

Übung Nr. 3

```
TCPIP MSG10 ==> SOURCE DATA SET = SYS1.LOCAL.VTAMLST(USSTCPIP)

02/04/01                W E L C O M E   T O                16:54:39

      SSSSSS   //   3333333  9999999  0000000
     SS       //   33   33   99   99   00   00
    SS       //           33   99   99   00   00
   SSSS     //   33333  9999999  00   00
    SS     //           33           99   00   00
   SS //   33   33   99   99   00   00
 SSSSSS //   3333333  9999999  0000000

YOUR TERMINAL NAME IS :                YOUR IP ADDRESS IS : 217.002.090.033

      APPLICATION DEVELOPMENT SYSTEM
      OS/390 RELEASE 2.7.0

===> ENTER "L " FOLLOWED BY THE APPLID YOU WISH TO LOGON TO.  EXAMPLE "L TSO"
      FOR TSO/E OR "L C001" FOR THE CICSC001 CICS APPLICATION.

l tso
```

Wir wollen ein einfaches „Hello World“ CICS Programm schreiben.

Wir erinnern uns: TSO ist ein OS/390 Subsystem. CICS ist ein weiteres OS/390 Subsystem. Jedes der beiden Subsysteme hat eine eigene Benutzerschnittstelle (eine eigene Shell). Um eine CICS Anwendung zu erstellen, müssen wir mit beiden Subsystemen arbeiten: Mit TSO, um die Anwendung zu erzeugen, und mit CICS, um die Anwendung (unter dem CICS Subsystem) auszuführen. Da OS/390 ein Multi-User Betriebssystem ist (multisession-fähig), können wir gleichzeitig eine TSO Session und eine CICS Session auf unserem NT Arbeitsplatzrechner laufen lassen. Jede Session läuft in einem eigenen Fenster.

Wir starten unseren 3270 Emulator zunächst für eine TSO Session, und später ein zweites Mal für eine CICS Session.

Wir fangen mit der TSO Session an und loggen uns ein.

Menu RefList Utilities Help

Data Set Utility

A Allocate new data set	C Catalog data set
R Rename entire data set	U Uncatalog data set
D Delete entire data set	S Data set information (short)
blank Data set information	M Allocate new data set
	V VSAM Utilities

ISPF Library:

Project . . SPRUTH
Group . . . CICS
Type TEST04

Other Partitioned, Sequential or VSAM Data Set:

Data Set Name . . .
Volume Serial . . . (If not cataloged, required for option "C")
Data Set Password . . (If password protected)

Option ==>

F1=Help F3=Exit F10=Actions F12=Cancel

Wir arbeiten uns zum Data Set Utility Screen vor und erzeugen (Allocate) einen neuen Partitioned Data Set SPRUTH.CICS.TEST04

Menu RefList Utilities Help

Data Set Utility

A Allocate new data set	C Catalog data set
R Rename entire data set	U Uncatalog data set
D Delete entire data set	S Data set information (short)
blank Data set information	M Allocate new data set
	V VSAM Utilities

ISPF Library:

Project . . SPRUTH
Group . . .
Type

Other Partitioned, Sequential or VSAM Data Set:

Data Set Name . . . 'SPRUTH.LIB'
Volume Serial . . . (If not cataloged, required for option "C")
Data Set Password . . (If password protected)

Option ==>

F1=Help F3=Exit F10=Actions F12=Cancel

Außerdem brauchen wir noch einen Partitioned Data Set mit dem vorgegebenen Namen „userid.LIB“, dessen Members von der Entwicklungsumgebung während der CICS Programmentwicklung mit Daten gefüllt werden. Für den Benutzer Spruth hat dieser den Namen „SPRUTH.LIB“.

Dieser Name besteht aus einem „Projekt“ Begriff und einem „Group“ Begriff. Es fehlt der „Type“ Begriff. Wenn wir das Typ Feld leer lassen, kann es sein, daß TSO dies nicht akzeptiert. Zur Abhilfe tragen wir den Namen SPRUTH.LIB (mit Hochzeichen) in die Zeile „Data Set Name“ ein. Damit wird auch dieser Data Set allocated.

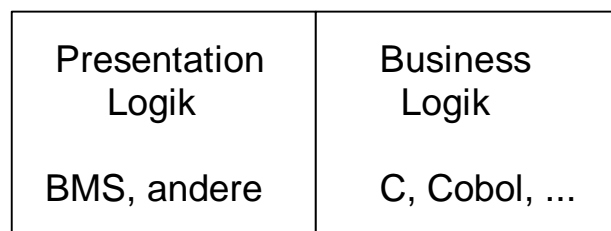
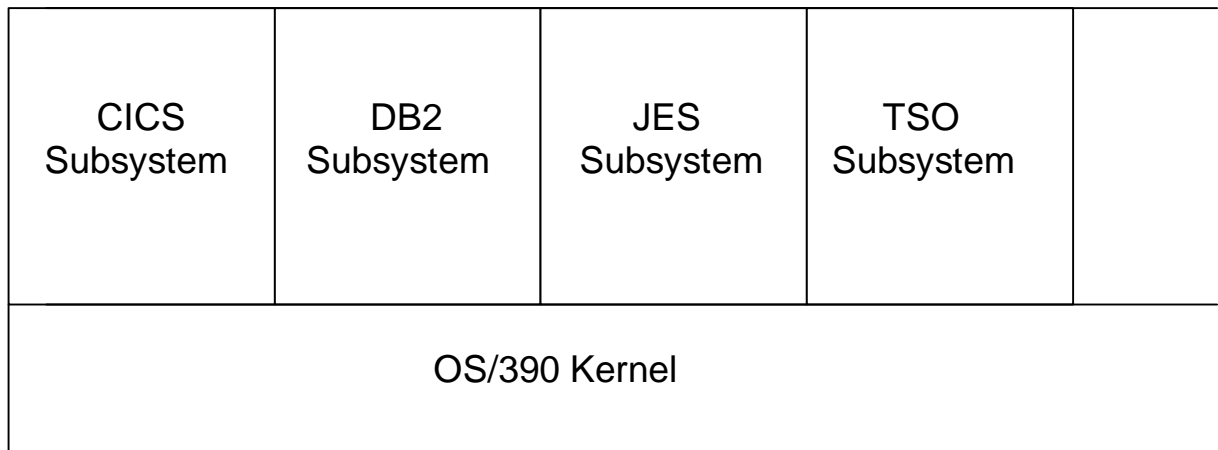
Unsere Anwendung besteht aus zwei Programmteilen und einem JCL Script für die Übersetzung. Wir erstellen diese als Members in dem neuen Partitioned Data Set SPRUTH.CICS.TEST04.

Die Entwicklung von CICS Programmen erlaubt viele Freiheiten. Falls gewünscht, kann beliebig wilder Spagetti Code erzeugt werden.

Ein sauber strukturiertes CICS Programm besteht aus zwei Teilen: Business Logik und Presentation Logik. Business Logik ist der Teil, in dem Berechnungen erfolgen und Daten in einer Datenbank gelesen/geschrieben werden. Presentation Logik ist der Teil, in dem die Ergebnisse der Berechnungen so aufgearbeitet werden, daß sie dem Benutzer in einer ansprechenden Art auf dem Bildschirm dargestellt werden.

Business Logik wird in Sprachen wie C++, COBOL, PL/1 usw. geschrieben. Für die Presentation Logik gibt es viele Alternativen. Die modernste Alternative benutzt Java Server Pages und einen Web Application Server um den Bildschirminhalt innerhalb eines Web Browsers darzustellen. Die älteste (und einfachste) Alternative verwendet das CICS BMS (Basic Mapping Support) Subsystem. BMS Programme werden in der BMS Sprache geschrieben. In unserem Beispiel wird die Business Logik in C, und die Presentationlogik in BMS geschrieben.

Wir fangen mit dem letzteren an, rufen den Edit Entry Panel Screen auf (siehe Bild 23 in unserem TSO Beispiel) und editieren ein Member „MAP04“ für den neu angelegten Partitioned Data Set SPRUTH.CICS.TEST04.



CICS Anwendung

```

File Edit Confirm Menu Utilities Compilers Test Help
-----
EDIT          SPRUTH.CICS.TEST04(MAP04) - 01.02          Columns 00001 00072
***** ***** Top of Data *****
==MSG> -CAUTION- Profile changed to CAPS ON (from CAPS OFF) because the
==MSG>          data does not contain any lower case characters.
==MSG> -Warning- The UNDO command is not available until you change
==MSG>          your edit profile using the command RECOVERY ON.
000001 //PREPARE JOB ( ),CLASS=A,MSGCLASS=H,MSGLEVEL=(1,1),NOTIFY=&SYSUID
000002 //ASSEM EXEC DFHMAPS,MAPNAME='S04SET',RMODE=24
000003 //SYSUT1 DD *
000004 S04SET DFHMSD TYPE=MAP,MODE=INOUT,LANG=C,STORAGE=AUTO,TIOAPFX=YES
000005 * MENU MAP.
000006 LABEL04 DFHMDI,SIZE=(24,80),CTRL=(PRINT,FREEKB)
000007 DFHMDF POS=(9,23),ATTRB=(ASKIP,NORM),LENGTH=34, X
000008 INITIAL='WELCOME TO THE MAGIC WORLD OF CICS'
000009 DFHMDF POS=(12,27),ATTRB=(ASKIP,NORM),LENGTH=26, X
000010 INITIAL='MAY THE FORCE BE WITH YOU!'
000011 DFHMSD TYPE=FINAL
000012 END
000013 /*
000014 //
Command ==> SUB          Scroll ==> PAGE
F1=Help F3=Exit F5=Rfind F6=Rchange F12=Cancel

```

Unser BMS Programm verwendet 3 Befehlstypen: DFHMSD, DFHMDI und DFHMDF.

Ein BMS Screen verwendet das 24 Zeilen x 80 Zeichen/Zeile 3270 Bildschirmformat. Eingabe und Ausgabe Daten werden als Felder innerhalb der 24 x 80 Matrix dargestellt, jeweils mit der Angabe: Zeilenadresse, Spaltenadresse und Feldlänge. Dies geschieht mit Hilfe des DFHMDF Befehls. Der DFHMDF Befehl in Zeile 000007 definiert ein Feld, welches in Zeile 9, Spalte 23 beginnt, 34 Zeichen lang ist, und mit dem Wert „Welcome to the magic World of CICS“ initialisiert wird.

Unser Beispiel BMS Programm enthält 2 derartige DFHMDF Befehle.

Der DFHMSD Befehl (Zeile 000004) definiert einen „Mapset“ mit dem Namen „S04SET“. Eine Transaktion involviert in der Regel mehrere unterschiedliche Screens, z.B einen Screen, in dem der Benutzer zu einer Eingabe aufgefordert wird und einen weiteren Screen, welcher die Ergebnisse der Anfrage wiedergibt. Alle Screens (Maps) eines Transaktionstyps werden zu einem Mapset zusammengefaßt.

Die einzelnen Maps (Screens) eines Mapsets werden durch den Befehl DFHMDI definiert und zur Kennzeichnung mit einer Label versehen. In unserem einfachen Hello World Beispiel besteht der Mapset aus einer einzigen Map, die in Zeile 000006 mit der Bezeichnung „Label04“ definiert wird.

Das Member SPRUTH.CICS.TEST04(MAP04) stellt in Wirklichkeit ein JCL Script dar. Im Gegensatz zu Bild 29 in unserem TSO Beispiel wird die zu verarbeitende File nicht mit INFILE='xxx.yyy.zzz' angegeben. Der JCL Befehl in Zeile 000003 „//SYSUT1 DD *“ besagt, daß die zu verarbeitende File unmittelbar danach folgt (Zeile 000004 bis 000012).

Wir geben auf der Kommandozeile den ISPF Befehl „SUB“ ein. Es wird die Prozedur DFHMAPS ausgeführt.

JCL findet den Member „DFHMAPS“ (Zeile 000002) in der Library SYS1.PROCLIB(DFHMAPS). Durch die Ausführung von DFHMAPS werden zwei Ausgabe Files erzeugt. Einmal wird der übersetzte BMS Quellcode in eine Member in einer MAPLIB mit dem Namen „CICSTS13.CICS.SDFHLOAD“ gestellt. Hier kann ihn die BMS Komponente des CICS Subsystems später finden.

```

File Edit Confirm Menu Utilities Compilers Test Help
-----
VIEW          SPRUTH.LIB(S04SET) - 01.00                      Columns 00001 00072
***** ***** Top of Data *****
==MSG> -Warning- The UNDO command is not available until you change
==MSG>          your edit profile using the command RECOVERY ON.
000001 union
000002 {
000003 struct {
000004     char          dfhms1Y12";
000005     } label04i;
000006
000007 struct {
000008     char          dfhms2Y12";
000009     } label04o;
000010
000011 } label04;
000012
***** ***** Bottom of Data *****

Command ===>
F1=Help      F3=Exit      F5=Rfind      F6=Rchange  F12=Cancel

                                Scroll ===> PAGE

```

Die zweite Ausgabe File wird als Member „S04SET“ in den Partitioned Data Set „SPRUTH.LIB“ gestellt.

Dies ist als Hilfe für die Erstellung des (in C zu schreibenden) Business Logik Programms gedacht. Alle Ein- und Ausgabedaten, die auf dem Bildschirm wiedergegeben werden sollen, werden ja bereits durch das BMS Programm definiert. Das Business Logik Programm bearbeitet diese Daten als C-Strukturen, und diese C-Strukturen werden von DFHMAPS während der Übersetzung von MAP04 gleich miterzeugt und in SPRUTH.LIB(S04SET) abgespeichert.

Beim Erzeugen des Business Logik Programms bietet es sich an, SPRUTH.LIB(S04SET) als Basis zu benutzen und zu vervollständigen. Hiermit ist sichergestellt (und Fehler ausgeschlossen), daß Presentation Logik und Business Logik identische Datendarstellungen benutzen.

In SPRUTH.LIB(S04SET) fällt das Sonderzeichen „Y“ auf. Dies hat etwas mit dem „eckige Klammern Problem“ bei der Erstellung von C-Programmen zu tun.

Wir erinnern uns: TSO speichert alle Daten im EBCDIC Format ab. Im EBCDIC Format gibt es zwei Darstellungen für die eckigen Klammern, nämlich

Character	ASCII hex	Richtiges EBCDIC hex	Falsches EBCDIC hex
Left Square ([)	x'5B'	x'AD'	x'BA'
Right Square (])	x'5D'	x'BD'	x'BB'
Character	ASCII hex	Richtiges EBCDIC hex	Falsches EBCDIC hex
Left Square ([)	x'5B'	x'AD'	x'BA'
Right Square (])	x'5D'	x'BD'	x'BB'

Der 3270 Emulator wird häufig hex BA und hex BB als exkige Klammern " [" und "] " darstellen, während hex AD und hex BD als Y und " dargestellt werden. Bei den meisten 3270 Emulatoren kann man die Darstellung ändern; hier ist das aber nicht erfolgt.

Wichtig ist, das Problem zu verstehen. Es gibt mehrere Möglichkeiten es anzugehen. Das Primitivverfahren ist folgendes:

Bei der Programmeingabe normal die Symbole "[" und "] " verwenden. Diese werden dann fälschlicherweise als x'BA und x'BB abgespeichert.

Nach Fertigstellung der Programmeingabe zwei globale ISPF "Change" Kommandos eingeben. Dies erfolgt durch Eingabe in der Kommandozeile:

```
C [ x'ad' all  
C ] x'bd' all
```

Es wird das Symbol "[" durch den hexadezimalen Wert hex "ad" ersetzt. Ähnlich für "] " durch hex "bd".

Statt "C" kann auch "change" eingegeben werden.

TCPIP MSG10 ==> SOURCE DATA SET = SYS1.LOCAL.VTAMLST(USSTCPIP)

02/04/01

W E L C O M E T O

08:55:40

```
      SSSSSS // 3333333 9999999 0000000
SS      // 33 33 99 99 00 00
SS      //      33 99 99 00 00
SSSS    // 33333 9999999 00 00
SS      //      33      99 00 00
SS      // 33 33 99 99 00 00
SSSSSSS // 3333333 9999999 0000000
```

YOUR TERMINAL NAME IS :

YOUR IP ADDRESS IS : 217.002.089.066

APPLICATION DEVELOPMENT SYSTEM
OS/390 RELEASE 2.7.0

==> ENTER "L " FOLLOWED BY THE APPLID YOU WISH TO LOGON TO. EXAMPLE "L TSO"
FOR TSO/E OR "L C001" FOR THE CICSC001 CICS APPLICATION.

l C001

Wir loggen uns mit C001 ein und rufen damit das OS/390 CICS Subsystem auf.

CICS C001 A06C001

IBM DEMONSTRATION SYSTEM 24:00:00

```
*****\ *****\ *****\ *****\
*****\ *****\ *****\ *****\
**\\\\\\**\ **\\\\ **\\\\\\**\ **\\\\\\**\
**\      \\\ **\ **\      \\\ **\      \\\
**\      **\ **\      **\ *****\
**\      **\ **\      **\ *****\
**\      **\ **\      **\ \\\\\\\**\
**\      **\ **\      **\ **\      **\
*****\ *****\ *****\ *****\
*****\\ *****\\ *****\\ *****\\
\\\\\\\\\\ \\\\\\\\\ \\\\\\\\\ \\\\\\\\\ TM
```

Der CICS Eingangsbildschirm erscheint.

Enter

```
CICS C001 A06C001      IBM DEMONSTRATION SYSTEM 24:00:00
```

```
*****\  *****\  *****\  *****\  
*****\  *****\  *****\  *****\  
**\\\\\\**\  **\\\\\  **\\\\\\**\  **\\\\\\**\  
**\      \\\  **\      \\\  **\      \\\  **\      \\  
**\      **\  **\      **\  **\      **\  **\      **\  
**\      **\  **\      **\  **\      **\  **\      **\  
**\      **\  **\      **\  **\      **\  **\      **\  
**\      **\  **\      **\  **\      **\  **\      **\  
*****\  *****\  *****\  *****\  
*****\  *****\  *****\  *****\  
\\\\\\\\\  \\\\\\\  \\\\\\\  \\\\\\\  TM
```

```
DFHAC2001 02/04/01 11:01:01 A06C001 Transaction ' ' is not recognized. Check  
that the transaction name is correct.
```

Eine Fehlermeldung (belanglos) erscheint. Mit Tab bewegen wir den Cursor auf die unterste Zeile.

```
CICS C001 A06C001      IBM DEMONSTRATION SYSTEM 24:00:00
```

```
*****\  *****\  *****\  *****\  
*****\  *****\  *****\  *****\  
**\\\\\\**\  **\\\\\  **\\\\\\**\  **\\\\\\**\  
**\      \\\  **\      \\\  **\      \\\  **\      \\  
**\      **\  **\      **\  **\      **\  **\      **\  
**\      **\  **\      **\  **\      **\  **\      **\  
**\      **\  **\      **\  **\      **\  **\      **\  
**\      **\  **\      **\  **\      **\  **\      **\  
*****\  *****\  *****\  *****\  
*****\  *****\  *****\  *****\  
\\\\\\\\\  \\\\\\\  \\\\\\\  \\\\\\\  TM
```

```
DFHAC2001 02/04/01 11:01:01 A06C001 Transaction ' ' is not recognized. Check  
that the transaction name is correct. ceda display group(*)
```

CICS erwartet, daß man eine (von vielen) Transaktionen aufruft. Die unterschiedlichen Transaktionen werden normalerweise durch die Eingabe einer aus vier Zeichen bestehenden Tranaktion ID aufgerufen.

Der CICS Kommandointerpreter ist ebenfalls als Transaktion implementiert. Er wird mit der Transaktion ID "CEDA" aufgerufen, gefolgt von einer Parameterliste, welche CICS Kommandos sowie Eingabedaten enthält.

Als Beispiel geben wir das Kommando "ceda display group(*)" ein

```

display group(*)
ENTER COMMANDS
GROUP
AOR2TOR
ARTT
ATC
CBPS
CEE
CICREXX
CSQ
CSQCKB
CSQSAMP
CTA1TCP
C001EZA
C001TCP
DBA1
DFH$ACCT
DFH$AFFY
DFH$AFLA
+ DFH$BABR

RESULTS: 1 TO 17
PF 1 HELP      3 END 4 TOP 5 BOT 6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL

SYSID=C001 APPLID=A06C001
TIME: 00.00.00 DATE: 01.035

```

Wenn unter CICS Anwendungen (Transaktionen) installiert werden, dann wird für jede Transaktion eine "Group" (Gruppe) angelegt. In der Group befinden sich typischerweise Members wie das Anwendungsprogramm selbst, der dazugehörige Mapset sowie ein Eintrag, der die Transaktion mit einer (normalerweise 4-stelligen) TRID (Transaktions ID) verknüpft.

Die Liste der bereits installierten Gruppen ist 17 Screens lang (läßt sich mit F8 und F7 anschauen).

```

CEDA DEFINE MAPSET(S04SET) GROUP(SPRUTH4)
ENTER COMMANDS
GROUP
AOR2TOR
ARTT
ATC
CBPS
CEE
CICREXX
CSQ
CSQCKB
CSQSAMP
CTA1TCP
C001EZA
C001TCP
DBA1
DFH$ACCT
DFH$AFFY
DFH$AFLA
+ DFH$BABR

RESULTS: 1 TO 17
PF 1 HELP      3 END 4 TOP 5 BOT 6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL

SYSID=C001 APPLID=A06C001
TIME: 00.00.00 DATE: 01.037

```

Wir definieren für unsere Transaktion eine eigene Gruppe "SPRUTH4" und den dazugehörigen Mapset als "S04SET". Hierzu überschreiben wir die oberste Zeile, die als Kommandozeile dient, mit dem CEDA Befehl "CEDA DEFINE MAPSET(S04SET) GROUP(SPRUTH4)". Bitte Großbuchstaben benutzen!

Enter

```

CEDA DEFINE MAPSET(S04SET) GROUP(SCRUTH4)
OVERTYPE TO MODIFY                                CICS RELEASE = 0530
CEDA DEFine Mapset( S04SET  )
  Mapset      : S04SET
  Group       : SCRUTH4
  Description  ==>
  RESident    ==> No                No | Yes
  USAge       ==> Normal            Normal | Transient
  USElpacopy  ==> No                No | Yes
  Status      ==> Enabled           Enabled | Disabled
  RSl         : 00                  0-24 | Public

I New group SCRUTH4 created.

DEFINE SUCCESSFUL                                SYSID=C001 APPLID=A06C001
PF 1 HELP 2 COM 3 END                          TIME: 00.00.00 DATE: 01.037
6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL

```

CICS teilt uns mit, daß die neue Gruppe "SCRUTH4" erstellt wurde. Führen wir nochmals den Befehl CEDA DISPLAY GROUP(*) aus, so finden wir in der dargestellten Liste den Eintrag SCRUTH4.

```

CEDA INSTALL GROUP(SCRUTH4)
OVERTYPE TO MODIFY                                CICS RELEASE = 0530
CEDA DEFine Mapset( S04SET  )
  Mapset      : S04SET
  Group       : SCRUTH4
  Description  ==>
  RESident    ==> No                No | Yes
  USAge       ==> Normal            Normal | Transient
  USElpacopy  ==> No                No | Yes
  Status      ==> Enabled           Enabled | Disabled
  RSl         : 00                  0-24 | Public

I New group SCRUTH4 created.

DEFINE SUCCESSFUL                                SYSID=C001 APPLID=A06C001
PF 1 HELP 2 COM 3 END                          TIME: 00.00.00 DATE: 01.037
6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL

```

Bis jetzt haben wir dem CICS Subsystem mitgeteilt, daß eine neue Gruppe „SCRUTH4“ und in ihr ein Mapset „S04SET“ existiert. Als nächster Schritt muß SCRUTH4 in der Anwendungsprogramm-bibliothek von CICS installiert werden. Dies geschieht mit dem dem INSTALL Kommando des CEDA Command-Line Interpreters. Er wird in die oberste Zeile eingegeben.

Enter

```

CEDA INSTALL GROUP(SPRUTH4)
OVERTYPE TO MODIFY
CEDA Install
All
Connection ==>
DB2Conn ==>
DB2Entry ==>
DB2Tran ==>
DOctemplate ==>
Enqmodel ==>
File ==>
Journalmodel ==>
LSrpool ==>
Mapset ==>
PARTitionset ==>
PARTner ==>
PROcesstype ==>
PROFile ==>
PROgram ==>
+ Requestmodel ==>

                                SYSID=C001 APPLID=A06C001
INSTALL SUCCESSFUL              TIME: 00.00.00 DATE: 01.037
PF 1 HELP                       3 END                    6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL

```

CEDA teilt mit, daß die Installation erfolgreich war.

Wir wechseln in das TSO Fenster zurück und rufen das Member SPRUTH.LIB(S04SET) auf

```

File Edit Confirm Menu Utilities Compilers Test Help
-----
VIEW          SPRUTH.LIB(S04SET) - 01.00                      Columns 00001 00072
***** Top of Data *****
==MSG> -Warning- The UNDO command is not available until you change
==MSG>          your edit profile using the command RECOVERY ON.
000001 union
000002 {
000003 struct {
000004     char          dfhms1Y12";
000005     } label04i;
000006
000007 struct {
000008     char          dfhms2Y12";
000009     } label04o;
000010
000011 } label04;
000012
***** Bottom of Data *****

Command ==>                                         Scroll ==> PAGE

```

Bei der Übersetzung des MAP BMS Quellcodes wurde für uns als Nebenprodukt im Member SPTUTH.LIB(S04SET) ein Template für unser Anwendungsprogramm erzeugt. Wir kopieren den Member in einen neuen Member PROG04 unteres Partitioned Data Sets SPRUTH.CICS.TEST04.

F3 Taste

```

Menu  Functions  Confirm  Utilities  Help
-----
VIEW          SPRUTH.LIB          Row 00001 of 00001
  Name      Prompt      VV MM      Changed      Size  Init  Mod  ID
  ----c---- S04SET1  *Viewed
           **End**

Command ==>
  F1=Help      F3=End      F4=Return    F5=Rfind     F6=Rchange   F10=Left
  F11=Right    F12=Cretriev

Scroll ==> PAGE

```

Wir kopieren dies Member, indem wir ein c (für copy) vor dem Member Namen eingeben.

Enter

```

RefList  Help
-----
                        COPY Entry Panel
                        More:      +

CURRENT from data set: 'WGT.LIB(WGTSET1)'

To Library              Replace option:
  Project . . . SPRUTH   Enter "/" to select option
  Group . . . CICS       Replace like-named members
  Type . . . TEST04

To Other Data Set Name
  Data Set Name . . .
  Volume Serial . . .   (If not cataloged)

NEW member name . . . PROG04 (Blank unless member to be renamed)

Options
  Sequential Disposition      Pack Option      SCLM Setting
  2 1. Mod                    1 1. Default    3 1. SCLM
  2. Old                      2. Pack         2. Non-SCLM

Command ==>
  F1=Help      F3=End      F4=Return    F5=Rfind     F6=Rchange   F10=Left
  F11=Right    F12=Cretriev

```

Wir geben als Ziel für den Kopiervorgang das Member PROG04 innerhalb unseres Partitioned Data Sets SPRUTH.CICS.TEST04 an. Dieses Member soll unser C Programm enthalten.

Enter.

Die Nachricht "*copied" erscheint. Wir schauen uns das Member SPRUTH.CICS.TEST04(PROG04) an.

```

File Edit Confirm Menu Utilities Compilers Test Help
-----
VIEW          SPRUTH.CICS.TEST04(PROG04) - 01.02          Columns 00001 00072
***** ***** Top of Data *****
==MSG> -CAUTION- Profile changed to NUMBER OFF (from NUMBER ON STD).
==MSG>          Data does not have valid standard numbers.
==MSG> -CAUTION- Profile changed to CAPS OFF (from CAPS ON) because data
==MSG>          contains lower case characters.
==MSG> -Warning- The UNDO command is not available until you change
==MSG>          your edit profile using the command RECOVERY ON.
000001 union
000002 {
000003 struct {
000004     char          dfhms1Y12";
000005     } label04i;
000006
000007 struct {
000008     char          dfhms2Y12";
000009     } label04o;
000010
000011 } label04;
000012
Command ===>          Scroll ===> PAGE
F1=Help      F3=Exit      F5=Rfind      F6=Rchange    F12=Cancel

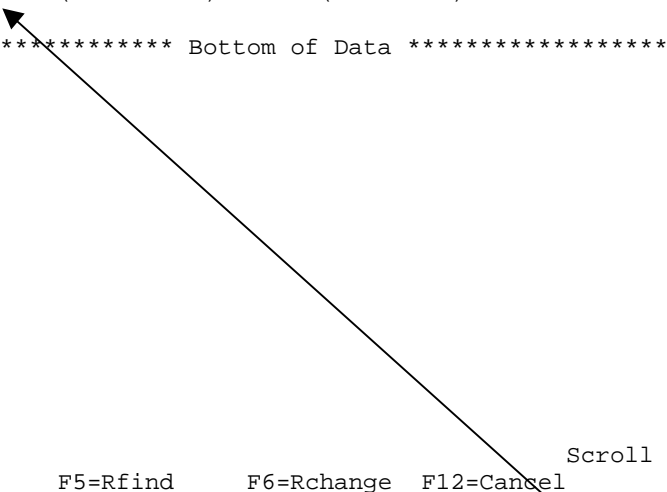
```

Das Template nimmt dort die Zeilen 000001 -000012 ein. Dann -

```

File Edit Confirm Menu Utilities Compilers Test Help
-----
VIEW          SPRUTH.CICS.TEST04(PROG04) - 01.02          Columns 00001 00072
000013 main()
000014 {
000015     EXEC CICS SEND MAP("label04") MAPSET("s04set") ERASE;
000016 }
***** ***** Bottom of Data *****

```



```

Command ===>          Scroll ===> PAGE
F1=Help      F3=Exit      F5=Rfind      F6=Rchange    F12=Cancel

```

- fügen wir in Zeile 000013 - 00016 (nächster Screen) die "main()" Routine unseres Anwendungsprogrammes hinzu. Sie enthält einen einzigen Befehl, einen CICS Befehl "SEND MAP". Dieser bewirkt, daß die Map mit der Adresse "LABEL04" aus dem Mapset "S04SET" mit Hilfe des 3270 Protokolls an den Bildschirm des Endbenutzers übertragen wird.

F3 Taste

```

File Edit Confirm Menu Utilities Compilers Test Help
-----
VIEW          SPRUTH.CICS.TEST04(START04) - 01.03          Columns 00001 00072
***** ***** Top of Data *****
==MSG> -CAUTION- Profile changed to NUMBER ON STD (from NUMBER OFF).
==MSG>          Data has valid standard numbers.
==MSG> -CAUTION- Profile changed to CAPS ON (from CAPS OFF) because the
==MSG>          data does not contain any lower case characters.
==MSG> -Warning- The UNDO command is not available until you change
==MSG>          your edit profile using the command RECOVERY ON.
000010 //CICSPRE  JOB ( ),CLASS=A,MSGCLASS=H,MSGLEVEL=(1,1),NOTIFY=&SYSUID,
000020 //          TIME=1440
001100 //          EXEC PROC=CTOCICS,REG=0M
001300 //TRN.SYSIN DD DISP=SHR,DSN=SPRUTH.CICS.TEST04(PROG04)
001400 //LKED.SYSIN DD *
001500          NAME PROG04(R)
***** ***** Bottom of Data *****

Command ==> sub          Scroll ==> PAGE
F1=Help    F3=Exit      F5=Rfind    F6=Rchange  F12=Cancel

```

Dieses Programm soll nun übersetzt werden. Dazu wird für den Partitioned Data Set "SPRUTH.CICS.TEST04" ein neues Member "START04" erstellt. Dies ist eine JCL File, welche die Prozedur CTOCICS (Compile to CICS) enthält. CTOCICS ruft zunächst den CICS Precompiler auf, der alle CICS Befehle in C Befehle übersetzt. Anschließend wird der C Compiler aufgerufen, der ein Maschinenprogramm erstellt und in eine für das CICS Subsystem zugängliche Library stellt. START04 wird mit dem Kommando "sub" ausgeführt.

Sub eingeben. Wir warten, bis der Job ausgeführt wurde.

Als nächsten Schritt wechseln wir wieder von der TSO Session zur CICS Session.

```

CEDA DEFINE PROGRAM(PROG04) GROUP(SPRUTH4)
OVERTYPE TO MODIFY
CEDA Install
All
Connection ==>
DB2Conn ==>
DB2Entry ==>
DB2Tran ==>
DOctemplate ==>
Enqmodel ==>
File ==>
Journalmodel ==>
LSrpool ==>
Mapset ==>
PARTitionset ==>
PARTner ==>
PROcesstype ==>
PROfile ==>
PROgram ==>
+ Requestmodel ==>

          SYSID=C001 APPLID=A06C001
INSTALL SUCCESSFUL          TIME: 00.00.00 DATE: 01.037
PF 1 HELP          3 END          6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL

```

Wir definieren für die Gruppe SPRUTH4 unser Anwendungsprogramm als PROG04, indem wir den entsprechenden CEDA Befehl in die oberste Zeile schreiben.

Enter

```

CEDA DEFINE PROGRAM(PROG04) GROUP(SCRUTH4)
OVERTYPE TO MODIFY                                CICS RELEASE = 0530
CEDA DEFine PROGram( PROG04 )
  PROGram      : PROG04
  Group        : SCRUTH4
  DDescription  ==>
  Language      ==>          CObol | Assembler | Le370 | C | Pli
  REload       ==> No       No | Yes
  RESident     ==> No       No | Yes
  USAge        ==> Normal   Normal | Transient
  USElpacopy   ==> No       No | Yes
  Status       ==> Enabled  Enabled | Disabled
  RSl          : 00        0-24 | Public
  CEdf         ==> Yes      Yes | No
  DAtalocation ==> Below    Below | Any
  EXEckey      ==> User     User | Cics
  COncurrency  ==> Quasirent Quasirent | Threadsafe
  REMOTE ATTRIBUTES
  Dynamic      ==> No       No | Yes
+  REMOTESystem ==>

                                SYSID=C001 APPLID=A06C001
DEFINE SUCCESSFUL                    TIME: 00.00.00 DATE: 01.037
PF 1 HELP 2 COM 3 END                6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL

```

CEDA will einiges von uns wissen. Wir übernehmen alle Default Werte und geben als Sprache Le370 an.

```

CEDA DEFINE PROGRAM(PROG04) GROUP(SCRUTH4)
OVERTYPE TO MODIFY                                CICS RELEASE = 0530
CEDA DEFine PROGram( PROG04 )
  PROGram      : PROG04
  Group        : SCRUTH4
  DDescription  ==>
  Language      ==> Le370    CObol | Assembler | Le370 | C | Pli
  REload       ==> No       No | Yes
  RESident     ==> No       No | Yes
  USAge        ==> Normal   Normal | Transient
  USElpacopy   ==> No       No | Yes
  Status       ==> Enabled  Enabled | Disabled
  RSl          : 00        0-24 | Public
  CEdf         ==> Yes      Yes | No
  DAtalocation ==> Below    Below | Any
  EXEckey      ==> User     User | Cics
  COncurrency  ==> Quasirent Quasirent | Threadsafe
  REMOTE ATTRIBUTES
  Dynamic      ==> No       No | Yes
+  REMOTESystem ==>

                                SYSID=C001 APPLID=A06C001
DEFINE SUCCESSFUL                    TIME: 00.00.00 DATE: 01.037
PF 1 HELP 2 COM 3 END                6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL

```

Was ist denn Le370 für eine Sprache ? Nun, Le370 ist überhaupt keine Sprache, sondern eine Laufzeitumgebung. CICS braucht an dieser Stelle in Wirklichkeit nicht die Angabe der Quellsprache unseres Anwendungsprogramms (wir haben es ja bereits übersetzt), sondern die Angabe der Laufzeitumgebung des von uns verwendeten Compilers. Alle modernen OS/390 Compiler verwenden eine gemeinsame Laufzeitumgebung, die auf den Namen Le370 getauft wurde.

Enter

```

OVERTYPE TO MODIFY                                CICS RELEASE = 0530
CEDA DEFINE PROGram( PROG04  )
  PROGram      : PROG04
  Group        : SPRUTH4
  DDescription  ==>
  Language     ==> Le370          CObol | Assembler | Le370 | C | Pli
  RELoad       ==> No            No | Yes
  RESident     ==> No            No | Yes
  USAge        ==> Normal        Normal | Transient
  USElpacopy   ==> No            No | Yes
  Status       ==> Enabled        Enabled | Disabled
  RSl          : 00              0-24 | Public
  CEdf         ==> Yes           Yes | No
  DAtalocation ==> Below         Below | Any
  EXEckey      ==> User          User | Cics
  COncurrency  ==> Quasirent     Quasirent | Threadsafe
  REMOTE ATTRIBUTES
  Dynamic      ==> No            No | Yes
+ REMOTESystem ==>

                                           SYSID=C001 APPLID=A06C001
DEFINE SUCCESSFUL                          TIME: 00.00.00 DATE: 01.037
PF 1 HELP 2 COM 3 END                      6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL

```

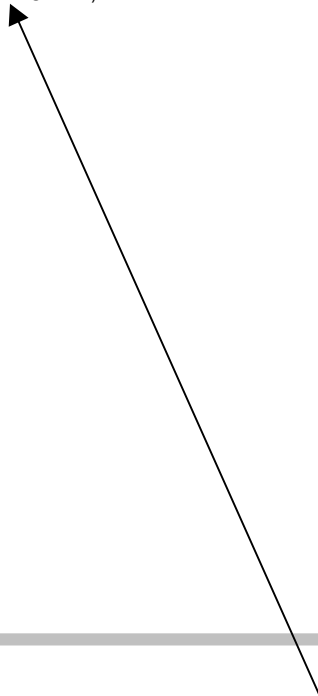
Es erscheint der obige Bildschirm

F3 eingeben

```

CEDA DEFINE PROGRAM(PROG04) GROUP(SPRUTH4)
STATUS:  SESSION ENDED

```



In diesen Bildschirm geben wir in die oberste Zeile den nächsten CEDA Befehl ein

```
CEDA DEFINE TRANS(SP04) GROUP(SPRUTH4)
STATUS:  SESSION ENDED
```

Unsere Transaktion soll wie alle anderen Transaktion vom Bildschirm über eine 4-stellige Transaktions ID aufgerufen werden. Wir wählen hierfür die ID „SP04“ und teilen diese Wahl mit Hilfe des CEDA DEFINE Befehls mit. Genauso wie PROG04 wird dies Bestandteil von GROUP4.

Enter

```
DEFINE TRANS(SP04) GROUP(SPRUTH4)
OVERTYPE TO MODIFY                                CICS RELEASE = 0530
  CEDA DEFine TRANSaction( SP04 )
    TRANSaction ==> SP04
    Group       ==> SPRUTH4
    DDescription ==>
    PROGram     ==>
    TWasize     ==> 00000          0-32767
    PROFile     ==> DFHCICST
    PARTitionset ==>
    STATus      ==> Enabled      Enabled | Disabled
    PRIMedsize  : 00000          0-65520
    TASKDATAloc ==> Below       Below | Any
    TASKDATAKey ==> User        User | Cics
    STORageclear ==> No         No | Yes
    RUNaway     ==> System      System | 0 | 500-2700000
    SHutdown    ==> Disabled    Disabled | Enabled
    ISolate     ==> Yes         Yes | No
    Brexit      ==>
+ REMOTE ATTRIBUTES
  S PROGRAM OR REMOTESYSTEM MUST BE SPECIFIED.
                                           SYSID=C001 APPLID=A06C001
PF 1 HELP 2 COM 3 END                    6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL
```

CEDA will mehrere Angaben von uns und schlägt eine Reihe von Default Werten vor.

```

DEFINE TRANS(SP04) GROUP( SPRUTH4)
OVERTYPE TO MODIFY                                CICS RELEASE = 0530
CEDA DEFINE TRANSAction( SP04 )
TRANSAction ==> SP04
Group        ==> SPRUTH4
Description  ==>
PROGm       ==> PROG04
TWAsize     ==> 00000          0-32767
PROfile     ==> DFHCICST
PARTitionset ==>
STATus      ==> Enabled      Enabled | Disabled
PRIMedsize  : 00000          0-65520
TASKDATAloc ==> Below        Below | Any
TASKDATAKey ==> User         User | Cics
STOrageclear ==> No          No | Yes
RUNaway     ==> System       System | 0 | 500-2700000
SHUTDOWN    ==> Disabled     Disabled | Enabled
ISolate     ==> Yes          Yes | No
Brexite     ==>
+ REMOTE ATTRIBUTES
S PROGRAM OR REMOTESYSTEM MUST BE SPECIFIED.
                                           SYSID=C001 APPLID=A06C001

PF 1 HELP 2 COM 3 END                    6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL

```

Wir übernehmen sie alle und geben in die Zeile "PROGm" den Namen unseres Anwendungsprogramms, nämlich "PROG04" ein.

Enter

```

OVERTYPE TO MODIFY                                CICS RELEASE = 0530
CEDA DEFINE TRANSAction( SP04 )
TRANSAction : SP04
Group       : SPRUTH4
Description ==>
PROGm       ==> PROG04
TWAsize     ==> 00000          0-32767
PROfile     ==> DFHCICST
PARTitionset ==>
STATus      ==> Enabled      Enabled | Disabled
PRIMedsize  : 00000          0-65520
TASKDATAloc ==> Below        Below | Any
TASKDATAKey ==> User         User | Cics
STOrageclear ==> No          No | Yes
RUNaway     ==> System       System | 0 | 500-2700000
SHUTDOWN    ==> Disabled     Disabled | Enabled
ISolate     ==> Yes          Yes | No
Brexite     ==>
+ REMOTE ATTRIBUTES
                                           SYSID=C001 APPLID=A06C001
DEFINE SUCCESSFUL                               TIME: 22.00.13 DATE: 01.037
PF 1 HELP 2 COM 3 END                          6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL

```

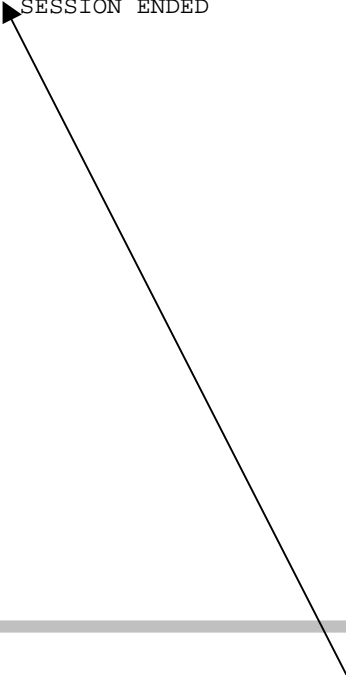
Die Meldung "DEFINE SUCCESSFUL" erscheint.

F3 Taste betätigen

```
CEDA DEFINE TRANS(SP04) GROUP(SPRUTH4)
STATUS:  SESSION ENDED
```

Der obige Bildschirm erscheint. Wir haben den Namen unseres Anwendungsprogramms und eine dazugehörige Transaktions ID an CICS bekanntgegeben. Jetzt müssen diese in die CICS Programmbibliothek übernommen (installiert) werden.

```
CEDA INSTALL GROUP(SPRUTH4)
STATUS:  SESSION ENDED
```



Wir geben in die oberste Zeile das CEDA INSTALL Kommando ein.

Enter

```
INSTALL GROUP( SPRUTH4)
OVERTYPE TO MODIFY
CEDA Install
All
Connection ==>
DB2Conn ==>
DB2Entry ==>
DB2Tran ==>
DOctemplate ==>
Enqmodel ==>
File ==>
Journalmodel ==>
LSrpool ==>
Mapset ==>
PARTitionset ==>
PARTner ==>
PROcesstype ==>
PROfile ==>
PROgram ==>
+ Requestmodel ==>

                                SYSID=C001 APPLID=A06C001
INSTALL SUCCESSFUL              TIME: 22.02.15 DATE: 01.037
PF 1 HELP                       3 END                6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL
```

CEDA bestätigt, daß die Installation erfolgreich war.

F3 Taste betätigen

```
CEDA INSTALL GROUP( SPRUTH4)
STATUS:  SESSION ENDED
```

Das war es. Der obige Bildschirm erscheint. Unsere Transaktion ist als Teil der CICS Anwendungsbibliothek installiert worden und kann nun aufgerufen und damit ausgeführt werden.

Hierzu löschen wir die oberste Zeile (die CEDA Kommando Zeile) ganz, und rufen unsere Anwendung auf, indem wir eben dort unsere Transaktions ID, nämlich SP04, eingeben.

```
WELCOME TO THE MAGIC WORLD OF CICS
```

```
MAY THE FORCE BE WITH YOU!
```

Und hier ist der Erfolg unserer Bemühungen. Eine echte CICS Transaktion, auch wenn sie nicht sehr viel leistet.

Enter

```
WELCOME TO THE MAGIC WORLD OF CICS
```

```
MAY THE FORCE BE WITH YOU!
```

```
DFHAC2001 02/04/01 11:31:55 A06C001 Transaction '' is not recognized. Check  
that the transaction name is correct.
```

Dies terminiert die Bildschirmausgabe unserer Transaktion und erzeugt wieder eine (belanglose) Fehlermeldung.

_WELCOME TO THE MAGIC WORLD OF CICS

MAY THE FORCE BE WITH YOU!

DFHAC2001 02/04/01 11:31:55 A06C001 Transaction '' is not recognized. Check that the transaction name is correct. CEDA DISPLAY GROUP(SCRUTH4)

Alle Bestandteile unserer Transaktion sind in der Gruppe SCRUTH4 gespeichert. Wir schauen sie uns an.

Enter

DISPLAY GROUP(SCRUTH4)

ENTER COMMANDS

NAME	TYPE	GROUP	DATE	TIME
S04SET	MAPSET	SCRUTH4	01.034	24.00.00
PROG04	PROGRAM	SCRUTH4	01.034	24.00.00
SP04	TRANSACTION	SCRUTH4	01.034	24.00.00

RESULTS: 1 TO 3 OF 3

SYSID=C001 APPLID=A06C001
TIME: 00.00.00 DATE: 01.035

PF 1 HELP 3 END 4 TOP 5 BOT 6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL

Die Gruppe Spruth besteht aus den drei Komponenten S04SET, PROG04 und SP04, die wir unter CICS definiert und anschließend installiert haben.

Das war es. Mit HTML wäre es sicher schneller gegangen. Aber wir haben die Basis geschaffen, wirklich leistungsfähige CICS Transaktionen zu erzeugen, etwas, was mit HTML nicht möglich ist.

The End