**Unternehmensberatung**
Siegrid Houben
Oststraße 14
D 4052 Korschenbroich 2

*Computer anwendungen sind expertensache*

# Anschluß von Analysengeräten an einen Prozeßleitrechner

- Wolfgang Houben -

## 1) Zusammenfassung

Analysengeräte haben sehr unterschiedliche Übertragungsprotokolle daher entsteht ein hoher Softwareaufwand, wenn diese Geräte an Prozeßrechner angeschlossen werden müssen. Hier soll gezeigt werden, wie dieser Aufwand durch Anwendung eines in FORTRAN geschriebenen Unterprogrammpaketes erheblich reduziert werden kann.

## 2) Aufgabenstellung

Der Betrieb einer industriellen biologischen Kläranlage erfordert zur Aufrechterhaltung des Betriebs, zur Optimierung der Fahrweise und zur Einhaltung der staatlichen Umweltschutzauflagen heute eine umfangreiche Wasseranalytik. Die Menge der heute erforderlichen Analysen läßt sich durch Analyse im Labor nur noch zu ganz erheblichen Kosten durchführen. Daher sind die Betreiber solcher Kläranlagen bestrebt, die Analysen von automatischen Prozeßanalysensystemen durchführen zu lassen. Vor einigen Jahren reichte hierzu die Installation eines Gaschromatographen mit automatischer Probennahme abwechselnd am Kläranlageneinlauf und -auslaß aus. Die in den vergangenen Jahren ständig verringerten Grenzwerte, aber auch die Notwendigkeit zu immer kostengünstigerem Betrieb der Anlagen hat den Umfang der erforderlichen Prozeßanalytik ständig steigen lassen. Einer unserer Kunden hatte in den vergangenen Jahren drei verschiedene Analysengeräte von verschiedenen Softwareherstellern an seinen in der Kläranlage installierten Prozeßrechner[1] anschließen lassen.

Die automatischen Analysengeräte senden in festem Zyklus Ihre Meßergebnisse in Form von Protokollen (siehe Beispiele) spontan über eine serielle Leitung mit Baudraten zwischen 1200 und 9600 Baud. Da diese Protokollschnittstellen in der Regel nicht zur Kommunikation mit einem Rechner konzipiert wurden, sondern zur Ausgabe auf einem Drucker, sind Protokollanforderung, Blockcheck, Längsparity u.ä. oft nicht vorhanden. Der Prozeßrechner muß also in die Lage versetzt werden, die eingehenden Protokolle (auch mehrere gleichzeitig) korrekt zu empfangen und zu verarbeiten.

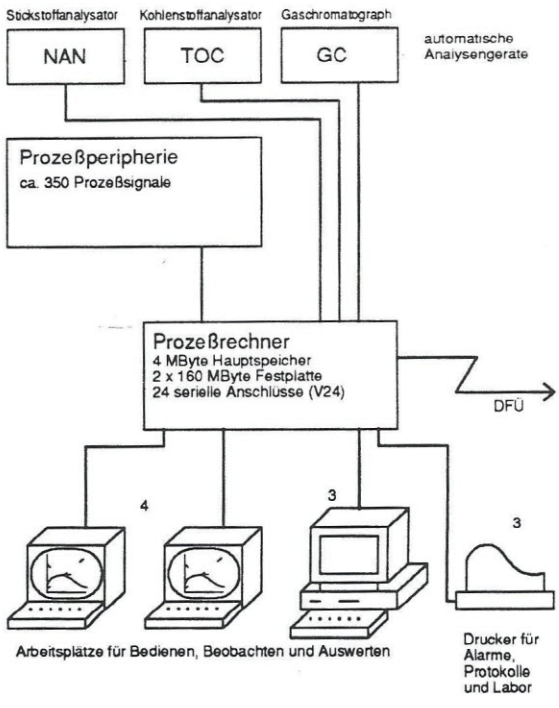


Bild 1, ursprüngliche Konfiguration

### Protokollbeispiel (NAN-Gerät)

```
D1992 02-10 14-14<LF><CR>
A0001001 14294<LF><CR>
S0001001 2.47          mg/Kg<LF><CR>
N0001000 2.47          mg/Kg<LF><CR>
D1992 02-10 14-42<LF><CR>
A0002001 2782712<LF><CR>
S0002001 481.96        mg/Kg<LF><CR>
N0002000 481.96        mg/Kg<LF><CR>
D1992 02-10 15-10<LF><CR>
A0003001 2716116<LF><CR>
S0003001 470.43        mg/Kg<LF><CR>
N0003000 470.43        mg/Kg<LF><CR>
                .
                .
D1992 02-10 18-24<LF><CR>
A9999001 2716116<LF><CR>
S9999001 470.43        mg/Kg<LF><CR>
N9999000 470.43        mg/Kg<LF><CR>
KALIBRIEREN<LF><CR>
```

[1] eine PDP 11/93 der Firma Digital Equipment mit dem Betriebssystem RSX-11M-PLUS

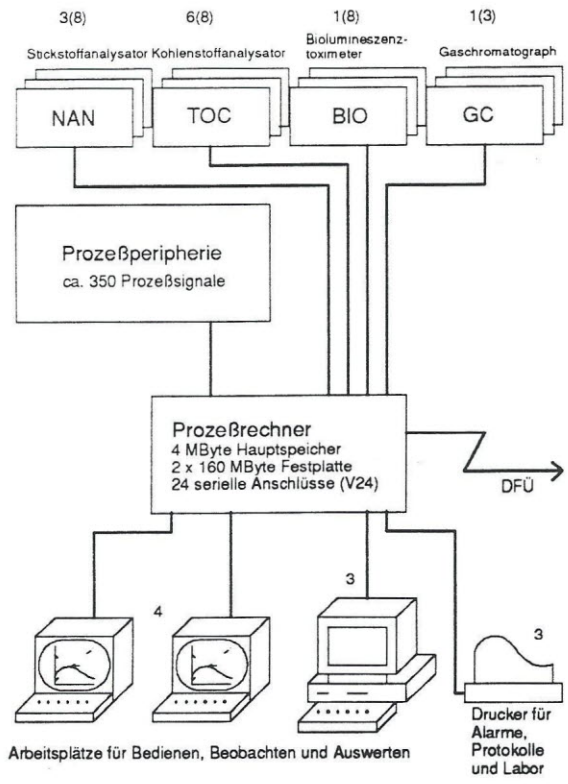


Bild 2, neue Konfiguration

**Protokollbeispiel (TOC-Gerät)**

```
CM0,1,07(JUL)-27-1990,20:19,2,TOC/ST
CM0,E
CC0,1,SM/TC1**,0020,05,0000,1,0,**
CC0,2,SM/TC1**,0020,05,1,*,**
CC0,3,SM/TC1**,010.0
CC0,E
CM1,1,07(JUL)-27-1990,20:19,001
CM1,2,08095,40.47,40.47/1.00,0020,05
CM1,E
CM2,1,07(JUL)-27-1990,20:19,001
CM2,2,07874,39.00,39.00/1.00,0020,05
CM2,3,1.47
CM2,E
CM1,1,07(JUL)-27-1990,20:25,002
CM1,2,06714,33.56,33.56/1.00,0020,05
CM1,E
CM2,1,07(JUL)-27-1990,20:25,002
CM2,2,08040,27.00,27.00/1.00,0020,05
CM2,3,6.56
CM2,E
```

Die Erweiterung der Analytik um einen weiteren Gerätetyp (BIO[2]) sowie der geplante Anschluß weiterer TOC[3]- und NAN[4]-Geräte wurde zum Anlaß genommen, für den Anschluß der Analysengeräte ein Gesamtkonzept zu erarbeiten. Die Erfahrung mit den bisherigen Treiberprogrammen hatten u.a. gezeigt, daß man bei Änderungen von Meßbereichen, Retentionszeiten und anderen Parametern immer wieder auf die Unterstützung durch die Softwarelieferanten angewiesen war. Daraus folgten für eine Gesamtlösung die nachstehenden Anforderungen.

## 3) Anforderungen

- Erstellung der Programme (soweit wie möglich) in einer höheren Programmiersprache
- Parameteränderungen ohne Änderung der Programme durch das Betreuungspersonal
- Unterstützung von maximal 8 Geräten je Gerätetyp
- Überwachung der Analysengeräte auf Einhaltung der Taktzeiten
- Überwachung der Meßergebnisse auf Einhaltung der Bereichsgrenzen
- Auswertung der Eich- und Testmessungen der Analysengeräte
- Änderung der Parametrierung ONLINE möglich, ohne Programmierung
- Anschluß weiterer gleichartiger Geräte ohne Programmierung
- Reduzierung des Aufwandes beim Anschluß neuer Gerätetypen

2 **Biolumineszenztoximeter**
3 **Total Organic Carbon Analyser**
4 **Stickstoffanalysator**

## 4) Lösungsweg

Als Programmiersprachen standen auf dem Prozeßrechner C, FORTRAN-77 und PASCAL zur Verfügung. Die Wahl fiel auf FORTRAN-77, da diese den besten Zugriff auf die Betriebssystemfunktionen bot und auch die existierenden Treiber zum überwiegenden Teil in FORTRAN erstellt waren.

Zur Lösung der beschriebenen Aufgabenstellung wurden von uns neben der eigentlichen Treibersoftware für das neu anzuschließende Gerät, auf die hier nur kurz eingegangen werden soll, die bestehenden Treiber gründlich überarbeitet und zwei allgemein verwendbare Module zum Empfang der Daten und zur Parametrierung der Treiber erstellt.

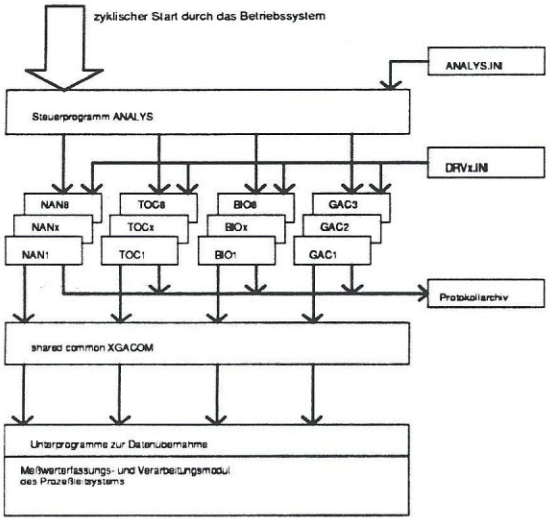


Bild 3, Aufbau des Treiberpaketes

Bild 3 gibt einen Überblick über den Aufbau des gesamten Treiberpaketes. Für jeden Gerätetyp wurde ein separates Treibermodul als eigenständige Task entwickelt. Diese Task wird für jedes angeschlossene Gerät vom Steuerprogramm ANALYS, dessen Ablaufplan in Bild 4 zu sehen ist, einmal installiert und gestartet.

Die Treiberprogramme wurden einheitlich nach der in Bild 5 gezeigten Struktur aufgebaut. Unterschiedlich sind jetzt nur noch die jeweils individuellen Protokollbearbeitungen.

Über die Betriebssystemschnittstelle erfragt der Treiber beim Start seinen Namen, in dem die Nummer des Analysengerätes enthalten ist. Über diese Nummer erfolgt die Kommunikation mit dem Steuerprogramm, die Auswahl der INI-Datei und die Datenverwaltung im *shared common*.

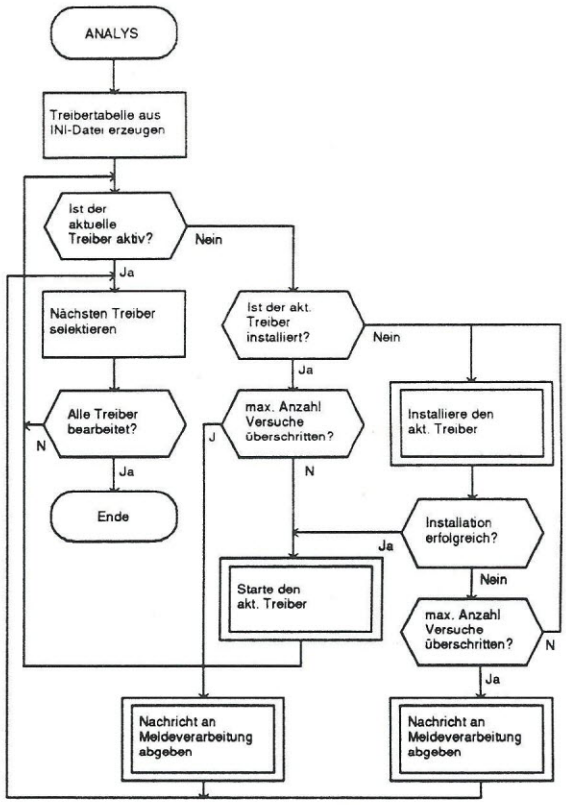


Bild 4, Ablaufplan des Programms ANALYS

Über die Steuerprogrammschnittstelle erhält der Treiber beim Systemshutdown seine Stopmeldung.

Über die INI-Dateischnittstelle werden beim Treiberstart die allgemeinen Parameter und die Schnittstellenparameter gelesen, vor jedem empfangenen Protokoll wird die Meßstellenkonfiguration neu gelesen.

Das Ein-/Ausgabe Modul empfängt die spontan gesendeten Protokolle, überwacht deren zeitgerechten Empfang und führt die Taktzeitüberwachung der angeschlossenen Geräte durch.

In der Protokollbearbeitung erfolgt die Auswertung der empfangenen Protokollzeilen und die Umwandlung der empfangenen Daten in die Gleitkommadarstellung.

Soweit möglich wird auch eine einfache Plausibilitätskontrolle der empfangenen Datensätze durchgeführt. Für betriebliche Zwecke erfolgt eine Archivierung der empfangenen Datensätze auf dem Plattenspeicher.

Die umgewandelten und geprüften Daten werden über einen *shared common* an das Prozeßleitsystem übergeben.


Bild 5, Aufbau eines Treiberprogramms

Bei auftretenden Fehlern werden dem Meldesystem des Prozeßrechners und an das Prozeßleitsystem entsprechende Fehlermeldungen übergeben.

## 5) Die entstandenen Werkzeuge

### 5.1) Ein-/Ausgabe Routinen

Für den Empfang der Daten wurde ein beim Kunden im Quellcode vorhandenes MACRO-11 Assemblerprogramm erweitert, und überarbeitet und dokumentiert, auf das hier nicht weiter eingegangen werden soll.

### 5.2) INI-Datei Schnittstelle

Für die Parametrierung erstellten wir für FORTRAN-77[5] eine Programmbibliothek[6], mit der die in einer Textdatei stehenden Parameter gelesen, konvertiert und geschrieben werden können. Das Format der Parameterdatei und der Programmaufrufe wurde an die in WINDOWS verwendeten Initialisierungsdateien und an den WINDOWS-Aufruf *GetPrivateProfileString* angelehnt.

Die Textdateien zur Parametrierung der Treiberprogramme können nun vom Betreuungspersonal mit einem Texteditor bearbeitet werden.

Durch diese Vorgehensweise wurde die aufwendige Erstellung individueller Dialogprogramme zur Parametrierung des Treiberpaketes vermieden.

Erstellt wurden die folgenden Unterprogramme:

GETPPS Lesen von Parametern

TRENN Zerlegen und umwandeln von Parameterstrings

PUTPPS Schreiben von Parametern

---

5 **unter dem Betriebssystem RSX-11M-PLUS auf der PDP11, der PDP11 FORTRAN-Compiler ist quellcodekompatibel zu VAX-FORTRAN**

6 **die Module sind auf Anfrage auch für andere Rechner und Compiler verfügbar**

### 5.2.1) Das Unterprogramm GETPPS

**Syntax:**

```
CALL GETPPS( ILUN, FILNAM,
     APPNAM, KEYNAM, STRBUF,
     STRLEN, DEFKEY )
```

**Beschreibung:**

Dieses Unterprogramm kopiert einen String aus der in *FILNAM* angegebenen Initialisierungsdatei in einen Puffer, der durch den Parameter *STRBUF* angegeben wird. Im Parameter *STRLEN* wird die Anzahl der übergebenen Zeichen oder ein entsprechender Fehlercode übergeben. Das Unterprogramm durchsucht die Datei nach einer Schlüsselanweisung, die mit der vom Parameter *KEYNAM* vorgegebenen Bezeichnung unter dem Programmkopf der vom Parameter *APPNAM* vorgegebenen Anwendung übereinstimmt. Wird diese gefunden, wird der zugehörende String in den Puffer kopiert. Wurde die Schlüsselanweisung nicht gefunden, wird die durch den Parameter *DEFKEY* angegebene Standardzeichenkette in den Puffer kopiert. Ein Stringeintrag in der Initialisierungsdatei muß folgende Form haben:

```
[Anwendungsname]
Schlüsselname = string
```

Ist *KEYNAM* gleich '*', wird bei jedem Aufruf die jeweils nächste Zeile nach [Anwendungsname] übergeben. Ist keine weitere Zeile vorhanden, wird *STRLEN* <0 übergeben.

***APPNAM*:**

(CHARACTER*x oder CHARACTER-Konstante) ist ein String mit dem Namen einer Anwendung, der in der Initialisierungsdatei erscheint.

***KEYNAM*:**

(CHARACTER*x oder CHARACTER-Konstante) ist ein String mit dem Namen eines Schlüssels, der in der Initialisierungsdatei unter *APPNAM* erscheint.

***DEFKEY*:**

(CHARACTER*x oder CHARACTER-Konstante) ist ein String mit dem Standardwert der Schlüssels, der dann eingesetzt wird, wenn die Schlüsselanweisung nicht gefunden werden kann.

***STRBUF*:**

(CHARACTER*x) ist eine CHARACTER Variable oder ein BYTE-Array, in das die übergebenen Zeichen kopiert werden.

***STRLEN*:**

(INTEGER*2) ist die Länge der übergebenen Zeichenkette (ohne den anschließenden NULL-Terminator) oder im Fehlerfall eine negative Fehlernummer.

***ILUN*:**

(INTEGER*2) ist eine FORTRAN *Logical Unit Number,* über die die notwendigen Dateizugriffe abgewickelt werden. Auf die Initialisierungsdatei wird von GETPPS *'shared'* und *'readonly'* zugegriffen. Nach Rückkehr aus dem Unterprogramm GETPPS ist die Initialisierungsdatei geschlossen und die *Logical Unit Number* steht für andere Operationen wieder zur Verfügung.

***FILNAM*:**

(CHARACTER*x oder CHARACTER-Konstante) enthält den Dateinamen der Initialisierungsdatei, eventuell mit vollständiger DEVICE- und DIRECTORY-Angabe.

**Hinweise:**

Bei *STRBUF* ist zu beachten, daß GETPPS keine Prüfung der maximalen Länge des übergebenen Strings durchführt. Die Variable *STRBUF* muß also ausreichend groß dimensioniert werden (i.d.R. reichen 80 Zeichen).

GETPPS läßt die Schreibweise der Anwendungs- und Schlüsselnamen unberücksichtigt, so daß die Strings in *APPNAM* und *KEYNAM* in einer beliebigen Kombination von Groß- und Kleinschreibung eingegeben werden können.

In der Initialisierungsdatei werden Zeilen, die mit einem Semikolon beginnen, als Kommentar überlesen.

**Beispiel:**

```
CALL GETPPS ( ILUN, 'BIO1.INI',
     'Schnittstelle', 'Leitung',
     STRBUF, STRLEN, 'NL:' )
```

der vorstehende Aufruf liest den Parameter *Leitung* unter der Überschrift *Schnittstelle* aus der nachfolgenden Datei:

```
[Schnittstelle]
Leitung   = TT14:
Baudrate  = 9600
Wartezeit = 900
```

### 5.2.2) Das Unterprogramm TRENN

**Syntax:**

```
CALL TRENN ( STRBUF, STRLEN,
     DELIM, F1, P1
     [, F2, P2 [, Fn, Pn ] ] )
```

**Beschreibung:**

Dieses Unterprogramm zerlegt die in *STRBUF* abgespeicherte Zeichenkette mit der Länge *STRLEN* in durch das in *DELIM* abgelegte Trennzeichen getrennte Elemente und wandelt diese entsprechend dem in *Fx* angegebenen Formatschlüssel um und speichert das Ergebnis in *Px*. Die Anzahl der Fx-Px Parameterpaare ist variabel und wird vom Unterprogramm TRENN selbständig erkannt.

***STRBUF*:**

(CHARACTER*x) ist eine CHARACTER Variable oder ein BYTE-Array, in dem die umzuwandelnde Zeichenkette übergeben wird.

***STRLEN*:**

(INTEGER*2) ist die Länge der übergebenen Zeichenkette (ohne den anschließenden NULL-Terminator).

***DELIM*:**

(CHARACTER*1 oder CHARACTER-Konstante) ist das Trennzeichen, mit die einzelnen Elemente des Parameterstrings getrennt sind.

***Fx*:**

(CHARACTER*2 oder CHARACTER*3) ist der Formatcode, nach dem das zugehörige Element umgewandelt werden soll. Folgende Formatcodes werden derzeit unterstützt:

| | |
|---|---|
| Cn | : Zeichenkette der Länge n |
| CZn | : Zeichenkette der Länge n <NULL> terminiert |
| I2 | : INTEGER*2 |
| I4 | : INTEGER*4 |
| R4 | : REAL*4 |
| R8 | : REAL*8 |

***Px*:**

ist der Name der Variablen, in der der umgewandelte Wert abgelegt wird. Der Datentyp muß zu der in *Fx* angegebenen Umwandlung passen.

**Hinweise:**

Eine Prüfung des Datentyps für den Parameter *Px* kann nicht durchgeführt werden.

**Beispiel:**

```
CALL GETPPS ( ILUN, 'GAC1.INI',
     'Komponenten','K1', STRBUF,
     STRLEN, 'NL:' )
CALL TRENN (  STRBUF, STRLEN,
     ',', 'C8', KNAME(i), 'I2',
     RTZ1(i), 'I2', RTZ2(i) )
```

Das vorstehende Programmfragment liest den zum Schlüsselwort *K1* gehörenden Parameterstring unter der Überschrift *Komponenten* aus der nachfolgenden Datei:

```
[Komponenten]
K1   = CH4,110,135
K2   = CH6,145,160
```

und zerlegt diesen Parameterstring in die Komponenten *CH4*, *110*, *135* und speichert die Ergebnisse in den Variablen *KNAME(i)*, *RTZ1(i)* und *RTZ2(i)*.

### 5.2.3) Das Unterprogramm PUTPPS

**Syntax:**

```
CALL PUTPPS ( ILUN, FILNAM,
     APPNAM, KEYNAM, STRBUF,
     IERR )
```

**Beschreibung:**

Dieses Unterprogramm kopiert einen String der in *STRBUF* gespeichert ist in die in *FILNAM* angegebene Initialisierungsdatei. Das Unterprogramm durchsucht die Datei nach einer Schlüsselanweisung, die mit der vom Parameter *KEYNAM* vorgegebenen Bezeichnung unter dem Programmkopf der vom Parameter *APPNAM* vorgegebenen Anwendung übereinstimmt. Wird diese gefunden, wird der in *STRBUF* gespeicherte String in die Datei kopiert. Wird *KEYNAM* nicht gefunden, wird der Schlüsselname neu in die Datei eingefügt. Wird *APPNAM* nicht gefunden, werden sowohl der Anwendungsname als auch der Schlüsselname neu in die Datei eingefügt.

***APPNAM*:**

(CHARACTER*x oder CHARACTER-Konstante) ist ein String mit dem Namen einer Anwendung, der in der Initialisierungsdatei erscheint.

***KEYNAM*:**

(CHARACTER*x oder CHARACTER-Konstante) ist ein String mit dem Namen eines Schlüssels, der in der Initialisierungsdatei unter *APPNAM* erscheint.

***STRBUF*:**

(CHARACTER*x) ist eine CHARACTER Variable oder ein BYTE-Array, aus dem die übergebenen Zeichen kopiert werden.

***ILUN*:**

(INTEGER*2) ist eine FORTRAN *Logical Unit Number*, über die die notwendigen Dateizugriffe abgewickelt werden. Auf die Initialisierungsdatei wird von PUTPPS *'shared'* und *'readonly'* zugegriffen. Nach Rückkehr aus dem Unterprogramm PUTPPS ist die Initialisierungsdatei geschlossen. Die *Logical Unit Number* steht für andere Dateizugriffe wieder zur Verfügung.

***FILNAM*:**

(CHARACTER*x oder CHARACTER-Konstante) enthält den Dateinamen der Initialisierungsdatei, eventuell mit vollständiger DEVICE- und DIRECTORY-Angabe.

***IERR*:**

(INTEGER*2) enthält nach Ausführung des Unterprogramms PUTPPS den Status der durchgeführten Operation. Folgende Werte werden übergeben:

<= 0 : Operation fehlerhaft
= 1 : KEYDEF wurde ersetzt
= 2 : KEYDEF wurde nicht gefunden;
Der Schlüsselname wurde neu in die Datei eingetragen.
= 3 : APPNAM wurde nicht gefunden;
Der Applikationsname und der der Schlüsselname wurden neu in die Datei eingetragen.

**Hinweise:**

Zum Eintragen der geänderten Schlüsseldefinition wird die Initialisierungsdatei vom Unterprogramm PUTPPS vollständig kopiert. Für diese Schreiboperation wird die *Logical Unit Number* ILUN+1 verwendet. Auch diese *Logical Unit Number* steht nach Ausführung des Unterprogramms PUTPPS für andere Dateizugriffe wieder zur Verfügung.

Da von PUTPPS die gesamte Datei kopiert wird ist bei längeren Initialisierungsdateien der Zeitaufwand eventuell nicht vernachlässigbar.

Beispiel:

```
CALL PUTPPS ( ILUN, 'GAC1.INI',
     'Komponenten', 'K1',
     'CH4,112,133', IERR )
```

Das vorstehende Programmfragment schreibt den zum Schlüsselwort *K1* gehörenden Parameterstring *'CH4,112,133'* unter der Überschrift *Komponenten* in die Initialisierungsdatei mit dem Namen *GAC1.INI*.

### 6) Erfahrungen

Die beschriebene Unterprogrammbibliothek wurde von uns bisher in vier weiteren Projekten erfolgreich eingesetzt.

## 7) Programmbeispiel eines Treiberprogramms unter Verwendung von GETPPS und TRENN.

```
C     ***********************************************************************
      PROGRAM   NAN
C     ***********************************************************************
C
C      Aufgabe des Programms :
C
C     Treiber fuer Stickstoff-Analysengeraet TC5000
C
C     Freigabe :
C
C      Aenderungen / Erweiterungen
C
C     Rel  |Datum        |Autor    |Beschreibung
C     ------------------------------------------------------------------
C     1.1  16-Feb-1991
C     2.0  08-SEP-1991             Anpassungen fuer ANALYS
c     3.0  12-Mar-1992   W. Houben Neues Protokoll des TC5000
c     3.1  17-Mar-1992   W. Houben Aufbau ueberarbeitet

C     ------------------------------------------------------------------
C     allgemeine INCLUDE-Datei
C     ------------------------------------------------------------------

      implicit none                           ! Vollstaendige Deklaration

      include   'sy:[3,321]gdrv.inc/NOLIST'

C     ------------------------------------------------------------------
C     VARIABLEN
C     ------------------------------------------------------------------

      REAL*8    NANNTB (CHAMAX)               ! Namentabelle
      REAL*4    NANZWS (CHAMAX)               ! Zwischenspeicher
      BYTE      BUFF (64)                     ! Eingangsstring von Schnittst.
      INTEGER*4 AREA                          ! Wert fuer Area
      INTEGER*2 KALI                          ! Kalibrierkennung

c     ------------------------------------------------------------------
c     Parameter
c     ------------------------------------------------------------------

      parameter t        = .true.
      parameter f        = .false.
      parameter IFLG= 1
      parameter INULL    = 0
      parameter NTEL= 64
      parameter IBS = CR

C     ------------------------------------------------------------------
C     FORMATANGABEN
C     ------------------------------------------------------------------

333   FORMAT    (BZ,I4)
444   FORMAT    (BZ,I<N>)
555   FORMAT    (BZ,F<N>.0)
```

```
C    ======================================================================
C    PROGRAMMSTART
C    ======================================================================

     CALL ERRSET (64,t,f,t,f,1)              ! disable input
                                             ! conversion error
     CALL ERRSET (68,t,f,t,f,1)              ! disable variable format
                                             ! expression value error
     CALL ERRSET (72,t,f,f,f,1)              ! disable floating overflow
     CALL SWERLN ('CO',0,IERR)               ! switch Errors to co:
     CALL INITSK ('IP:[3,3]',PNAME,TNAME,TINDEX)     ! Task-Init
     CALL INILIN (PARLUN,PNAME,SERLUN,IERR)          ! Init Schnittstelle
     IF (IERR.LE.0) CALL EXIT                ! Initialisierungsfehler
     CALL MAPINI                             ! MAP to IMPAS Database
C
C    lese Event Flag # aus dem INI-File
C
     call getpps (parlun,pname,'Basisparameter','Eventflag',buff,1,'68')
     if (1.le.0) stop 'Parameterfile Error'
     call trenn (buff,1,',',endfla,'I2')
     if ((endfla.lt.65.or.endfla.gt.96)
     1               .and.(endfla.ne.0)) stop 'kein Group Global Flag'

C    ======================================================================
C    HAUPTSCHLEIFE
C    ======================================================================

 10  if (endfla.ne.0) then                   ! beenden signalisiert?
        call readef (ENDFLA,I)
        if (I.eq.0) then
            call msg (-1,'Treiber Ende durch Flag ',ENDFLA)
            call exit
        end if
     end if
C
C    Messstellennamen einlesen
C
     call getpps (PARLUN,PNAME,'Messstellen','*',buff,1,0)
     if (1.gt.0) THEN
         call trenn (buff,1,',',k,'I2',MSTNAM,'C8')
         NANNTB(k) = MSTNAM
         goto 10
     END IF
C
C    Timeout Intervall einlesen
C
     call getpps (PARLUN,PNAME,'Basisparameter','Intervall',buff,1,'2400')
     decode (1,'(I<1>)',buff(1)) ITOU
C
C    Debugstatus einlesen
C
     call getpps (PARLUN,PNAME,'Basisparameter','Debug',buff,1,'nein')
     IF ((buff(1).eq.'j').or.(buff(1).eq.'J')) THEN
      imsg = -1
       ELSE
        imsg = 0
         END IF
C
C    Startmeldung ausgeben
C
     call msg (imsg,'Start Messung Geraet ',TINDEX)
```

```
C    ------------------------------------------------------------------
C    Lesen bis CR
C    ------------------------------------------------------------------

 20  CALL INPOUT (SERLUN,IFLG,ITOU,INULL,INULL,BUFF,NTEL,IBS,IERR)

     if (ierr.eq.-15) then                  ! Timeout melden
        CALL MSG (IMSG,'Timeout = ',ITOU)   ! *** DEBUG ***
        DO 40 I = 2,CHAMAX                  ! alle Eintraege auf
 40        XGABUF (I,TINDEX,OFFNAN) = GRNZBR    ! Bruch (BR) stellen
     else if (ierr.le.0) then               ! Alle anderen Fehler
        GOTO 20                             ! ignorieren
     else                                   ! Datensatz bearbeiten
        CALL MSG (IMSG,'Input=/A16',BUFF,' IERR=',IERR) ! *** DEBUG ***
        ILAN = IERR
        IF (BUFF(1).EQ.STX) THEN            ! STX entfernen
            DO 23 I =1,ILAN
 23            BUFF(I) = BUFF(I+1)
            ILAN = ILAN-1
        END IF

C    ------------------------------------------------------------------
C    Eingabestring auf gueltigen Eintrag untersuchen
C    ------------------------------------------------------------------

C    ------------------------------------------------------------------
        if (buff(1).eq.'M'.or.              ! Kalibriernachricht
     1      buff(1).eq.'D') then            ! und Analysenuhrzeit
           goto 20                          ! ignorieren
C    ------------------------------------------------------------------
        else if (buff(1).eq.'A') then       ! Ident = 'A' : Area
           CALL SUB1 (BUFF,ILAN,M,N)
           call msg (imsg,'A: ILAN=',ILAN,' M=',M,' N=',N)
           DECODE (N,444,BUFF(M)) AREA
           DECODE (4,333,BUFF(2)) KALI
           IF (KALI.EQ.9999) THEN
            NANZWS(4) = AREA/1000           ! KALIBRIERWERT
             ELSE
              NANZWS(1) = AREA/1000         ! Messwert
               END IF
           GOTO 20                          ! auf neue Eingabe warten
C    ------------------------------------------------------------------
        else if (buff(1).eq.'S') then       ! Ident = 'S' : Concentrat.
200        CALL SUB1 (BUFF,ILAN,M,N)
           call msg (imsg,'S: ILAN=',ILAN,' M=',M,' N=',N)
           DECODE (4,333,BUFF(2)) KALI
           IF (KALI.EQ.9999) THEN
            DECODE (N,555,BUFF(M)) NANZWS(5)    ! KALIBRIERWERT
             ELSE
              DECODE (N,555,BUFF(M)) NANZWS(2)  ! Messwert
               END IF
           GOTO 20                          ! auf neue Eingabe warten
C    ------------------------------------------------------------------
        else if (buff(1).eq.'N') then       ! Ident = 'N' : Mean Value
300        CALL SUB1 (BUFF,ILAN,M,N)
           call msg (imsg,'N: ILAN=',ILAN,' M=',M,' N=',N)
           DECODE (4,333,BUFF(2)) KALI
           IF (KALI.EQ.9999) THEN
            DECODE (N,555,BUFF(M)) NANZWS(6)    ! KALIBRIERWERT
             ELSE
              DECODE (N,555,BUFF(M)) NANZWS(3)  ! Messwert
               END IF
           call OPEPRT (PARLUN,PNAME,IDATUM,PRTLUN) ! Protodatei oeffnen
```

```
c
c          Werte an IMPAS uebergeben und als Protokoll ausgeben
c
           CALL MESCOM (IMSG,NANNTB,NANZWS,IDATUM,TINDEX,OFFNAN,PRTLUN)
           WRITE (PRTLUN,11) CR,LF              ! Leerzeile
           CLOSE (UNIT=PRTLUN)                  ! Protokolldatei schliessen
           CALL MSG (IMSG,'Ende der Berechnung')     ! *** Debug ***
C     ------------------------------------------------------------------
         else                                   ! Alles andere ignorieren
           GOTO 20
         end if                                 ! <<BUFF>>
       END IF                                   ! <<IERR>>

       GOTO 10                                  ! weiter mit naechster
                                                ! Analyse
       END
```