Macro and SVC Services Guide

z390
ASM

Automated Software Tools Corporation

Function Table

SVC		FUNCTION
DEC	HEX	
1	01	**WAIT**
2	02	**POST**
11	0B	**TIME (and date)**
11	0B	**GETIME (VSE)**
40	28	**GETENV**
46	2E	**TTIMER**
47	2F	**STIMER**
52	34	**CMDPROC**
53	35	**WTO**
103	67	**XLATE**
160	A0	**WTOR**
170	AA	**CTD**
171	AB	**CFD**
172	AC	**SYSTRACE**

Supporting Macros

COMRG	Address Communications region (VSE)

Time periods

mS milliseconds   0.001 seconds (one thousandth)
uS microseconds 0.000001 seconds (one millionth)
nS nanoseconds   0.000000001 seconds (one billionth)

Event Control Block

There is no macro or DSECT describing it, I thought a little doc might be useful.

Bits 0-1
- 00   The initial state, WAIT requires both these bits to be zero.
- 10   After a Macro specifying an ECB is issued (eg. WTOR), this wait state is set.
- 01   Set to this state internally or by the POST Macro indicates that the event is complete or that the task in a wait state is to be resumed. It is valid to test for this state using a bit test instruction like TM.
- 11   Invalid.

Bits 2-31
  Completion code, set internally or by the POST Macro.

TIME (and date)

```
name TIME   type,addr,LINKAGE=,DATETYPE=,CLOCKTYPE=
  Obtain the time and/or date in various formats.
```

Formats--part 1...LINKAGE=SVC (default)
```
  name TIME
   --or--
  name TIME DEC
```
Returns: Time in GR0 as HHMMSSTH
- Hours, mins and secs to 2 decimal places.
- The values are unsigned packed decimal: eg. X'21420654' = 21:42:06.54
- The MVO instruction can be used after storing the register to convert it to standard packed decimal format.

Date in GR1 as CCYYDDDF
- Century, year, day number and sign
- The values are signed decimal:
  - CC is (almost) the century number.
  - YY the year number.
  - DDD the day number within the year.
  - F   the positive sign.

eg. X'0106003F' = 3rd January 2006
After storing, AP DATE,=P'1900000' can be used to convert to a 4-digit year.

name TIME BIN
Returns: Time in GR0 in hundredths of a second since midnight in binary.
        Date in GR1 as above.

name TIME TU
Returns: Time in GR0 in timer units of 26.04166uS since midnight in binary.
        Date in GR1 as above.

name TIME INS
Returns: Instruction count in GR1 (64 bit value).

name TIME MIC,label
 --or--
name TIME MIC,(reg)
Returns: Time in units of 1uS in binary since midnight.
        The time is stored at the 8 bytes specified.
        Date in GR1 as above.

name TIME NS,label
 --or--
name TIME NS,(reg)
Returns: Time in units of 1nS in binary since midnight.
        The time is stored at the 8 bytes specified.

name TIME STCK,label
 --or--
name TIME STCK,(reg)
Returns: Time in units of 1uS in binary since midnight.
        The time is stored at the 8 bytes specified and uses only bits 0-51 of the
        8-byte field.
        Date in GR1 as above.

```
      name TIME TS,label
       --or--
      name TIME TS,(reg)
      Returns: A string of 29 bytes at the label or pointed to by reg.
               The format is "YYYY-MM-DD HH:MM:SS.NNNNNNNNN".


      name TIME CLOCK,label,CLOCKTYPE=STCK
       --or--
      name TIME CLOCK,(reg),CLOCKTYPE=STCK
      Returns: Time in units of 1uS in binary since 1st January 1900.
               The time is stored at the 8 bytes specified and
               uses only bits 0-51 of the 8-byte field.


      name TIME CLOCK,label,CLOCKTYPE=STCKE
       --or--
      name TIME CLOCK,(reg),CLOCKTYPE=STCKE
      Returns: Time in units of 1uS in binary since 1st January 1900.
               The time is stored at the 16 bytes specified:
                  Byte  0   :   Zero
                  Bytes 1-13  :  The time
                  Bytes 14-15 : Programmable field set by the SCKPF instruction and not
                               currently implemented.

               The time uses only bits 8-111 of the 16-byte field
               with bits 8-59 being the value in microseconds.


      name TIME CLOCK,label,CLOCKTYPE=JAVA
       --or--
      name TIME CLOCK,(reg),CLOCKTYPE=JAVA
      Returns: Time in units of 1mS in binary since 1st January 1970.
               The time is stored at the 8 bytes specified.
```

DATETYPE can be set for any of the formats below.
All the formats are 4 bytes and the values are unsigned packed decimal.
The MVO instruction can be used to convert it to standard packed decimal format.
   YYYY the year number.
   DDD   the day number within the year.
   DD   the day number within the month.
   MM   the month number.

   YYYYDDD (default) stored as 0YYYYDDD
   MMDDYYYY
   DDMMYYYY
   YYYYMMDD

name TIME ,label,LINKAGE=SYSTEM
 --or--
name TIME ,(reg),LINKAGE=SYSTEM
 --or--
name TIME DEC,label,LINKAGE=SYSTEM
 --or--
name TIME DEC,(reg),LINKAGE=SYSTEM
Returns: Time as HHMMSSTH
      The time is stored at the 4 bytes specified.
       Hours, mins and secs to 2 decimal places.
       The values are unsigned packed decimal:
        eg. X'21420654' = 21:42:06.54
       The MVO instruction can be used to convert
       it to standard packed decimal format.

      The date is stored at label+8 or 8(reg).

name TIME BIN,label,LINKAGE=SYSTEM
 --or--
name TIME BIN,(reg),LINKAGE=SYSTEM
Returns: The time is stored at the 4 bytes specified in hundredths of a second since
        midnight

        The date is stored at label+8 or 8(reg).

```
          name TIME MIC,label,LINKAGE=SYSTEM
           --or--
          name TIME MIC,(reg),LINKAGE=SYSTEM
          Returns: Time in units of 1uS in binary since midnight.
                   The time is stored at the 8 bytes specified.

                   The date is stored at label+8 or 8(reg).

          name TIME STCK,label.LINKAGE=SYSTEM
           --or--
          name TIME STCK,(reg),LINKAGE=SYSTEM
          Returns: Time in units of 1uS in binary since midnight.
                   The time is stored at the 8 bytes specified and uses only bits 0-51 of the
                   8-byte field.

                   The date is stored at label+8 or 8(reg).

          name TIME STCKE,label.LINKAGE=SYSTEM
           --or--
          name TIME STCKE,(reg),LINKAGE=SYSTEM
          Returns: Time in units of 1uS in binary since midnight.
                   The time is stored at the 16 bytes specified:
                      Byte  0   :   Zero
                      Bytes 1-13 :  The time
                      Bytes 14-15 : Programmable field set by the SCKPF instruction and not
                                    currently implemented.

                   The time uses only bits 8-111 of the 16-byte field
                   with bits 8-59 being the value in microseconds.

                   Note: The DATETYPE parameter is ignored.

     Register Usage:
        R0 = Code for units and date type
        R1 = Result area

     GR15 has a return code:
        0  TIME ok
        4  Invalid request
```

GETIME type
   Obtain the time in various formats (VSE only).

   type
     STANDARD (default)
     Returns: Time in GR1 as 0HHMMSSc
           Hours, mins and secs in packed format.

     BIN
     Returns: Time in GR1 in seconds since midnight in binary.

     TU
     Returns: Time in GR1 in timer units of 26.04166uS since midnight in binary.

     MIC
     Returns: Time in units of 1uS in binary since midnight.
           The time is stored in the GR0/GR1 register pair.

   GR15 has a return code:
     0 GETIME ok
     4 Invalid request

STIMER

   a) Wait for an interval of time.

   name STIMER WAIT,BINTVL=label
   name STIMER WAIT,DINTVL=label
   name STIMER WAIT,MICVL=label
   name STIMER WAIT,TUINTVL=label

   b) Start a timer and continue.
      When the time expires the exit routine is invoked.

   name STIMER REAL,exit,BINTVL=label
   name STIMER REAL,exit,DINTVL=label
   name STIMER REAL,exit,MICVL=label
   name STIMER REAL,exit,TUINTVL=label

Only one STIMER can be waiting for expiry at any moment.

In each case the label points to a number of timer units.
BINTVL is a fullword with 100th of a second units.
      Maximum value of X'7FFFFFFF' is approx. 249 days.
DINTVL is a doubleword PL8'HHMMSSth', where th is 2 decimal positions of seconds.
      Maximum value of 99595999 is approx. 4 days.
MICVL is a doubleword with microsecond units.
      Maximum value of X'7FFFFFFFFFFFFFFF' is nearly 300,000 years !
TUINTVL is a fullword with 26.04166uS units.
      Maximum value of X'7FFFFFFF' is approx. 16 hours.

exit
  Can be label or (reg).
  When the time expires, the exit routine is invoked.
  GR15 has the address of the exit routine.
  Other registers must be assumed to be destroyed.

Note: STIMER REAL is measuring clock time, and not the time that
      the z390 program is executing.

Register Usage:
  R0 = Code for timer units and exit address
  R1 = Address of the timer units
  R15 = By implication, exit routine address

TTIMER

Test or cancel a previously set STIMER REAL

name TTIMER cancel,type,addr

cancel is optional
  CANCEL means that the STIMER timing is terminated.

type is optional
  TU (default) returns the remaining time in GR0 as 4 bytes
  in timer units of 26.04166uS. addr is ignored.

MIC,addr
Using MIC requires addr which may be specified as label or
(reg). The remaining time is returned at the doubleword addr in microseconds.

Example 1:
  Cancel the current STIMER REAL, return the remaining time in GR0 in timer units.
    TTIMER CANCEL

Example 2:
  Return the remaining time in microseconds at REMAIN.
    TTIMER ,MIC,REMAIN
    ...
    REMAIN DS D

Register Usage:
  R0 = Code for timer units, returned value
  R1 = Address of returned timer units

GR15 has a return code:
  0 TTIMER ok
  4 TU units remaining exceed 31 bits

CMDPROC

Open, close, read and write from a command processor.

The command processor has also been called a DOS window or
Command Prompt.

With the CMDPROC macro, you can issue commands like CD or DIR,
receive the replies from those commands line by line and start
other programs.

There is a limit of 10 command processors that can be open at
any time. The limit is only to protect the operating system
(eg. Windows) from storage depletion.

In all cases below, ID may be defined as a numeric value or in a
general register. ie. ID=2 or ID=(R5).

ID may range from 0 (default) to 9.
If the ID exceeds 9 then an abend SFFF will occur

name CMDPROC START,ID=,CMDLOG=
  Open a command processor and assign an identifier.

  CMDLOG=YES (Default)
    All output from the command processor is written to the log.

  CMDLOG=NO
    All output is saved in a memory queue. Use this option if you intend to use
    CMDPROC READ to retrieve command processor messages.

    If the memory queue exceeds the MAXQUE value (default 1000)
    then the memory queue is written to the log and CMDPROC=YES
    is assumed. An error message is generated.

name CMDPROC STOP,ID=
  Close a previously opened command processor.

name CMDPROC WRITE,label,ID=
    --or--
name CMDPROC WRITE,literal,ID=
    --or--
name CMDPROC WRITE,(reg),ID=
  Send a command to a previously opened command processor.

  label or (reg) points to a constant which terminates with X'00'
  or be defined as a double-quoted string within a standard C-type constant.

  literal is a double-quoted string within a standard C-type constant preceded by an
  equals sign.

  eg. using label
  name   CMDPROC WRITE,CMD1,ID=5

      ...
  CMD1   DC   C'DIR /X',X'00'
   --or--
  CMD1   DC   C'"DIR /X"'

eg. using literal
name   CMDPROC WRITE,=C'"DIR /X"',ID=5

name CMDPROC READ,label,len,ID=,WAIT=
Obtain the output, a line at a time, from the result of
a command issued by CMDPROC WRITE from a previously
opened command processor.

label is the receiving area and may be specified as (reg).

If len is specified it determines the maximum length that is passed to your program.
The default is the implied length of the receiving field.
Maximum value is 4095 bytes.

len may be specified as (reg).
   Maximum value is 2G bytes.

If label is specified as (reg), then len is mandatory.
WAIT (default 500) is the time in milliseconds before the READ will terminate if no
        output from the command processor is available to be read.
        Maximum value is 4094 (4 seconds).
        A value of 4095 means wait indefinitely.

WAIT may be specified as (reg).
   Maximum value is X'7FFFFFFF' (about 24 days).

Register Usage:
   R0 = Operation code and ID
   R1 = Command area
   R2 = Length
   R3 = Wait value
   R15 = Formation of ID and return code

GR15 has a return code:
    0 = CMDPROC ok
    4 = READ terminated as WAIT time has expired
    8 = READ terminated because the command processor has ended
   16 = Command Processor abnormally ended (see log message)

WTO

Display a message on the GUI console.

The record descriptor word (RDW) defines the variable length
text message generated by the WTO macro.
eg.
   DC AL2(len,0),C'text'
   len includes the 4 bytes for the RDW.

Formats:
  name WTO 'text'
    The RDW that describes the message is generated internally.

  name WTO 'text',MF=L   (list form)
    No text is written to the console, only the RDW is generated.
    This allows a 'collection' of messages to be constructed which can be used by the
    execute form.
  name WTO MF=E   (execute form 1)
    GR1 must be preloaded with the address of an RDW previously
    generated with the list form of WTO.

  name WTO MF=(E,label) (execute form 2)
   --or--
  name WTO MF=(E,(reg)) (execute form 2)
    label or (reg) points to an RDW previously generated with the list form of the WTO.

Register Usage:
  R1 = Branch around RDW or parm pointer

XLATE

Translates data to EBCDIC or ASCII.

name XLATE area,len,TO=

area may be specified as label or (reg).

len may be specified as a number or (reg).
   Maximum numeric value is 4095 bytes.
   Maximum register value is 2G bytes.

TO=A convert area to ASCII.
TO=E convert area to EBCDIC.

Register Usage:
   R0 = Area address and codes
   R1 = Length

WTOR

Display a message on the GUI console and receive a response.

name WTOR 'text',reply,len,ecb,MF=
   --or--
name WTOR "text",reply,len,ecb,MF=
   The RDW (see WTO) that describes the message is generated internally.
   The text appears on the console.

   reply
      specified as label or (reg), is the field into which the
      reply is put. The reply appears on the console.

   len
      Maximum length of reply.
      If reply is specified as (reg) then len is mandatory.
      If len is omitted, then the implied length of reply is used.

   ecb
      specified as label or (reg), by convention defined as DC F'0'.

      After the WTOR macro, instruction execution can proceed until
      the reply is completed by the user (commonly the Return key).

MF=
  Default is MF=I.
  Specify MF=L to just construct the parameter list
  Specify MF=(E,addr) or (E,(reg)) to execute the WTOR with a previously generated
  parameter list.

Example 1:
  Implied length, named ECB, wait for reply immediately...

```
    WTOR 'Enter your name',NAME,,MYECB
    WAIT ECB=MYECB
    ...
NAME  DC   CL40' '
MYECB DC   F'0'
```

Example 2:
  Register notation, maximum length, no wait for reply...

```
    LA   R5,NAME
    LA   R6,MYECB
    WTOR 'Enter your name',(R5),40,(R6)
    ... other processing
    TM   MYECB,X'40'
    BO   GOTREPLY
    ...
NAME  DC   CL40' '
MYECB DC   F'0'
```

Register Usage:
  R0 = Reply address
  R1 = Branch around RDW
  R14 = Reply length
  R15 = ECB address

WAIT

a) Wait for one **ECB** completion.
name WAIT num,ECB=

b) Wait for one or more ECB completions.
name WAIT num,ECBLIST=

ECB or ECBLIST must be specified.

num is optional and defaults to 1
   For ECB= num must be 1 or omitted.

   For ECBLIST= num is the minimum number of ECBs that must be posted before the
   WAIT is complete. This value must, of course, be less or equal to the number of
   ECBs in the list. An abend SF05 will occur if this is not the case.

ECB=
   Specified as label or (reg).
   The location of a single 4-byte ECB.

ECBLIST=
   Specified as label or (reg).
   The location of a sequence of 4-byte addresses, each of which
   points to a 4-byte ECB. The last 4-byte address must have bit 0 set to 1.

Note: For DECBs, use the CHECK macro rather than WAIT, otherwise
     error routines may not be correctly invoked.

Example:
  Wait for 2 out of 3 ECBs.
    WAIT 2,ECBLIST
  ...
  ECBLIST DC A(ECB1)
       DC A(ECB2)
       DC A(X'80000000'+ECB3)
  ECB1   DC F'0'
  ECB2   DC F'0'
  ECB3   DC F'0'

Register Usage:
  R0 = Number of ECBs
  R1 = ECB address

POST

Signal the completion of one ECB.
name POST ecb,code

ecb is required
  Specified as label or (reg).
  The location of a single 4-byte ECB.

code is optional and defaults to zero
  Specified as a value (eg. 14 or X'123') or as (reg).
  The completion code is placed in bits 2-31 of the ECB.

Register Usage:
  R0 = Event completion code
  R1 = ECB address

CTD

Convert a binary or floating point value to a printable format.

name CTD type,IN=input,OUT=output,LINKAGE=

type
This is a numeric value which determines the operation to be carried out. Equates are automatically generated. The value of type also determines the length of the input field.
type may be specified in a register eg. (R5).

Value	Equate	Length	Description
1	CTD_INT128	16	binary
2	CTD_EH	4	short HFP
3	CTD_EB	4	short BFP
4	CTD_DH	8	long HFP
5	CTD_DB	8	long BFP
6	CTD_LH	16	extended HFP
7	CTD_LB	16	extended BFP
8	CTD_DD	8	long DFP
9	CTD_ED	4	short DFP
10	CTD_LD	16	extended DFP

IN=
The input field may be specified as a literal eg. IN==DH'3.8', a label, a register pointer eg. IN=(R4) or a register eg. IN=R4

For some types, input from a register implies the use of a
register pair as follows:

value	equate	register specified
1	CTD_INT128	Any even general register, input is from the even/odd pair
2	CTD_EH	)
3	CTD_EB	) Any
4	CTD_DH	) floating
5	CTD_DB	) point
8	CTD_DD	) register
9	CTD_ED	)
6	CTD_LH	) The first floating point register
7	CTD_LB	) of a valid pair, input is from the
10	CTD_LD	) the register pair.

OUT=
  The output field may be specified as a label or a register
  pointer eg. OUT=(R4)

  The output field is always 45 bytes, and is initialised to
  blanks. Not all 45 bytes may be used.

  The output field will be ASCII if the ASCII option on
  CALL EZ390 is used, otherwise EBCDIC.

  The output field has the following format in this sequence...
  -       If the value is negative
  n...n Digits preceding the decimal point
          If the value is less than 1 and there is no exponent, then 0 is output. eg. 0.04
  .   Decimal point if there are decimal positions
  n...n Digits following the decimal point if the value is not a whole number
  E   Indicates an exponent follows
  -       Indicates a negative exponent
  nnnn The exponent value, 1-4 digits

Examples:
    zero   0
    root2   1.4142...
    -root2   -1.4142...
    50!   3.0414...E64
    2 power -50   8.8817...E-16

LINKAGE=
    SVC   (default) invokes SVC 170
    CALL generates a CALL to module FPCONMFC

Register Usage:
    R0 = Parameter formation
    R1 = Parameter list
    R14 = Subroutine call
    R15 = Subroutine address and return code

GR15 has a return code:
    0 CTD ok
    8 Invalid data address

CFD

Convert a printable format number to a binary or floating point value.

name CFD type,IN=input,OUT=output,LINKAGE=

type
  This is a numeric value which determines the operation to be
  carried out. Equates are automatically generated. The value of
  type also determines the length of the output field.
  type may be specified in a register eg. (R5).

Value	Equate	Length	Description
21	CFD_INT128	16	binary
22	CFD_EH	4	short HFP
23	CFD_EB	4	short BFP
24	CFD_DH	8	long HFP
25	CFD_DB	8	long BFP
26	CFD_LH	16	extended HFP
27	CFD_LB	16	extended BFP
28	CFD_DD	8	long DFP
29	CFD_ED	4	short DFP
30	CFD_LD	16	extended DFP

IN=
  The input field may be specified as a label or a register
  pointer eg. (R4).

  The input field must be in ASCII if the ASCII option on
  CALL EZ390 is used, otherwise EBCDIC.

The input field is always 45 bytes, and has the following format
in this sequence...

        Optional preceding blanks
+/-   Valid sign
n...n Digits preceding the decimal point
.   Decimal point if there are decimal positions
n...n Digits following the decimal point if the value is not a whole number
E   Indicates an exponent follows
-     Indicates a negative exponent
nnnn The exponent value, 1-4 digits

For CFD_INT128 all correct forms are accepted and any decimal
places are discarded eg. 129E-1 = 12.9 = 12

OUT=
The output field may be specified as a label, a register
pointer eg. OUT=(R4) or a register eg. OUT=R4

For some types, output to a register implies the use of a
register pair as follows:

Value	Equate	Register specified
21	CFD_INT128	Any even general register, output is to the even/odd pair
22	CFD_EH	)
23	CFD_EB	) Any
24	CFD_DH	) floating
25	CFD_DB	) point
28	CFD_DD	) register
29	CFD_ED	)
26	CFD_LH	) The first floating point register
27	CFD_LB	) of a valid pair, output is to the
30	CFD_LD	) register pair.

LINKAGE=
SVC (default) invokes SVC 171
CALL generates a CALL to module FPCONMFC

Register Usage:
  R1 = Parameter list
  R14 = Subroutine call
  R15 = Subroutine address and return code

GR15 has a return code:
  0 CFD ok
  8 Invalid data address
  12 invalid input data or number too large for format type

GETENV

Get an environment variable.

Environment variables are created using the SET statement in a batch process. eg. SET MYDATA=C:\MYDATA.TXT

GETENV extracts the string in a program.

name GETENV setname
name GETENV (reg)

setname is the label of a null terminated string or the string can be pointed to by reg.
eg. SETNAME DC C'MYDATA',X'00'

GETENV acquires a storage area for the variable and sets the address in GR2. The string is terminated with X'00'.

Register Usage:
  R0 = Function code
  R1 = setname pointer
  R2 = Address of variable
  R15 = Return code

GR15 has a return code:
  0 GETENV ok
  4 setname is null
  8 variable is null

SYSTRACE

Turn on and off tracing options.

SYSTRACE options

If options are omitted, all tracing is turned off.
The following options are allowed, others are ignored:
    E   Instruction
    G   GETMAIN/FREEMAIN
    Q   QSAM I/O
    T   TCPIO I/O
    V   VSAM I/O

eg. Turn on all traces except QSAM
     SYSTRACE EGTV

COMRG

VSE only.
Establish addressability to the Communications region in the ZCVT.

COMRG REG=(reg)
  If REG is omitted it defaults to GR1.
  It is the users responsibility to provide a DSECT to map the
  COMRG.

Register Usage:
  R1 = Address the ZCVT
  reg used in REG parm.

Change Summary

June 10, 2011
  Added MF= to WTOR
  CMDPROC WAIT parameter has indefinite wait when 4095
  Added new SYSTRACE

November 23, 2008
  Update to CMDPROC return codes

June 27, 2008
  Added TIME INS

January 18, 2008
  Added maximum timer values to STIMER WAIT
  Minor correction to STIMER register usage
  Added abend and return code sections
  CMDPROC CMDLOG=YES/NO

September 28, 2007
  Added GETENV
  Added VSE Macros COMRG and GETIME
  Added TIME TS

July 10, 2007
  Correction to ID= and added abend SFFF to CMDPROC

March 8, 2007
  WTO corrections
  CMDPROC ID=(reg)
  All macros now list possible general register usage

Acknowledgements

Thanks to David Bond of Tachyon Software LLC for FPCONVRT on which
the CFD and CTD macros were based.

Trademarks

IBM, CICS, VSAM and VSE are registered trademarks of International Business Machines Corporation.

Credits

Author: Melvyn Maltz
Shipping date: June 10, 2011
z390 version : V1.5.04
zCICS version: V10