

CICS Cobol Tutorial 1

Cobol Hello World unter CICS

© Abteilung Technische Informatik, Institut für Informatik, Universität Leipzig
© Abteilung Technische Informatik, Wilhelm Schickard Institut für Informatik, Universität Tübingen

Ziel dieser Aufgabe ist es, ein "Hello World"-Programm in COBOL zu schreiben und Daten mittels CICS auf dem Bildschirm auszugeben. CICS ist ein z/OS Transaktions-Server, der Transaktionen ausführt. Transaktionen sind eine spezielle Art von Prozessen, die entweder zu 100 %, oder aber gar nicht ausgeführt werden. Auf einem Mainframe Rechner werden etwa 80 % aller Prozesse als Transaktionen ausgeführt, die meisten davon mittels des CICS Transaktionsservers.

Informationen über CICS finden Sie auch in Kapitel 8 dieses Buches.

Hinweis: Dieses Tutorial wurde unter Verwendung der Benutzer-ID "PRAK085" erstellt. In allen Dateinamen müssen Sie "PRAK085" durch ihre eigene Benutzer-ID ersetzen.

Aufgabe: *Beschäftigen Sie sich mit diesem Tutorial und lösen Sie gewissenhaft alle kursiv geschriebenen und eingerahmten Aufgaben.*

Inhalt

- 1. Einführung**
- 2. Presentation Logic**
- 3. CICS Anwendungsprogramm**
- 4. Definition und Installation des Programms in CICS**
- 5. Ausführung des Programms als CICS Transaktion**
- 6. Logoff**

Anhang

Hinweise

Die CICS-Tutorien sind etwas schwerer als die vorangegangenen Tutorien. Hierzu noch ein paar Tips:

- Nervige Fehlersuchen sind in den CICS-Tutorien normal, und Bestandteil des Lerneffektes.
- Falls "JCL Error" zurückgegeben wird, befindet sich der Fehler mit großer Wahrscheinlichkeit im JCL-Script und nicht im Cobol-Programm.
- Gehen Sie Schritt für Schritt vor; d.h. bringen Sie erst das Cobol-Programm zum Laufen, wenn das BMS-Programm fehlerfrei läuft und beginnen Sie erst mit den CICS-Installationen, sobald das Cobol-Programm fehlerfrei übersetzt wurde u.s.w.
- Falls der ISPF-Editor ungewollt gültigen Programmcode mit überflüssigen Zeilen-Nummern überschreibt, dann geben Sie bitte im ISPF-Editor auf der Kommandozeile "NUMBER OFF" ein (siehe <http://mvs.wiu.edu/stumvs/TSO/TSOChangingYourEditProfile.html>).
- Sie können NUR den Transaktionsnamen X< Prak-Nr.>, also z.B. X613 oder X609 für die User Ids prak613 oder prak609 verwenden. Bei anderen Transaktionsnamen wird die Fehlermeldung "DFHAC2033 22:14:09 CICS You are not authorized to use transaction W607" zurückgegeben.
- Wenn man SUB auf einen Mapset anwendet, dann werden aus den selbst gewählten Namen für BMS-Eingabefelder neue Variablen gebildet (die zukünftige Schnittstelle zwischen Mapset und Programm), welche unter Umständen verbotene Schlüsselwörter werden. Aus dem Feldnamen "C" wird z.B. "CF" und aus "I" wird z.B. "IF" gebildet, beides verbotene Schlüsselwörter.

Deshalb sind "C", "I" sowie "G" als mögliche Namen für Felder in einer BMS-Map verboten.

Als Symptom entsteht "MAXCC=12". Wenn man in die Joblog-File hineinschaut, findet man z.B. 'CF is a reserved word related to language'.

1. Einführung

Das Erstellen einer neuen Anwendung verwendet in der Regel eine Entwicklungsumgebung für das Schreiben und Übersetzen des neuen Programms und eine Produktionsumgebung für das Ausführen des Programms. In der letzten Übung war TSO und ISPF die Entwicklungsumgebung, JES die Ausführungsumgebung für Compile und Link, und TSO die Ausführungsumgebung für die Programmausführung. In der vorliegenden Übung verwenden wir wieder TSO und ISPF als Entwicklungsumgebung, JES als Ausführungsumgebung für Compile und Link, und CICS als Produktionsumgebung (Ausführungsumgebung für die Programmausführung).

TSO ist ein z/OS-Subsystem. CICS ist ein weiteres z/OS-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 z/OS ein Multi-User-Betriebssystem ist (multisession-fähig), können wir gleichzeitig eine TSO-Session und eine CICS-Session auf unserem Arbeitsplatzrechner laufen lassen. Jede Session läuft in einem eigenen Fenster.

Wir starten einen 3270-Emulator zunächst für eine TSO-Session und loggen uns ein. Wir öffnen den "Data Set Utility"-Screen und erzeugen (Allocate) einen neuen Partitioned Dataset: "PRAK085.CICS.COBOl". Dabei verwenden wir die in der nachstehenden Aufgabe angegebenen Parameter.

Außerdem brauchen wir noch einen Partitioned Dataset mit dem vorgegebenen Namen "PRAK085.LIB", dessen Members von der Entwicklungsumgebung während der CICS-Programmentwicklung mit Daten gefüllt werden. Verwenden Sie bei dessen Anlage ebenfalls die unten angegebenen Parameter.

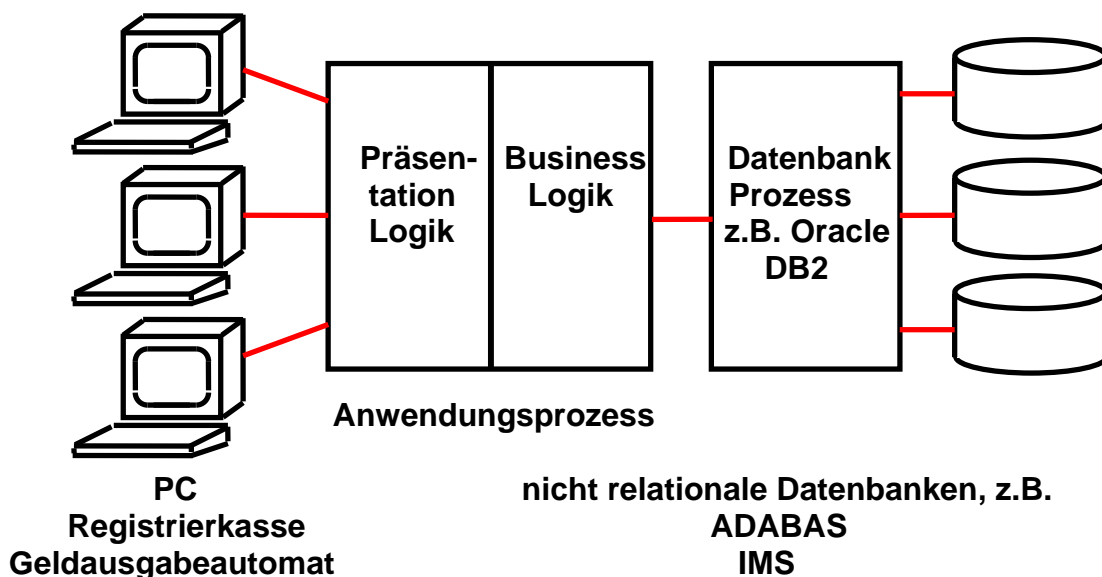
Dieser Name besteht aus einem "Project"-Begriff und einem "Group"-Begriff. Es fehlt der "Type"-Begriff. Wenn wir das Typ-Feld leer lassen, wird TSO dies nicht akzeptieren. Deshalb tragen wir den Namen 'PRAK085.LIB' (mit Hochkommas!) in die Zeile "Data Set Name" ein. Damit wird auch dieser Dataset angelegt.

Aufgabe: Legen Sie die Datasets "PRAK085.CICS.COBOl" , "PRAK085.LIB" an. Verwenden Sie dazu folgende Parameter:

<i>Space units</i>	<i>KILOBYTE</i>	<i>Record format</i>	<i>FB</i>
<i>Primary quantity . .</i>	<i>16</i>	<i>Record length</i>	<i>80</i>
<i>Secondary quantity</i>	<i>1</i>	<i>Block size</i>	<i>320</i>
<i>Directory blocks . .</i>	<i>2</i>	<i>Data set name type</i>	<i>: PDS</i>

Unsere Anwendung besteht aus zwei Programmteilen und einem JCL-Script für die Übersetzung. Wir erstellen diese als Members in dem neuen Partitioned Dataset "PRAK085.CICS.COBOL".

Ein sauber strukturiertes CICS-Programm besteht aus zwei Teilen: Business Logic und Presentation Logic. Während die Geschäftslogik die eigentliche Arbeit verrichtet, dient die Präsentationslogik dazu, die Ergebnisse der Geschäftslogik auf eine attraktive Art auf dem Bildschirm darzustellen.



Der Server Teil einer Client/Server Anwendung besteht aus zwei Teilen: einem Anwendungsprozess und einem Datenbankprozess.

Dargestellt ist eine logische Struktur. 2-Tier, 3-Tier oder n-Tier Konfigurationen unterscheiden sich dadurch, wie diese Funktionen auf physische Server abgebildet werden.

Einige der Alternativen sind:

- In einer 2-Tier Konfiguration befindet sich der Anwendungsprozess auf dem gleichen Rechner wie der Klient.
- In einer 3-Tier Konfiguration befinden sich Anwendung und Datenbank als getrennte Prozesse auf getrennten Rechnern.
-

2. Presentation Logic

Business Logic wird in Sprachen wie C, C++, COBOL, PL/I, Java, ASSEMBLER, REXX usw. geschrieben. Für die Presentation Logic gibt es viele Alternativen. Eine sehr moderne Alternative benutzt Java Server Pages und einen Web Application Server, um den Bildschirminhalt innerhalb eines Web-Browsers darzustellen. Eine weitere sehr moderne Alternative benutzt XML, SOAP und Web Services. Die älteste (und einfachste) Alternative verwendet das CICS-BMS (Basic Mapping Support)-Subsystem und eine 3270 Bildschirmausgabe. Dies setzen wir hier ein. BMS-Programme werden in der BMS-Sprache geschrieben. In unserem Beispiel wird die Business Logic in COBOL und die Presentation Logic in BMS geschrieben.

Wir fangen mit der Presentation Logic an, rufen den "Edit Entry Panel Screen" auf und editieren ein Member "MAPCO01" für den neu angelegten Partitioned Dataset "PRAK085.CICS.COBO1" (s. Abbildung 1). Man beachte den Buchstaben "O" und die Ziffer "0" in diesem Membernamen (!!!)

```

File Edit Confirm Menu Utilities Compilers Help
-----
EDIT          PRAK085.CICS.COBO(L(MAPCO01) - 01.05          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.
000100 //PRAK085M JOB (),CLASS=A,MSGCLASS=H,MSGLEVEL=(1,1),NOTIFY=&SYSUID,
000200 //          REGION=4M
000300 //ASSEM EXEC DFHMDS,MAPNAME='MSET085',RMODE=24
000400 //SYSUT1 DD *
000500 MSET085 DFHMDS TYPE=MAP,MODE=INOUT,LANG=COBOL2,STORAGE=AUTO,          *
000600          TIOAPFX=YES
000700 *          MENU MAP
000800 MAP085 DFHMDS SIZE=(24,80),CTRL=(PRINT,FREEKB)
000900          DFHMDF POS=(9,23),ATTRB=(ASKIP,NORM),LENGTH=34,          *
001000          INITIAL='WELCOME TO THE MAGIC WORLD OF CICS'
001100          DFHMDF POS=(12,33),ATTRB=(ASKIP,NORM),LENGTH=16,          *
001200          INITIAL='MEMBER PRAK085 !'
001300          DFHMDS TYPE=FINAL
001400          END
001500 /*
001600 //
Command ===>          Scroll ===> PAGE
F1=Help          F3=Exit          F5=Rfind          F6=Rchange          F12=Cancel

```

Abbildung 1: Das BMS-Programm

Die sehr einfache von CICS verwendete BMS Sprache besteht nur aus den drei Befehlen **DFHMDS**, **DFHMDS** und **DFHMDF**, mehr wird für die Beschreibung der BMS Maps nicht benötigt. Es ist nachvollziehbar, dass graphische Bildschirm-Darstellungen eine komplexere Beschreibungssprache (z.B. HTML, Java usw.) benötigen. (Es existieren auch Werkzeuge, die BMS Code direkt in HTML Code umwandeln).

Die BMS Sprache wird mit Hilfe von drei Assembler Macros implementiert, die alle mit den Buchstaben DFHM anfangen. Ein BMS Programm ist also in Wirklichkeit ein Assembler Programm.

Unser BMS-Programm verwendet 3 Befehlstypen: DFHM**SD**, DFHM**DI** und DFHM**DF**. Ein BMS-Bildschirm verwendet das 24 Zeilen x 80 Zeichen / Zeile 3270-Bildschirmformat. Die Ein- und Ausgabemasken werden als Felder innerhalb der 24x80-Matrix dargestellt, jeweils mit der Angabe: Zeilenadresse, Spaltenadresse und Feldlänge. Dies geschieht mit Hilfe des DFHM**DF**-Befehls. Der DFHMDF-Befehl in den Zeilen 000900 und 001000 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 DFHM**SD**-Befehl (Zeile 000500) definiert einen "Mapset" mit dem Namen "MSET085". (Bitte benutzen Sie hier die letzten 3 Ziffern Ihrer User ID.) Eine Transaktion involviert in der Regel mehrere unterschiedliche Screens (Maps): Z.B. einen Screen, in dem der Benutzer zu einer Eingabe aufgefordert wird und einen weiteren Screen, welcher die Ergebnisse der Anfrage wiedergibt. Alle Maps (Screens) eines Transaktionstyps werden zu einem Mapset zusammengefasst.

Die einzelnen Maps (Screens) eines Mapsets werden durch den Befehl DFHM**DI** definiert und zur Kennzeichnung mit einer Label versehen. In unserem einfachen "Hello World"-Beispiel besteht der Mapset aus einer einzigen Map, die in Zeile 000800 mit der Bezeichnung "MAP085" definiert wird.

Das Member "MAPCO01" stellt in Wirklichkeit ein JCL-Script dar. Im Gegensatz zu Tutorial 2 wird das zu verarbeitende File nicht mit INFILE='xxx.yyy.zzz' angegeben. Der JCL-Befehl in Zeile 000400 "//SYSUT1 DD *" besagt, dass das zu verarbeitende File unmittelbar danach folgt (Zeilen 000500 bis 001400).

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

JCL findet das Member "DFHMAPS" (Zeile 000300) in der Library "ADCD.Z18.PROCLIB(DFHMAPS)". Durch die Ausführung von "DFHMAPS" werden zwei Ausgabe-Files erzeugt. Einmal wird der übersetzte BMS-Quellcode in einen Member in eine CICS-interne Library mit dem Namen "DFH310.CICS.PRAKLOAD" gestellt. Hier kann ihn die BMS-Komponente des CICS-Subsystems später finden.


```

File Edit Confirm Menu Utilities Compilers Test Help
-----
EDIT          PRAK085.LIB(MSET085) - 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.
==MSG> -CAUTION- Profile is set to STATS ON. Statistics did not exist for
==MSG>          this member, but will be generated if data is saved.
000001      01  MAP085I.
000002      02  FILLER PIC X(12).
000003      01  MAP085O REDEFINES MAP085I.
000004      02  FILLER PIC X(12).
***** Bottom of Data *****

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

```

Abbildung 2: Das Member "MSET085"

Das zweite Ausgabe-File wird als Member "MSET085" in den Partitioned Dataset "PRAK085.LIB" gestellt (s. Abbildung 2). Alle Ein- und Ausgabe-Masken, die auf dem Bildschirm wiedergegeben werden sollen, werden ja bereits durch das BMS-Programm definiert. Das Business Logik-Programm bearbeitet diese Daten als COBOL-Strukturen, und diese werden von DFHMAPS während der Übersetzung von "MAPCO01" gleich miterzeugt und in "PRAK085.LIB(MSET085)" abgespeichert.

Aufgabe: *Modifizieren Sie Ihr BMS-Programm derart, dass es den Transaktionsnamen "U<Login-Nr.>", "TUTORIAL 3 IN COBOL" sowie den Namen oder die Namen der Autoren enthält. Dieses BMS-Programm soll später einen Screen ähnlich der Abbildung 29 generieren. (Hinweis: Die Ausführung des JCL-Scriptes erfolgt fehlerfrei, wenn dieses mit der Statusmeldung "MAXCC=0" beendet wird).*

3. CICS Anwendungsprogramm

Wir rufen den Editor auf und erstellen in COBOL das Anwendungsprogramm COB085 (s. Abbildung 3). Letzteres enthält das CICS-Statement "EXEC CICS SEND MAP('MAP085') ...

Dies ist das einzige EXEC CICS Statement in unserem Cobol Programm. Normalerweise würden viele EXEC CICS Statements vorhanden sein, die vor allem etwas mit dem Zugriff auf Daten zu tun haben würden.

```
File Edit Confirm Menu Utilities Compilers Test Help
-----
EDIT          PRAK085.CICS.COBO(LCOB085) - 01.03          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.
000100      IDENTIFICATION DIVISION.
000200      PROGRAM-ID. COB085.
000300      ENVIRONMENT DIVISION.
000400      DATA DIVISION.
000500      WORKING-STORAGE SECTION.
000600      COPY MSET085.
000700      LINKAGE SECTION.
000800      PROCEDURE DIVISION.
000900          EXEC CICS SEND MAP('MAP085')
001000                      MAPSET('MSET085')
001100                      FROM(MAP085O)
001200                      ERASE
001300      END-EXEC.
001400      GOBACK.
***** Bottom of Data *****

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

Abbildung 3: Das Anwendungsprogramm "COB085"

Dieses bewirkt, dass die Map mit dem im BMS-Programm definierten Mapnamen "MAP085" aus dem Mapset "MSET085" mit Hilfe des 3270-Protokolls an den Bildschirm des Endbenutzers übertragen wird.

```
File Edit Confirm Menu Utilities Compilers Test Help
-----
EDIT          PRAK085.CICS.COBO(COBSTA03) - 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.
000100 //PRAK085C JOB ( ),CLASS=A,MSGCLASS=H,MSGLEVEL=(1,1),NOTIFY=&SYSUID,
000200 //          REGION=4M
000300 //STEP1 EXEC COBCICS,PARM.TRN='COBOL3'
000400 //TRN.SYSIN DD DISP=SHR,DSN=&SYSUID..CICS.COBO(COB085)
000500 //COB.SYSLIB DD DSN=&SYSUID..LIB,DISP=SHR
000600 //LKED.SYSIN DD *
000700 NAME COB085(R)
000800 /*
***** ***** Bottom of Data *****

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

Abbildung 4: JCL-File "COBSTA03"

Dieses Programm soll nun übersetzt werden. Dazu wird für den Partitioned Dataset "PRAK085.CICS.COBO" ein neues Member "COBSTA03" erstellt (s. Abbildung 4). Dies ist ein JCL-File, das die Prozedur COBCICS enthält. COBCICS ruft zunächst den CICS-Precompiler auf, der alle CICS-Befehle in COBOL-Befehle übersetzt. Anschließend wird der COBOL-Compiler aufgerufen, der ein Maschinenprogramm erstellt und in eine für das CICS-Subsystem zugängliche Library stellt.

Wir geben "SUB" ein und warten, bis der Job ausgeführt wurde (s. Abbildung 4).

4. Definieren und Installieren des Programms in CICS

Wir entwickeln unsere Anwendung unter TSO. Wir wollen sie unter CICS laufen lassen. Dazu muss sie als Teil des CICS-Subsystems installiert werden. Es ist komfortabel, mit zwei z/OS-Sessions gleichzeitig zu arbeiten. Dazu kann man einfach noch ein zweites 3270 Emulator Fenster öffnen, mit dem man CICS startet.

```
z/OS Z18 Level 0609                                IP Address = 88.64.144.189
                                                    VTAM Terminal = SC0TCP14

Application Developer System

          // 0000000  SSSSS
          //  OO    OO SS
zzzzzzz //  OO    OO SS
          zz  //  OO    OO SSSS
          zz  //  OO    OO  SS
          zz  //  OO    OO  SS
zzzzzzz //  0000000  SSSS

System Customization - ADCD.Z18.*

===> Enter "LOGON" followed by the TSO userid. Example "LOGON IBMUSER" or
===> Enter L followed by the APPLID
===> Examples: "L TSO", "L CICS", "L IMS3270"

L CICS
```

Abbildung 5: Logon-Bildschirm

Wir loggen uns mit "L CICS" (einschließlich Eingabetaste) ein und rufen damit das CICS-Subsystem auf (s. Abbildung 5). Das CICS Subsystem verwendet eine ganz andere Shell und Benutzeroberfläche als TSO.

```
                                Signon to CICS                                APPLID A06C001

----- WELCOME AT UNIVERSITY OF LEIPZIG -----                        -JEDI-
BITTE TRANSAKTION <CESF LOGOFF> ZUM AUSLOGGEN BENUTZEN!                -CICS-

Type your userid and password, then press ENTER:

  Userid . . . PRAK085          Groupid . . .
  Password . . . *****
  Language . . .

New Password . . .

DFHCE3520 Please type your userid.
F3=Exit
```

Abbildung 6: Signon to CICS-Bildschirm

Der Signon to CICS-Bildschirm erscheint (s. Abbildung 6). Hier müssen wir unseren TSO-Loginnamen sowie das entsprechende Passwort eingeben. Die Eingabetaste führt uns in den nächsten Screen (s. Abbildung 7).




DFHCE3549 Sign-on is complete (Language ENU).

10:41:29 IBM-3278-2

Abbildung 7: Sign-on is complete

Mit der "Tab"-Taste bewegen wir den Cursor auf die unterste Zeile.

A screenshot of a terminal window with a grey border. The text is displayed in a monospaced font. At the bottom left, it says "DFHCE3549 Sign-on is complete (Language ENU)". To the right of this, the command "CEDA DISPLAY GROUP(*)" is entered and highlighted with a green rectangular box. At the bottom right, the time "10:42:29" and the identifier "IBM-3278-2" are visible.

```
DFHCE3549 Sign-on is complete (Language ENU). CEDA DISPLAY GROUP(*)
```

10:42:29 IBM-3278-2

Abbildung 8: Beispielkommando "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 Transaktions-ID aufgerufen.

Genauso wie eine Linux ftp shell andere Commands als die Linux bash shells benutzt, verwendet CICS ebenfalls einen anderen Command Interpreter als z.B. TSO oder z/OS Unix System Services.

Der CICS-Kommandointerpreter ist ebenfalls als Transaktion implementiert (etwas anderes als Transaktionen kann CICS nicht ausführen). Er wird mit der Transaktions-ID "CEDA" aufgerufen, gefolgt von einer Parameterliste, welche CICS-Kommandos sowie Eingabedaten enthält. Als Beispiel geben wir das Kommando "CEDA DISPLAY GROUP(*)" gefolgt von der Eingabetaste ein (siehe Abb. 9).

```
display group(*)
ENTER COMMANDS
  GROUP
  AOR2TOR
  ARTT
  ATC
  CBPS
  CEE
  CICREXX
  CICSJADP
  CICS0ADP
  CSQ
  CSQCKB
  CSQSAMP
  CTA1TCP
  C001EZA
  C001TCP
  DAVIN15
  DAVIN4
+ DAVIN8

                                     SYSID=C001 APPLID=A06C001
RESULTS: 1 TO 17                               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
```

Abbildung 9: Auflistung der Gruppen

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

Es werden die ersten 17 Gruppen angezeigt (s. Abbildung 9). Durch Betätigen der F8 Taste können Sie sich weitere Gruppen ansehen.


```
CEDA DEFINE MAPSET(MSET085) GROUP(PRAK085) 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
```

Abbildung 10: Definition des Mapsets "MSET085" und der Gruppe "PRAK085"

Wir definieren für unsere Transaktion eine eigene Gruppe "PRAK085" und den dazugehörigen Mapset als "MSET085". Hierzu überschreiben wir die oberste Zeile, die als Kommandozeile dient, mit dem CEDA-Befehl "CEDA DEFINE MAPSET(MSET085) GROUP(PRAK085)" (s. Abbildung 10, **bitte Großbuchstaben benutzen!**)

Um zu bestätigen drücken wir die Eingabetaste.

```

CEDA DEFINE MAPSET(MSET085) GROUP(PRAK085)
OVERTYPE TO MODIFY
CEDA DEFINE Mapset( MSET085 )
Mapset      : MSET085
Group       : PRAK085
Description ==>
RESident    ==> No           No | Yes
USAge       ==> Normal      Normal | Transient
USElpacopy  ==> No         No | Yes
Status      ==> Enabled     Enabled | Disabled
RS1         : 00           0-24 | Public

I New group PRAK085 created.
DEFINE SUCCESSFUL
PF 1 HELP 2 COM 3 END
SYSID=C001 APPLID=A06C001
TIME: 00.00.00 DATE: 01.037
6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL

```

Abbildung 11: Bestätigung der Erstellung von Gruppe und Mapset

CICS teilt uns mit, dass die neue Gruppe "PRAK085" erstellt sowie der Mapset erfolgreich definiert wurde (s.

Abbildung 11).

Führen wir nochmals den Befehl "CEDA DISPLAY GROUP(*)" aus, so finden wir in der dargestellten Liste den Eintrag "PRAK085" (erfordert ein Durchblättern der Liste unter Benutzung der Taste F8).

Nachdem der Mapset unter CICS definiert wurde, definieren wir jetzt das COBOL-Programm "COB085".

```
CEDA DEFINE PROGRAM(COB085) GROUP(PRAK085)
```

```
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
```

Abbildung 12: Definition des Programms "COB085"

Wir definieren für die Gruppe "PRAK085" unser Anwendungsprogramm "COB085", indem wir den entsprechenden CEDA-Befehl

"CEDA DEFINE PROGRAM(COB085) GROUP(PRAK085)"

in die oberste Zeile schreiben und anschließend die Eingabetaste betätigen (s. Abbildung 12).

CEDA DEFINE PROGRAM(COB085) GROUP(PRAK085)

```
OVERTYPE TO MODIFY                                CICS RELEASE = 0530
CEDA DEFINE PROGRAM( PROG01 )
  PROGRAM      : COB085
  Group       : PRAK085
  Description  ==>
  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
```

Abbildung 13: Definition von "COB085"

CEDA will einiges von uns wissen (s. Abbildung 13). Wir übernehmen alle Default-Werte und geben als Sprache "Le370" an (s. Abbildung 14).

```

CEDA DEFINE PROGRAM(COB085) GROUP(PRAK085)
OVERTYPE TO MODIFY
CICS RELEASE = 0530
CEDA Define PROGram( COB085 )
PROGRAM      : COB085
Group       : PRAK085
DEscription ==>
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
RSI        : 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

```

Abbildung 14: Auswahl der Sprache

Was ist denn "Le370" für eine Sprache? "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 z/OS-Compiler verwenden eine gemeinsame Laufzeitumgebung, die den Namen "Le370" (oder LE390) trägt. Mit Betätigung der Eingabetaste erscheint der nächste Screen.

```

OVERTYPE TO MODIFY                                CICS RELEASE = 0530
CEDA DEFINE PROGram( COB085  )
PROGRAM      : COB085
Group       : PRAK085
DEscription ==>
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
RS1        : 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

```

Abbildung 15: Erfolgreiche Definition

Nach dem Betätigen der Eingabetaste erscheint der Bildschirm in Abbildung 15. Wir verlassen die Definition mit der Eingabe von F3, als Ergebnis davon erscheint "SESSION ENDED" (s. Abbildung 16).

```
CEDA DEFINE PROGRAM(COB085) GROUP(PRAK085)
STATUS: SESSION ENDED
```

#

Abbildung 16: Definition wurde beendet

In diesen Bildschirm geben wir in die oberste Zeile den nächsten CEDA-Befehl, gefolgt von der Eingabetaste, ein (s. Abbildung 17).

```
CEDA DEFINE TRANS(X085) GROUP(PRAK085)  
STATUS:  SESSION ENDED
```

Abbildung 17: Definition der Transaktion X085

Unsere Transaktion soll wie alle anderen Transaktionen vom Bildschirm über eine 4-stellige Transaktions-ID aufgerufen werden. Wir wählen hierfür die ID "X085" und teilen diese Wahl mit Hilfe des "CEDA DEFINE"-Befehls mit (s. Abbildung 17). Genauso wie der Mapset "MSET085" und das Programm "COB085" wird dies Bestandteil der Gruppe "PRAK085". Abschließend betätigen wir die Eingabetaste.


```

DEFINE TRANS(X085) GROUP(PRAK085)
OVERTYPE TO MODIFY                                CICS RELEASE = 0530
CEDA Define TRANSAction( X085 )
TRANSAction ==> X085
Group       ==> PRAK085
Description ==>
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
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

```

Abbildung 18: Der Definitions-Screen

CEDA will mehrere Angaben von uns und schlägt eine Reihe von Default-Werten vor (s. Abbildung 18).

```

DEFINE TRANS(X085) GROUP(PRAK085)
OVERTYPE TO MODIFY
CEDA DEFINE TRANSAction( X085 )
TRANSAction ==> X085
Group ==> PRAK085
DEscription ==>
PROGram ==> COB085
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

```

Abbildung 19: Eingabe der Parameter

Wir übernehmen alle Default-Werte und geben in die Zeile "PROGram" den Namen unseres Anwendungsprogramms, nämlich "COB085" ein und bestätigen mit der Eingabetaste (s. Abbildung 19).

```

OVERTYPE TO MODIFY
CICS RELEASE = 0530
CEDA DEFINE TRANSACTION( X085 )
TRANSACTION      : X085
GROUP            : PRAK085
DESCRIPTION      ==>
PROGRAM          ==> COB085
TWSIZE          ==> 00000          0-32767
PROFILE         ==> DFHCICST
PARTITIONSET    ==>
STATUS          ==> Enabled          Enabled | Disabled
PRIMESIZE      : 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

                                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

```

Abbildung 20: Erfolgreiche Definition

Die Meldung "DEFINE SUCCESSFUL" erscheint (s. Abbildung 20).

Wir verlassen dieses Menü mit F3.

```
CEDA DEFINE TRANS(X085) GROUP(PRAK085)
STATUS: SESSION ENDED
```

Abbildung 21: SESSION ENDED

Der Bildschirm in der Abbildung 21 erscheint. Wir haben CICS den Namen unseres Mapsets, Anwendungsprogramms und eine dazugehörige Transaktions-ID bekanntgegeben. Jetzt müssen diese drei Komponenten in die CICS-Programmbibliothek übernommen (installiert) werden.

```
CEDA INSTALL GROUP(PRAK085)
```

```
STATUS:  SESSION ENDED
```

Abbildung 22: Aufruf der Installation

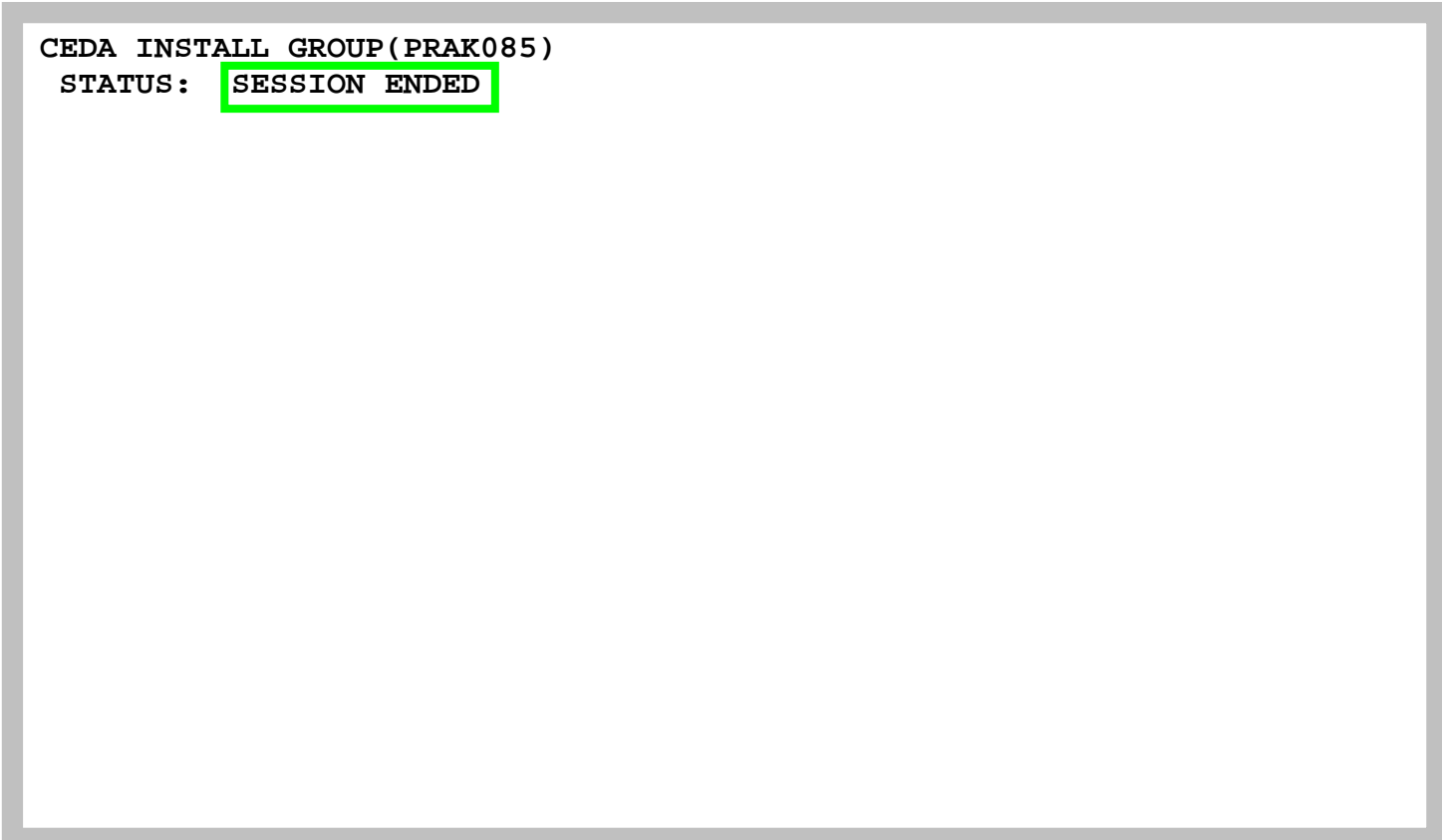
Wir geben in die oberste Zeile das CEDA-INSTALL-Kommando ein und drücken die Eingabetaste (s. Abbildung 22).

```
INSTALL GROUP(PRAK085)
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
                                TIME: 22.02.15 DATE: 01.037
INSTALL SUCCESSFUL
PF 1 HELP          3 END          6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL
```

Abbildung 23: Installation war erfolgreich

CEDA bestätigt, dass die Installation erfolgreich war (s. Abbildung 23). Wir verlassen mit F3 dieses Menü.



```
CEDA INSTALL GROUP(PRAK085)  
STATUS: SESSION ENDED
```

Abbildung 24: Beendete Installation

Der obige Bildschirm erscheint (s. Abbildung 24). 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-Kommandozeile) ganz und rufen unsere Anwendung auf, indem wir dort unsere Transaktions-ID, nämlich "X085", eingeben.

5. Ausführen des Programms als CICS Transaktion



WELCOME TO THE MAGIC WORLD OF CICS

GROUP PRAK085

Abbildung 25: Ausgabe der Transaktion "X085" auf dem Bildschirm

Nach dem Betätigen der Eingabetaste erscheint unsere CICS-Transaktion auf dem Monitor (s. Abbildung 25).

Wir betätigen die Eingabetaste, um zum nächsten Screen zu gelangen.

WELCOME TO THE MAGIC WORLD OF CICS

GROUP PRAK085

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

Abbildung 26: Fehlermeldung

Dies terminiert die Bildschirmausgabe unserer Transaktion und erzeugt wieder eine (belanglose) Fehlermeldung (s. Abbildung 26).

WELCOME TO THE MAGIC WORLD OF CICS

GROUP PRAK085

DFHAC2001 02/04/01 11:31:55 A06C001 Transaction '' is not recognized. Check that the transaction name is correct. **CEDA DISPLAY GROUP(PRAK085)**

Abbildung 27: Ansicht aller gespeicherter Daten in Group "PRAK085"

Alle Bestandteile unserer Transaktion sind in der Gruppe PRAK085 gespeichert. Wir schauen sie uns an, indem wir den Befehl "CEDA DISPLAY GROUP(PRAK085)" eingeben und anschließend die Eingabetaste drücken (s. Abbildung 27).

DISPLAY GROUP(PRAK085)

ENTER COMMANDS

NAME	TYPE	GROUP	DATE	TIME
MSET085	MAPSET	PRAK085	01.034	24.00.00
COB085	PROGRAM	PRAK085	01.034	24.00.00
X085	TRANSACTION	PRAK085	01.034	24.00.00

SYSID=C001 APPLID=A06C001

RESULTS: 1 TO 3 OF 3

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

Abbildung 28: Ausgabe aller Komponenten der Gruppe "PRAK085"

Die Gruppe "PRAK085" besteht aus den drei Komponenten "MSET085", "COB085" und "X085", die wir unter CICS definiert und anschließend installiert haben.

Aufgabe: Installieren Sie eine CICS-Transaktion "U<Login-Nr.>, die eine Ausgabe ähnlich der

Abbildung 29 liefert und führen Sie diese dann aus. (Hinweis: Die Ausführung der beiden JCL-Scripte erfolgt fehlerfrei, wenn diese mit der Statusmeldung "MAXCC=0" beendet werden). Die Ausgabe Ihrer Transaktion muss unbedingt den Transaktionsnamen, "TUTORIAL 3 IN COBOL" sowie den Namen oder die Namen der Autoren enthalten. Benutzen Sie als CICS-Gruppennamen Ihren z/OS-Login-Namen (z.B. "PRAK085" oder "PRAK100").

Ersetzen Sie in den Bezeichnern für Ihren Mapset, Ihren Map sowie Ihr Cobol-Programm die Ziffern "085" durch die Nummer Ihres Prakt<xx> oder PRAK<xxx>-Accounts. Wenn Sie das nicht tun und mehrere Teilnehmer dieses Tutorial gleichzeitig bearbeiten, kommt es zu sehr unschönen Effekten, wie z.B. dass Sie den Screen von jemand anderem anstelle Ihres eigenen als Ergebnisscreen erhalten.

Aufgabe: Erzeugen Sie einen Screenshot Ihres Fensters, welches die Bildschirmausgabe von CICS enthält (Ihre Version der

Abbildung 29). Achten Sie darauf, dass das Bild nicht mehr als 250 KByte Speicherplatz belegt. Sehr gut ist das JPEG-Format, das mit weniger als 90 KByte auskommt. Schicken Sie Ihrem Tutor dieses Bild im Bitmap- oder JPEG-Format zu. Die Daten Ihrer Arbeit dürfen nicht gelöscht werden, damit der Tutor Ihre Transaktion aufrufen kann.

Aufgabe: Gehen Sie vom CUSTOMPAC MASTER APPLICATION MENU aus mittels SD.ST ins SDSF. Es erscheint eine Liste mit Logfiles. Löschen Sie alle Logfiles in der Print Queue, um so Server-Ressourcen wieder freizugeben. Einen Job dürfen Sie dort allerdings nicht killen: den in der Execution Queue. Denn das ist der Job, durch den Sie jetzt gerade eingeloggt sind. Für jedes Logfile, das Sie löschen möchten, geben Sie "p" (purge) in die Spalte NP sowie die entsprechende Zeile ein. Mit der Eingabetaste ist jetzt das Löschen eines jeden Logfiles noch einmal zu bestätigen. Möchten Sie mehrere Logfiles auf einmal im Block löschen, so ist das durch Eingabe von "//p" in die Spalte NP (erstes zu löschendes Logfile), die Eingabe von "/" in die Spalte NP (letztes zu löschendes Logfile) sowie Enter möglich. Ein bestätigendes zweites Enter löscht nun den gesamten ausgewählten Logfile-Block.

AUSGABE DER TRANSAKTION X085


TUTORIAL 3 IN COBOL

AUTOR: MAX MUSTERMANN

Abbildung 29: Ausgabe der Transaktion X085 der obigen Aufgabe

6. Logoff

Um uns aus CICS auszuloggen, betätigen wir so oft die Taste F3, bis die Meldung "STATUS: SESSION ENDED" ausgegeben wird. In die Zeile darüber geben wir die Logoff-Transaktion "CESF LOGOFF", gefolgt von der Eingabetaste, ein (s. Abbildung 30).



```
CESF LOGOFF
STATUS:  SESSION ENDED
```

Abbildung 30: Ausloggen aus CICS

Die CICS Shell besteht neben CEDA aus aus einer Reihe weiterer Transaktionen. Wenn Sie CICS auf einem z/OS Rechner installieren, werden diese Transaktionen automatisch mit installiert. CEDA, CESSN und CESF sind wichtig für Sie; den Rest werden Sie selten brauchen.

CBAM	Business Transaction Server browser
CDBC	database control menu
CDBI	database control inquiry
CDBM	database control interface
CEBR	temporary storage browse
CEBT	master terminal (alternate CICS)
CECI	command-level interpreter
CECS	command-level syntax-checking transaction
CEDA	resource definition online
CEDF	and
CEDX	the execution diagnostic facility
CEMT	master terminal
CEOT	terminal status
CESF	sign off
CESN	sign on
CEST	supervisory terminal
CETR	trace control
CIND	in-doubt testing tool
CLER	Language Environment run-time options
CMAC	messages and codes display
CMSG	message switching
CRTE	remote transactions
CSFE	terminal and system test
CSPG	page retrieval
CWTO	write to operator
DSNC	CICS DB2 transaction