1932          75 s   0
1933  s1931   62    17
1934          29    45     p
1935          43    10
1936          37    45     4D p
1937          08    10     - abs(x)
```

arctan is calculated by the method given in Lance: Numerical Methods, pag. 41. -z is calculated as follows:
N:= (- (sqrt(2) - 1)/2)x(-x) + 0.5
T:= N + 1 - (-x)
-z:= T/N.

```
1938          44    10
1939          0A s 544
1940          20 s 113
1941          08     0,B  N
1942          20 s 104
1943          01    10
1944          0B     0,B
1945          07 s   0
1946          08    10    z
1947          0A    10
1948          08     0,B  z↑2
1949          55 s  10
1950          44 s1976,C  -
1951  s1954   55 s2046,C  | Polynomial calculation
1952          0A     0,B
1953          04 s1976,C  |
1954          53 s1951    -
1955          0A    10
1956          06     6
1957          73 s1959    positive
1958          41     6
1959  s1957   0E    17
1960          18     0,B
1961          40    17
1962          0D s   8
1963  s1971   00 s 117
1964          06     0,B
1965          10 s 655    Exit
1966  s1918   21 s 128
1967          11 s 655    abs(x) < 2↑(-11), exit
1968          50 s1928
1969  s1973   40 s1990
1970  s1975   18     0,B
1971          50 s1963
1972  s1920                abs(x) > 2↑30
              40    10
1973          11 s1969    x > 0
1974          41 s1990
1975          10 s1970
```

```
arctancoeff.

1976  s1950,C
      s1953,C
            + .207 106 780 491
1978        - .011 844 612 778
1980        + .001 219 146 913
1982        - .000 148 781 957
1984        + .000 018 803 703
1986        - .000 001 795 498
1988  s1928
            B 1921F  B B5444  pi/16
1990  s1927
      s1969
      s1974
            B 6487E  B D5111  pi/4

ln (section 11.8.4.1).

1992        55 s   1,D
1993        16 s 616     Take value of parameter
1994        00 s 101
1995        58    13     128 - x2
1996        01    13
1997        0E    17
1998        08     0,B   x1x2^p
1999        40    17
2000        0D s   8
2001        01 s 111
2002        06    13     128 - x2 + p
2003        41     0,B
2004        11 s  59     argument ≤ 0
2005        16 s 454
2006  s 477             - log base 2(x) in AR and MR
2006        0A s2018
2007        0E    13
2008  s1696 18     0,B
2009        43     0,B
2010        11 s1868     Result = 0
2011        60    13
2012        20    17
2013        0D s   8
2014        20 s 130
2015        26     1,B
2016        10 s 655     Exit
2017        13 s   0
2018  s2006             B A746F  B 00417  - log nat(2)
```

gälb
sulgu Fröberg

exp (section 11.5.4.1).

```
2020          55 s   1,D
2021          16 s 616     Take value of parameter
2022          75 s  20
2023          00 s 101
2024          18     0,B   x1
2025          58    15     128 - x2
2026          44     0,B
2027          4A s2044
2028          4E    17
2029          08     0,B   - x1/(2xln(2))
2030          60    15
2031          21 s 132
2032          0C s   8
2033          20    17
2034          51 s2041     x2 ≥ 39
2035          29    45
2036          21 s 143
2037          11 s 487     x2 ≤ 31
2038          68    12     sign:= plus
2039          40     0,B
2040          10 s 491
2041    s2034 40     0,B
2042          11 s 324     argument is large, negative
2043          10 s  59          -    -    -  , positive, alarm
2044    s2027
              B A3AAE  B26B52  1/(2xln(2))
2046    s 250
        s 259
        s1771 00 s1024
2047          01 s   0
```

Locations in the core store used by the wired program.

```
   0          00     0
   1          00     0
   2    s 162
        s 183
        s 867
        s 957
        s 987
        s 999
        s1064
        s1066
        s1102
        s1184
        s1200
        s1206
        s1426
        s1431
        s1444
        s1565
        s1569
        s1631
        s1670              General working location
```

3	s 163			
	s 164			
	s 168			
	s 170			
	s 177			
	s 179			
	s 181			
	s 185			
	s1042			
	s1052			
	s1067			
	s1098			
	s1182			
	s1187			
	s1236			
	s1441			
	s1567			
	s1668			General working location
4	s 34			
	s 45			
	s1052	17	257	Used as instruction by <u>for</u> real (entry from 3).
5	s 40	17	361	- - - - - - - -
6	s 158			
	s 182			
	s 765			
	s 767			
	s 798			
	s 807			
	s 815			
	s 819			
	s 887			
	s 903			
	s 905			
	s 949			
	s 971			
	s 984			
	s1006			
	s1048			
	s1216			
	s1420			
	s1421			
	s1451			
	s1453			
	s1525			
	s1551			
	s1554			
	s1584			
	s1664			
	s1929			
	s1956			
	s1958			General working location

```
 7  s 820
    s 821
    s 909
    s 951
    s 963
    s 966
    s1017
    s1020
    s1040
    s1041
    s1115
    s1123
    s1258
    s1263
    s1267
    s1269
    s1273
    s1312
    s1320
    s1328
    s1331
    s1348
    s1382
    s1398
    s1503
    s1508
    s1512
    s1515
    s1536
    s1663
    s1739
    s1741                 General working location
 8  s  36 17   264        Used as instruction by for real
 9        17  1053          -  -       -     -  -  -
10  s 209
    s 213
    s 223
    s 225
    s 229
    s 232
    s 240
    s 372
    s 373
    s 411
    s 412
    s 419
    s 428
    s 457
    s 460
    s 462
    s 463
    s 471
    s 499
```

```
      s 500
      s 507
      s 508
      s 510
      s 511
      s 872
      s 876
      s 894
      s 906
      s 935
      s 939
      s 961
      s 975
      s1108
      s1128
      s1138
      s1218
      s1476
      s1523
      s1531
      s1809
      s1816
      s1818
      s1820
      s1822
      s1853
      s1857
      s1913
      s1922
      s1924
      s1930
      s1935
      s1937
      s1938
      s1943
      s1946
      s1947
      s1955
      s1972          General working location
   11 s 207
      s 215
      s 227
      s 228
      s 230
      s 875
      s 891
      s 918
      s 944
      s1090
      s1097
      s1178
      s1226
      s1228
      s1365
      s1366          General working location
```

12	s 450	
	s 514	
	s 870	
	s 910	
	s 911	
	s 922	
	s 923	
	s 968	
	s1109	
	s1117,B	
	s1120,B	
	s1122,B	
	s1124,B	
	s1126,B	
	s1164	
	s1214	
	s1360	
	s1370	
	s2038	General working location
13	s 270	
	s 276	
	s 280	
	s 292	
	s 296	
	s 307	
	s 309	
	s 319	
	s 378	
	s 379	
	s 387	
	s 426	
	s 427	
	s 433	
	s 473	
	s 575	
	s 576	
	s 880	
	s 890	
	s 913	
	s 969	
	s 983	
	s1240	
	s1242	
	s1995	
	s1996	
	s2002	
	s2007	
	s2011	General working location

14	s 874	
	s 893	
	s 916	
	s1113	
	s1116	
	s1141	
	s1160	
	s1167	
	s1169	
	s1217	
	s1227	
	s1430	
	s1456	
	s1458	
	s1483	
	s1490	
	s1507	General working location
15	s 273	
	s 275	
	s 298	
	s 302	
	s 308	
	s 320	
	s 332	
	s 334	
	s 342	
	s 352	
	s 354	
	s 358	
	s 370	
	s 392	
	s 406	
	s 435	
	s 479	
	s 496	
	s 520	
	s 904	
	s 932	
	s 945	
	s1107	
	s1292	
	s1341	
	s1485	
	s1486	
	s1506	
	s1537	
	s1796	
	s1805	
	s1807	
	s1824	
	s1834	
	s1835	
	s1836	
	s1891	

	s1892			
	s1893			
	s1910			
	s1911			
	s1914			
	s2025			
	s2030			General working location
16	s 730			
	s 731			
	s 786			
	s 787			
	s 791			
	s1074			
	s1077			
	s1080			
	s1188			
	s1207			
	s1492			
	s1494			
	s1501			
	s1511	00	1	Working location in pos. 1 - 11. Pos. 0 and 12 - 19 = 0
17	s 346			
	s 350			
	s 380			
	s 384			
	s 442			
	s 444			
	s 475			
	s 482			
	s 728			
	s 739			
	s 769			
	s1134			
	s1137			
	s1215			
	s1219			
	s1496			
	s1500			
	s1800			
	s1802			
	s1862			
	s1866			
	s1912			
	s1917			
	s1926			
	s1933			
	s1959			
	s1961			
	s1997			
	s1999			
	s2012			
	s2028			
	s2033	00		Working location in pos. 0 - 11. Pos. 12 - 19 = 0

```
18  s1617
    s1619
    s1620 13     0     Output case for medium not in use (section
                       11.8.3): no case: 13, upper: 11 20, lower: 51 20.
19        00     0     Output sum for medium not in use (section
                       11.8.3).
20  s1682
    s1684
    s1686
    s1688 17     1,D
21  s2022              Entry from s522 in case of exp
          17   655
22  s1285
    s1286
    s1290
    s1291
    s1598 75     0     Working location in pos. 1 - 11. Pos. 0 = 0
23  s 399
    s 401
    s1288 0C     0     Working location in pos. 1 - 11. Pos. 0 = 0
24  s 653 2A     0
25        10    27     Constant used for take value
26        16     0     Working location in pos. 1 - 11. Pos. 0 = 0
27     25
    s  25
    s 189
    s 192
    s 197
    s 198
    s 210
    s 626
    s 631
    s 633
    s 647
    s 648
    s 649
    s 651
    s1088
    s1092
    s1201
    s1213
    s1695
    s1698
    s1710 00     0     General working location
28  s 652 10     2,B   Return via stack after take value
29  s 597
    s 743
    s 747
    s 770
    s 781
    s 788
    s 793 55  2046,B   55 (2048 - arrow),B (section 11.8.1)
```

```
30  s 818
    s 823
    s 898
    s 943
    s1192
    s1203
    s1232
    s1239
    s1428
    s1677 35      0     Working location in pos. 1 - 11. Pos. 0 = 0
31  s 714
    s 716
    s 814
    s1233
    s1337
    s1355
    s1427
    s1678 55      0     Working location in pos. 1 - 11. Pos. 0 = 0
32  s 659
    s 670
    s 693
    s 754
    s 810
    s 813
    s 836
    s 846
    s 981
    s1027
    s1031
    s1033
    s1060
    s1062
    s1196
    s1679
    s1744 10     1,C    Return jump
 33 s 615
    s 634
    s 635
    s 642
    s 643
    s 710
    s 719 10     27     During take name: 00 24
 34 s1424
    s1459
    s1462
    s1468
    s1475
    s1489
    s1495
    s1499
    s1504
    s1514
    s1662 0D      0     Working location in pos. 1 - 11. Pos. 0 = 0
```

35	s1434			
	s1445			
	s1541			
	s1550			
	s1556			
	s1572			
	s1586			
	s1657	00	0	Working location in pos. 0 - 11
36	s1035			for real
	s1037			
	s1038	75	0	Switch used by for real
37	s 942			
	s 977			
	s 978			
	s 982			
	s1007			
	s1105			
	s1106			
	s1185			
	s1433			
	s1480			
	s1487			
	s1539			
	s1540			
	s1652	35	0	Working location in pos. 1 - 11. Pos. 0 = 0
38	s1262			
	s1315			
	s1329	79	0	The input instruction
39	s 431			
	s 432			
	s 864			
	s 869			
	s 979			
	s1101			
	s1190			
	s1437			
	s1482			
	s1538			
	s1558			
	s1653	55	0	Working location in pos. 1 - 11. Pos. 0 = 0
40	s1440			
	s1488			
	s1491			
	s1649			
	s1656	55	0	Working location in pos. 1 - 11. Pos. 0 = 0
41	s 154			
	s 155			
	s 204			
	s 218			
	s 774			
	s 777			
	s1645			
	s1648			
	s1674	75	0	Working location in pos. 1 - 11. Pos. 0 = 0

```
42 s 200
   s 202
   s 451
   s 478
   s1449
   s1497
   s1518
   s1651 75     0    Working location in pos. 1 - 11. Pos. 0 = 0
43 s 839
   s 843
   s1481
   s1509
   s1520
   s1627
   s1636 75     0    Working location in pos. 0 - 11
44 s 959             2↑(-20)x(address of first free drum track)
                     (section 11.8.1)
45 s 282
   s 286
   s 311
   s 315
   s 408
   s 410
   s 484
   s 491
   s 580
   s 584
   s 586
   s1811
   s1813
   s1934
   s1936
   s2035 4D     0    Working location in pos. 1 - 11. Pos. 0 = 0
46 s1840
   s1843
   s1897
   s1900 4C     0    Working location in pos. 1 - 11. Pos. 0 = 0
47 s 627
   s 628
   s 840
   s 841 62     0    Working location in pos. 1 - 11. Pos. 0 = 0
48 s 737
   s 751
   s 799
   s 801
   s 816
   s 865
   s 974 35          35 SR (stack reference, section 11.8.1)
49 s 740
   s 742
   s 746
   s 782
   s 794 00          Current store limit (section 11.8.1)
```

```
50 s1339
   s1351
   s1353
   s1359                Input string (section 11.8.2)
52 s 749
   s 773,C
   s 803
   s 805 00             2↑(-11)xdrum procedure depth (section 11.8.1)
53 s 764,C
   s 766,C              2↑(-19)xdrum procedure block number
                        (section 11.8.1)
54 s 722
   s 735
   s 750
   s 764,C
   s 766,C
   s 802                2↑(-19)xdrum block number (section 11.8.1)
55 s 952
   s 958                2↑(-20)xaddress of last drum track used
56 s1372
   s1412
   s1417
   s1608
   s1609
   s1691
   s1733
   s1736 00     1       Medium changer parameter (section 11.8.3)
57 s1270
   s1272
   s1325
   s1335                Input sum (section 11.8.2)
58 s1143
   s1261
   s1283
   s1346
   s1383
   s1384
   s1396                Input case: upper: 00, lower: 40 (section 11.8.2)
59 s1275
   s1276 20     0       Working location in pos. 0 - 11
60 s1610
   s1615                Current output medium: tryk: 51 20, skrv: 11 20
                        (section 11.8.3)
                        Current output instruction: tryk: 7A, skrv: 5A
61 s1595
62 s1601
   s1606
   s1616
   s1618                Output case for medium in use (section 11.8.3):
                        no case: 13, upper: 11 20, lower: 51 20
63 s1596
   s1718
   s1721
   s1724                Output sum for medium in use
```

The following constants will be used by the running ALGOL program:

64	00	0	zero and false
65	6D	0	....
66	00	,C	true
67	01		
68	00	2	
69	00	64	
70	00	0	2↑(-20)×(address of first track of identifier list (section 12)
71	00	11	