MUSIL PROGRAMMER'S REFERENCE CARD

The information in these pages is based on data contained in "MUSIL Programming Guide" RCSL 42-i 0344 and "RC 3600 File System - System Programmer's Guide" RCSL 44-RT 1278.

MUSIL CHARACTER SET

The following characters are used for identifiers:

A through Z capital letters
0 through 9 digits

First character must be letter, the first 7 characters and the length are significant.

The following symbols are used for punctuation:

! !	surrounding comments
# #	surrounding sequences of byte values
< >	surrounding numeric representation of character
' '	surrounding character string
'	separating radix and number
,	separating elements of lists
:	follows labels; separates list of identifiers and their type
=	separating identifier and constant; separating identifier and type
;	separates statements
space	
.	denotes record variable components
↑	denotes reference to file record
:=	separating variable and expression
( )	surrounding length of string type; surrounding simple expressions

RC 3600 File System

The following symbols are used for arithmetic expressions:

BYTE	operand: string; result: integer. Take first byte.
WORD	operand: string; result: integer. Take first and second byte.
+	operands: integer; result: integer. Addition.
-	operands: integer; result:integer. Subtraction.
*	operands: integer; result: integer. Multiplication.
/	operands: integer; result: integer. Integer division.
SHIFT	operands: integer; result: integer. First operand logical shift, second operand positions.
EXTRACT	operands: integer; result: integer. First operand mask out, second operand positions from right.
AND	operands: integer; result: integer. Logical and.

The following symbols are used in comparison operations (operands: strings or integers, for strings the length compared is computed as for assignments):

>	greater than
>=	greater than or equal to
=	equal to
<>	not equal to
<=	less than or equal to
<	less than

MUSIL PROGRAM

A MUSIL program may contain CONST section where constants are defined, a TYPE section where data types are defined, a VAR section where variables and files are defined, a PROCEDURE section where procedures are declared, and a STATEMENT part where algorithmic actions are defined.

MUSIL CONST SECTION

```
CONST                          !examples of constant definitions!
A= 17,                         !A 17(10) integer!
B= -33,                        !B -33(10) integer!
C=8'77,                        !C 63(10) integer!
D='EXAMPLE'                    !D is a string!
E='<10>NL text NL<10>',        !E is a string!
F=#128 0 0 1 0 2 #;            !F is a string!
```

MUSIL TYPE SECTION

Note that a data type must be declared before a reference can be made to its associated identifier.

```
TYPE                           !examples of type definitions!
LINE=     STRING (20);
PLINE=    RECORD
            L1: LINE;
            L2: LINE;
            L3: LINE
          END;
IN FILE 'MTO', 14, 1, 600, FB OF PLINE;
```

MUSIL VAR SECTION

```
VAR                            !examples of variable definitions!
G:          INTEGER;
H, I, J:    INTEGER;
K, L:       LINE;              !string (20) see the examples
                               under TYPE!
M:          PLINE;             !record, see the examples under
                               TYPE!
N, O:       STRING(30);
P:          IN;                !file, see the examples under
                               TYPE!
Q:          RECORD             !a record definition!
              CCW:  STRING(1);
              DATA: STRING(19)
            END;
R:          RECORD             !a record definition!
              I1: INTEGER;
              S1: STRING(16);
              J1: STRING(2)
            END;
```

```
S:      RECORD      !a record definition!
          TOTAL:  STRING(20);
          COL10:  STRING(1) FROM 10;
          ILAST:  INTEGER   FROM 19
        END;
T:      FILE 'PTR', 1,2,600,U;
        GIVEUP PTRERROR, 8'143777
        OF STRING(600);
U:      FILE 'PTP', 1,2,600,FB;
        GIVEUP PTPERROR, 8'143777;
        CONV F      !conversion table!
        OF PLINE;
```

MUSIL PROCEDURE SECTION

```
PROCEDURE P0001(VAR X: STRING(8));
CODEBODY;
PROCEDURE DATE (VAR X: STRING(8));
CODEBODY P0003;
PROCEDURE PTRERROR;
BEGIN
  !statement(s)!
END;
```

MUSIL STATEMENT PART

The ASSIGNMENT statement serves to replace the current value of a variable by a new value indicated by an expression.

```
G :=A;      !G becomes 17, see the examples under CONST!
N :=D;      !N becomes EXAMPLE, see the examples under
            CONST!
U :=E;
U .L2:=E;   !see the examples under CONST and TYPE!
U .L3:=E;   !see the examples under CONST and TYPE!
H :=A*C+B;
```

In assignment between string types, the number of bytes moved is determined by

(1):=(2)	string(L2)	file2↑
string(L1)	min(L1,L2)	L1
file1↑	L2	file1.zlength

In the following e means integer expression constant or variable, s means statement, int means integer variable,str means string variable, strc means string variable or constant and f means file.

The following statements are used to control PROGRAM FLOW:

```
GOTO label              !unconditional jump!
IF e THEN s;            !conditional execution!
IF e THEN s1 ELSE s2;
REPEAT s UNTIL e;       !repetitive statement!
WHILE e DO s;           !repetitive statement!
```


The following statements are used to CONVERT DATA:

```
BINDEC(e, str);          !converts an integer to a 5 cha-
                         racter decimal string followed by
                         a binary zero byte!
CONVERT(strc1, strc2, strc3, e);
                         !converts e bytes from strc 1 to
                         strc 2 using strc 3 as table!
DECBIN(strc, int);       !converts a numeric string to an
                         integer!
TRANSLATE(strc1, strc2, strc3);
                         !converts first byte of strc1 to
                         first byte of strc2 using strc3
                         as table!
```

The following statements are used to MOVE DATA:

```
INSERT(e1, strc, e2);    !places value of e1 mod 256 in
                         strc indexed by e2!
MOVE(strc1, e1, strc2, e2, e3);
                         !moves e3 bytes from strc1
                         indexed by e1 to strc2 indexed by
                         e2!
```

The following statements are used to control TRANSPUT THROUGH ZONES:

```
CLOSE(f, e);             !if e<>0 then the driver is re-
                         leased, otherwise not!
GETREC(f, int);          !make a new record available!
INBLOCK(f);              !read a whole block!
INCHAR(f, int);          !make a new character available!
OPEN(f, int);            !open a file in mode int!
OUTBLOCK(f);             !write a whole block!
OUTCHAR(f, e);           !write out one byte value (e)!
OUTTEXT(f, strc);        !write out a string strc!
PUTREC(f, e);            !make room for new output record!
REPEATSHARE(f);          !repeat last erroneous operation.
                         Note: should only be used inside
                         giveup procedures!
SETPOSITION(f, e1, e2); !position a file to a certain
                         tape mark=e1 and block=e2
                         position!
TRANSFER(f, e1, e2);     !transfer a buffer of length =e1
                         bytes with mode operation = e2 to
                         the driver specified by f!
WAITTRANSFER(f);         !wait for a buffer to be retur-
                         ned!
WAITZONE(f);             !wait for all buffers of file f
                         to be returned!
```

The following statements are used to control TRANSPUT WITH CAT 76 THROUGH ZONES:

```
INITCAT(f, e1, e2);      !e1=driveno, e2 irrelevant!
CREATEENTRY(f, e1, int2);
                         !creates a new file f.zname with
                         size e1, int2=0 means fixed
                         length, int2=1 means extensible!
REMOVEENTRY(f);          !removes file f.zname!
LOOKUPENTRY(f, str);     !moves the 32-byte entry de-
                         scriptor of file f.zname to str!
CHANGEENTRY(f, str);     !changes the entry descriptor of
                         file f.zname to the filename and
                         filelength of the entry descrip-
                         tor held in str!
SETENTRY(f, str);        !sets an entry with name and size
                         according to the values held in
                         the 32-byte entry descriptor held
                         in str!
```

The following statements are used to control OPERATOR COMMUNICATION WITHOUT ZONE:

```
OPIN(str);               !enables str to receive what the
                         operator types to the process!
OPMESS(strc);            !writes contents of strc on the
                         operator device!
OPSTATUS(e, strc);       !strc is supposed to hold a set
                         of texts separated by<0>. Text
                         numbers from strc are written
                         according to bits set in e!
OPWAIT(int);             !the process will wait for a
                         message from the operator, int is
                         the length of the message!
```

Input-Output statusword when communicating with operator without files, using the OPIN statement:

```
OPTEST                   !=0 if no message typed,
                         <>0 if a message has been typed.
                         OPWAIT will reset the value to 0!
```


MUSIL FILE DEFINITION

```
ident: FILE              filename, filekind, buffers,
                         bufferlength
                         [, datatype]0
                         [; GIVEUP procedurename, mask]1
                         [; CONV tablename]0
       OF                [recordtype scalartype];
```

MUSIL FILEKIND BITS

1b15	file character-oriented
1b14	file block-oriented
1b13	file positionable
1b12	file operations repeatable
1b11	file is a disc file
1b10	irrelevant
1b0	coroutine file

MUSIL FILE DATATYPE

U	(0) undefined format
UB	(1) undefined blocked format
F	(2) fixed format
FB	(3) fixed blocked format
V	(4) variable format (IBM compatible)
VB	(5) variable blocked format (IBM compatible)

MUSIL FILE STATUSWORD BITS

1b0	device disconnected
1b1	device off-line
1b2	device busy
1b3	device bit 1
1b4	device bit 2
1b5	device bit 3
1b6	driver reserved
1b7	end of file
1b8	device block length error
1b9	device data late
1b10	parity error
1b11	end medium
1b12	device position error
1b13	device driver missing
1b14	device time out
1b8+1b15	format error in getrec/putrec

MUSIL FILE STATUSWORD BITS FROM CAT 76

INITCAT

1b3+1b0	catalog i/o error
1b3+1b1	'sys' or 'map' not in catalog
1b3+1b6	illegal drive number

CREATEENTRY

1b3+1b0	catalog i/o error
1b3+1b11	entry with same name already exists
1b3+1b7	disc area of requested size not available
1b3+1b6	illegal drive number, wrong size, wrong type or disc not initialized
1b3+1b12	catalog full

REMOVEENTRY

1b3+1b0	catalog i/o error
1b3+1b1	entry does not exist
1b3+1b6	illegal name format or disc not initialized, or area process exists

LOOKUPENTRY

1b3+1b0	catalog i/o error
1b3+1b1	entry does not exist
1b3+1b6	disc not initialized

CHANGEENTRY

1b3+1b0	catalog i/o error
1b3+1b1	entry does not exist
1b3+1b7	not enough disc space for new size
1b3+1b6	illegal name format, disc not initialized, area process exists, wrong attribute, change of name of permanent file
1b3+1b11	entry with new name already exists

SETENTRY

1b3+1b0	catalog i/o error
1b3+1b6	wrong size, wrong attribute, disc not initialized
1b3+1b7	disc full
1b3+1b11	entry with same name exists

MUSIL FILE DESCRIPTOR RECORD COMPONENTS

ZNAME:	file name, string(6)
ZMODE:	operation mode, integer
ZKIND:	file kind, integer
ZMASK:	GIVEUP mask, integer
ZFILE:	current file number, integer
ZBLOCK:	current block number, integer
ZCONV:	conversion table address, integer
ZFORM:	record format = datatype, integer
ZREM:	remaining bytes in current buffer, integer
ZLENGTH:	record length, integer
ZFIRST:	address of first byte of current record, integer
ZTOP:	address of first byte after current record, integer
Z0:	statusword, integer
ZUSED:	address of currently used buffer
ZSHAREL:	length of buffer, integer

MUSIL CAT 76 ENTRY DESCRIPTOR

```
ident:    RECORD
            filename:  STRING(6);
            free1:     STRING(6);
            attribute: INTEGER;
            length:    INTEGER;
            filestart: INTEGER;
            filesize:  INTEGER;
            free 2:    STRING(12)
          END
```

MUSIL ZONE OPERATION MODE SURVEY

TTY	1	normal read mode
	3	write mode
	5	read mode with time out
	17	read mode with attention request
PTR	1	read binary
	5	read odd parity, deliver 7 bits
	9	read even parity, deliver 7 bits
PTP	3	punch binary
	7	punch odd parity
	11	punch even parity
LPT	3	print characters
	7	interpret first byte of share as CCW
MTn	1	read byte limit 12
	3	write
	5	read byte limit 0
	+8192	selects lower of two possible densities
	+4096	selects even parity
CDR	1	read binary bytes
	5	read binary punched cards
	21	read decimal punched cards
	33	read decimal punched cards and skip trailing blank columns (from column 10)
RDP	1	read binary bytes with conversion
	5	read binary punched cards
	9	read decimal punched cards, convert, skip trailing blank columns, and terminate with byte values <13><10>. A maximum of 72 card columns are delivered.
	21	read decimal punched cards and convert
	33	read decimal punched cards, convert, and skip trailing blank columns. A minimum of 10 columns is delivered.
	+256	selects the secondary hopper
	+64	demotes that the card in the wait station will be led out to stacker 2
	3	punch binary byte, no conversion
	7	punch binary word, no conversion
	11	punch decimal byte with conversion
	19	print decimal byte with conversion
	27	punch and print decimal byte with conversion
	+512	separates print data. The first 80 (51) bytes are interpreted as the punch data, and the rest as the print data. (If binary words, the first 160 (102) bytes are the punch data.) Only relevant in modes 3, 7 and 11.
	+256	selects stacker 2
	+64	the card in the wait station is led out to the appropriate stacker, and a card is fed before the write operation (read before write).
	+32	selects the primary hopper

CTn	(normal)	
	1	read (ECMA 34 vs 2)
	3	write (ECMA 34 vs 2)
	9	read (ECMA 34 vs 1)
	17	read, no check
	(special)	
	1	read (ECMA 34 vs 2)
	3	write (ECMA 34 vs 2)
	7	write (ECMA 34 vs 2) with control read
	9	read (ECMA 34 vs 1)
	11	write (ECMA 34 vs 1)
	15	write (ECMA 34 vs 1) with control read
	17	read, no check
	+4	added to a read mode causes continuous reading
DKPO	1	read sequential
	5	read random
	3	write sequential
	7	write random
	11	write/read sequential
	15	write/read random
FDO	1	read sequential
	5	read random
	17	read sequential nonskip
	21	read random nonskip
	3	write sequential
	7	write random
	11	write/read sequential
	15	write/read random
	+32	for logical position
PLT	3	interpret 4-bits output instruction
CPT	3	print characters
	7	interpret first byte of block as CCW
	15	output to VFU (6 lpi)
	31	output to VFU (8 lpi)
SP	3	print characters
	7	interpret first byte of block as CCW

COMPILER OPERATION PROCEDURE

Commands are underscored, messages not.

Command/Message	Explanation
MUSIL READY	compiler ready for commands
DISP	display of parameter values (here the standard values):
IN $PTR	source file
OUT $PTP	object file
LIST	list file (here none)
INCOD	codeprocedure file (here none)
NAME MAIN	process name
IDENT	ASCII ident (here none)
OPCOM TTY	process operator device
MODIF	modifications (here none)
INIT	parameters set to standard values
START	compilation will start

Parameter Modification:

IN<file>	e.g. $PTR, $MT0:3, TEXT:1
OUT <file>	e.g. $PTP, $MT1:1, OBJ:1
LIST <file>	e.g. $LPT, LIST:1
INCOD <file>	e.g. $PTR, CODES:1
NAME <text>	1 to 5 characters, e.g. EDIT
IDENT <text>	0 to 5 characters, e.g. EDNEW
OPCOM <device>	e.g. TTY, OCP, TTY1
MODIF <letters>	C for coroutines (process description structure), B for one message buffer per share (included in C), N for no process description
<file>	denotes a file. Files controlled by drivers: $<driver>. Multifile drivers: $<driver>:<filenumber>. Files controlled by CAT(76): <filename> or <filename>:<drive> when other than drive 0.
<text>	denotes an ASCII text. Up to 5 characters are significant.
<device>	denotes an operator device driver
<letters>	denote modifications

COMPILER OPERATION PROCEDURE (continued)

Drivers supported by the compiler are:

PTR, PTP, TTY, MT0, MT1, CDR, RDP,
CT0, CT1, FD0, FD1, LPT, CPT, SP.

Copy, inclusion of source texts from other files:

```
PRIME:    BEGIN
          source text
          $COPY SEGM1
          source text
          $COPY SEGM2
          source text
          END

SEGM1:    source text
          $END

SEGM2:    source text
          $END
```

Code Conversion:

May be used in connection with source file, copy files, and list file. Conversion table program names are:

CITAB	for normal source file
CCTAB	for copied source file(s)
CLTAB	for list file

Command Error Messages:

ILLEGAL COMMAND	command unknown
DRIVER MISSING	driver or CAT (76)-system missing
ILLEGAL DEVICE	file cannot be used for input or output
NO INPUT FILE	input disc file does not exist
FILE NO MISSING	filenumber for a multible file driver is missing
ILLEGAL MODIF ITEM	other MODIF letters than B, C or N are used

Normal Messages During Compilation:

MUSIL COMPILER 3	written to list file at start
<name>	<name>= process name
SIZE <size>	written to list file at end, if no errors in source text, <size> = number of $bytes_{10}$.
***SIZE <size>	no object file specified and codeprocedure not included

COMPILER OPERATION PROCEDURE (continued)

Error Messages During Compilation:

<file> STATE <status> REP?	indicates <status> error on <file>. Answer NO terminates compilation, other answers cause repetition of last operation.
COREOVERFLOW	lack of core, compilation is terminated

Error Messages During Codeprocedure Loading:

STATUS EM	codeprocedure(s) missing
INCOD ERROR	no INCOD device specified
END ERROR	start block missing
SUM ERROR	sumcheck error
ILL ERROR	inconsistent block

Syntax Error Messages In the List File:

020202	number overflow in constant
020301	illegal character in source
030102	illegal use of <nnn> in string
040105	name conflict in CONST section
040205	name conflict in TYPE section
040302	syntax error in TYPE section, ident not followed by =
040405	name conflict in VAR section
040602	procedure head not followed by a ;
050102	type is no identifier
050202	(missing
050203	length undefined for string
050502	) missing after string
050604	undefined type identifier. Note that a name shall be declared before any use.
050702	improper termination of file specification
051002	field of structured type or too long
051102	incorrect use of FROM or integer field starting at odd byte address
051205	name conflict in GIVEUP procedure
051304	conversion table undeclared
051406	conversion table type error
060206	multiple defined table
060302	variable not identifier, or multiple
060402	. is not followed by identifier or by undeclared field
060504	identifier undefined
060606	type error with BYTE or WORD
060702	relational operator missing
061002	procedure statement missing)
061102	type error in procedure parameter
061306	illegal number of parameters
061406	type error with operator
061506	overflow of work registers, expression too complex

COMPILER OPERATION PROCEDURE (continued)

Syntax Error Messages In the List File Which Cause Skipping of Program Parts:

000040	syntax in section delimiter
000041	syntax in CONST section
000043	type specification incorrectly terminated
000044	variable declaration incorrect
000045	variable declaration incorrectly terminated
000046	procedure heading incorrect
000047	incorrect parameter format
000051	syntax in field list
000052	syntax in file declaration
000063	incomprehensible statement
000064	incorrect label declaration
000065	incomprehensible expression

LIST OF RESERVED MUSIL WORDS

AND	GOTO	RECORD
		REMOVEENTRY
BEGIN	IF	REPEAT
BINDEC	INBLOCK	REPEATSHARE
BYTE	INCHAR	
	INITCAT	SETENTRY
CHANGEENTRY	INSERT	SETPOSITION
CLOSE	INTEGER	SHIFT
CODEBODY		STRING
CONV	LOOKUPENTRY	
CONVERT		THEN
CONST	MOVE	TRANSFER
CREATEENTRY		TRANSLATE
	OF	TYPE
DECBIN	OPEN	
DO	OPIN	UNTIL
	OPMESS	
ELSE	OPSTATUS	VAR
END	OPTEST	
EXTRACT	OPWAIT	WAITTRANSFER
	OUTBLOCK	WAITZONE
FILE	OUTCHAR	WHILE
FROM	OUTTEXT	WORD
GETREC	PROCEDURE	
GIVEUP	PUTREC	

For detailed information consult also the RC 3600 Operating Guide.

ASCII CODE TABLE

	Decimal Representation	7-Bit Octal Code	Character
**	0	000	NUL
**	1	001	SOH
**	2	002	STX
**	3	003	ETX
**	4	004	EOT
**	5	005	ENQ
**	6	006	ACK
**	7	007	BEL
**	8	010	BS
**	9	011	HT
**	10	012	LF
**	11	013	VT
**	12	014	FF
**	13	015	CR
**	14	016	SO
**	15	017	SI
**	16	020	DLE
**	17	021	DC1
**	18	022	DC2
**	19	023	DC3
**	20	024	DC4
**	21	025	NAK
**	22	026	SYN
**	23	027	ETB
**	24	030	CAN
**	25	031	EM
**	26	032	SUB
**	27	033	ESC
**	28	034	FS
**	29	035	GS
**	30	036	RS
**	31	037	US
	32	040	SP
	33	041	!
	34	042	"
	35	043	#
	36	044	$
	37	045	%
	38	046	&
	39	047	'
	40	050	(
	41	051	)
	42	052	*
	43	053	+
	44	054	,
	45	055	-
	46	056	.
	47	057	/
	48	060	0
	49	061	1
	50	062	2
	51	063	3
	52	064	4
	53	065	5
	54	066	6
	55	067	7
	56	070	8
	57	071	9
	58	072	:
	59	073	;
	60	074	<
	61	075	=
	62	076	>
	63	077	?
	64	100	@
	65	101	A
	66	102	B
	67	103	C
	68	104	D
	69	105	E
	70	106	F
	71	107	G
	72	110	H
	73	111	I
	74	112	J
	75	113	K
	76	114	L
	77	115	M
	78	116	N
	79	117	O
	80	120	P
	81	121	Q
	82	122	R
	83	123	S
	84	124	T
	85	125	U
	86	126	V
	87	127	W
	88	130	X
	89	131	Y
	90	132	Z
***	91	133	
***	92	134	
***	93	135	
	94	136	↑
	95	137	←
	96	140	'
	97	141	a
	98	142	b
	99	143	c
	100	144	d
	101	145	e
	102	146	f
	103	147	g
	104	150	h
	105	151	i
	106	152	j
	107	153	k
	108	154	l
	109	155	m
	110	156	n
	111	157	o
	112	160	p
	113	161	q
	114	162	r
	115	163	s
	116	164	t
	117	165	u
	118	166	v
	119	167	w
	120	170	x
	121	171	y
	122	172	z
***	123	173	
***	124	174	
***	125	175	
	126	176	~
	127	177	DEL

** Special control characters.
Will be interpreted in accordance with actual device specifications.

*** Reserved for national characters:

RCSL 42-i 0519