Interpreter testoutput.

It is possible to have testoutput on the lineprinter (LPT) from to interpreter intt5, RCSL: 43-GL 1390. In case of normal program running the interpreter int05, RCSL: 43-GL 1294 should be used.

The lpt driver should always be loaded together with the intepreter. In case of lpt errors the action is repetition until lpt is ready. If testoutput is wanted on another driver the filename lpt should be changed. This name is found as the first word of the loaded module. A list command will give the address as current nmax.

The testoutput depends on how the keyboardswitches are set. It is possible to select the testoutput in accordance to listing on next page.

The testoutput primary consists of the operation word (op) and the modification word (modif). The modif is especially used in connection with the procedures takeaddress and takevalue. Here the two rightmost bits are extracted indicating how the addresses and values should be fetched. For every takeaddress and takevalue the modif is shifted two bits to the right. The functions of the procedures are:

```
; procedure takevalue(modif, value);
; gets the value of an integer.
; 2 bits modif:  00 value = instr(pc); incr(pc);
; -  -    -  :  01 value = r.cur.
; -  -    -  :  10 value = word(instr(pc)); incr(pc);
; -  -    -  :  11 value = r.cur.
;              call:              return:
; ac0          modif              modif shift (-2).
; ac1                             value
; ac2          cur                cur
; ac3          link               destroyed

;procedure takeaddress(modif, address);
; gets the address of an integer addressed by pc and incr(pc).
; 2 bits modif:  00 addr = word(pc). ! integer !
; -  -    -  :  01 addr =  -     . ! string !
; -  -    -  :  10 addr =  -     . ! file   !
; -  -    -  :  11 addr =  -     . addr = zone(cur+addr(0:7)).zfirst + addr(8:15).
;              call:              return:
; ac0          modif              modif shift (-2).
; ac1                             address.
; ac2          cur                cur.
; ac3          link               destroyed.
```

switch 0 set: testoutput format = op modif zone zused zshared ztop zrem zfirst zlength zformat z0

op	i/o-instruction:	op	catalog-instruction:
200	getrec	346	create entry
201	putrec	347	lookup entry
202	waittransfer	350	change entry
203	repeatshare	351	remove entry
204	transfer	352	init catalog
205	inblock	353	update catalog
206	outblock		
207	inchar		
210	backspace		
211	outspace		
212	outchar		
213	outnl		
214	outend		
215	outtext		
216	outoctal		
217	setposition		
220	close		
221	open		
222	waitzone		

switch 1 set: op instruction testoutput format = op. modif. PC+1 PC+2 PC+3 PC+4 PC+5 PC+6

52 move(from addr=takeaddr(PC+2)+takevalue(PC+3),to addr=takeaddr(PC+4)+takevalue(PC+5),count=takevalue(PC+6)),modif=PC+1

60 go code: index=-modif, addr=cur+index, modif=PC+1, goto word(addr) (call of the codeprocedure)

switch 2 set: op instruction testoutput format = op. modif. PC+1 PC+2 PC+3 PC+4

36 moveword (to addr = takeaddr(PC+1), value = takevalue(PC+2))

37 movestring (from addr = takeaddr(PC+1),to addr = takeaddr(PC+2),count=takevalue(PC+3))

40 compareword, not used

41 comparestring (addr of string 1=takeaddr(PC+1),addr of string 2=takeaddr(PC+2),count=takevalue(PC+3))

42 store register (store r in addr = PC+1)

43 translate(from byte addr=takeaddr(PC+1),to byte addr=takeaddr(PC+2),tableaddr=takeaddr(PC+3))

44 convert(from string addr=takeaddr(PC+1), to string addr=takeaddr(PC+2),tableaddr=takeaddr(PC+3),count=takevalue(PC+4))

switch 3 set: op instruction testoutput format = op. modif. PC+1 PC+2 PC+3

45 opmess(addr of string = takeaddr(PC+1))

46 opin (addr of string variable = takeaddr(PC+1))

47 opwait (addr of variable = (PC-1))

50 call (addr of first cell in the procedure, used to save the link = PC+1(return PC),content of return PC = PC+2)

51 optest

52 see move (switch 1)

53 opstatus(status=takevalue(PC+1), addr of errorstring = takeaddr(PC+2))

54 bindec(value of bin = takevalue(PC+1), addr of dec=takeaddr(PC+2))

55 decbin(addr of dec = takeaddr(PC+1), store binvalue in addr =PC+2)

56 insert (bytevalue = takevalue(PC+1), to addr=takeaddr(PC+2)+takevalue(PC+3))

57 goto (store goto addr =PC+1 in PC and continue)

switch 4 set: op instruction testoutput format = op. modif. PC+1

11 and constant, constant = PC+1

12 load - , -

13 + - , -

14 ÷ - , -

15 shift - , -

16 extract - , -

17 * - , -

20 / - , -

21 and variable, addr of variable = PC+1

22 load - , -

23 + - , -

24 ÷ - , -

25 shift - , -

26 extract - , -

27 * - , -

30 / - , -

31 load negative (value =-takevalue (PC+1))

32 load byte (addr = takeaddr(PC+1), getbyte)

33 load byteword (addr = takeaddr(PC+1), getbyte getbyte)

34 jump 1. goto addr = PC-1 (modif=0)
2. if a=b (modif= 9, 10 of 11) then goto addr = PC+1
3. if a< b (modif = 11, 12 or 13) then goto addr=PC+1
4. if a > b (modif = 10, 12 or 14) then goto addr=PC+1

35 link 1. exit from procedure: addr used to save "return PC" = PC+1
2. exit from zone giveup: addr <0, zone = -addr, PC = savePC. zone = zsize, link = savelink. zone=zbuff

switch 5 set: op instruction testoutput format = op modif.

0 stop

1 and direct, modif = value

2 load - , -

3 + - , -

4 ÷ - , -

5 shift - , -

6 extract - , -

7 * - , -

10 / - , -

```
MUSIL COMPILER/2
0000
0001 ! PROGRAM USED TO GIVE AN EXAMPEL OF TESTOUTPUT FROM INTERPRETER !
0002 CONST
0003 TEXT1='TT', TEXT2='SS', MOVEOK='<10>MOVE, OK', MOVEF='<10>MOVE, FAULT';
0004 VAR
0005 RESULT: INTEGER;
0006 OUT:    FILE 'XXXXX0',62,1,512,UB; GIVEUP FERROR, 8'177777 OF STRING(1);
0007
0008 PROCEDURE WRITERESULT;
0009 BEGIN
0010   IF RESULT=15 THEN OPMESS(MOVEOK); IF RESULT=16 THEN OPMESS(MOVEF);
0011 END;
0012
0013 PROCEDURE FERROR;
0014 BEGIN
0015   ! DUMMY !
0016 END;
0017
0018 BEGIN
0019 ! NORMAL INSTRUCTION !
0020 RESULT:=15;
0021 MOVE(TEXT1,0,TEXT2,0,2);
0022 IF TEXT1<>TEXT2 THEN RESULT:=16;
0023 WRITERESULT;
0024
0025 ! CATALOG INSTRUCTION !
0026 INITCAT(OUT,0,1);
0027
0028 END;
SIZE: 00430
```

Interpreter testoutput:

```
-000036-000002-007746-000017-025000-000021                          result:=15
£-000052-000000-000021-017654-000000-017660-000000-000002           move
£-000041-000005-017654-017660-000002-016011                         comparestring
£-000034-000011-010460                                              jump
£-000050-000000-010410-165001-007747                                call writeresult
£-000022-000002-007746                                              load result
£-000004-000017                                                     -15
£-000034-000014-010420                                              jump
£-000045-000001-017664-011002-007746                                opmess(moveok)
£-000022-000002-007746                                              load result
£-000004-000020                                                     -16
£-000034-000014-010427                                              jump
£-000035-000000-010410                                              link          initcat
£-000352-000001-007747-010001-001000-000000-000000-000000-000001-000001-043105
£-000000-000003                                                     stop
```