RCSL : 44-RT 820
Author : J. Bloch-Petersen
Edited : 25.4.1974

MUSIL CODEPROCEDURES.

Adition to: MUSIL-manual RCSL :44-RT 740.
MUS-manuals RCSL 44-RT 759.
MUSIL Compiler operating guide RCSL 44-RT 766.

Keywords: RC 3600 MUS System Software, Programming Languages.

Abstract: Syntax Rules for MUSIL-procedures coded in assembler language, compilation, error description and example.

Codeprocedures.

In order to use procedures coded in assembler language in connection with musil programs, the user is assumed to be familiar with the general principles of the MUS-system as well as MUS-programmers guide RCSL 44-RT 759.

Declaration.

The following adition is made to the MUSIL manual 44-RT 740.

<procedure declaration> :: = <procedure heading > <statement part> |
<codeprocedure heading> codebody ;

The codeprocedure heading gives the identification of the assembler coded procedure. .TITL <identifier> should be used.
The formal parameter sections specify kind and type of valid parameters in call.

NB.: No check is made that the assembler procedure uses these correctly.
NB.: A maximum of 5 parameters are allowed.

<codeprocedure heading> :: = procedure <identifier> ; |
procedure <identifier>
(<formal parameter section>
{ ; <formal parameter section> } *) ;

<formal parameter section> :: = <parameter group> |
const <parameter group>
{ ; <parameter group } * |

var <parameter group>
{ ; <parameter group> } * |

file <file group>

<parameter group> :: = <identifier> { , <identifier> } *
: <type identifier>

<file group> :: = <identifier> { , <identifier> } *

A parameter group without preceding specifier implies const parameters.

Conventions for assembler coded part.

The code should be assembled using ASMUS 8 RCSL 43-GL 98.
The following initial statements should be included:

```
.TXTM 1        ;    packing left to right
.RDX 10        ;    decimal
.NREL          ;    no .ZREL code allowed
```

It is advicable to use labels of the form:

CPXXX XXX any numeric or alfabetic

in order to avoid conflict with ASMUS definitions.

Environment.

The following definitions may be used to get access to interpreter variables and constants. In process description:

```
save 2 = save 1 + 1     ;    workcell
save 3 = save 2 + 1     ;      -
save 4 = save 3 + 1     ;      -
save 5 = save 4 + 1     ;      -
     R = save 5 + 1     ;    result register (contain values of expressions)
    pc = R + 1          ;    program counter (at entry it points to first
                        ;    parameter word. At exit it should point to
                        ;    the word after last parameter).
    op = pc + 1         ;    operator message or 0 if no opin pending.
 .oper = op + 1         ;    A block of 3 words containing name of ope-
                        ;    rator driver.
    zn = .oper + 4      ;    addresses of zone descriptors of program
```

The following interpreter procedures may be used:

```
; procedure return ;
; exits to interpreter from codeprocedure
;                call:
; ac 0           irr.
; ac 1           irr.
; ac 2           cur
; ac 3           irr.
; call
;                jmp@   MZSTART + 1
```

```
; procedure takeaddress (var bits: integer ; address: integer) ;
; takes the address of a var or file parameter; the address is either a byte ad-
; dress for a string, or a word address for an  integer or file. (The file address
; points to the zone descriptor of the file)
;                       call:     return:
; ac 0                  bits      bits shift (2)
; ac 1                   -        address
; ac 2                  cur       cur
; ac 3                  link      destroyed
; call
;                       jsr @     MZSTART + 2

; procedure takevalue (var bits, value: integer) ;
; takes the value of a const integer parameter.
;                       call:     return:
; ac 0                  bits      bits shift (2)
; ac 1                   -        value
; ac 2                  cur       cur
; ac 3                  link      destroyed
; call
;                       jsr @     MZSTART + 3
```

Entry to procedure:

When the procedure is called, entry takes place to first location with

ac 0: bits
ac 1: irr
ac 2: cur
ac 3: irr

pc points to first parameter.

Procedure statements.

No change.

Parameters.

FILE		bits = 00 word (pc + param) = abs. address of zone descriptor
VAR	STRING	bits = 01 word (pc + param) = byteaddress bits = 11 word (pc + param) = zone number displacement (final address zn (zone number) .zfirst + displacement)
	INTEGER	bits = 10 word (pc + param) = abs. address of value
CONST	INTEGER	bits = 00 word (pc + param) = constant value bits = 10 word (pc + param) = abs. address of value bits = 11 R.cur = value Note: no word (pc + param) is present.

Compilation.

When the MUSIL program is correctly compiled the message :

LOAD "PROCNAME"

will be printet on operator device, which means that the rel. binary codeprocedure will be loaded from ready input device by striking NL.
If there are more codeprocedures this will be repeated.

Error description:

END ERROR	appears if a relocatable block contains no proper start code.
SUM ERROR	appears if a relocatable block contains a checksum error.
ILL ERROR	appears of wordcount in a relocatable block is greater than 16 or one of the relocation codes are greater than 3.
LOAD "PROCNAME"	repeating of Load text indicates, that no proper titleblock is met. E.g. No titleblock or wrong title. This error is always non. fatal.
FATAL	If any relocatable blocks were outputtet before the detected error, the error message is followed by the text "FATAL", which means that the final relocatable output is not proper.

After errormessages the Load text will be repeated, and the user can start again.
If the user wants to stop loading he can set input device local and strike NL.

```
MUSIL COMPILER

0000            ! CODEPROCEDURE  EXAMPLE  !
0001  CONST
0002  PIP=      2'10001000,0000,0000,
0003  POP=      7,
0004  TXT=      "<10>WHAT CODECALL DO YOU WANT? ",
0005  TEXT=     "<10>CODEBODY TEST",
0006  STWO=     "TWO",
0007  SONE=     "ONE ",
0008  STEST=    "TEST",
0009  CL=       "CLEAR",
0010  TWOTXT=   "CALL TWO: ",
0011  ONETXT=   "CALL ONE: ",
0012  RETURNED= " NOW RETURNED<10>",
0013  OUTS=     "THIS IS *TEST*, PRESS A BUTTON<0>",
0014  TESTTXT=  "CALL TEST: ";
0015
0016  VAR
0017  OPSTRING:          STRING(40);
0018  OPLENGTH:          INTEGER;
0019  TTY:      FILE 'TTY',1,1,40,06
0020            OF STRING(40);
0021
0022  PROCEDURE  ONE;
0023  CODEBODY;
0024
0025  PROCEDURE  TWO(VAR  STRG: STRING(1));
0026  CODEBODY;
0027
0028  PROCEDURE  CLEAR;
0029  CODEBODY;
0030
0031  PROCEDURE TEST(CONST PIP,POP: INTEGER;
0032                 VAR STRG: STRING(1);
0033                 FILE OUT);
0034  CODEBODY;
0035
0036  BEGIN
0037  OPRESS(TEXT);
0038  0: OPRESS(TXT); OPINT(OPSTRING); OPWAIT(OPLENGTH);
0039     IF OPSTRING = SONE THEN
0040     BEGIN
0041       OPRESS(ONETXT);
0042       ONE;
0043       OPRESS(RETURNED);
0044     END;
0045     IF OPSTRING = STWO THEN
0046     BEGIN
0047       OPRESS(TWOTXT);
0048       TWO(OPSTRING);
0049       OPRESS(OPSTRING);
0050       OPRESS(RETURNED);
0051     END;
0052     IF OPSTRING = STEST THEN
0053     BEGIN
0054       OPSTRING := OUTS;
0055       OPRESS(TESTTXT);
0056       TEST(PIP,POP,OPSTRING,TTY);
0057       OPRESS(RETURNED);
0058     END;
0059     IF OPSTRING = CL THEN CLEAR;
0060     GOTO 0;
0061  END;
```

```
0001  ONE

                ; RC-GLOSTRUP  74.01.30
                ;
                ;  MUSIL CODE-COMPILER TEST:
                ;
                ;  CODEPROCEDURE:
                ;  CALL:                        RETURN:  JMP¤   MZSTART+1
                ;  AC0: MODIFBITS               AC0:
                ;  AC1:                         AC1:
                ;  AC2: CUR                     AC2:     CUR
                ;  AC3:                         AC3:
                ;
                ;  PROCEDURE TAKEADDRESS(MODIF'ADDRESS);
                ;  CALL:  JSR¤  MZSTART+2  RETURN:
                ;  AC0: MODIF                   AC0:     MODIF SHIFT(-2)
                ;  AC1:                         AC1:     ADDRESS
                ;  AC2: CUR                     AC2:     CUR
                ;  AC3: LINK                    AC3:     DESTROYED
                ;
                ;  PROCEDURE TAKEVALUE(MODIF'VALUE);
                ;  CALL:  JSR¤  MZSTART+3  RETURN:
                ;  AC0: MODIF                   AC0:     MODIF SHIFT(-2)
                ;  AC1:                         AC1:     VALUE
                ;  AC2:                         AC2:     CUR
                ;  AC3: LINK                    AC3:     DESTROYED
                ;

                .TITL   ONE
        000012  .RDX    10
                .NREL

00000,020411 STONE:  LDA     0       WK      ;
00001,061135         DOAS    0       29      ; OUTCHAR(OCP);
00002,010407         ISZ             WK      ;
00003,024112         LDA     1       .64     ;
00004,106404         SUB     0,1     SZR     ;  IF CHAR = 64 THEN
00005,000403         JMP             .+3     ;
00006,024135         LDA     1       .48     ;
00007,044402         STA     1       WK      ;  CHAR:=48;
00010,002235         JMP¤            MZSTART+1
00011,000060 WK:     48

        000000,.END  STONE
```

```
0001  TWO

                .TITL   TWO
        000012  .RDX    10
                .NREL
        000001  .TXTM   1

                ; CODEPROCEDURE TWO(OPSTRING);

00000,006236 STTWO:  JSR¤            "ZSTART+2; TAKEADDRESS(MODIF'ADDR
00001,135220         MOVZR   1'3                 ;
00002,030414         LDA     2           "       ;
00003,020126         LDA     0           .10     ;
00004,040411         STA     0           WK      ;
00005,025000         LDA     1           +0'2    ;
00006,045400         STA     1           +0'3    ;
00007,151400         INC     2'2                 ;
00010,175400         INC     3'3                 ;
00011,014404         DSZ                 WK      ;
00012,000773         JMP                 .-5     ;
00013,030040         LDA     2           CUR     ;
00014,002235         JMP¤                MZSTART+1;
00015,000012 WK:     10
00016,000017,T:      .+1
                     .TXT    .HALLO' HERE IS TWO.
00017,044101
00020,046114
00021,047454
00022,020110
00023,042522
00024,042440
00025,044523
00026,020124
00027,053517
00030,000000

        000000;.END     STTWO
```

```
0001  CLEAR

                    .TITL   CLEAR
             000012 .RDX    10
                    .NREL

 00000,102400 STCL: SUB     0'0
 00001,061032       DOA     0       26          ; CLEAR LAMPS;
 00002,020126       LDA     0       .10         ;
 00003,061135       DOAS    0       ??          ; CLEAR DISPLAY;
 00004,002235       JMP@                ;START+1;

        000000,.END STCL
```

```
0001  TEST

                .TITL   TEST
        000012  .RDX    10
                .NREL

                ; CODEPROCEDURE TEST(PIP'POP'OPSTRING'TTY);

00000,006237 STTES:  JSR¤    MZSTART+3       ; TAKEVALUE(MODIF'VALUE);
00001,065032         DOA     1       2       ;
00002,006237         JSR¤    MZSTART         ; TAKEVALUE(MODIF'VALUE);
00003,044422         STA     1       GEM1    ;
00004,006236         JSR¤    MZSTART+2       ; TAKEADDRESS(MODIF'ADDRE
00005,044421         STA     1       GEM2    ;
00006,006236         JSR¤    MZSTART+2       ; TAKEADDRESS(MODIF'ADDRE
00007,131000         MOV     1'2             ;
00010,020121         LDA     0       .3      ;
00011,006221         OPEN                    ;
00012,020414         LDA     0       GEM2    ;
00013,006215         OUTTEXT                 ;
00014,006220         CLOSE                   ;
00015,030040         LDA     2       CUR     ;
00016,060433         DIA     0       27      ;
00017,101005         MOV     0'0     SNR     ;
00020,000776         JMP             .-2     ;
00021,061032         DOA     0       26      ;
00022,020403         LDA     0       GEM1    ;
00023,061011         DOA     0       TTO     ;
00024,002235         JMP¤            MZSTART+1;

00025,000000 GEM1:   0
00026,000000 GEM2:   0

       000000,.END   STTES
```