Description of the object program (a sequel to EWD102)

Obsolete are EWD101, page 5, top half of page 6, last paragraph of page 7 and first paragraph page 8. (Complex assignment.) From page 14, the lines SE7, SE8 and SE9 should be deleted.

Exhaustive list of the formal locations

Formal locations occupy four consecutive words in the stack, designated by f[0], f[1], f[2] and f[3] (in order of increasing address). The word f[1] will contain the so-called parameter code, a characteristic bit pattern, specifying the nature of the actual parameter. Layout of the parameter code will be decided later.

1. A real constant

f[0] F = MS[2]
f[1] parameter code
f[2]] constant value in
f[3]] floating point representation.

2. An integer constant

f[0] G = MS[3]
f[1] parameter code
f[2] unused
f[3] integer value.

3. A real in the stack (< 32K)

f[0] F = M[a]
f[1] parameter code
f[2] M[a] = F
f[3] unused.

4. An integer in the stack (< 32K)

f[0] G = M[a]
f[1] parameter code
f[2] M[a] = G
f[3] unused.

5. Pro forma: A real in the stack above 32 K

f[0] SE36 High stack real
f[1] parameter code
f[2] a
f[3] unused.

6. Pro forma: An integer in the stack above 32K

f[0] SE37 High stack integer
f[1] parameter code
f[2] a
f[3] unused.

7. An integer array

f[0] array word
f[1] parameter code
f[2] SE38 Left subscripted integer
f[3] SE39 Right subscripted integer.

8. A real array

f[0] array word
f[1] parameter code
f[2] SE40 Left subscripted real
f[3] SE42 Right subscripted real.

9. A complete array

f[0] array word
f[1] parameter code
f[2] SE42 Left subscripted ** complex
f[3] SE43 Right subscripted complex.

10. A procedure or switch

f[0] unused
f[1] parameter code
f[2] invariant starting address
f[3] context D.

Layout 10 will be used when the actual parameter is a procedure identifier. The parameter code distinguishes between non-type, integer, real, complex, boolean or switch (= label procedure).

11. An arithmetic procedure as expression

f[0] SE44 Arithmetic procedure as expression
f[1] parameter code
f[2] invariant starting address
f[3] context D.

The above formal location contents are never generated directly; it is created by SE20 (Check formal arithmetic) and SE30 (Check formal complex) when the actual parameter is approcedure of arithmetic type. The parameter code will distinguish between arithmetic and complex.

12. A complex procedure as expression

f[0] SE45 Complex procedure as expression
f[1] parameter code
f[2] invariant starting address
f[3] context D.

13. Non simple implicit subroutine

f[0] SE46 Non simple implicit subroutine
f[1] parameter code
f[2] invariant starting address
f[3] context D.

These formal locations are derived from actual parameters which are
a) a subscripted variable (integer, real boolean or complex)
b) a arithmetic, complex or boolean expression, more trivial than a single identifier or constant
c) a designational expression, more complicated than a simple label.

The parameter code will distinguish between these cases.

14. Simple implicit subroutine

f[0] SE47
f[1] parameter code
f[2] invariant starting address
f[3] context D.

Parameter code as above.

15. Label

f[0] SE48 Actual label value
f[1] parameter code
f[2] invariant target address
f[3] target LRP.

16. A complex in the stack

f[0] SE55 Stack complex
f[1] parameter code
f[2] a
f[3] unused.

17. A complex array in arithmetic specification

When formal locations of type 9 are met by the system entry SE22 "Check formal arithmetic array" the right hand value is destroyed giving:

f[0] array word
f[1] parameter code
f[2] SE42 Left subscripted complex
f[3] SE56 Alarm arithmetic value of complex array.

Possible structures of left hand values

(Inppictures stack upside down.)

Left hand value for an integer

on address a < 32K:	on address a ⩾ 32K:	on array segment:
M[B-1]: M[a] = G	M[B-2]: a	invariant address
B →:	M[B-1]: SE49	M[B-1]: SE50
	B →	B → .....

Remark. Number of words occupied by invariant address still undecided.

SE49 Assign to high stacked integer
SE50 Assign to integer on segment.

Left hand value for a real

on address a < 32K:	on address a ⩾ 32K:	on array segment:
M[B-1]: M[a] = F	M[B-2]: a	invariant address
B →	M[B-1]: SE51	M[B-1]: SE52
	B →	B → .....

SE51 Assign to high stacked real
SE52 Assign to real on segment.

Left hand value for a complex

on address a:	on array segment:
M[B-2]: a	invariant address
M[B-1]: SE53	M[B-1]: SE54
B →	B → .....

SE53 Assign to stcked complex
SE54 Assign to complex on segment.

Creation of the left hand values described

Indexing a non formal array

After the indexing routine has been called the object code continues with

SE 38 Left subscripted integer or

SE 40 Left subscripted real or

SE 42 Left subscripted complex,

depending on the type declaration of the array (which is known to the translator).

Depending on the outcome of the indexing process, SE38 creates one of the three left hand values for an integer. The indexing process delivers location information only (i.e. physical or invariant address), but not the type of the array: this is known to the translator and brought to effect by calling the appropriate system entry. For the moment we assume that the indexing process leaves its outcome in some of the registers.

Indexing a formal array

After the indexing routine has been called, the object code continues with

DO(f[2])

refering again to the formal location of the array offered.

Processing of the left hand values described

The processing of a left hand value may only be given in the object program when the right hand value offered is of the correct type.

The assignment itself will be commanded by the instruction

DO(MC[-1])

and the stack will be shortened in accordance with the extent of the left hand value processed.

In the case of an integer assignment the integer value must be stored in G, the assignment itself will leave this value unchanged.

In the case of a real assignment the real value must be stored in F, where it must remain unchanged.

In the case of a complex assignment the real part of the complex value will stand in the F-register, the imaginary part in two universal locations of the program. The assignment will leave this complex value unchanged.

Creation of the right hand values due to subscripting

In the case of a right hand value required from an arithmetic array the call of the indexing routine will be followed by SE39 or SE41, Right subscripted integer or real, respectively.
It will deliver the value in the F-register.

When subscripting a complex array, the indexing call will be followed by

```
B + 2
SE 43
```

In the two places left blank will be filled by the imaginary part. The real part will be handed over in the F-register.

The creation of a right hand value of a formal array

If the specification of the array is arithmetic,

```
DO(f[3])
```

referring to the formal locations, will put the value required in F.

If the specification of the array is complex, the indexing sequence will be followed by

```
F = 0
MC[0] = F
DO(f[3])
```

The zero placed on top of the stack will be overwritten by the imaginary part of the value delivered, when the actual array is indeed a complex one. The real part will be delivered in F.

The universal locations FLV and FRV

Per program two universal locations FLV and FRV (Formal Left/Right hand Value) play a role in the information transmission from actual parameters, when the formal parameter has been specified arithmetic or complex.

These locations have standard values in order to speed up the transmission in the case of simple arithmetic actuals. The standard values are:

FRV : "U, S = 0"

or any faster skip instruction available, and

FLV : SE57 "Standard value for FLV".

The body of SE57 will be:

```
G = MS[+2]        take instruction to be stacked
S = M[B-1]        save link
M[B-1] = G        stack left hand value
GOTOR(S) ⇒        return.
```

To get the left hand value of a formal parameter specified arithmetic or complex, the body will eventually invoke

```
DOS(f[0])
DO (FLV).
```

To get the right hand value of a formal parameter specified arithmetic the body will contain

```
DOS(f[0])
DO (FRV),
```

which pair will put the value required in the F-register.

The Universal location FCV

Per program there is a third universal location, called FCV (Formal Complex Value).

To get the right hand value of a parameter specified to be complex, the body will contain

```
   F = 0
MC[0] = F
DOS(f[0])
DO(FCV)
```

After return x the imaginary part of the value will be on top of the stack, the real part in the F-register.

Also FCV has a standard value, viz.

```
FCV : DO(FRV).
```

All actual parameters of type arithmetic leave FCV unchanged. When the actual value is asked for by the above mechanism, it will ensure that the complex value will be properly delivered.

Advanced use of the FRV, FLV and FCV

As long as actual parameters specify arithmetic quantities, FCV will keep its standard value, FCV will first be left out of discussion.

Implicit subroutines may end by changing FRV and FLV; if so, they will be filled by system entries which, upon execution return FRV and FLV to their standard values.

The implicit subroutine, describing a subscripted variable

First we shall discuss the case of the non-formal array. After the call of the indexing routine, the result of this proves may be stored in the registers.

The idea is, that the choice of a left or right hand value will be postponed until return to the body has been completed. It will do so by filling in FLV and FRV in the appropriate way. At the same time,

we can choose some other universal locations, the Universal Address UA to transmit the outcome of the indexing.

If the array has been declared integer, the implicit subroutine will fill

FLV with SE58 "Formal Left Subscripted Integer" and
FLR with SE59 "Formal Right Subscripted Integer".

These system entries start by resetting FLV and FRV to their standard values, by picking up from the UA locations the quantities, as delivered by the indexing process and from then onwards they merge with SE38 and SE39 respectively.

If the array has been declared real, the implicit subroutine will fill

FLV with SE60 "Formal Left Subscripted Real" and
FRV with SE61 "Formal Right Subscripted Real"

which, after the same introduction merge with SE40 and SE41 respectively.

If the array has been declared complex, the implicit subroutine will fill

FLV with SE62 "Formal Left Subscripted Complex"
FRV with SE 64 "Alarm complex value in arithmetic"
FCV with SE63 "Formal Right Subscripted Complex"

SE62 and SE63 will reset all three to their standard values, will pick up the UA-contents and will merge with SE42 and SE43 respectively.

If the array is formal, from the three transmission mechanism one has to be chosen on account of the type of the actual array, as can be derived from its parameter code.

The above can be condensed as follows.
After indexing an declared integer, real or complex array, there follows one of the following system entries.

SE65 "Transmit subscripted integer"

SE66 "Transmit subscripted real"

SE67 "Transmit subscripted complex".

They will save the outcome of the indexing process in the UA-locations and will FLV, FRV (and the last also FCV) in the appropriate way.

After indexing a formal array, S can be saved in a UA-location, the instruction

S = f[1].

puts the parameter code of the array in the S-register and the following system entry

SE68 "Transmit subscripted formal"

will effectively select one of the previous three on account of the bit pattern in the S-register.

We code SE55 "stack complex".

```
SE55:     FLV:=  "SE 55.1"      ]
          FRV:= ""SE64          ]   transports via F(G)
          FCV:=  "SE 55.2       ]
            S =  MS[+2]           take address a in S
          GOTOR (MC[-1]) ⟹
```

```
SE 55.1:  FLV:=  "SE57"         ]   reset to standard
          FRV:=  "U, S = 0"     ]   values via F(and G).
          FCV:=  "DO(FRV)"      ]
            F =  MS[0]          ]   transportation im. part
        M[B-3] =  F             ]
            F =  MS[2]            real part
          GOTOR (MC[-1]) ⟹
```

```
SE 55.2:  FLV:=  "SE57"         ]   reset to standard
          FRV:=  "U, S = 0"     ]   values via F(G)
          FCV:=  "DO(FRV)"      ]
            G =  M[B-1]           save link
        M[B-1] =  S               store address a
            S =  "SE53"
       MC[0] =  S                 complete left hand value
          GOTOR(G)          ⟹
```

Cumulative list of system entries in EWD105

SE 36	High stack real
SE 37	High stack integer
SE 38	Left subscripted integer
SE 39	Right subscripted integer
SE 40	Left subscripted real
SE 41	Right subscripted real
SE 42	Left subscripted complex
SE 43	Right subscripted complex
SE 44	Arithmetic procedure as expression
SE 45	Complex procedure as expression
SE 46	Non simple implicit subroutine
SE 47	Simple implicit subroutine
SE 48	Actual label value
SE 49	Assign to high stacked integer
SE 50	Assign to integer on segment
SE 51	Assign to high stacked real
SE 52	Assign to real on segment
SE 53	Assign to stacked complex
SE 54	Assign to complex on segment
SE 55	Stack complex
SE 56	Alarm "arithmetic value of complex array"
SE 57	Standard Value for FLV
SE 58	Formal left subscripted integer
SE 59	Formal right subscripted integer
SE 60	Formal left subscripted real
SE 61	Formal right subscripted real
SE 62	Formal left subscripted complex
SE 63	Formal right subscripted complex
SE 64	Alarm "complex value in arithmetic"
SE 65	Transmit subscripted integer
SE 66	Transmit subscripted real
SE 67	Transmit subscripted complex
SE 68	Transmit subscripted formal.