1 Preface

The *open*UTM Universal Transaction Monitor is a comprehensive middleware platform, offering a wealth of options for designing and implementing transaction-oriented OLTP applications, as well as the functionality of a complete message queuing system.

Thanks to its optimum performance, sophisticated security functions, and high availability, *open*UTM is also suitable for situations in which conventional OLTP systems have long been pushed to their limits.

*open*UTM forms a secure, efficient framework for modern, multi-tier client/server architectures. Among other things, it controls global transactions, optimizes the utilization of system resources (memory, CPU, etc.), manages parallel access, takes care of access control, and sets up network connections.

The name "*open*UTM" says it all:

open	... because *open*UTM complies with the reference model for Distributed Transaction Processing (DTP) defined by X/Open and supports the open interfaces standardized by X/Open.
Universal	... because *open*UTM links different environments and is designed for use in the most varied scenarios: it integrates heterogeneous networks, platforms, resource managers, and applications.
Transaction	... because *open*UTM guarantees complete global transaction management in accordance with the classical ACID properties of atomicity, consistency, isolation and durability.
Monitor	... because *open*UTM not only offers "pure" transaction processing, but also allows for the management of distributed, enterprise-wide IT solutions.

1.1 Summary of contents and target group

This manual is intended to support programmers writing *open*UTM applications in Assembler in their work. It is a supplement to the *open*UTM manual “Programming Applications with KDCS for COBOL, C and C++”.

A basic knowledge of the operating system and *open*UTM, as well as of the core manual “Programming Applications with KDCS for COBOL, C and C++” is required. For more detailed information, the *open*UTM manuals “Generating and Administering Applications”, “Messages, Debugging and Diagnostics” and “Concepts and Functions” should be consulted.

This manual describes the language-specific points to be observed when writing Assembler program units.

The manual contains a sample program written in Assembler of the KDCS call INIT.

The Assembler data structures are listed in the chapter “Assembler data structures” on page 17.

README file

You will find information on your BS2000 computer in the Release Notes (file name SYSFGM.*product.version.language)* or in a README file (file name SYSRME.*product.version.language)*. Please ask your systems support for the user ID under which the README file is stored. You can view the README file with the /SHOW-FILE command or in an editor, or you can print it to a standard printer with the following command:

```
/PRINT-DOCUMENT filename, LINE-SPACING=*BY-EBCDIC-CONTROL
```

or, for SPOOL versions prior to 3.0A:

```
/PRINT-FILE FILE-NAME=filename,LAYOUT-CONTROL=
    PARAMETERS(CONTROL-CHARACTERS=EBCDIC)
```

2 Assembler program units

UTM program units can be written in Assembler.

– You define Assembler program units that are not ICLS-compatible during generation by means of the KDCDEF control statement PROGRAM ...,COMP=ASSEMB.

– In the case of program units which support ILCS (InterLanguage Communication Services), you have to specify PROGRAM ...,COMP=ILCS (see the *open*UTM-Manual "Generating and Handling Applications").

For the purpose of generating Assembler program units, *open*UTM provides macros that enable you to:

– call KDCS functions
– satisfy linkage conventions
– write programs more easily.

The macros are stored in the SYSLIB.UTM.050.ASS library.

The ZSTRT, ZCALL and ZEND macros do not generate shared code. Information on how to write shareable program units in Assembler is given in section “Shareable Assembler modules” on page 10.

These macros only need to be used if you want to create compatible KDCS programs.

If you want to make Assembler programs ILCS-compatible, you have a choice of two methods (see the table “Notes on the table:” on page 4 for more information on this subject, and see the CRTE manual for the ILCS conventions).

With the aid of ILCS it is possible to link program units from several source codes in different programming languages. When passing parameters or accessing common data structures, it is absolutely essential that the data representations are identical. You will find a list of all compiler and runtime systems that allow you to link mixed sources in the “Generating and Handling Applications” manual for *open*UTM (BS2000/OSD).

For UTM V3.1 and higher, UTM no longer sets the 2**7 bit (stop bit) in the last address in the parameter list passed in register 1 when calling a program unit.

2.1 Compilers, runtime systems and generation options

The following table shows the compilers (assemblers), runtime systems and generation options which you can use to create Assembler program units and execute them in a UTM user program.

– The first column of the table contains all the compiler versions that can be used to create the object modules of the program unit.

– The second column contains the versions of the runtime systems in which these program units run smoothly.

– The third column contains the values of the COMP operand of the UTM generation statement PROGRAM that you need to specify in the KDCDEF generation in order to integrate these program units into the application configuration.

Assembler compiler	**Runtime system**	**PROGRAM..., COMP=**	
ASSGEN	—	ASSEMB	1.
ASSEMB ≥ V30	—	ASSEMB	1.
ASSEMBH V1.0 through V1.2	—	ASSEMB	1.
ASSEMBH V1.2A	ASSEMBH V1.2A	ILCS	2.

Notes on the table:

1. If you specify COMP=ASSEMB, then you must **not** use the ASSEMBH runtime system. The reason for this is that ASSEMBH runtime system versions 1.1 and higher use the ILCS. The result is a mixture of non-ILCS and ILCS program units, and this is not allowed.

2. The Assembler program **must** be ILCS-compatible.
   There are two ways to make an Assembler program ILCS-compatible:

   – You can use the Assembler macros ZSTRT, ZCALL and ZEND while specifying ZSTRT ILCS=YES. Please note that the specification ZSTRT ILCS=NO (not ILCS-compatible) is the default value!

   – You can use the macros@ENTR ... ILCS=YES..., @PASS and @EXIT (see also the “ASSEMBH” manual)

   The compiler and the runtime system must have a correction status greater than or equal to 10.

2.2 Structure of an Assembler program unit

The programming rules for Assembler are the same as for program units in other programming languages:

– The first UTM call must be an INIT.

– Program unit execution ends with the UTM call PEND. After this, control is not returned to the program unit.

– Each program unit that ends a dialog step must contain an MPUT.

– The code must be serially reusable (see also the Core Manual “Programming Applications with KDCS for COBOL, C and C++”).

Start of program

Assembler program units are subroutines of the KDCROOT main routine.

To satisfy linkage conventions, each program unit must begin with the ZSTRT macro.

You also have to make the communication area (KB) and the standard primary working area (SPAB) addressable, as well as any other shareable areas that have been defined with the KDCDEF statement AREA (see the *open*UTM-Manual “Generating and Handling Applications”).

UTM passes the addresses of the parameters in a parameter list. The address of the parameter list is stored in register 1.

Word 1: address of the KB
Word 2: address of the SPAB
Word 3: address of the first shareable area

.

.

.

Word *n*+2: address of the nth shareable area

Example

The start of a program unit might, for example, look like this:

```
prgnam  ZSTRT BASIS=r1,REGS=n,PARM=savpar
         USING kckb,r2
         L     r2,0(R1)
         USING spab,r3
         L     r3,4(R1)
         USING area1,r4
         L     r4,8(R1)
            .
            .
            .
```

Where

prgnam	is the entry point of the program unit as specified at generation in the PROGRAM statement.
kckb	is the name of the DSECT which describes the KB.
r2	is the register which addresses the KB.
spab	is the name of the DSECT which describes the SPAB.
r3	is the register which addresses the SPAB.
area1	is the name of the DSECT which describes the first shareable area.
r4	is the register which addresses *area1*.

For information on the entries *r1*, *n* and *savpar* see chapter “Macros” on page 13ff.

2.3 Calling UTM from a program unit

Calling UTM functions

Calling UTM from a program unit involves the following steps:

1. Write the necessary specifications to the KDCS parameter area. These can be found in the descriptions of the function call concerned in the Core Manual “Programming Applications with KDCS for COBOL, C and C++”.
   UTM provides the KCPAA macro. KCPAA generates a DSECT with the structure of the KDCS parameter area. The names of the parameters are identical to those in the COBOL COPY element KCPAC. If the function call uses a message area (NB), this area must generally be predefined (MPUT, FPUT). The KDCS parameter area and the NB should be placed in the SPAB.
2. Call the ZCALL macro:

   ```
   ZCALL KDCS,kcpaa[,nb]
   ```

   where

   kcpaa is the name of the KDCS parameter area.

   nb is the name of the message area.
3. If desired, evaluate the return information from UTM: return codes in the communication area KB and data in the message area NB.

Example

The following excerpt from a program unit shows an example of an INIT call:

```
INITA  EQU    *
       MVC    KCOP,INIT
       LA     R9,80
       STH    R9,KCLA
       LA     R9,138
       STH    R9,KCLM
       ZCALL  KDCS,KCPAA
       CLC    KCRCCC,=C'000'
       BNE    ERRORS
       .
```

The program unit containing this INIT call uses an 80-byte KB program area and a 138-byte SPAB. Following the INIT call, the UTM error code is queried.

This example makes use of the programming aids described in chapter “Assembler data structures” on page 17ff.

Calling Assembler subroutines

Assembler program units generated with PROGRAM...,COMP=ASSEMB can call Assembler subroutines. The following points should be borne in mind:

– from Assembler program units, subroutines are called by means of the ZCALL macro
– the return from the subroutine is programmed with the ZEND macro

By the same token, ILCS program units generated with PROGRAM...,COMP=ILCS can also call Assembler subroutines, provided the ILCS program units satisfy the ILCS conventions. This is only possible if the following @ macros of the ASSEMBH compiler are used (see the "ASSEMBH-User Guide"):

– @ENTR with ILCS=YES
– @PASS
– @EXIT

or the UTM macros:

– ZSTRT with ILCS=YES
– ZCALL
– ZEND

From within ILCS program units you can call any ILCS-compatible subroutine, even subroutines written in another ILCS-supported language.

For further information on calling subroutines from program units see the Core Manual “Programming Applications with KDCS for COBOL, C and C++”.

2.4 Compiling and linking ILCS-compatible Assembler programs

When compiling an Assembler program with ZSTRT ILCS=YES, you must also assign

$TSOS.SYSLIB.ASSEMBH.012

as an additional macro library. This library contains the @ENTR, @PASS and @EXIT macros used by the Z macros.

When linking the UTM application, you must assign the library

$TSOS.SYSLIB.ASSEMBH.012

This library contains the Assembler runtime system.

Additional information can be found in chapter 4, “Runtime System for Structured Programming”, of the “ASSEMBH V1.2A” User Guide.

2.5 Shareable Assembler modules

A shareable Assembler module can only be loaded in a common memory pool (i.e. in class 6 memory). It may only be a submodule of the UTM program units. It must not be defined as a UTM program unit in the KDCDEF control statement PROGRAM.

Specifying LOAD=(POOL,*poolname*) in the MODULE statement determines where the shareable Assembler module is to be loaded. Further entry points can be defined with the KDCDEF control statement ENTRY.

If you are working with BLS (Binder-Loader-Starter), you must use the LOAD-MODULE generation statement instead of the MODULE statement.

Assembler modules defined in the PROGRAM statement as UTM program units cannot be shareable.

Shareable modules must not contain any external addresses, V-type constants, local register save areas or local working areas.

The ZSTRT, ZCALL and ZEND macros must **not** be used.

Communication between UTM and these shareable modules in the common memory pool takes place exclusively via UTM program units which are defined in the PROGRAM statement and which, in turn, communicate with the shareable modules in accordance with the usual conventions for submodule branching (BALR Rx,Ry).

If a UTM program unit contains a reference to a shareable module, different entries in the PROGRAM statement are required, depending on whether or not you are working with the BLS interface:

– without the BLS interface:
  you have to specify the operand LOAD=STARTUP in the PROGRAM statement

– with the BLS interface:
  the program unit must be contained in a load module described in the LOAD-MODULE statement. By means of the parameter LOAD-MODE=STARTUP or LOAD-MODE=ONCALL you stipulate that the load module is to be dynamically loaded as an autonomous unit.
  You must specify the name of the load module in the LOAD-MODULE parameter of the PROGRAM statement.

For further information on generating shareable modules see the *open*UTM-Manual "Generating and Handling Applications".

Example

1. In the following example the UTM application does not use the BLS functions.

   The Assembler program unit TPRGASS calls the shareable Assembler module SHARASS. TPRGASS is defined in the PROGRAM statement and SHARASS in the MODULE statement.

```
TPRGASS  ZSTRT
          .
          .
         L     R15,=V(SHARASS)
         BALR  R14,R15
          .
          .
         ZCALL KDCS,PARM1,PARM2
          .
          .
         END


SHARASS  CSECT PUBLIC
         USING *,R15
          .
          .
         BR    R14
          .
          .
         END
```

2. In this example the UTM application uses the BLS functions.

   Both modules, TPRGASS and SHARASS, are linked using the BINDER to form an LLM (link and load module) with the name LMODASS with private and public slices.

   The LOAD-MODULE statement defines the name, version and attributes of the LMODASS load module:

```
LOAD-MODULE    LMODASS, VERSION=xxx, LIB= lmod-lib,
               LOAD-MODE = (POOL, poolname, STARTUP)
```

   The PROGRAM statement defines the name and attributes of the program unit and specifies a name for the load module:

```
PROGRAM    TPRGASS, COMP= ASSEMB
           LOAD-MODULE= LMODASS
```

3 Macros

This chapter describes macros made available by *open*UTM for the programming of Assembler program units.

ZSTRT - Start program and pass parameters

The ZSTRT macro must be the first statement in an *open*UTM program unit or subroutine. It performs the following functions:

– generate CSECT with entry point

– save register contents

– assign and load base address register

– generate save area

– load address of save area in register 13 and save parameter address

– generate C-type constant with name of program unit or subroutine

– equate register numbers with R0-R15

Format

Name	Operation	Operands
name	ZSTRT	[BASIS={(r1) | Rr1}] [,REGS=n] [,PARM={savpar | (r2) | Rr2}] [,ILCS={YES | NO}]

Meanings of the operands

name is the name of the program unit or subroutine.

BASIS=(*r1*) *r1* is the number of the register to be assigned as the base register.

BASIS=Rr1	base address register; Rr1 is the name of this register (formed from “R” and the register number), e.g. (3) or R3 (optional). If there are two or more base address registers, the number refers to the first one. The registers are numbered consecutively. The register numbers must be between 3 and 12. Default value: (12) or R12
REGS=*n*	“*n*” base address registers are used (optional). Default value: 1
PARM=*savpar*	*savpar* is the address of a word in which the contents of register 1 are saved (optional).
(*r2*)	*r2* is the number of a register into which register 1 is to be reloaded.
Rr2	Rr2 is the register name (formed from “R” and the register number) into which register 1 is to be reloaded. Default: register 1 is not saved. **Note** Register 1 contains the address of the parameter list. It is not changed by ZSTRT.
ILCS=YES/NO	If YES is specified, the ILCS convention is used for program linkage. In this case, the program unit must be generated with COMP=ILCS. Default value: NO

Possible MNOTES

Module name missing.

Syntax error in parameter.

ZCALL - Call UTM or subroutine

The ZCALL macro performs the following functions:

– generate V-type constants with the destination address
– store return address
– generate parameter list
– branch to subroutine

Format

Name	Operation	Operands
[name]	ZCALL	{ KDCS | subnam | (r1) } [,{ par | /par | (r2) }] [,...]

Meanings of the operands

name is the symbolic address of the macro (optional).

KDCS is the UTM function call.

subnam is the call of subroutine with the name *subnam* (destination address).

(*r1*) is the number or name of a register with the destination address.

par The address of *par* is entered in the parameter list (optional).
A maximum of up to 78 parameters are allowed.
If no operand is specified, then register 1 remains unchanged in conjunction with ILCS=NO, and it is set to 0 with ILCS=YES.
Otherwise, ZCALL writes the address of the parameter list in register 1.

/par The address of *par*, defined with a DSECT, is entered.

(*r2*) The address stored in register *r2* is entered.
The parameter list must be specified contiguously.
Default: no parameter list is generated and register 1 remains unchanged.

Possible MNOTEs

Branch address is missing.

Syntax error in parameter.

ZEND - Terminate subroutine

ZEND controls the return to the calling program. A return code can be passed at the same time. ZEND performs the following functions:

– reload registers

– return to the calling program

– transfer return code

Format

Name	Operation	Operands
[name]	ZEND	[RC={num / (r1)}]

Meanings of the operands

[*name*]	is the symbolic address of the macro (optional).
RC=	is the return code specification to be passed to the calling program in register 15.
num	1 to 4096.
(*r1*)	The return code is moved from register *r1* to register 15. Use of register 13 is prohibited.
	Default: no return code, register 15 is cleared.

Possible MNOTEs

Operand has more than 5 positions (return code larger than 5 positions).

Numeric operand portion incorrect (alpha characters in register notation).

Return code greater than 4096.

Register number greater than 15.

Register 13 used.

RC= is not supported when ILCS=YES is specified.

4 Assembler data structures

The following data structures simplify programming. They make it easier for you to modify or swap program units:

KCAPROA	Defines an optional second parameter area for the APRO call. This area is used for selecting specific OSI TP function combinations.
KCATA	Defines the UTM attribute functions for FHS.
KCKBA	Creates a DSECT with the structure of the KB header in the KDCS communication area. A KB program area, which you have to define yourself, can be connected to this KB header. UTM passes the address of the real KB communication area in a parameter list (see page 5). Calling with KCKBA C suppresses the DSECT statement.
KCPAA	Creates a DSECT with the structure of the KDCS parameter area. It is best to locate the KDCS parameter area in the SPAB. For how to address the SPAB, see page 5. Calling with KCPAA C suppresses the DSECT statement. Calling with KCPAA PREFIX=*xx*: This selects a different prefix than the default prefix KC. This function is necessary if, for example, the layout of a second KDCS parameter area is required for an INFO call with KCOM=CK. Note that KCPAA must be aligned on a doubleword boundary.
KCOPA	Defines constants with the names of the KDCS operations.
KCDFA	Defines the KDCS screen output functions.
KCINIA	Defines a second parameter area for the INIT call (necessary only with INIT PU). In this parameter area UTM returns the information requested with INIT PU.
KCINFA	Defines data structures for returning information with the UTM call INFO.
KCMSGA	Defines a DSECT with the data structure of the UTM messages.
KCDADA	Defines data structures for the DADM call.
KCPADA	Defines data structures for the PADM call.
KCINPA	Defines a DSECT for setting up the parameter area for the event exit INPUT.

KCCFA Defines the second parameter passed by UTM for the event exit INPUT. In this parameter UTM passes the contents of the control fields of screen formats to the program unit. For this reason, this second parameter is also known as the control fields area.

These data structures are stored in the SYSLIB.UTM.050.ASS library.

The data structures for KCKBA and KCPAA are presented in the following.

Data structure KCKBA

```
************************************************************************
*                                                                      ***
*      COPYRIGHT (C) SIEMENS NIXDORF INFORMATIONSSYSTEME AG 1992       ***
*                    ALL RIGHTS RESERVED                               ***
*      COPYRIGHT (C) SIEMENS AG 1998 ALL RIGHTS RESERVED               ***
*                                                                      ***
************************************************************************
*      SIEMENS AG openUTM  5.0                                         ***
MACRO            500       980708   51311101
&NAME    KCKBA &C,&CSECT=NO
         SPACE
************************************************************************
*                                                                      *
*        KDCS COMMUNICATION AREA     (KB)                              *
*                                                COPY:  KCKBA          *
************************************************************************
         SPACE
************************************************************************
*        KDCS KB HEADER                                                *
************************************************************************
         SPACE
         AIF   (&C EQ 'C').P1
&NAME    DSECT                          KDCS COMMUNICATION AREA
         AGO   .P2
.P1      ANOP
         AIF   (&CSECT EQ 'NO').P3
&NAME    AMODE ANY
&NAME    RMODE ANY
&NAME    CSECT
         AGO   .P2
.P3      ANOP
&NAME    DS     0D                      KDCS COMMUNICATION AREA
.P2      ANOP
         SPACE
KCKBKOPF DS    0CL68                    KB HEADER
KCBENID  DS    D                        . USER IDENTIFICATION
KCVORG   DS    0CL24                    . CONVERSATION-SPECIFIC DATA:
KCTACVG  DS    CL8                        . TRANSACTION CODE
KCDATVG  DS    0CL9                       . DATE:
KCTAGVG  DS    CL2                           . DAY
KCMONVG  DS    CL2                           . MONTH
KCJHRVG  DS    CL2                           . YEAR
KCTJHVG  DS    CL3                           . DAY OF YEAR
KCUHRVG  DS    0CL6                        . TIME:
KCSTDVG  DS    CL2                           . HOUR
```

```
KCMINVG  DS    CL2                        . MINUTE
KCSEKVG  DS    CL2                        . SECOND
KCKNZVG  DS    CL1                       . CONVERSATION ID
KCAKTUEL DS    OCL16                   . DATA TO CURRENT PROGRAM RUN:
KCTACAL  DS    CL8                       . TRANSACTION CODE
KCUHRAL  DS    OCL6                      . TIME:
KCSTDAL  DS    CL2                          . HOUR
KCMINAL  DS    CL2                          . MINUTE
KCSEKAL  DS    CL2                          . SECOND
KCAUSW   DS    CL1                     . A = CARD IN READER
KCTAIND  DS    CL1                     . TRANSACTION INDICATOR
KCLOGTER DS    CL8                     . NAME OF UTM TERMINAL (LTERM)
KCTERMN  DS    CL2                     . TERMINAL MNEMONIC
KCLKBPB  DS    H                       . MAXIMUM LENGTH OF
*                                      . KB PROGRAM AREA
KCSTA    DS    OCL3                    . STACK INFORMATION:
KCHSTA   DS    H                       . CURRENT STACK LEVEL
KCDSTA   DS    CL1                     . CHANGE IN STACK LEVEL
         DS    CL1
KCPRIND  DS    CL1                     . PROGRAM INDICATOR
KCOF1    DS    CL1                     . OSI-TP FUNCTION1
KCCP     DS    CL1                     . CLIENT PROTOCOL
KCTARB   DS    CL1                     . TRANSACTION IS MARKED ROLLBACK
KCYEARVG DS    CL4                     . YEAR START CONVERSATION
         SPACE
         DS    CL12
         SPACE
*************************************************************************
*        KDCS RETURN AREA                                               *
*************************************************************************
         SPACE
KCRFELD  DS    OCL24                 KDCS RETURN AREA
KCRI     DS    CL2                     . RETURN IDENTIFICATION
*                                      . (NOT USED)
         ORG   KCRI
KCRDF    DS    H                       . RETURN DEVICE FEATURE
KCRLM    DS    H                       . RETURN LENGTH
KCRINFCC DS    CL3                     . INFO CALL ERROR CODE
         ORG   KCRINFCC
KCRSTATE DS    OCL2                    . CONVERSATION AND
*                                      . TRANSACTION STATUS
KCRST    EQU   KCRSTATE
KCVGST   DS    CL1                     . CONVERSATION STATUS
KCTAST   DS    CL1                     . TRANSACTION STATUS
         DS    CL1                     . NOT USED
         ORG   KCRINFCC
KCRSIGN  DS    OCL3                    . STATUS OF SIGN-ON:
KCRSIGN1 DS    CL1                     . PRIMARY CODE
```

```
KCRSIGN2 DS    CL2                    . SECONDARY CODE
KCRMGT   DS    CL1                    . RETURN INFO MGET
KCRC     DS    0CL8                   . RETURN CODES:
KCRCCC   DS    CL3                      . KDCS ERROR CODE
KCRCKZ   DS    CL1                      . INDICATOR
*                                       . P=PRODUCTION, T=UTM-T
KCRCDC   DS    CL4                      . ADDITIONAL ERROR CODE FROM
*                                       . UTM (NOT COMPATIBLE)
KCRMF    DS    CL8                   .RETURN MESSAGE FORMAT
KCRPI    DS    CL8                    . RETURN CONVERSATION ID
         ORG   KCRPI
KCRUS    DS    CL8                    . RETURN USER (SIGN ST)
         SPACE
*************************************************************************
*        KDCS KB PROGRAM AREA                                           *
*************************************************************************
         SPACE
KCKBPRG  EQU   *                     KB PROGRAM AREA
         SPACE 2
         MEND
```

Data structure KCPAA

```
************************************************************************
*                                                                      ***
*      COPYRIGHT (C) SIEMENS NIXDORF INFORMATIONSSYSTEME AG 1992       ***
*                    ALL RIGHTS RESERVED                               ***
*      COPYRIGHT (C) SIEMENS AG 1998 ALL RIGHTS RESERVED               ***
*                                                                      ***
************************************************************************
*      SIEMENS AG openUTM  5.0                                         ***
         MACRO            500       980708   51311102
&NAME    KCPAA &C,                                                      -
               &PREFIX=KC
         SPACE
************************************************************************
*                                                                      *
*        KDCS STANDARD PRIMARY WORKING AREA                            *
*        (SPAB)                                                        *
*                                                                      *
************************************************************************
         SPACE
         AIF   (&C EQ 'C').P1
&NAME    DSECT                KDCS STANDARD PRIMARY WORKING AREA (SPAB)
         SPACE
         AGO   .P2
.P1      ANOP
&NAME    DS    OF             KDCS STANDARD PRIMARY WORKING AREA (SPAB)
.P2      ANOP
&PREFIX.OP     DS    CL4            . OPERATION CODE
&PREFIX.OM     DS    CL2            . OPERATION MODIFICATION
&PREFIX.LA     DS    H              . LENGTH OF AREA
&PREFIX.LKBPRG EQU   &PREFIX.LA          . LENGTH OF KB PROGRAM AREA
&PREFIX.LM     DS    H              . LENGTH OF MESSAGE
&PREFIX.LPAB   EQU   &PREFIX.LM          . LENGTH OF SPAB
&PREFIX.RN     DS    CL8            . REFERENCE NAME
*                                   . TAC/LTERM/STORAGE AREA
&PREFIX.MF     DS    CL8            . MESSAGE FORMAT
&PREFIX.LT     EQU   &PREFIX.MF         . NAME OF UTM TERMINAL
&PREFIX.US     EQU   &PREFIX.MF         . USER ID
&PREFIX.PA     EQU   &PREFIX.MF         . NAME OF THE PARTNER
*                                       . APPLICATION
&PREFIX.DF     DS    H              . SCREEN FUNCTION
&PREFIX.LI     EQU   &PREFIX.DF         . LENGTH OF INIT AREA
&PREFIX.EXTENT DS    OCL14          . EXTENTION SINCE V3.0
*
&PREFIX.DPUT   DS    OCL10          . FOR DPUT-CALL:
&PREFIX.MOD    DS    CL1            . MODE: A=ABSOLUTE,R=RELATIVE
```

```
*                                       .       SPACE= NO TIME
&PREFIX.TAG    DS    CL3                . DAY
&PREFIX.STD    DS    CL2                . HOUR
&PREFIX.MIN    DS    CL2                . MINUTE
&PREFIX.SEK    DS    CL2                . SECOND
               DS    CL4                . NOT USED
*
         ORG   &PREFIX.EXTENT
&PREFIX.APRO   DS    OCL8               . FOR APRO-CALL:
&PREFIX.PI     DS    CL8                . CONVERSATION ID
&PREFIX.OF     DS    CL1                . OSI-TP FUNCTIONS
               DS    CL5                . NOT USED
*
         ORG   &PREFIX.EXTENT
&PREFIX.PADM   DS    OCL11              . FOR PADM-CALL:
&PREFIX.ACT    DS    CL3                . ACTION (ON/OFF/CON/DIS)
&PREFIX.ADRLT  DS    CL8                . ADDRESSED LTERM, DESTINATION
               DS    CL3                . (NOT USED)
*
         ORG   &PREFIX.MF
&PREFIX.MCOM   DS    OCL24              . FOR MCOM-CALL:
&PREFIX.POS    DS    CL8                . DESTINATION IN POSITIVE CASE
&PREFIX.NEG    DS    CL8                . DESTINATION IN NEGATIVE CASE
&PREFIX.COMID  DS    CL8                . COMPLEX IDENTIFICATION
*
         ORG   &PREFIX.EXTENT
&PREFIX.SGCL   DS    OCL12              . FOR SIGN CL CALL:
&PREFIX.LANGID DS    CL2                . LANGUAGE ID
&PREFIX.TERRID DS    CL2                . TERRITORY ID
&PREFIX.CSNAME DS    CL8                . CODED CHARACTER SET NAME
               DS    CL2                . (NOT USED)
*
         ORG
         SPACE
&PREFIX.PAREND EQU   *                  . END OF PARAMETER AREA
         SPACE 2
         MEND
```

Index

Contents

*open*UTM V5.0 (BS2000/OSD)

Programming Applications with KDCS for Assembler

Target group
This manual is intended to support programmers of *open*UTM applications in Assembler.

Contents
This manual describes the language-specific features involved in writing Assembler program units. It supplements the Core Manual “Programming Applications with KDCS for COBOL, C and C++”.

Edition: November 1998

File: utm_ass.pdf

Copyright © Siemens AG 1998.

All rights reserved.
Delivery subject to availability; right of technical modifications reserved.

All hardware and software names used are trademarks of their respective manufacturers.

Information on this document

On April 1, 2009, Fujitsu became the sole owner of Fujitsu Siemens Computers. This new subsidiary of Fujitsu has been renamed Fujitsu Technology Solutions.

This document from the document archive refers to a product version which was released a considerable time ago or which is no longer marketed.

Please note that all company references and copyrights in this document have been legally transferred to Fujitsu Technology Solutions.

Contact and support addresses will now be offered by Fujitsu Technology Solutions and have the format *...@ts.fujitsu.com*.

The Internet pages of Fujitsu Technology Solutions are available at *http://ts.fujitsu.com/*...
and the user documentation at *http://manuals.ts.fujitsu.com*.

Copyright Fujitsu Technology Solutions, 2009

Hinweise zum vorliegenden Dokument

Zum 1. April 2009 ist Fujitsu Siemens Computers in den alleinigen Besitz von Fujitsu übergegangen. Diese neue Tochtergesellschaft von Fujitsu trägt seitdem den Namen Fujitsu Technology Solutions.

Das vorliegende Dokument aus dem Dokumentenarchiv bezieht sich auf eine bereits vor längerer Zeit freigegebene oder nicht mehr im Vertrieb befindliche Produktversion.

Bitte beachten Sie, dass alle Firmenbezüge und Copyrights im vorliegenden Dokument rechtlich auf Fujitsu Technology Solutions übergegangen sind.

Kontakt- und Supportadressen werden nun von Fujitsu Technology Solutions angeboten und haben die Form *...@ts.fujitsu.com*.

Die Internetseiten von Fujitsu Technology Solutions finden Sie unter *http://de.ts.fujitsu.com/*..., und unter *http://manuals.ts.fujitsu.com* finden Sie die Benutzerdokumentation.

Copyright Fujitsu Technology Solutions, 2009