LZH Datenbank
und
Datenerfassung

dokumentiert im Auftrag der
Landeshaupstadt München
Baureferat Hochbau 6
Landsberger Straße 6

von

Dr.-Ing. Ernst Oeser
Technischer Berater für Rechneranwendungen
Bahnhofstr. 11a
D-85622 Feldkirchen

INHALT

EINLEITUNG

Dieses Dokument beschreibt das Design und die Implementierung der Datenbank für die Leitzentrale Haustechnik (LZH) , die Programme zur kontinuierlichen Datenerfassung der Daten, die von den GA-Knoten an die LZH geschickt werden, und einige PC-Anwendungen, die eine Kommunikation zwischen einer Bedienstation und den GA-Knoten ermöglichen. Im einzelnen werden folgende Punkte behandelt:

- Design und Implementierng der Datenbank

- Kommunikation mit der LZH
  - Konzept der Kommunikation
  - Das Erfassungsprogramm ´gakserver´
  - Das Programm ´userver´ zur Verwaltung von Client-Applikationen
  - Das Programm ´tsynch´ zur Korrektur der Systemzeit
  - Das interne Protokoll
  - Die verwendeten Datagramme des internen Protokolls
  - Das Programm ´gakimport´ zum Einlesen von Datenpunkten von einer Diskette
- PC-Programme
  - Der lokale PC-DDE-Server-Prozeß ´gakdde´
  - Das Formular ´Anzeige-Modus´
  - Das Formular ´FND1.0-Kommunikation´
  - Das Formular ´Istwert-Abfrage´
  - Hinweise für Entwickler

DESIGN UND IMPLEMENTIERUNG DER DATENBANK

Vorgaben

Für das Design und die Implementierung der Datenbank für die Leitzentrale Haustechnik war das Dokument **Fachliches Feinkonzept für die Leitzentrale Haustechnik** (Verfasser Dipl.-Ing. W. Fries) in der Fassung vom 26. Juli 1995 maßgeblich; im folgenden wird es kurz ´Feinkonzept´ genannt.

Betriebssystem

Die Datenbank wurde erstellt auf einem SUN-Server unter dem Betriebssystem Solaris. Im folgenden wird dieser Rechner der LZH-Server genannt (LZH: Leitzentrale Haustechnik).

X.25-Kommunikation

Die Kommunikationsweg von den in den Objekten installierten GA-Knoten zum LZH-Server verläuft zunächst über ISDN von den GA-Knoten zu einem ISDN/X.25-Router. Der Router setzt das ISDN-Protokoll in das X.25-Protokoll um, und schickt die Daten in diesem Format zum LZH-Server.

Damit der LZH-Server die X.25 Datenströme verarbeiten kann, ist er mit einer X.25-Schnittstellenkarte und der entsprechenden Treibersoftware ausgestattet. Die Treibersoftware ist nicht im Lieferumfang des Betriebssystems Solaris enthalten. Sie muß von einer seperaten CDROM installiert werden, wobei die Beschaffung einer Lizenz über die SUN-Hotline erforderlich ist.

Die im LZH-Server verwendete Schnittstellenkarte enthält vier X25-Schnittstellen. Für die Kopplung mit dem ISDN/X.25 wird die erste Schnittstelle (HSI-0) benutzt. Sie ist über einen Schnittstellenkonverter (X.21 to RS-449) mit einem Modem verbunden, der die Übertragung zum ISDN/X.25-Router übernimmt. Die Schnittstelle muß nach jedem Start der X.25 Treibersoftware auf die Leitungscharakteristiken der Verbindung zum ISDN-Router neu eingestellt werden. Dazu dient das Shell-Script /etc/rc2.d/S88hsi_init‘, welches nach dem Booten des LZH-Servers automatisch ausgeführt wird. Dieses Script muß auch immer dann ausgeführt werden, wenn man das X.25-System auf dem LZH_Server neu starten will.

Datenbanksystem

Als Datenbanksystem wurde Oracle in der Version 7.2.2 verwendet. Es stand als Speichermedium ein Raid-System für die alleinige Benutzung durch die Datenbank zur Verfügung.

Installation der Datenbank

Die Installation des Datenbanksystems wurde in folgenden Schritten durchgeführt:

Als Benutzer ´root´:

Zunächst wurden einige Systemparameter gemäß den Vorgaben des Oracle-Installations-Handbuches angepaßt. Dazu editiert man die Datei ´/etc/system´ und erzeugt eine leere Datei mit dem Namen ´/reconfigure´. Das bewirk, daß beim nächsten Booten die neuen Werte der Datei ´/etc/system´ berücksichtigt werden. Die notwendigen Einstellungen für die Verwendung von Oracle in ´/etc/system´ wurden wie folgt eingetragen:

```
set shmsys:shminfo_shmmax=128000000
set shmsys:shminfo_shmmin=1
set shmsys:shminfo_shmmni=100
set shmsys:shminfo_shmseg=10
set shmsys:shminfo_semmns=200
set shmsys:shminfo_semmni=70
set shmsys:shminfo_semmsl=25
```

Einfügen der Port-Number 1525 (nötig für SQL*Net) in die Datei ´/etc/services´:

```
orasrv 1525/tcp oracle
```

Anlegen der Benutzergruppe ´dba´.

Dann wurde das $HOME-Directory des Benutzers ´oracle´ angelegt. Es ist das Root-Directory der Slice ´/dev/c0t0d0s6´ des RAID-Arrays, es wird als ´/oracle´ gemounted (Siehe Datei /etc/fstab). (Die weiteren Slices des RAID-Arrays wurden für die Tablespaces der Datenbank verwendet.):

```
# cd /oracle
# mkdir orahome
# chown oracle orahome
# chgrp dba    orahome
# chmod 755    orahome
# cd orahome
# mkdir odoc
# chown oracle odoc
# chgrp dba    odoc
# chmod 755    odoc
```

Anlegen des Benutzers ´oracle´: Sein $HOME-Directory ist das oben erwähnte Verzeichnis ´/oracle´. Er muß zur Gruppe ´dba´ gehören. Es wurde die Korn-Shell (ksh) als Standard-Shell ausgewählt.

Dann als Benutzer ´oracle´ weiterarbeiten.

Zunächst sind die benötigten Umgebungsvariablen gemäß Oracle-Installationshandbuch zu setzen. Siehe hierzu die Datei ´/oracle/.profile´. Wichtig ist, daß während des folgenden Installationsvorganges die Variable ´NLS_LANG´ nicht gesetzt ist. Dann legt man die CD ´Oracle 7 V7.2.2.3.0´ in das CD-Laufwerk ein. Sie wird automatisch gemounted. Anschließend wird der ´Oracle Installer´ wie folgt aufgerufen:

```
$ cd /cdrom/oracle/orainst
$ ./orainst
```

Im Verlauf der interaktiven Installation wurden die folgenden Optionen ausgewählt:

- Frech install
- Produkte:
  - Oracle UNIX Installer
  - PL/SQL V2.2.2.3.0
  - RDBMS V7.2.2.3.0
  - PROC*C V2.1.2.0.0
  - SQL*PLUS V3.2.2.0.0
  - SQL*Net V2
  - TCP/IP Adapter
- bei der Landessprache den Wert `we8dec´ wählen
- als Instance-Name (SID) ´HB6´ wählen
- bei den Tablespaces wurden folgende Größen gewählt:
  - RBS 200MB
  - SYSTEM 100MB
  - TEMP 200MB
  - TOOLS 15MB
  - USER 3000MB

Anschließend als Benutzer ´root´ das script ´root.sh´ ausführen:

```
# cd /oracle/orahome/orainst
# sh ./root.sh
```

Dann die Datei ´/var/opt/oracle/oratab´ editieren.

Dann wieder als Benutzer ´oracle´ einloggen und die Patches für Oracle installieren. Dazu die bei der Installation hochgefahrene Datenbank herunterfahren (´dbshut´) und dann die Installtions-CD mit dem Befehl ´eject´ auswerfen lassen. Dann die Patch CD ´V7.2.2.4´ einlegen.

```
$ cd /cdrom/oracle#1
$ ./README
```

ausführen und anschließend die Hinweise ausführen:

```
$ cd /oracle/orahome/rdbms/lib
$ make –f oracle.mk install
```

Dann die Datei ´/oracle/orahome/dbs/initHB6.ora´ anpassen und mit folgendem Befehl dafür sorgen, daß administartive Aufgaben auch über Netzwerk möglich werden:

```
$  orapwd file=/oracle/orahome/dbs/orapwHB6 –
$> password=bahnhof entries=30
```

Installation der Konfigurationsdateien für SQL*Net V2. Die benötigten Dateien werden auf einem PC mit dem Oracle Network Manager Tool erzeugt und müssen anschließend von dort in das Verzeichnis ´/var/opt/oracle´ kopiert werden. Folgende Dateien werden benötigt:

- `/var/opt/oracle/listener.ora`
- `/var/opt/oracle/tnsnames.ora`
- `/var/opt/oracle/sqlnet.ora`

Datenbankdesignwerkzeuge

Das Datenbankdesign wurde vollständig mit dem Oracle Werkzeug **Designer/2000** erstellt auf einem PC unter Windows NT 4.0. Dabei wurde wie folgt vorgegangen:

Zunächst wurden mit Hilfe des **Entity Relationship Diagrammer** die Entitäten und ihre Beziehungen zueinander formuliert. Daraus wurden mit Hilfe des **Database Design Wizard** die Datenbanktabellen innerhalb des Repositories des Designers/2000 generiert.

Mit Hilfe des **Data Diagrammers** wurden die Tabellen und ihre Spalten um weitere im Feinkonzept geforderte Eigenschaften ergänzt. Insbesondere wurde versucht, mit Hilfe der im **Designer/2000** zur Verfügung stehende Werkzeuge möglichst viele der geforderten Eigenschaften der Datenfelder so festzulegen, daß sie bereits vom Datenbanksystem bei einer Eingabe oder Änderung überprüft werden; siehe hierzu den Abschnitt 5.2 des Feinkonzepts, **ARZFI**-Notation. Es zeigte sich, daß bei weitem nicht alle im Feinkonzept geforderten Eigenschaften durch das Design der Datenbank abgedeckt werden können. Die vollständige Implementierung aller dieser Forderungen muß durch spezielle Anwenderprogramme realisiert werden.

Schließlich wurden mit Hilfe des **Server (DDL) Generators** Script-Dateien erzeugt, die dazu dienten, die Tabellen inklusive aller Constraints, Indices und Sequences auf dem SUN-Datenbankserverrechner mit Hilfe des Programms **SQLPLUS** anzulegen. Bei der Generierung des DDL-Scripts erzeugt der **Designer/2000** Script-Dateien mit folgenden Datei-Endungen:

*.con Definition der Constraints
*.ind Definition der Indices
*.seq Definition von Oracle Sequences
*.tab Definition der Tabellen und ihrer Spalten
*.sql enthält Aufrufe zur Ausführung der oben genannten Dateien

Alle Script-Dateien liegen im Quelltext vor und sollten als primäre Referenz und Auskunft über die Tabellen benutzt werden. In vielen Fällen enthält eine Datei mit der Endung *.tab die Beschreibung mehrerer Tabellen, weil man bei der Entwicklung Tabellen, die eine enge Beziehung zueinander haben, in einem Arbeitsgang implementiert und anschließend die Implementierung getestet hat. Die vollständige Liste der Script-Dateien mit der Endung *.tab findet man im Anhang zu diesem Dokument. Diese Liste enthält auch die richtige Reihenfolge, die einzuhalten ist, wenn

man alle Tabellen ausgehend von einer leeren Datenbank völlig neu erstellen möchte.

Neben den Script-Dateien, die eine detaillierte Referenz für die Tabellen darstellen, dient ein Entity-Relationship-Diagramm (Größe ca. 1 qm) als gute Übersicht über das gesamte Tabellensystem und den vielfachen Beziehungen der Tabellen untereinander. Es sollte in jedem Falle bei der Entwicklung von Anwenderprogrammen zur Verfügung stehen.

Allgemeine Implementierungsdetails

Die im Feinkonzept als ´*** Verweis ***´ beschriebenen Datentypen sind entweder als ´Foreign Key´ implementiert, wenn es sich um einen Verweis auf einen einzigen Datensatz handelt, oder als eigenständige Tabelle, wenn auf eine zugehörige Liste verwiesen wird. Eine Implementierung als Liste erfolgte auch dann, wenn im Feinkonzept eine Formulierung wie ´n-mal ***Verweis***´ gewählt wurde.
Datums- und Zeitangaben wurden grundsätzlich als eine Tabellenspalte mit dem datenbankinternen Datentyp ´DATE´ implementiert.
Bitfelder (im Feinkonzept mit dem Datentyp m x B*n bezeichnet) wurden in vielen Fällen in einzelne numerische Spalten aufgeteilt und Spaltennamen vergeben, die der Bedeutung des Bits (B*1) oder der Bitmuster (B*2, B*3, ...) entsprechen.

Spezielle Implementierungsdetails

Datenpunktstamm (5.6.1)

Tabelle: DATENPUNKTE
Diese Tabelle war im Feinkonzept als Tabelle mit weiteren sogenannten ´Ergänzungsstrukturen´ beschrieben, wobei der Aufbau der Ergänzungsstrukturen abhängig vom Typ des Datenpunktes ist. Implementiert wurde diese Forderung durch eine einzige Tabelle, in der neben den Spalten, die für alle Datenpunkttypen vorgeschrieben sind, zusätzlich die Spalten aller möglichen Ergänzungsstrukturen optional enthalten sind. Ferner wurde auch die zu jedem Datenpunkt gehörende Wartungsstruktur (s. 5.6.1.3.2 im Feinkonzept) in die Tabelle DATENPUNKTE mit aufgenommen.

Ergänzungsstruktur für Melde- und Schaltpunkte (5.6.1.3.1.1)

In der Tabelle DATENPUNKTE enthalten. Die Spalte ´Kriterienblock´ ist durch die Tabelle MEKBLOCK implementiert. Die Spalten ´Meldungs-Profile´ und ´Meldungs-Maske´ werden durch die Tabelle ACTUALMELDUNGEN implementiert.

Ergänzungsstruktur für Meßpunkte (5.6.1.3.1.2)

In der Tabelle DATENPUNKTE enthalten.

Ergänzungsstruktur für Stellpunkte (5.6.1.3.1.3)

In der Tabelle DATENPUNKTE enthalten.

Ergänzungsstruktur für Zählpunkte (5.6.1.3.1.4:

In der Tabelle DATENPUNKTE enthalten.

Ergänzungsstruktur für virtuelle Datenpunkte (5.6.1.3.1.5)

In der Tabelle DATENPUNKTE enthalten. Die Berechnungsformeln sind als Verweise auf die Tabelle FORMELN implementiert.

Ergänzungsstruktur für die Gruppenbildung (5.6.1.3.1.6)

Tabelle GRUPPENVERWENDUNGEN
Aus der Ergänzungsstruktur für die Gruppenbildung (5.6.1.3.1.6 im Feinkonzept) wurden die Texte der Berechnungsformeln herausgenommen und in eine eigene zusätzliche Tabelle mit der Bezeichnung ´FORMELN´ verlagert, auf die über 3 Fremdschlüssel von der Tabelle DATENPUNKTE zugegriffen werden kann. Die verbleibenden Spalten der Ergänzungsstruktur für die Gruppenbildung werden in der Datenbankbankimplementierung durch die Tabelle GRUPPENVERWENDUNGEN realisiert. In dieser Tabelle ist eine weitere Spalte (´Verwendung´) hinzugekommen, die angibt, für welchen Anwendungsfall die Gruppenbildung erfolgte (Beispiel: Gruppenbildung für das Wartungswesen).

Wartungsstruktur/Wartungsstamm (5.6.1.3.2)

In der Tabelle DATENPUNKTE enthalten. Die Spalte ´Kriterienblock´ wurde mit der Tabelle ´DKBLOECKE´ implementiert.

Referenzliste auf virtuelle und durch Gruppenbildung definierte Datenpunkte (5.6.1.3.3)

Diese Datenstruktur wurde nicht implementiert, da sie aus redundanter Information besteht.

Referenzliste auf Schemate (5.6.1.3.4)

Die zugehörige Tabelle wurde nicht implementiert. Die Information ist bis auf die Positionsnummer redundant. Die Positionsnummer wurde in die Tabelle der Referenzliste der Symbolfenster übertragen.

Referenzliste der Schaltprogramme (5.6.1.3.5)

Die zugehörige Tabelle wurde wegen redundanter Information nicht implementiert.

Firmenstamm (5.6.2.1)

Tabelle: Firmen
Die Verweise auf die Qualifikationen und die Vertretungen wurden durch eigenständige Tabellen implementiert. Somit können beliebig viele Qualifikationen bzw. Vertretungen angegeben werden.

Modellstamm (5.6.2.2)

Tabelle: BAUTEILE

Programme (5.6.2.3.1)

Tabelle: Programme

Data-Dictionaries (5.6.2.3.2)

Tabelle: DICTIONARIES

Prozeduren (5.6.2.3.3)

Tabelle: PROZEDUREN

Ergebnisse (5.6.2.3.4)

Tabelle: Ergebnisse

Ausgabe und Export von Ergebnissen (5.6.2.3.5)

Tabelle: WEITERVERARBEITUNG

Meldungs-Profile (5.6.2.4)

Tabelle: MELDUNGSPROFILE

Bedienstationen und Ausgabegeräte(5.6.2.5)

Tabelle: BEDIENSTATIONEN
Über die zusätzlich zum Feinkonzept implementierte Tabelle BEDBAUTEILE lassen sich die einzelnen Bauteile einer Bedienstation angeben. Die Spalten Standard-Textmodus, Standard-Grafikmodus, Ref.-Liste weitere Textmodi, ISDN-Profil, Fx-Konfiguration der Datenstruktur Bedienstation wurden nicht implementiert.

Gerätemodi (5.6.2.5.1)

Nicht implementiert.

Referenzliste weiterer Textmodi (5.6.2.5.2)

Nicht implementiert.

Benutzerstamm (5.6.2.6)

Tabelle BENUTZER
Die Spalte ´Ref.-Liste Privilegien-Profile´ ist über die Tabelle BPRIVILEGIEN implementiert.

Privilegien-Profile (5.6.2.6.1)

Referenzliste der Privilegien-Profile (5.6.2.6.2):
Implementiert mit den Tabellen
BPRIVILEGIEN, PRIVILEGIEN, OPRIVILEGIEN, FPRIVILEGIEN und FUNKTIONEN

Wartungspersonal (5.6.2.7)

Tabelle: WARTUNGSPERSONEN

Objektstamm (5.6.2.8)

Tabelle: OBJEKTE
Tabelle: GAKNOTEN
Diese Datenstruktur wurde mit zwei Tabellen implementiert: Tabelle OBJEKTE und Tabelle GAKNOTEN. Dies war notwendig, weil eine 1:1 Zuordnung zwischen Objekt und GA-Knoten nicht immer gegeben ist.

In der Tabelle OBJEKTE sind die Zusatzkennungen, der Stadtbezirk und die Vermessungskoordinaten nicht implementiert. Die Straßennummer ist durch einen Verweis auf das Straßenverzeichnis ersetzt.

In der Tabelle GAKNOTEN sind die Zähler der Meldungen nicht implementiert. Über die Tabelle GAKBAUTEILE lassen sich die Bauteile eines GA-Knotens angeben.

Tabelle: EDS

Über die Tabelle ´EDS´ kann einem Objekt eine Liste von elektronischen Dokumenten zugewiesen werden. Die Tabelle ´EDS´ enthält eine Spalte für den Namen der Datei, in der das elektronische Dokument abgelegt ist.

Verbindungs-Profile (5.6.2.9)

Tabelle: VERBINDUNGSPROFILE

Schemata (5.6.2.10)

Tabelle: SCHEMATA
Die Spalte ´Objekt-Kennung´ wurde nicht implementiert. Die Spalte ´Ref.-Liste der einzublendenden Symbole´ gehört nicht in diese Datenstruktur, sondern in die Datenstruktur Refernzliste der Symbolfenster (5.6.2.10.1) und wurde auch so implementiert.

Referenzliste der Symbolfenster (5.6.2.10.1)

Tabelle: FENSTER
Die Spalte ´Objekt-Kennung´ wurde nicht implementiert. Es wurde zusätzlich die Spalte ´Positionsnummer´ aus der Datenstruktur ´Referenzliste auf Schemata‘ (5.6.1.3.4) in diese Tabelle übernommen.

Referenzliste der einzublendenden Symbole (5.6.2.10.2)

Tabelle: FENSTERSYMBOLE

Symbol-Bibliothek (5.6.2.10.3)

Tabelle: SYMBOLE

Schaltprogramme (5.6.2.11)

Tabelle: SCHALTPROGRAMME
Die Spalte ´Ref.Liste Zeitschaltpunkte´ wurde nicht implementiert. Die Spalte ´Ereignis-Schaltprogramm´ wurde implementiert als ein Verweis auf die Tabelle ´FORMELN´.

Referenzliste der Zeitschaltpunkte (Zeitschaltprogramme) (5.6.2.11.1)

Tabelle: ZEITSCHALTPROGRAMME

Schaltprogramm-Kalender (5.6.2.11.2)

Tabelle: SCHALTKALENDER

Dimensions-Texte (5.6.2.12)

Implementiert über eine Auswahlliste mit der Kennung ´DIMENSION´.

Attribut-Texte (5.6.2.13)

Tabelle: ATTRIBUTTEXTE

Anweisungstexte für Meldungen und Alarme (5.6.2.14)

Tabelle: ANWEISUNGSTEXTE

Wartungsanweisungen (5.6.2.15)

Tabelle: WARTUNGSANWEISUNGEN
Die Spalte ´Kriterienblock´ wurde über die Tabelle WKBLOECKE implementiert.
Die Planzeit ist in Minuten anzugeben.

Qualifikationstexte (5.6.2.16)

Tabelle: QUALIFIKATIONSTEXTE

Kriterientexte (5.6.2.17)

Tabelle: KRITERIENTEXTE

Hilfetexte (5.6.2.18)

Tabelle: HILFETEXTE

Fehlertexte für Bedien- und Funktionsfehler (5.6.2.19)

Tabelle: FEHLERTEXTE

Fehlertexte für GA-Knoten (5.6.2.20)

Tabelle: GAFEHLERTEXTE

Fehlertexte für Systemfehler (5.6.2.21)

Tabelle: SYSTEMFEHLERTEXTE

Auswahllisten-Stamm (5.6.2.22)

Tabelle: AUSWAHLEN

Auswahllisten (5.6.2.22.1)

Tabelle: AUSWAHLPOSITIONEN

Bedien-Makros (5.6.2.23)

Tabelle: MAKROS

Chronologische Daten von den GA-Knoten (5.7.1)

Tabelle: MESSWERTE
Der eindeutige Schlüssel dieser Tabelle ist ein Zähler, der durch eine Oracle-Sequence generiert wird. Die Spalten ´DP_#0´, ´DP_#1´ und ´DP_#2´ (FND Datenpunkttabellen), die jeweils für sich mehrere Werte repräsentieren, wurden in ihre einzelnen Werte aufgetrennt. Da die Tabelle MESSWERTE start anwachsen kann, wurde für sie ein eigener ´Tablespace´ eingerichtet, um den Speicherplatz besser verwalten und beobachten zu können.

Chronologisch abgesetzte Meldungen (5.7.2.1)

Tabelle: MELDUNGEN
Der eindeutige Schlüssel der Tabelle MELDUNGEN ist ein Zähler, der durch eine Oracle-Sequence erzeugt wird.

Referenzliste der Empfänger (5.7.2.1.1)

Tabelle: EMPFAENGER

Statusprotokolle (5.7.2.1.2)

Tabelle: STATUSPROTOKOLLE
Der eindeutige Schlüssel der Tabelle STATUSPROTOKOLLE ist ein Zähler, der durch eine Oracle-Sequence erzeugt wird.

Freie Bemerkungsprotokolle (5.7.2.1.3)

Tabelle: BEMERKUNGSPROTOKOLLE
Der eindeutige Schlüssel der Tabelle BEMERKUNGSPROTOKOLLE ist ein Zähler, der durch eine Oracle-Sequence erzeugt wird.

Wartungsmeldungen (5.7.3)

Tabelle: WARTUNGSMELDUNGEN
Der eindeutige Schlüssel der Tabelle WARTUNGSMELDUNGEN ist ein Zähler, der durch eine Oracle-Sequence erzeugt wird. Die Spalte ´Kriterienblock´ wird durch die Tabelle ´WMKBLOECKE´ implementiert.

Status- und freie Bemerkungsprotokolle für Wartungsmeldungen (5.7.3.1)

Tabelle:WSTATUSPROTOKOLLE
Tabelle: WBEMERKUNGSPROTOKOLLE
Der eindeutige Schlüssel beider Tabellen wird über Zähler, die durch jeweils eine Oracle-Sequence inkrementiert werden, implementiert.

Historie (5.7.4)

Tabelle: HISTORIEN
Der eindeutige Schlüssel dieser Tabelle wird durch einen Zähler implementiert (Oracle-Sequence).

Benutzermailbox (5.7.5)

Tabelle: MAILBOXEN

Formulare zum Testen der Tabellen

Für alle oben beschriebenen Datenbanktabellen wurden Formulare entwickelt, mit deren Hilfe das Datenbankdesign mit den erstellten Tabellen getestet werden kann. Diese Formulare wurden mit Hilfe des Formulargenerators von **Designer/2000** erstellt und mit dem Formulargenerator des **Developers/2000** von Oracle überarbeitet. So entstanden Formulare, die die typischen Bedienmerkmale von Oracle Forms enthalten: es lassen sich die Tabellen beauskunften, neue Zeilen können eingegeben, abgeändert oder gelöscht werden. Die entstandenen Formulare decken aber bei weitem nicht die im Feinkonzept geforderte Funktionalität der Bedienoberfläche ab.

Tabelle der SQL-DDL-Script-Dateien

Dateiname	Realisierte Tabellen
ATT.*	Attributtexte
AUS.*	AUSWAHLEN AUSWAHLPOSITIONEN
BEN.*	BENUTZER
QUA.*	QUALIFIKATIONSTEXTE
FIR.*	FIRMEN FQUALIFIKATIONEN FVERTRETUNGEN
BAUTEIL.*	BAUTEILE
STRASSE.*	STRASSEN
GAK.*	GAKBAUTEILE GAKNOTEN
OBJ.*	OBJEKTE OBJEKTNUTZUNGEN
VER.*	VERBINDUNGEN
VBP.*	EINBRUCHVERBINDUNGEN GEFAHRVERBINDUNGEN LZVERBINDUNGEN VERBINDUNGSPROFILE
TEXTE.*	HILFETEXTE SYSTEMFEHLERTEXTE GAKFEHLERTEXTE FEHLERTEXTE
KRI.*	KRITERIENTEXTE
ANW.*	ANWEISUNGSTEXTE
MPR.*	MELDUNGSPROFILE MKBLOECKE
WAR.*	WARTUNGSANWEISUNGEN WKBLOECKE
FML.*	FORMELN
DAT.*	ACTUALMELDUNGEN DATENPUNKTE DKBLOECKE INFOMELDUNGEN MEKBLOECKE
MW.*	MESSWERTE
SYM.*	SYMBOLE
SCH.*	SCHEMATA
FEN.*	FENSTER FENSTERSYMBOLE
GRU.*	GRUPPENVERWENDUNGEN
BED.*	BEDIENSTATIONEN

WAR.*	WARTUNGSPERSONEN
MDG.*	MELDUNGEN STATUSPROTOKOLLE BEMERKUNGSPROTOKOLLE EMPFAENGER
WTM.*	WARTUNGSMELDUNGEN WMKBLOECKE WSTATUSPROTOKOLLE WBEMERKUNGSPROTOKOLLE
PRO.*	PROGRAMME
DIC.*	DICTIONARIES
PZD.*	PROZEDUREN
ERG.*	ERGEBNISSE
WEI.*	WEITERVERARBEITUNGEN
SPG.*	SCHALTPROGRAMME
ZEI.*	ZEITSCHALTPROGRAMME
KAL.*	SCHALTKALENDER
MBX.*	MAILBOXEN
FUN.*	FUNKTIONEN
PRI.*	FPRIVILEGIEN OPRIVILEGIEN PRIVILEGIEN
BPR.*	BPRIVILEGIEN
MAK.*	MAKROS
FOR.*	FORMELVERWENDUNGEN
EDS.*	EDS

KOMMUNIKATION MIT DER LZH

Dieses Kapitel beschreibt die Kommunikation der Leitzentrale Haustechnik (LZH) mit den Benutzern vor Ort und den Datenerfassungsgeräten in den zu überwachenden Objekten. Dabei werden die folgenden Komponenten betrachtet:

- LZH-Server
  Hier handelt es sich um einen zentralen UNIX Server, auf dem die Datenbank der LZH implementiert ist.
- Bedienstationen
  Dies sind z.Zt. PCs unter Windows NT, die bei den einzenen Sachbearbeitern vor Ort installiert sind, und von denen aus die Dienste der LZH abgerufen werden können. Die Bedienstationen können über das ISDN-Netz der LHM oder der Telekom eine Verbindung mit der LZH aufnehmen. Für diese Art der Verbindung wird das TCP/IP Protokoll verwendet.
- GA-Knoten
  Diese sogenannten Gebäude-Automatisierungs-Knoten dienen zur Erfassung und Steuerung der Meßdaten vor Ort. Es handelt sich um PCs, die über eine ISDN-Schnittstelle den LZH-Server anrufen oder von diesem angerufen werden können. Das Übertragungsprotokoll ist FND1.0 (mit Erweiterungen).

Eine Bedienstation kann mit der Leitzentrale Haustechnik (LZH) über das Netzwerk der LHM kommunizieren. Die Verbindung wird über das ISDN-Netz von der Bedienstation angewählt. Über einen TCP/IP Protokollstack wird der Datentransfer abgewickelt. Wenn auf der Bedienstation Microsoft Windows als Betriebssystem eingesetzt wird, müssen die entsprechenden Treiber zur Realisierung eines TCP/IP Protokollstacks installiert sein. Zur Kommunikation zwischen Programm und TCP/IP dient in einer solchen Konfiguration die Bibliothek WINSOCKET.DLL.

Die Kommunikation zwischen einer Bedienstation und der LZH läßt sich in folgende Teilbereiche untergliedern:

- Zugriffe auf die Datenbank der LZH
  Diese Zugriffe werden durch die vom Datenbankhersteller ORACLE gelieferten Softwarekomponenten realisiert. Dazu ist es notwendig. neben dem TCP/IP Protokollstack auf der Bedienstation die Client-Komponenten von ORACLE SQL*Net Version 2 zu installieren.

- Kommunikation mit den GA-Knoten
  Eine Bedienstation kann nicht direkt mit einem GA-Knoten kommunizieren, da nur der LZH-Server einen Kommunikationsweg mit den GA-Knoten aufbauen kann. Der LZH-Server stellt den Bedienstationen aber eine Reihe von Diensten zur verfügung, mit dessen Hilfe der indirekte Zugriff einer Bedienstation auf die GA-Knoten möglich wird. Diese Dienste sind:

  - Auslesen von Werten, die ein GA-Knoten vor Ort gemessen hat

- Steuerung der GA-Knoten durch eine Bedienstation
- Darstellung von spontanen Meldungen der GA-Knoten

Im folgenden beschreiben wir die Möglichkeiten, die eine Bedienstation hat, um über die LZH Daten mit den GA-Knoten auszutauschen. Vorab erläutern wir zum besseren Verständnis die Prozesse und die Kommunikationsprotokolle, die auf dem LZH-Server implementiert sind, um den Datenverkehr zu verwalten.

PROZESSE DER LZH

Für die LZH wurden zur Verwaltung des Datenverkehrs zwischen Bedienstation und GA-Knoten folgende Prozesse entwickelt:

- gakserver
  Dieser Prozeß übernimmt die folgenden Aufgaben, die weiter unten näher beschrieben werden:
  - Abwicklung des Datenverkehrs zwischen der LZH und den GA-Knoten
  - Speicherung von Meßwerten, die die GA-Knoten senden
  - Bearbeiten von Aufträgen, die von den Bedienstationen erzeugt werden und an die GA-Knoten weitergeleitet werden müssen
  - Aufträge an den tsynch-Prozeß (siehe unten) senden, wenn die von einem GA-Knoten empfangene Uhrzeit von der Systemzeit der UNIX-Systemzeit des LZH-Servers abweicht
  - Steuern des Ablauf des gakserver-Prozesses über eine Operator-Schnittstelle

- userver
  Der userver-Prozeß wickelt die folgenden Tätigkeiten ab:
  - Kommunikation mit den Bedienstation über eine UNIX socket Schnittstelle
  - Weitergabe von Benutzeraufträgen an den gakserver-Prozeß
  - empfangen von Daten des gakserver-Prozesses und weiterleiten an die Bedienstationen
  - Steuerung des Ablaufs des userver-Prozesses über eine Operator-Schnittstelle

- tsynch
  Der tsynch-Prozeß dient zur Korrektur der Uhrzeit des LZH-Servers.

- gakimport
  Dieses Programm dient zum Importieren von Datenpunktinformationen eines GA-Knotens in die Datenbanktabelle DATENPUNKTE von einer Diskette.

GAKSERVER

In diesem Kapitel werden die Aufgaben des gakserver-Prozesses beschrieben. Dies sind:

- Abwicklung des Datenverkehrs zwischen der LZH und den GA-Knoten
- Speicherung von Meßwerten, die die GA-Knoten senden
- Bearbeiten von Aufträgen, die von den Bedienstationen erzeugt werden und an die GA-Knoten weitergeleitet werden müssen
- Aufträge an den tsynch-Prozeß (siehe unten) senden, wenn die von einem GA-Knoten empfangene Uhrzeit von der Systemzeit der UNIX-Systemzeit des LZH-Servers abweicht
- Steuern des Ablauf des gakserver-Prozesses über eine Operator-Schnittstelle

Datenverkehr mit den GA-Knoten

Der Datenverkehr benutzt das FND 1.0 Protokoll [3] mit den Ergänzungen DIN V32735 [4].

Physikalisch wird der Datentransport durch folgende Komponenten realisiert:

- ISDN-Schnittstellenkarte des GA-Knotens
  Jeder GA-Knoten ist mit einer ISDN-Schnittstellenkarte ausgestattet, die verbunden ist mit dem

- Telephonnetz der LHM oder der Telekom.

- ISDN/X.25-Router
  Will ein GA-Knoten eine Verbindung mit der LZH herstellen, so muß er zunächst die Nummer eines speziellen ISDN/X.25 Routers anwählen. Dieser Router ist in der Telefonzentrale der LHM am Roßmarkt installiert. Er wandelt ankommende Datenpakete vom ISDN-Protokoll in X.25-Pakete um und schickt diese über seine X.25-Schnittstelle an die

- X.25-Schnittstellenkarte des LZH-Servers
  Der LZH-Server empfängt die X.25-Pakete über seine X.25-Schnittstelle und bearbeitet sie entsprechend ihrem Inhalt.

Der umgekehrte Datenfluß - vom LZH-Server zu einem GA-Knoten - verläuft entsprechend: Der LZH-Server schickt X.25-Datenpakete über seine X.25-Schnittstellenkarte an den ISDN/X.25-Router. Dieser wandelt die X.25-Pakete um in ISDN-Pakete und schickt sie über das Telephonnetz an die GA-Knoten.

Zur Abwicklung des Datenverkehrs wurde eine Bibliothek von Funktionen entwickelt. Diese Bibliothek heißt ´libx25.a´ und benutzt ihrerseits die vom Systemhersteller SUN beigestellte Bibliothek ´/opt/SUNWconn/x25/lib/liblsx25.a´. ´liblsx25.a´ enthält Funktionen, die die auf dem UNIX Streams-Konzept aufbauenden X.25 Netzwerk-Dienste implementieren.

Zur internen Verwaltung aller GA-Knoten, mit denen der gakserver-Prozeß zur Zeit eine Verbindung unterhält, dient die Datenstruktur GAK (siehe Datei gakserver.h). Für jeden GA-Knoten legt der gakserver-Prozeß eine solche Struktur an und fügt sie in eine Liste aller GA-Knoten ein. Wenn die Verbindung zum GA-Knoten geschlossen wird, wird der Eintrag aus der Liste entfernt und der verwendete Speicherbereich wieder frei gegeben.

Speicherung der Meßwerte

Die von den GA-Knoten ermittelten Meßwerte, die dauerhaft in der LZH abgespeichert werden sollen, werden zunächst von den GA_Knoten vor Ort zwischengespeichert. Von Zeit zu Zeit übermittelt der GA-Knoten dann seinen Pufferinhalt an die LZH. Die Anrufe werden vom gakserver-Prozeß entgegengenommen und bearbeitet. Dabei laufen die folgenden Teilschritte ab:

- Es wird geprüft, ob der Anrufer der LZH bekannt ist. Zur Identifizierung dient die Spalte CALLINGPARTYNUMBER der Datenbanktabelle GAKNOTEN. Falls der Anrufer nicht registriert ist, wird die Verbindung nicht aufgebaut und der Anruf abgewiesen.
- Es wird geprüft, ob der GA-Knoten in der Datenbanktabelle GAKNOTEN als gestört markiert ist. Ist das der Fall, dann wird der Anruf abgewiesen und die Verbindung nicht aufgebaut.
- Der GA-Knoten übermittelt in seinem X.25 Call Indication Datagramm im User Data Field die aktuelle Uhrzeit. Der gakserver-Prozeß prüft, ob diese von der aktuellen Systemzeit des UNIX-Systems abweicht und schickt dem Prozeß tsynch gegebenenfalls einen Auftrag zur Angleichung der Systemzeit.
- Der gakserver-Prozeß legt eine neue Datenstruktur GAK für den anrufenden GAK-Knoten an und stellt die X.25-Verbindung (SVC) mit dem GA-Knoten her.

Wenn der GA-Knoten vom gakserver-Prozeß die Bestätigung erhalten hat, daß sein Anruf akzeptiert wurde, beginnt er mit der Übertragung seiner Meßwerte. Sie werden in Datagrammen entsprechend FND1.0 oder als Karteipunkte (Erweiterung der FND-Spezifikation) übertragen.

Der gakserver-Prozeß wertet die Datagramme in folgenden Teilschritten aus:

- Prüfung auf korrekte Syntax der Datagramme. Falls ein Datagramm erkannt wird, das nicht FND-konform ist, wird:
  - ein Eintrag in das Errorlog geschrieben
  - die gemäß FND erforderliche Bestätigung (Ack-APDU) an den sendenden GA-Knoten nicht zurückgeschickt
  - der GA-Knoten in der Datenbanktabelle GAKNOTEN als gestört markiert

- Datagramme, die im Blockformat empfangen wurden, werden in einzelne Datagramme gemäß FND1.0 zerlegt und wie diese dann weiter bearbeitet.
- Der Meßwert wird in die Datenbank eingetragen. Dabei werden die folgenden Datenbanktabellen verändert:
  - MESSWERTE
    In diese Tabelle wird der neue Meßwert eingefügt.
  - GAKNOTEN
    Die Spalten LETZTE_UEBETRAGUNG und GAGESTARTET werden aktualisiert.
  - DATENPUNKTE
    Der aktuelle Meßwert wird in die entsprechenden Spalten eingetragen, die Spalte DATUMPARAMETER wird aktualisiert. Wenn ein zugehöriger Datenpunkt in der Tabelle DATENPUNKTE nicht gefunden wird, dann wird ein neuer Datenpunkt für das fiktive Objekt ´ZZZZ´ angelegt.

  Die Änderungen an den Tabellen GAKNOTEN und DATENPUNKTE sind mit Hilfe eines Datenbanktriggers implementiert. Der Trigger hat den Namen ´OESER.MWINSERT´.

  Es kann vorkommen, daß die Datagramme der GA-Knoten unzulässige Werte für gewisse Tabellenspalten enthalten. Da die Datenbanktabellen so angelegt sind, daß unzulässige Werte nicht gespeichert werden können, ergibt sich beim Versuch, diese Werte abzuspeichern, ein vom Datenbanksystem erkannter Fehler. In diesem Fall kann der Meßwert nicht abgespeichert werden. Es wird ein entsprechender Eintrag im Errorlog erzeugt, und der GA-Knoten wird auf ´gestört´ gesetzt.

- Wenn die Abspeicherung des Meßwertes in der Datenbank erfolgreich war, wandelt der gakserver-Prozeß das FND-Datagramm in eine interne Dartellung um und schickt es an den userver-Prozeß, damit dieser es eventuell an eine darauf wartende Bedienstation weiterleiten kann.

- Die Behandlung des vom GA-Knoten gesendeten Datagramms beendet der gakserver-Prozeß durch das Senden einer positiven Bestätigung (Ack-APDU) an den GA-Knoten. Dieser kann daraufhin das nächste Datagramm senden.

Bearbeitung von Benutzeraufträgen

Der gakserver-Prozeß stellt Funktionen zur Verfügung, die die notwendige Kommunikation zwischen einer Bedienstation und einem GA-Knoten unterstützen. Da der gakserver-Prozeß keine direkte Verbindung zu einer Bedienstation unterhält, verläuft die Kommunikation immer über den userver-Prozeß. Dabei wird ein internes Protokoll verwendet, das weiter unten beschrieben wird.

Die folgenden Funktionen werden vom gakserver-Prozeß unterstützt:

- Kommunikation mit einem GA-Knoten gemäß FND1.0:
  Lesen und Ändern von Parametern eines Datenpunktes innerhalb des GA-Knotens vor Ort
- Definieren einer Istwertabfrage
  Diese Funktion ermittelt aus dem Auftrag des Benutzers die beteiligten GA-Knoten, legt für jeden dieser GA-Knoten eine Struktur vom Typ GAK (siehe ´gakserver.h´) an und merkt sich für jeden GA-Knoten in einem Ausgabepuffer die Datagramme (gemäß Karteipunktdefinition), die beim Starten der Istwert-Abfrage an den jeweiligen GA-Knoten zu senden sind.
- Starten einer Istwertabfrage
  Schickt die mit der Funktion ´Definieren einer Istwertabfrage´ erzeugten Ausgabepuffer an die beteiligten GA-Knoten.
- Stoppen einer Istwertabfrage
  Schickt den beteiligten GA-Knoten den Befehl zum Stoppen der Istwert-Abfrage und baut die internen Datenstrukturen zur Verwaltung der Istwertabfragen wieder ab.

Steuerung des Ablaufs

Der Prozeß ´gakserver´ ist als UNIX-Server Prozeß implementiert: Nachdem er gestartet wurde, läuft er im Hintergrund ab und wird auch dann nicht abgebrochen, wenn der Benutzer, der den Prozeß gestartet hat, sich ausloggt. Der Prozeß wird durch die ausführbare Datei ´/udd/h6ga/exe/gakserver.exe´ implementiert und sollte über das Shell-Script ´/udd/h6ga/exe/gakserver´ gestartet werden. Dieses Script sorgt dafür, daß der Prozeß nicht mehrfach gestartet werden kann. Das Shell-Script ´gakserver´ kann mit den folgenden optionalen Parametern aufgerufen werden:

```
$ gakserver
  [-say]
  [-trace]
  [-log    LogFileName]
  [-error  ErrorLogFileName]
```

Die Bedeutung der Parameter ist:

- `-say:`
  Es wird ein ausführliches Protokoll erstellt. Das Protokoll wird in die Datei ´gakserver.log´ geschrieben, wenn nicht mit der Option `-log` eine andere Datei als Logdatei angegeben wird.
- `-trace:`
  Es wird das Standard- und das Fehlerprotokoll nicht nur in die dafür vorgesehenen Dateien abgespeichert, sondern zusätzlich auch auf dem Bildschirm ausgegeben, an dem der Benutzer den Prozeß gestartet hat.
- `-log LogFileName:`
  Mit dieser Option kann die Ausgabe des Standardprotokolls auf eine Datei mit einem frei wählbaren Namen umgeleitet werden. Die Voreinstellung für den Namen der Log-Datei ist ´gakserver.log´.
- `-error ErrorLogFileName:`
  Mit dieser Option kann der Name der Datei angegeben werden, in die Fehlermel-

dungen geschrieben werden sollen. Die Voreinstellung heißt ´gakserver_error.log´.

Während des Ablaufs des gakserver-Prozesses kann dieser mit Hilfe des Programms ´cxn´ wie folgt gesteuert werden:

- `cxn gakserver stop`
  Mit diesem Befehl läßt sich der gakserver-Prozeß terminieren.
- `cxn gakserver`
  Mit diesem Befehl läßt sich feststellen, ob der gakserver-Prozeß noch aktiv ist. Arbeitet der gakserver-Prozeß noch einwandfrei, dann erhält man als Antwort ´Prozeß ist aktiv´.
- `cxn gakserver status`
  Gibt Auskunft über die aktuellen Protokoll- und Fehlerprotokolldateien und ob der Trace-Modus eingeschaltet ist.
- `cxn gakserver say`
  Gibt Auskunft, ob ein Standard- oder das ausführliche Protokoll eingeschaltet ist.
- `cxn gakserver say on`
  `cxn gakserver say off`
  Schaltet das ausführliche Protokoll ein oder aus.
- `cxn gakserver trace`
  Gibt Auskunft über den aktuell eingestellten Trace-Modus
- `cxn gakserver trace on`
  `cxn gakserver trave off`
  Schaltet den Trace-Modus ein oder aus. Wenn der Trace-Modus eingeschaltet ist, wird die Ausgabe, die auf die Standard- und die Fehlerprotokolldatei geschrieben wird, zusätzlich auch am Bildschirm ausgegeben, an dem das ´cxn´-Kommando eingegeben wurde.
- `cxn gakserver gaknoten`
  Mit diesem Befehl läßt sich eine Liste der GA-Knoten ausgeben, zu denen der gakserver-Prozeß eine aktive X.25-Verbindung unterhält, oder für die er Daten für noch nicht abgeschlossene Istwert-Abfragen verwalten muß.
- `cxn gakserver log`
  Gibt Auskunft, in welche Datei das Standard-Protokoll ausgegeben wird.
- `cxn gakserver log NewLogFileName`
  Mit diesem Befehl kann die Ausgabe des Standard-Protokolls auf eine andere Datei umgelenkt werden. Dabei wird die aktuelle Standard-Protokoll-Datei geschlossen und das Protokoll in der Datei mit dem neuen Namen fortgesetzt.
- `cxn gakserver error`
  Gibt Auskunft, in welche Datei das Fehler-Protokoll ausgegeben wird.
- `cxn gakserver error NewErrorLogFileName`
  Mit diesem Befehl kann die Ausgabe des Fehler-Protokolls auf eine andere Datei umgelenkt werden. Dabei wird die aktuelle Fehler-Protokoll-Datei geschlossen und das Fehler-Protokoll in der Datei mit dem neuen Namen fortgesetzt.

TSYNCH

Der Prozeß ´tsynch.exe´ dient zum Nachstellen der System-Zeit des LZH-Servers. Da dies nur von einem Benutzer mit ´root´-Privilegien durchgeführt werden kann, wurde dafür ein eigener Prozeß entwickelt. Er wird beim Booten des UNIX-Systems gestartet und beim Abschalten wieder gestoppt. Die entsprechenden Shell-Scripts findet man in:

- `/etc/init.d/tsynch.sh`
- `/etc/rc3.d/S02tsync`
- `/etc/rc0.d/K46tsync`

Die korrekte Uhrzeit wird im User-Data-Field eines X.25-Datagramms immer dann übermittelt, wenn ein GA-Knoten beim LZH-Server anruft. Der Anruf wird vom gakserver-Prozeß bearbeitet. Wenn der gakserver-Prozeß eine Abweichung von mehr als 5 Sekunden zwischen der vom GA-Knoten gesendeten und der aktuellen Systemzeit des UNIX-Systems feststellt, sendet der gakserver-Prozeß eine entsprechende Meldung an den tsynch-Prozeß. Dieser stellt dann die Systemzeit entsprechend nach. Die Kommunikation innerhalb des LZH-Servers zwischen gakserver- und tsynch-Prozeß verläuft über ein UNIX Named Pipe (FIFO).

USERVER

Der Prozeß userver, der auf dem LZH-Server vom Systemadministrator gestartet wird, übernimmt alle Aufgaben, die die Kommunikation zwischen den Bedienstationen und der LZH betreffen.

Die Aufgaben des userver-Prozesses sind:

- bereitstellen der Kommunikationswege zu den Bedienstationen
- kommunizieren mit dem gakserver-Prozeß
- bearbeiten von Kommandos der Bedienstationen
- den Bedienstationen spontane Meldungen zuschicken
- bearbeiten der Kommandos zur Ablaufsteuerung

Kommunikation mit den Bedienstationen

Der Datenaustausch zwischen einer Bedienstation und LZH-Server kann über ein lokales Netzwerk (LAN) oder über das ISDN-Telephonnetz der LHM oder der Telekomm erfolgen. Zu diesem Zweck ist in jede Bedienstation entweder eine Netzwerkkarte (Ethernet) oder ein ISDN-Adapter eingebaut. Die Bedienstationen enthalten Softwarekomponenten, die den Aufbau eines TCP/IP Protokollstacks ermöglichen.

Wenn die Bedienstation nicht mit dem lokalen Netzwerk des LZH-Servers verkabelt ist, sondern über ISDN den LZH-Server erreichen muß, dann wählt die Bedienstation beim Verbindungsaufbau zunächst einen Router an. Dieser Router ist auf der einen Seite an das ISDN-Netz der LHM und auf der anderen Seite mit dem lokalen Netzwerk des LZH-Servers verbunden. Er setzt eingehende ISDN-Datenpakete in TCP/IP um und schickt sie an den LZH-Rechner weiter. Der umgekehrte Datenweg - LZH schickt Daten zu einer Bedienstation, die sich über ISDN eingewählt hat - verläuft entspechend: in diesem Falle schickt der LZH-Server die Daten über seine lokale Ethernet-Schnittstelle an den Router, der für den weiteren Transport an die eingewählte Bedienstation sorgt.

Als Übertragungsverfahren zwischen einer Bedienstation und dem userver-Prozeß des LZH-Servers dient das Internet IP Protokoll. Der Datenaustausch wird über connectionsless UNIX Sockets (UDP) abgewickelt. Nach dem Starten des userver-Prozesses wartet dieser auf der Port-Nummer 4402 auf eingehende Aufträge. Für den Aufbau des Sockets auf der Seite der Bedienstation sollte die Vergabe der lokalen Port-Nummer dem TCP/IP Protocol-Stack überlassen bleiben (nicht die Funktion bind() verwenden).

Jedes Programm einer Bedienstation, das die Leistungen des LZH-Servers in Anspruch nehmen will, muß sich zunächst beim userver-Prozeß anmelden. Für jedes so angemeldete Programm legt der userver-Prozeß eine Datenstruktur vom Typ RMTUSER (siehe ´userver.h´) an.

Bei der Anmeldung muß das Programm eine sogenannte SessionId mitliefern. Sie ermöglicht das mehrfache Anmelden über einen gemeinsamen Socket. Dies erlaubt den Einsatz von Server-Prozessen auf den Bedienstationen, die den Zugang zum LZH-Server zusammenfassen (Siehe GAKDDE-Server).

Bei einer erfolgreichen Anmeldung teilt der userver-Prozeß dem neu angemeldeten Programm eine eindeutige sogonannte ´lz_subid´ zu. Die lz_subid wird für die Zuordnung der Aufträge der Benutzer innerhalb der GA-Knoten benötigt.

Der userver-Prozeß überwacht von Zeit zu Zeit, ob die angemeldeten Benutzerprogramme noch aktiv sind. Dadurch wird vermieden, daß Benutzer, die sich nicht ordnungsgemäß abgemeldet haben, weiterhin in den internen Tabellen des userver-Prozesses geführt werden. Zur Überwachung sendet der userver-Prozeß an das zu überwachenden Benutzerprogramm ein kurzes Datagramm, das von diesem unverzüglich beantwortet werden muß. Bleibt die Antwort aus, dann streicht der userver-Prozeß das Anwenderprogramm aus seiner Liste.

Jedes Datagramm, das eine Bedienstation dem userver-Prozeß schickt, wird von diesem durch ein positives oder negatives Bestätigungsdatagramm beantwortet.

Kommunikation mit dem gakserver-Prozeß

Die Ausführung der Kommandos der Bedienstationen erfordert in vielen Fällen die Steuerung der GA-Knoten. Dies übernimmt im LZH-Server der gakserver-Prozeß. Der userver-Prozeß muß deshalb die Kommandos der Bedienstationen in entsprechende Kommandos umsetzen, die der gakserver-Prozeß versteht, und an den gakserver-Prozeß weiterschicken.

Auf der anderen Seite empfängt der gakserver-Prozeß spontane Meßwerte der GA-Knoten oder Meßwerte für Istwert-Abfragen, die an die Bedienstationen weitergeleitet werden müssen. Dies kann im LZH-Server nur der userver-Prozeß bewerkstelligen. Deshalb schickt der gakserver-Prozeß diese Daten zunächst an den userver-Prozeß, der sie dann an die entsprechenden Bedienstationen verteilt.

Die Kommunikation zwischen userver- und gakserver-Prozeß innerhalb des LZH-Servers verläuft über UNIX Named Pipes (FIFOs). Das dazu benutzte Protokoll wird im Kapitel ´interne Protokolle´ beschrieben.

Kommandos der Bedienstationen

Der userver-Prozeß versteht die folgenden Aufträge der Bedienstationen:

- Anmelden (siehe oben)
- Abmelden (siehe oben)
- Start des Anzeigemodus:
  Hiermit kann die Bedienstation angeben, daß sie an einlaufenden Meßwerten interessiert ist, die dem userver-Prozeß vom gakserver-Prozeß zugesendet werden. In Zukunft sendet der userver-Prozeß diese Werte an die Bedienstation.

- Stop des Anzeigemodus:
  Die Ausgabe von weiteren Meßwerten des Anzeigemodus wird abgeschaltet.
- Ausgabe der aktuellen Uhrzeit
  Mit Hilfe dieser Funktion kann die Bedienstation die Systemzeit des LZH-Servers abfragen.
- Definition von Istwert-Abfragen:
  Hiermit kann die Bedienstation Istwert-Abfragen definieren. Der Auftrag wird an den gakserver-Prozeß weitergeleitet, da dieser für die Bearbeitung von Istwert-Abfragen zuständig ist.
- Istwert-Abfrage starten:
  Startet die zuvor definierte Istwert-Abfrage an den dafür zuständigen GA-Knoten. Der Auftrag wird an den gakserver-Prozeß weitergeleitet. Der userver-Prozeß leitet einlaufende Meßwerte, die zu dieser Istwert-Abfrage gehören, in Zukunft an die Bedienstation weiter.
- Istwert-Abfrage stoppen:
  Die laufende Istwert-Abfrage wird gestoppt. Auch dieses Kommando wird an den gakserver-Prozeß weiter geschickt.
- FND1.0 Kommando:
  Dieses Kommando dient zum Auslesen und Ändern von Datenpunkten eines GA-Knotens vor Ort. Das Kommando wird an den gakserver-Prozeß weiter geschickt. Dieser sendet es an den betroffenen GA-Knoten. Der GA-Knoten sendet ein Antwort-Datagramm zurück, das über den Weg gakserver -> userver wieder an die Bedienstation ausgegeben wird.

Spontane Meldungen

Der userver-Prozeß erhält vom gakserver-Prozeß laufend spontane Meldungen, die er weiter bearbeiten muß. Folgende Meldungen müssen bearbeitet werden:

- Spontane Meßwerte:
  Diese Werte werden von den GA-Knoten erzeugt, wenn sie Werte in der Datenbank permanent abspeichern wollen. Die Meßwerte sind keinem Auftrag einer Bedienstation zugeordnet (lz_subid = 0). Sie müssen aber an die Programme solcher Bedienstationen weitergeleitet werden, die den Anzeigemodus aktiviert haben.
- Meßwerte aus Istwerte-Abfragen:
  Diese Werte werden von GA-Knoten erzeugt, für die eine Bedienstation eine Istwert-Abfrage gestartet hat. Solche Werte sind mit einer lz_subid ungleich 0 markiert und werden vom userver-Prozeß an das zugehörige Programm der Bedienstation weitergeleitet.

Ablaufsteuerung

Der userver-Prozeß ist als UNIX-Server-Prozeß implementiert. Er läuft im Hintergrund ab und wird auch dann nicht beendet, wenn der Benutzer, der ihn gestartet hat, sich ausloggt. Der Prozeß wird durch das ausführbare Programm ´/udd/h6ga/exe/userver.exe´ implementiert. Zum starten sollte das Shell-Script

´/udd/h6ga/exe/userver´ verwendet werden. Es verhindert, daß der userver-Prozeß mehrfach gestartet werden kann. Das Shell-Script ´userver´ kann mit den folgenden optionalen Parametern aufgerufen werden:

```
$ userver
  [-say]
  [-trace]
  [-log LogFileName]
  [-error ErrorLogFileName]
```

Die Parameter haben die folgende Bedeutung:

- `-say:`
  Es wird ein ausführliches Protokoll erstellt. Das Protokoll wird in die Datei ´userver.log´ geschrieben, wenn nicht mit der Option `–log` eine andere Datei als Logdatei angegeben wird.
- `-trace:`
  Es wird das Standard- und das Fehlerprotokoll nicht nur in die dafür vorgesehenen Dateien abgespeichert, sondern zusätzlich auch auf dem Bildschirm ausgegeben, an dem der Benutzer den Prozeß gestartet hat.
- `-log LogFileName:`
  Mit dieser Option kann die Ausgabe des Standardprotokolls auf eine Datei mit einem frei wählbaren Namen umgeleitet werden. Die Voreinstellung für den Namen der Log-Datei ist ´userver.log´.
- `-error ErrorLogFileName:`
  Mit dieser Option kann der Name der Datei angegeben werden, in die Fehlermeldungen geschrieben werden sollen. Die Voreinstellung heißt ´userver_error.log´

Während des Ablaufs des userver-Prozesses kann dieser mit Hilfe des Programms´cxn´ wie folgt gesteuert werden:

- `cxn userver stop`
  Mit diesem Befehl läßt sich der userver-Prozeß terminieren.
- `cxn userver`
  Mit diesem Befehl läßt sich feststellen, ob der userver-Prozeß noch aktiv ist. Arbeitet der userver-Prozeß noch einwandfrei, dann erhält man als Antwort ´Prozeß ist aktiv´.
- `cxn userver status`
  Gibt Auskunft über die aktuellen Protokoll- und Fehlerprotokolldateien, den Trace-Modus und die angemeldeten Benutzerprogramme.
- `cxn userver say`
  Gibt Auskunft, ob ein Standard- oder das ausführliche Protokoll eingeschaltet ist.
- `cxn userver say on`
  `cxn userver say off`
  Schaltet das ausführliche Protokoll ein oder aus.
- `cxn userver trace`
  Gibt Auskunft über den aktuell eingestellten Trace-Modus
- `cxn userver trace on`
  `cxn userver trace off`
  Schaltet den Trace-Modus ein oder aus. Wenn der Trace-Modus eingeschaltet ist, wird die Ausgabe, die auf die Standard- und die Fehlerprotokolldatei geschrieben

wird, zusätzlich auch am Bildschirm ausgegeben, an dem das ´cxn´-Kommando eingegeben wurde.

- `cxn userver log`
  Gibt Auskunft, in welche Datei das Standard-Protokoll ausgegeben wird.
- `cxn userver log NewLogFileName`
  Mit diesem Befehl kann die Ausgabe des Standard-Protokolls auf eine andere Datei umgelenkt werden. Dabei wird die aktuelle Standard-Protokoll-Datei geschlossen und das Protokoll in der Datei mit dem neuen Namen fortgesetzt.
- `cxn userver error`
  Gibt Auskunft, in welche Datei das Fehler-Protokoll ausgegeben wird.
- `cxn userver error NewErrorLogFileName`
  Mit diesem Befehl kann die Ausgabe des Fehler-Protokolls auf eine andere Datei umgelenkt werden. Dabei wird die aktuelle Fehler-Protokoll-Datei geschlossen und das Fehler-Protokoll in der Datei mit dem neuen Namen fortgesetzt.

INTERNES PROTOKOLL

Dieses Kapitel beschreibt die Formate des internen Protokolls für die Übertragung von Informationen,die zwischen einer Bedienstation und dem userver-Prozeß der LZH und zwischen dem userver- und dem gakserver-Prozeß innerhalb des LZH-Servers ausgetauscht werden. Zunächst sollen einige allgemeine Prinzipien der Kommunikation beschrieben werden:

Die Kommunikation zwischen Bedienstation und userver-Prozeß des LZH-Servers erfolgt über UNIX sockets, wobei das UDP-Protokoll benutzt wird (connectionless socket). Die Bedienstation schickt Meldungen an den userver-Prozess der LZH unter Angabe der Portnummer 4402 (symbolisch IPPORT_GAK) und der IP-Adresse 10.2.26.31 (symbolisch Server20).

Alle Datagramme des Protokolls enthalten nur Zeichen des ISO-8859 Zeichensatzes (Obermenge von ASCII). Probleme beim Verschicken über das Netzwerk mit unterschiedlichen Rechnerarchitekturen sollten deshalb nicht auftreten. In den Datagrammen sind Datenfelder mit folgenden Datentypen erlaubt:

Typ	**Format**	**Verwendung**
SID	%4.4d	Session-Id im Bereich 1 .. 99999
TYP	%5.5d	Datagrammtyp 0 .. 9999
N*1	%3.3d	Ganze Zahlen im Bereich 0 .. 255
I*2	%6.6d	Ganze Zahlen im Bereich -32768 .. +32767
R*4	%14.6e	Gleitkommazahl, z.B.: -1.234567e-002
D*8	TTMMJJJJ	Datum (Tag, Monat, Jahr)
Z*6	HHMMSS	Zeit (Stunde, Minute, Sekunde)
T*x		für Texte variabler Länge mit folgendem Aufbau: %3.3d Länge gefolgt von soviel Textzeichen, wie die voranstehende Länge angibt

NULL-Werte im Sinne von 'nicht vorhanden' werden durch den Buchstaben 'N' angegeben. Für Werte vom Typ 'T*x' darf dann nur 'N' als Längenangabe vorhanden sein. Will man dagegen einen Text der Länge 0 definieren, so ist das Längenfeld ' 0' zu besetzen. Einige Beispiele:

```
Typ N*1 '123' Das Datenfeld hat den Wert 123.
        '  0' Das Datenfeld hat den Wert 0.
        'N'   Der Wert des Datenfeldes ist undefiniert.

Typ T*x '   4Hugo' Das Datenfeld hat den Wert 'Hugo'.
        '   0'     Das Datenfeld ist ein String der Länge 0.
        'N'        Der Wert des Datenfeldes ist undefiniert.
```

Die in den Beispielen verwendeten Apostrophzeichen (') dienen hier nur zur Verdeutlichung; sie sind in den Meldungen NICHT enthalten.

Session-Id

Jede Meldung beginnt mit einer Kennzahl für die Sitzung der Bedienstation (Session-Id). Dies ist notwendig, damit von einer Bedienstation aus mehrere unabhängige "Dialoge" mit dem userver-Prozeß der LZH geführt werden können. Die Session-ID ist eine maximal 5-stellige Zahl. Sie ist im Format %5.5d anzugeben. Die Software der Bedienstation muß für jede eigenständige Sitzung eine eindeutige Nummer erzeugen.

Meldungstyp

Auf die Session-Id folgt eine 4-stellige Zahl im Format %4.4d: dies ist der Typ des Datagramms. Er legt fest, welche funktionelle Bedeutung das Datengramm hat und welche Datenfelder es enthalten muß. Wir beschreiben die verschiedenen Datagramme detailliert in einem eigenen Kapitel weiter unten und geben zunächst nur eine Übersicht über die möglichen Meldungstypen:

Typ	Symbolische Bezeichnung	Verwendung
0	GAMDG_UNDEFINIERT	nur der Vollständigkeit halber
1	GAMDG_ANMELDEN	Anmeldung durch Bedienstation
2	GAMDG_ABMELDEN	Abmeldung durch Bedienstation
3	GAMDG_FND10_CMD	FND 1.0 Kommando von Bedienstation
4	GAMDG_FND10_RSP	FND 1.0 Response eines GA-Knotens
5	GAMDG_FND10_USM	FND 1.0 spontaner Meßwert vom GA-Knoten
6	GAMDG_REJECT	FND 1.0 Reject (Ablauf-Fehler)
7	GAMDG_ERROR	FND 1.0 Error (Syntax-Fehler)
8	GAMDG_ACK	Acknowledge
9	GAMDG_NAK	No Acknowledge
10	GAMDG_TIME_REQUEST	Abfrage der Systemzeit der LZH durch Bedienstation (nur für Testzwecke)
11	GAMDG_TIME	Antwort der LZH auf GAMDG_TIME_REQUEST
12	GAMDG_ARE_YOU_THERE	Abfrage der LZH, ob Programm an der Bedienstation noch aktiv
13	GAMDG_I_AM_HERE	Antwort der Bedienstation auf GAMDG_ARE_YOU_THERE
14	GAMDG_START_ANZMOD	Bedienstation startet den Anzeigemodus
15	GAMDG_STOP_ANZMOD	Bedienstation beendet den Anzeigemodus
16	GAMDG_IW_DEFINE	Istwertabfrage: Datenpunkt(e) hinzufügen
17	GAMDG_IW_START	Istwertabfrage: starten
18	GAMDG_IW_STOP	Istwertabfrage: stoppen

Quittung

Ein Datagramm, das eine Bedienstation an den userver-Prozeß der LZH schickt, wird vom userver-Prozeß analysiert und bearbeitet. Wenn die Bearbeitung erfolgreich abgeschlossen wurde, schickt der userver-Prozeß ein Quittungsdatagramm des Typs GAMDG_ACK (= Datagramm wurde akzeptiert), bzw. GAMDG_NAK (= Datagramm wurde nicht akzeptiert) an die Bedienstation zurück.

Eine Bedienstation muß stets auf ein Quittungsdatagramm warten, bevor es ein weiteres Datagramm sendet. Dieses Verfahren stellt sicher, daß der Eingangspuffer des userver-Prozesses nicht überläuft.

Datagramme, die der userver-Prozeß an die Bedienstationen sendet, erfordern keine Quittung nach erfolgreichem Empfang durch die Bedienstationen.

DATAGRAMME

In diesem Kapitel beschreiben wir die Datagramme, die zwischen dem userver-Prozeß des LZH-Rechners und den Bedienstationen ausgetauscht werden. Die Datagramme werden auch für den Datenaustausch zwischen dem gakserver- und dem userver-Prozeß innerhalb des LZH-Servers verwendet.

- GAMDG_UNDEFINIERT (0)
  Verwendung:
  Dieser Meldungstyp ist nur der Vollständigkeit halber aufgeführt. Eine Meldung mit diesem Typ ist unzulässig.

  Format:

SID	Session-Id
TYP	Datagrammtyp

- GAMDG_ANMELDEN (1)
  Verwendung:
  Anmeldung eines Benutzers an einer Bedienstation. Eine Bedienstation muß den Dialog mit der LZH immer mit diesem Datagramm beginnen. Der Dialog bleibt solange bestehen, bis die Bedienstation das Datagramm GAMDG_ABMELDEN schickt oder bis der userver-Prozeß der LZH gemerkt hat, daß die Bedienstation nicht mehr antwortet. Eine Bedienstation kann gleichzeitig mehrere Dialoge mit der LZH führen. Jeder dieser Dialoge muß mit einem eigenen Datagramm vom Typ GAMDG_ANMELDEN eingeleitet werden. Zur Unterscheidung zwischen mehreren Dialogen der gleichen Bedienstation dient die Session-Id, die von der Bedienstation beliebig vergeben werden kann.

  Format:

SID	Session-Id
TYP	Datagrammtyp
T*x	Benutzername

- GAMDG_ABMELDEN (2)
  Verwendung:
  siehe unter GAMDG_ANMELDEN

  Format:

SID	Session-Id
TYP	Datagrammtyp
T*x	Benutzername

- GAMDG_FND10_CMD (3)
  Verwendung:
  FND 1.0 Kommando einer Bedienstation

Format:

SID	Session-Id
TYP	Datagrammtyp
N*1	lz_subid (dieser Wert wird vom userver-Prozeß nicht beachtet)
T*x	Objektkennung
T*x	dp_id (Datenpunktadresse)
N*1	fct_id/tab_id
N*1	prd/rpr/emq
N*1	mis/msk
N*1	atb_id
R*4	actual
R*4	nominal
N*1	dimension
N*1	update_control
N*1	fixed
R*4	last_save
R*4	alarm_low
R*4	warning_low
R*4	warning_high
R*4	alarm_high

- GAMDG_FND10_RSP (4)
  Verwendung:
  FND 1.0 Response eines GA-Knotens

  Format:

SID	Session-Id
TYP	Datagrammtyp
N*1	lz_subid
T*x	Objektkennung
T*x	dp_id (Datenpunktadresse)
T*x	Beschreibung des Datenpunktes
N*1	dp_type/dp_subtype
D*8	Datum der Messung
Z*6	Zeit der Messung
N*1	fct_id/tab_id
N*1	prd/rpr/emq
N*1	val/apr/mis/msk
N*1	letzter val/apr/mis/msk (zur Identifizierung einer Veränderung)
N*1	info/event
N*1	letzter info/event (zur Identifizierung einer Veränderung)
N*1	atb_id
N*1	actual
N*1	nominal/dimension
T*x	atb_id/actual oder dimension (als Klartext)
T*x	atb_id/nominal (als Klartext)
R*4	actual (float)
R*4	nominal (float)
R*4	last_save
N*1	update_control

N*1	fixed
R*4	alarm_low
R*4	warning_low
R*4	warning_high
R*4	alarm_high

- GAMDG_FND10_USM (5)
  Verwendung:
  spontane Meldung eines GA-Knotens

  Format:

SID	Session-Id
TYP	Datagrammtyp
N*1	lz_subid
T*x	Objektkennung
T*x	dp_id (Datenpunktadresse)
T*x	Beschreibung des Datenpunktes
N*1	dp_type/dp_subtype
D*8	Datum der Messung
Z*6	Zeit der Messung
N*1	val/apr/mis/msk
N*1	letzter val/apr/mis/msk (zur Identifizierung einer Veränderung)
N*1	info/event
N*1	letzter info/event (zur Identifizierung einer Veränderung)
N*1	atb_id
N*1	actual
N*1	nominal/dimension
T*x	atb_id/actual oder dimension (als Klartext)
T*x	atb_id/nominal (als Klartext)
R*4	actual (float)
R*4	nominal (float)
R*4	last_save
N*1	update_control
N*1	fixed
R*4	alarm_low
R*4	warning_low
R*4	warning_high
R*4	alarm_high

- GAMDG_REJECT (6)
  Verwendung:
  FND 1.0 Reject (Ablauf-Fehler; empfangen von einem GA-Knoten)

  Format:

SID	Session-Id
TYP	Datagrammtyp
N*1	lz_subid
T*x	Objektkennung
T*x	dp_id (Datenpunktadresse)
T*x	Text entsprechend fct_id/tab_id/exc_1/exc_2

- GAMDG_ERROR (7)
  Verwendung:
  FND 1.0 Error (Syntax-Fehler; empfangen von einem GA-Knoten)

  Format:

SID	Session-Id
TYP	Datagrammtyp
N*1	lz_subid
T*x	Text entsprechend fct_id/tab_id/exc_1/exc_2

- GAMDG_ACK (8)
  Verwendung:
  Antwort der LZH für den Client als Bestätigung, daß das vorausgegangene Kommando der Bedienstation akzeptiert wurde.

  Format:

SID	Session-ID
TYP	Datagrammtyp

- GAMDG_NAK (9)
  Verwendung:
  Antwort der LZH für den Client mit der Aussage, daß das vorausgegangene Kommando der Bedienstation nicht akzeptiert wurde. Der in dem Datagramm enthaltene Fehlertext beschreibt dabei die Fehlersituation.

  Format:

SID	Session-ID
TYP	Datagrammtyp
T*x	Fehlertext

- GAMDG_TIME_REQUEST (10)
  Verwendung:
  Abfrage der Systemzeit des LZH-Servers durch eine Bedienstation.

  Format:

SID	Session-Id
TYP	Datagrammtyp

- GAMDG_TIME (11)
  Verwendung:
  Antwort der LZH auf das Datagramm GAMDG_TIME_REQUEST.

  Format:

SID	Session-Id
TYP	Datagrammtyp
T*x	Zeitinformation

- GAMDG_ARE_YOU_THERE (12)
  Verwendung:
  Abfrage durch den userver-Prozeß der LZH, ob eine Bedienstation noch aktiv

ist.

Format:
SID Session-Id
TYP Datagrammtyp

- GAMDG_I_AM_HERE (13)
  Verwendung:
  Antwort einer Bedienstation auf GAMDG_ARE_YOU_THERE. Wenn die Antwort ausbleibt, wird die Sitzung durch den userver-Prozeß beendet.

  Format:
  SID Session-Id
  TYP Datagrammtyp

- GAMDG_START_ANZMOD (14)
  Verwendung:
  Befehl einer Bedienstation an den userver-Prozeß, ab jetzt alle einlaufende Meßdaten, die den Auswahlkriterien entsprechen, zu schicken (Start des Anzeigemodus).

  Format:
  SID Session-Id
  TYP Datagrammtyp
  T*x regulärer Ausdruck zur Auswahl von Objekten
  T*x regulärer Ausdruck zur Auswahl von Datenpunktadressen
  N*1 nur Daten von Datenpunkten dieses Typs sollen gesendet werden

- GAMDG_STOP_ANZMOD (15)
  Verwendung.
  Befehl einer Bedienstation an den userver-Prozeß, das Senden von Meßwerten für den Anzeigemodus einzustellen.

  Format:
  SID Session-Id
  TYP Datagrammtyp

- GAMDG_IW_DEFINE (16)
  Verwendung:
  Hinzufügen von Datenpunkten zu einer Istwertabfrage. Mit Hilfe dieses Datagramms definiert eine Bedienstation, welche Datenpunkte in eine Istwertabfrage aufgenommen werden sollen. Die Istwertabfrage wird durch dieses Datagramm noch nicht gestartet. Mit weiteren Datagrammen vom Typ GAMDG_IW_DEFINE kann die Liste der Datenpunkte, für die Istwerte übertragen werden sollen, erweitert werden. Die Istwertabfrage wird durch den userver- und gakserver-Prozeß erst dann aktiviert, wenn die Bedienstation das Datagramm GAMDG_IW_START sendet. Mit dem Datagramm GAMDG_IW_STOP kann die Bedienstation eine laufende Istwertabfrage stoppen. Dieses Kommando löscht auch die Liste der zuvor mit den Datagrammen GAMDG_IW_DEFINE definierten Datenpunkte. Eine laufende

Istwertabfrage wird durch die LZH beendet, wenn die Bedienstation sich abmeldet (Datagramm GAMDG_ABMELDEN) oder als nicht mehr aktiv erkannt wird. Näheres zu den Parametern Zykluszeit, Synchronisationszeit und Verbindungsdauer findet man in [2], Seite 48.

Format:

SID	Session-Id
TYP	Datagrammtyp
N*1	lz_subid
T*x	Objektkennung (Es sind die Jokerzeichen * und ? erlaubt.)
T*x	Datenpunktadresse (Es sind die Jokerzeichen * und ? erlaubt.)
R*4	Zykluszeit im Minuten
N*1	Zeiteinheit für Synchronisationszeitpunkt
R*4	Synchronisationszeitpunkt
R*4	Verbindungsdauer in Minuten

- GAMDG_IW_START (17)
  Verwendung:
  Start einer Istwertabfrage durch eine Bedienstation. Es müssen zuvor mit Datagrammen vom Typ GAMDG_IW_DEFINE die Datenpunkte definiert worden sein, deren Istwerte ausgelesen werden sollen.

  Format:

SID	Session-Id
TYP	Datagrammtyp
N*1	lz_subid

- GAMDG_IW_STOP (18)
  Verwendung:
  Befehl einer Bedienstation an die LZH, die laufende Istwertabfrage zu stoppen. Zusätzlich löscht dieses Datagramm die Liste aller Datenpunkte der Istwertabfrage.

  Format:

SID	Session-Id
TYP	Datagrammtyp
N*1	lz_subid

GAKIMPORT

Das Programm 'gakimport.exe' dient zum Einlesen von Konfigurationsdateien in die Datenbank der LZH. Konfigurationsdateien sind ASCII-Dateien, die Objekte und Datenpunkte beschreiben, die zu einem GA-Knoten gehören. Die Dateien haben typischerweise die Endung '*.cfg'. Sie werden vom Meßgeräte-Hersteller auf MSDOS-formatierten Disketten geliefert.

Das Einlesen geschieht an einer Bedienstation in folgenden Teilschritten:

- Kopieren der Konfigurations-Datei von der Diskette auf eine Datei der LZH mit dem Dateiübertragungsprogramm 'ftp'. Dazu die unter Windows 95 (oder Windows NT) zur Verfügung stehende Version von 'ftp' benutzen.

- Einloggen in die LZH mit Hilfe von 'telnet'.

- Die vom PC auf die LZH kopierte Konfigurations-Datei mit Hilfe des LZH-Kommandos 'oem2iso.exe' vom DOS- (= OEM-) in den ISO8859-Zeichensatz umwandeln. 'oem2iso.exe' ist ein "typisches" UNIX Kommando: es liest den zu konvertierenden Text von 'stdin' und gibt den konvertierten Text auf 'stdout' aus. Zum Umkodieren einer Konfigurations-Datei muß deshalb eine Kommandozeile der folgenden Art verwendet werden:

  ```
  $ cat DOS_DATEI.cfg | oem2iso.exe > ISO8859_DATEI.cfg
  ```

  Das LZH-Kommando 'oem2iso.exe' findet man im Verzeichnis '/zutil'.

- Jetzt kann das Programm 'gakimport.exe' verwendet werden, um die Konfigurations-Datei im ISO8859-Format in die Datenbank der LZH einzulesen:

  'gakimport.exe' benötigt die folgenden Schalter:

-betreuer	NAME_DES_BETREUERS (max. 10 Zeichen)
-gaknoten	NAME_DES_GAKNOTENS (max. 10 Zeichen)
-eingabe	NAME_DER_KONFIGURATIONSDATEI

  Die Bezeichnung jedes Schalters muß mit einem - (Minus) Zeichen beginnen. Nach der Bezeichnung des Schalters folgt - durch wenigstens ein Leerzeichen abgetrennt - der Wert des Schalters.

  Die Schalter können abgekürzt werden bis auf den Anfangsbuchstaben.

  Die Schalter können mit großen oder kleinen Buchstaben angegeben werden. Der Wert eines Schalter muß die richtige Groß-Kleinschreibung enthalten.

  Die Reihenfolge der Schalter in der Kommandozeile ist beliebig.

Der Betreuer (Wert des -b Schalters) und der GA-Knoten (Wert des -g Schalters) müssen bereits in der Datenbank gespeichert sein, bevor '`gakimport.exe`' gestartet wird.

Beispiel für einen Programm-Aufruf:

```
$ gakimport.exe -GAK GAK1 -betreu FUNK -e herrnstr_iso.cfg
```

Wenn die Eingabedatei Objekte und Datenpunkte enthält, die in der Datenbank bereits enthalten sind, so werden diese Daten mit den Daten aus der Eingabedatei überschrieben. Man kann '`gakimport.exe`' also mit der gleichen Eingabedatei mehrfach laufen lassen.

Wenn in der Datenbank bereits Datenpunkte existieren, die zu dem angegebenen GA-Knoten gehören, aber für das fiktive Objekt ´ZZZZ´ eingetragen sind, dann werden diese Datenpunkte gelöscht. Diese Funktionalität wurde eingebaut, um alle Datenpunkte zu löschen, die während des Probebetriebs eines GA-Knotens vor dem Einspielen der Konfigurationsdatei durch das automatische Erfassungsprogramm in die Datenbank eingetragen wurden.

´`gakimport.exe`´ unterzieht die Konfigurationsdatei einer eingehenden Prüfung. Es werden in der Datenbank nur dann Daten abgespeichert, wenn ALLE Einträge der Konfigurationsdatei korrekt behandelt werden konnten.

GAKDDE

´gakdde´ ist ein unter Microsoft Windows 3.1 ablauffähiger Serverprozeß. Er enthält die beiden folgenden Funktionalitäten:

Zum einen ermöglicht er die Kommunikation der Bedienstation mit dem userver-Prozeß des LZH-Servers über eine WINSOCKET-Schnittstelle.

Zum anderen dient er als DDE-Server für Oracle Forms Applikationen, die sich die Zugriffsmöglichkeiten auf die vom LZH-Server verwalteten GA-Knoten zunutze machen wollen.

Wir erklären die Funktionsweise von gakdde am besten, wenn wir beschreiben, wie eine Oracle-Forms-Applikation sich die Leistungen von gakdde zunutze macht.

Nehmen wir als Beispiel die Forms-Applikation ´Anzeigemodus´. In dieser Applikation sollen alle einlaufenden Meßwerte der GA-Knoten auf dem Bildschirm einer Bedienstation dargestellt werden. Die Meßwerte werden vom userver-Prozeß des LZH-Servers zur Verfügung gestellt. Dieser Prozeß wartet nur darauf, daß sich jemand anmeldet und den Anzeigemodus einschaltet, um die Daten abzurufen und zu sichten. Genau das tut die Forms-Applikation ´Anzeigemodus´ in folgenden Teilschritten:

Zunächst wird die DDE-Kommunikation zwischen der Forms-Applikation und dem gakdde-Server durch Aufruf der unter Oracle-Forms zur Verfügung stehenden Funktion DDE.INITIATE gestartet. Dadurch werden zwei Aktionen ausgelöst:

Zum einen wird der gakdde-Prozeß gestartet, wenn er nicht bereits von einer anderen Applikation gestartet wurde. Das Starten von gakdde bewirkt, daß auf der Fensteroberfläche der gakdde-Server als Icon erscheint. Er bleibt auch während seiner ganzen Ausführungszeit nur als Icon sichtbar. Weiterhin wird die Kommunikation mit dem userver-Prozeß initialisiert (Aufruf der Funktionen WSAStartup() und socket())und festgelegt, wie asynchron vom userver-Prozeß eingehende Datagramme vom gakdde zu behandeln sind (Aufruf der Funktion WSAAsyncSelect()).

Zum anderen wird durch den Aufruf von DDE.INITIATE in der Forms-Applikation im gakdde-Server das Ereignis WM_DDE_INITIATE erzeugt. Es veranlaßt den gakdde-Server, für jede sich mit DDE.INITIATE anmeldende Forms-Applikation (DDE-Client) ein neues unsichtbares DDE-Client-Fenster anzulegen. Die DDE-Client-Fenster gehören einer anderen vom gakdde-Hauptprogramm registrierten Fensterklasse mit dem Namen ServerClass an. Sie benutzen als Event-Handler nicht den Event-Handler des Hauptfensters wndproc(), sondern die Routine servproc(). Jedes DDE-Client-Fenster merkt sich das Window-Handle der Forms-Applikation, die das DDE-Client-Fenster erzeugt hat. Es wird als Session-Id für die Kommunikation mit dem userver-Prozeß verwendet. Danach wird noch eine Datenstruktur für jedes DDE-Client-Window angelegt, daß zur Pufferung von eingehenden Daten des LZH-Servers dient. Zum Schluß sendet der gakdde-Prozeß der Forms Applikationen eine

Meldung zurück, aus der die Forms Applikation erkennen kann, an welches DDE-Client-Window es die weiteren DDE-Anforderungen stellen muß.

Jetzt ist der gakdde-Server für weitere Dienste vorbereitet und die Forms-Applikation kann mit den Dialog mit dem userver der LZH via gakdde-Server aufnehmen.

Zunächst installiert die Forms-Applikation einen Timer, der alle Sekunde abläuft. Er dient dazu, Daten abzurufen, die sich in der Zwischenzeit im Puffer des gakdde-Servers angesammelt haben. Sobald der Timer ablauft, fragt die Forms Applikation beim gakdde-Server nach, ob Daten für sie vorliegen. Dazu benutzt sie die Funktion DDE_REQUEST. Liegen Daten vor, so werden sie vom gakdde-Server an die Forms Applikation übertragen, die sie dann entsprechend ihres Anwendungszustandes interpretieren und gegebenenfalls am Bildschirm darstellen muß. Dieses Polling-Verfahren ist leider unvermeidbar, da es unter Forms keinen Multi-Tasking Mechanismus gibt. Würde man auf das Polling verzichten, dann wäre die Forms Applikation während der Wartezeit nicht bedienbar und eine Funktion nicht abbrechbar, wenn auf dem Kommunikationsweg irgendetwas schief läuft.

Wenn der Timer wieder einmal abgelaufen ist, übermittelt die Forms Applikation ihr erstes Datagramm – eine Anmeldung beim userver der LZH - mit Hilfe der DDE.POKE Funktion an den gakdde-Server. Der gakdde-Server fügt dem Datagramm die zuvor registrierte SessionId (= Window Handle der Oracle Forms Applikation) hinzu, sendet das Datagramm an den userver der LZH weiter und bestätigt gegenüber der Forms Applikation die korrekte Ausführung dieser Sequenz. Jetzt muß die Forms Applikation darauf warten, bis der userver-Prozeß an den gakdde-Prozeß ein Datagramm schickt, daß die Anmeldung positiv oder negativ bestätigt. Dies geschieht durch wiederholtes Nachfragen beim gakdde-Server, ob Daten vorliegen. Wenn diese schließlich eintreffen, werden sie vom gakkdde-Server mittels DDE.REQUEST übernommen und ausgewertet.

Jetzt kann die Oracle Forms Applikation ihren nächsten Befehl absetzen und das Spiel beginnt von neuem. Im Falle des Anzeigemodus wird die Oracle Forms Applikation das Kommado ´Start Anzeigemodus´ absetzen. Nachdem es ordnungsgemäß abgelaufen ist, übernimmt der Timer-Trigger innerhalb der Oracle Forms Applikation die Beschaffung (DDE.REQUEST) und Darstellung der beim gakdde-Prozeß eingelaufenen Meßwerte auf dem Bildschirm.

FORMULAR ANZEIGEMODUS

Mit Hilfe des Formulars Anzeigemodus (siehe Anhang A, Abb.1) kann der Benutzer einer Bedienstation die von den GA-Knoten zum LZH-Server gesendeten Meßwerte laufend beobachten.

Das Formular ist in zwei Abschnitte unterteilt:

Im oberen Bereich werden die einlaufenden Meßwerte in Tabellenform angezeigt. Der sichtbare Ausschnitt der Tabelle läßt sich in horizontaler und vertikaler Richtung mit Hilfe von Scrollbars einstellen. In den Spalten `Info` und `Event` werden die Bits gemäß FND 1.0 symbolisch dargestellt. In der Spalte `Meßzeit` wird aus Platzgründen vom Datum nur der Tag angezeigt.

Im unteren Bereich des Formulars befinden sich die Steuerungselemente für den Bediener:

Die Felder `Objekt, Adresse` und `Datenpunkttyp` dienen zur Filterung der einlaufenden Meßwerte. Falls irgendeines dieser Felder ausgefüllt ist, werden nur die Meßwerte angezeigt, die den Filterbedingungen genügen. Als Filterkriterien lassen sich auch Ausdrücke mit den Wildcards ´*´ (beliebig lange Zeichenfolge) und ´?´ (ein beliebiges Zeichen) angeben. Die Filterbedingungen werden bei jeder Aktivierung des `Start` Pushbuttons an den userver-Prozeß des LZH-Servers geschickt und von diesem auch ausgewertet.

Der Pushbutton `Start`/`Stop` dient zum Starten und Stoppen der Meßwertanzeige. Er wechselt seine Beschriftung je nach aktuellem Zustand: Wenn die Meßwertausgabe noch nicht gestartet wurde, enthält er die Beschriftung ´Start´. Der Benutzer kann dann mit einem Mouseklick die Meßwertausgabe starten. Die Beschriftung wechselt dabei in ´Stop´. Ein weiterer Mouseklick in diesem Betriebszustand stoppt die weitere Meßwertausgabe, wobei die Beschriftung wieder den Wert ´Start´ annimmt.

Mit dem Pushbutton `Schließen` wird das Formular Anzeigemodus beendet.

Zusätzlich zu den oben erwähnten Filterbedingungen läßt sich die Ausgabe der Meßwerte auf solche Datenpunkte beschränken, die zu einer ausgewählten Anzahl von Objekten gehören. In diesem Falle muß beim Aufruf des Formulars die Kommandozeile den Parameter `objfile=<dateiname>` enthalten. Der Parameter muß eine Textdatei benennen, die eine Liste der zu berücksichtigenden Objekte enthält. Die Textdatei muß pro Zeile die Bezeichnung eines Objektes (max. 4 Zeichen) enthalten.

Nach dem Aufruf des Formulars aus der Window NT Oberfläche erscheint zunächst das Formularfenster auf dem Bildschirm. Danach muß der Bediener noch einige Zeit warten, bis die automatisch ablaufende Initialisierungsprozedur abgelaufen ist. Sie besteht aus den folgenden Teilschritten:

- Aufbau der DDE-Verbindung zum gakdde-Prozeß (siehe oben)

- Anmeldung beim userver-Prozeß der LZH

Der Benutzer wird über den Fortgang der Initialisierung durch Textausgaben in der Statuszeile um unteren Bildschirmrand informiert. Während der Initialisierungsphase ist der Pushbutton `Start/Stop` nicht aktivierbar.

Hinweise für Entwickler

Das Formular wurde mit Hilfe von Oracle Forms Version 4.5 unter Windows NT 4.0 erstellt. Das Formular hat den Modulnamen `anzmod.fmb` bzw. `anzmod.fmx` (kompilierte Version). Die Dateien befinden sich auf dem Entwicklungs-PC der LHM im Verzeichnis `c:\lhmaxp`. Im Unterverzeichnis `c:\lhmaxp\anzmod` findet man einige als Textdateien exportierte, zum Formular gehörende PLS/SQL Trigger. Gleichlautende Kopien sind auch auf dem LZH-Server (SUN) im Verzeichnis `/udd/lzhente/lhmaxp` abgelegt.

Der Aufruf des Formulars von der Windows NT Kommandozeile lautet wie folgt:

```
C:\orant\bin\f45run32.exe anzmod <username/password> [objfile=<filename>]
```

FORMULAR FND1.0 KOMMUNIKATION

Das Formular FND1.0 Kommunikation (siehe Anhang A, Abb. 2) gestattet die direkte Abfrage und Veränderung von Datenpunkten innerhalb der GA-Knoten vor Ort in den Objekten auf der Ebene des FND 1.0 Protokolls. Als Beispiel mag das An- oder Abschalten der Beleuchtung eines Raumes von irgendeiner Bedienstation innerhalb des Netzes der LZH dienen.

Das Formular ist in drei Bereiche unterteilt:

- Oben: Tabelle mit ausgewählten Datenpunkten
- Mitte: Felder zur Anzeige/Veränderung von FND 1.0 Parametern
- Unten: Pushbuttons zur Ablaufsteuerung

Die Bearbeitung eines Datenpunktes durch den Bediener läuft in groben Zügen wie folgt ab:

- Auswahl des zu bearbeitenden Datenpunktes
- Eventuelle Änderung der Parameter des ausgewählten Datenpunktes
- Abschicken eines Kommandos an den GA-Knoten

Nach dem Aufruf des Formulars aus der Window NT Oberfläche erscheint zunächst das Formularfenster auf dem Bildschirm. Danach muß der Bediener noch einige Zeit warten, bis die automatisch ablaufende Initialisierungsprozedur abgelaufen ist. Sie besteht aus den folgenden Teilschritten:

- Aufbau der DDE-Verbindung zum gakdde-Prozeß (siehe oben)
- Anmeldung beim userver-Prozeß der LZH

Der Benutzer wird über den Fortgang der Initialisierung durch Textausgaben in der Statuszeile um unteren Bildschirmrand informiert. Während der Initialisierungsphase sind die Bedienelemente deaktiviert (mit halber Intensität dargestellt).

Auswahl eines Datenpunktes

Bevor ein Datenpunkt bearbeitet werden kann, muß dieser zunächst in der Datenbank DATENPUNKTE des LZH-Servers gesucht und seine aktuellen Parameter auf dem Formular sichtbar gemacht werden. Dies geschieht mit Hilfe der Bedienelemente im oberen Bereich des Formulars. Er besteht aus einer Tabelle und einer Leiste von Pushbuttons (rechts oben: `Definieren, Abbrechen, Ausführen`).

Der Benutzer muß zunächst den Pushbutton `Definieren` betätigen, um eine Abfrage an die Datenbank zu formulieren. Nach dem Mouseklick auf den Pushbutton

`Definieren` kann der Benutzer in die erste Zeile der Tabelle Filterkriterien eintragen. Also z.B. nur Datenpunkte auswählen, die zu einem vorgegebenen Objekt gehören. Bei der Definition der Filterkriterien sind auch Wildcard-Zeichen erlaubt: ´*´ steht für eine beliebig Zahl von Zeichen, ´?´ steht für genau ein Zeichen. Für die Spalte `Objekt` steht eine Werteliste zur Verfügung, die man mit der Funktionstaste `F9` aktivieren kann, wenn der Cursor sich in der Spalte `Objekt` befindet. Ein Hinweis, daß eine Werteliste zu diesem Feld existiert, findet man auch am unteren Rand des Formulars.

Nach der Definition der Datenbankabfrage muß der Benutzer diese ausführen. Dazu betätigt er den Pushbutton `Ausführen`. Dann wird in der Datenbank nach den angeforderten Datenpunkten gesucht und diese in der Tabelle dargestellt. Mit den Cursortasten (nach oben, nach unten, Bild nach oben, Bild nach unten) kann der Benutzer einen der in der Tabelle dargestellten Datenpunkte markieren. Der markierte Datenpunkt wird durch eine andere Hintergrundfarbe kenntlich gemacht. Gleichzeitig werden im mittleren Teil des Formulars die aktuellen FND 1.0 Parameter angezeigt, wie sie zur Zeit in der Datenbank des LZH-Servers gespeichert sind.

Der Pushbutton `Abbrechen` ist nur aktivierbar in der Phase der Definition einer Abfrage. Mit seiner Hilfe kann man die Definitions-Phase beenden, ohne die Abfrage auszuführen.

Die Auswahl der Datenpunkte läßt sich auf solche beschränken, die zu einer ausgewählten Anzahl von Objekten gehören. In diesem Falle muß beim Aufruf des Formulars die Kommandozeile den Parameter `objfile=<dateiname>` enthalten. Der Parameter muß eine Textdatei benennen, die eine Liste der zu berücksichtigenden Objekte enthält. Die Textdatei muß pro Zeile die Bezeichnung eines Objektes (max. 4 Zeichen) enthalten.

Anzeige/Änderung von Parametern

Im mittleren Bereich des Formulars werden die Parameter des Datenpunktes angezeigt, der im oberen Bereich markiert dargestellt ist. Die Darstellung ist abhängig vom Typ des Datenpunktes gemäß FND 1.0. Die Parameterwerte entsprechen den in der Datenbanktabelle DATENPUNKTE gespeicherten Werten zur Zeit der Abfrage. Diese müssen nicht unbedingt den gleichen Werten entsprechen, die der zuständige GA-Knoten zur Zeit kennt, weil sie der GA-Knoten bisher nicht zur permanenten Speicherung an den LZH-Server übertragen hat.

Wenn der Benutzer beabsichtigt, die Parameter nicht zu verändern, sondern nur die aktuell dem GA-Knoten bekannten Werte auslesen möchte, sind im mittleren Abschnitt des Formulars keine weiteren Einträge nötig. In diesem Fall löst der Benutzer das Auslesen der Parameter durch Betätigung des Pushbuttons `Lesen` im unteren Block aus. Dies hat zur Folge, daß von der Bedienstation über den LZH-Server dem betroffenen GA-Knoten ein Kommando geschickt wird, was diesen veranlaßt, die Werte des ausgewählten Datenpunktes zu bestimmen und über den umgekehrten Weg wieder an die Bedienstation zurückzuschicken. Während dieser Vorgang läuft, kann der Bediener des Formulars keine weitere Eingabe machen; alle Bedienelemente bis auf den Pushbutton `Abbrechen` werden während dieser Zeit deaktiviert. Sollte die Verbindung mit dem GA-Knoten nicht zustande kommen, erscheint nach einer

gewissen Zeit ein Fenster mit einer entsprechenden Fehlermeldung, nach deren Quittierung weitere Arbeiten mit dem Formular wieder möglich werden. Sollte überhaupt keine Reaktion erfolgen, so kann der Benutzer mit dem Pushbutton Abbrechen sich aus dieser Situation befreien. Wenn alles korrekt abläuft, werden nach dem Eintreffen der ausgelesenen Parameter diese im mittleren Teil des Formulars angezeigt.

Wenn der Benutzer die Parameter des ausgewählten Datenpunktes im zugehörigen GA-Knoten ändern will, dann tut er dies durch Ausfüllen oder Übertippen der freigegeben Felder in den mittleren Blöcken DP_#0, DP_#1 oder DP_#2. Alle nicht änderbaren Parameter sind mit halber Intensität dargestellt. Für eine Reihe von Feldern stehen Wertelisten zur Verfügung, erkenntlich daran, daß in der Statuszeile am unteren Bildschirmrand der Text ´Werteliste´ erscheint. Die Wertelisten können mit der Funktionstaste F9 aufgeblättert werden. Nach dem Ändern der Parameter in dem Bildschirmformular, muß der Bediener einen der Pushbuttons DP_#0, DP_#1 oder DP_#2 betätigen, um die Werte an den betroffenen GA-Knoten zu übertragen. Die Bedienung des Formulars bleibt auch hier während des Übertragungsvorganges solange gesperrt, bis eine Antwort eingetroffen ist. Mit einem Kommando lassen sich nur die Werte einer FND-Tabelle (dp_#0, dp_#1 oder dp_#2) verändern.

Hinweise für Entwickler

Das Formular wurde mit Hilfe von Oracle Forms Version 4.5 unter Windows NT 4.0 erstellt. Das Formular hat den Modulnamen fnd2.fmb bzw. fnd2.fmx (kompilierte Version). Die Dateien befinden sich auf dem Entwicklungs-PC der LHM im Verzeichnis c:\lhmaxp. Im Unterverzeichnis c:\lhmaxp\fnd2 findet man einige als Textdateien exportierte, zum Formular gehörende PLS/SQL Trigger. Gleichlautende Kopien sind auch auf dem LZH-Server (SUN) im Verzeichnis /udd/lzhente/lhmaxp abgelegt.

Der Aufruf des Formulars von der Windows NT Kommandozeile lautet wie folgt:

```
C:\orant\bin\f45run32.exe fnd2 <username/password> [objfile=<filename>]
```

FORMULAR ISTWERT-ABFRAGEN

Mit Hilfe des Formularesystems Istwert-Abfragen (siehe Abb.3 und 4 im Anhang A) lassen sich die aktuellen Meßwerte von Datenpunkte aus den zugehörigen GA-Knoten zyklisch auslesen.

Das Formularsystem besteht aus zwei Fenstern, die abwechselnd auf dem Bildschirm dargestellt werden können. Mit Hilfe des Formulars Istwert-Abfrage:definieren (Abb.3) kann der Bediener die Menge der Datenpunkte und den Abfragezyklus festlegen. Das Formular Istwert-Anzeige (Abb. 4) dient zur Anzeige der ausgelesenen Meßwerte.

Nach dem Aufruf des Formulars aus der Window NT Oberfläche erscheint zunächst das Formularfenster Istwert-Abfrage:definieren auf dem Bildschirm. Danach muß der Bediener noch einige Zeit warten, bis die automatisch ablaufende Initialisierungsprozedur abgelaufen ist. Sie besteht aus den folgenden Teilschritten:

- Aufbau der DDE-Verbindung zum gakdde-Prozeß (siehe oben)
- Anmeldung beim userver-Prozeß der LZH

Der Benutzer wird über den Fortgang der Initialisierung durch Textausgaben in der Statuszeile um unteren Bildschirmrand informiert.

Um eine Istwert-Abfrage durchzuführen, muß der Bediener die folgenden Aktionen durchführen:

- Mit Hilfe des Formulars Istwert-Abfrage:definieren kann er sich eine Liste der in der Datenbanktabelle DATENPUNKTE des LZH-Servers abgespeicherten Datenpunkte anzeigen lassen.
- In der angezeigten Liste markiert er die Datenpunkte, für die er die Istwert-Abfrage durchführen will.
- Dann gibt er an, in welchen Zeitabständen und zu welchen Zeitpunkten die Istwert-Abfrage durchgeführt werden soll.
- Danach gibt er die Abfrage zur Bearbeitung frei. Es werden dann alle Befehle, die zur Bearbeitung der Abfrage nötig sind, an die betroffenen GA-Knoten gesendet.
- Nach der Freigabe der Abfrage schaltet das Programm automatisch auf das Formular Istwert-Anzeige um, in dem die von den GA-Knoten gesendeten Meßwerte dargestellt werden. Der Benutzer kann von dem Anzeige-Fenster jederzeit wieder zurück in das Definitionsfenster schalten.

Abfrage definieren

Das Formular Istwert-Abfrage:definieren dient zur Festlegung, welche Datenpunkte ausgelesen werden sollen und in welchen Zeitabständen das geschehen soll.

Das Formular enthält die folgenden Blöcke:

- Oben: eine Tabelle mit einer Liste von Datenpunkten
- Unten links: einen Block, mit dessen Hilfe die Auswahl mehrerer Datenpunkte vereinfacht werden kann
- Unten Mitte: Felder zur Angabe von globalen Parametern zur Bestimmung des Abfragezyklus und der Verbindungsdauer mit den GA-Knoten
- Rechts unten: Pushbuttons zur Ablaufsteuerung

Die obere Tabelle dient zur Anzeige und Auswahl der Datenpunkte, für die eine Istwert-Abfrage durchgeführt werden soll.

Zunächst muß sich der Benutzer eine Liste von Datenpunkten anzeigen lassen, aus der er im zweiten Schritt dann die zu bearbeitenden Datenpunkte auswählen kann. Mit der Funktionstaste `F7` gelangt der Benutzer in den Abfrage-Modus der Datenbank: in diesem Zustand dient die erste Zeile der Tabelle als Filter. Nach Eingabe der Suchkriterien, bei denen auch die Wildcardzeichen ´*´ und ´?´ erlaubt sind, wird mit der Funktionstaste `F8` die Datenbanksuche gestartet. Die gefundenen Datenpunkte werden anschließend in der Tabelle angezeigt.

Im zweiten Schritt markiert der Benutzer die Datenpunkte, die er für die Istwert-Abfrage verwenden will. Die Markierung geschieht mit einem Doppelklick auf die betreffende Tabellenzeile. Markierte Datenpunkte werden mit einer grünen Hintergrundfarbe dargestellt. Ein erneuter Doppelklick auf eine Zeile löscht die Markierung wieder. Zur Erleichterung von Mehrfachmarkierungen dienen die Bedienelemente des Blocks Mehrfachauswahl (unten links):

- Pushbutton ´Nichts auswählen´: löscht alle Markierungen
- Pushbutton ´Alles auswählen´: markiert alle in der Tabelle angezeigten Zeilen
- Pushbutton ´Alles leeren´: löscht alle Einträge aus der Tabelle

Die Stellung der Radiobuttons ´Bereich´ und ´eine Zeile´ bestimmt, ob mit dem Doppelklick auf eine Zeile nur diese Zeile oder ein ganzer Bereich ausgewählt werden soll. Ist die Stellung ´Bereich´ ausgewählt, dann werden bei einem Doppelklick auf eine Zeile auch alle Zeilen markiert, die vor der aktuellen Zeile noch nicht markiert sind, bis man auf eine Zeile stößt, die bereits markiert ist. Entsprechendes gilt auch für das Entfernen einer Markierung. Das Markieren von Bereichen kann auch erreicht werden, wenn man während des Doppelklicks die Taste `Umschalt` oder `Strg` drückt.

Die Auswahl der Datenpunkte läßt sich auf solche beschränken, die zu einer ausgewählten Anzahl von Objekten gehören. In diesem Falle muß beim Aufruf des Formulars die Kommandozeile den Parameter `objfile=<dateiname>` enthalten. Der Parameter muß eine Textdatei benennen, die eine Liste der zu berücksichtigenden Objekte enthält. Die Textdatei muß pro Zeile die Bezeichnung eines Objektes (max. 4 Zeichen) enthalten.

Nach der Markierung der Datenpunkte muß der Benutzer bestimmen, zu welchen Zeiten die Istwert-Abfrage durchgeführt werden soll. Die Zyklus-Zeit gibt an, in

welchen Zeitabständen die Abfrage erfolgen soll. Die Angabe ist eine Gleitkommazahl in Minuten. Weiterhin kann angegeben werden, zu welchen Zeitpunkten die Abfrage erfolgen soll (Synch-Zeitpunkt). Wenn dieser Wert nicht ausgefüllt ist, erfolgt die Abfrage sofort. Für die Angabe des Synchronisationszeitpunktes steht eine Combobox zur Verfügung, aus der der Benutzer die zulässigen Werte bequem auswählen kann.

Die Parameter zur Zyklus-Zeit können entweder für jeden ausgewählten Datenpunkt innerhalb der Tabellenzeilen angegeben werden oder aber im Block ´globale Parameter für komplexe Abfragen´ als globale Parameter, die für alle ausgewählten Datenpunkte verwendet werden sollen, bei denen keine Angaben in den Tabellenzeilen gemacht wurden.

Weiterhin kann angegeben werden, wie lange die betroffenen GA-Knoten die Telephonverbindung zum LZH-Server während der Istwert-Abfrage aufrecht erhalten sollen. Die Angabe erfolgt in Minuten als Gleitkommazahl.

Abfrage ausführen

Wenn der Bediener alle Angaben zur Festlegung der Istwert-Abfrage gemacht hat, gibt er die Abfrage durch Aktivierung des Pushbuttons `Start/Anzeige` frei. Das bewirkt, daß alle notwendigen Kommandos über den LZH-Server zu den betroffenen GA-Knoten gesendet werden. Gleichzeitig wird das Istwert-Anzeige-Fenster aufgeblendet, damit der Benutzer die einlaufenden Istwerte beobachten kann.

Während der Anzeige der laufend eintreffenden Istwerte, kann der Bediener jederzeit zurück in das Definitions-Fenster schalten. Dazu dient der Pushbutton `zurück zur Definition` am unteren Rand des Anzeigefensters. Nach einer Umschaltung in das Definitionsfenster hat der Pushbutton `Start/Anzeige` die Beschriftung `Anzeige` angenommen, was andeuten soll, daß die Aktivierung dieses Pushbuttons in diesem Betriebszustand die erneute Aufschaltung des Anzeigefensters auslöst.

Wenn der Bediener die Istwert-Abfrage beenden will, dann verwendet er hierzu den Pushbutton `Stop` im Definitionsfenster. Danach kann eine neue Istwert-Abfrage definiert werden.

Die Istwert-Abfrage wird auch beendet, wenn der Benutzer mit dem Pushbutton `Beenden` die Applikation beendet.

Hinweise für Entwickler

Das Formular wurde mit Hilfe von Oracle Forms Version 4.5 unter Windows NT 4.0 erstellt. Das Formular hat den Modulnamen `istabf.fmb` bzw. `istabf.fmx` (kompilierte Version). Die Dateien befinden sich auf dem Entwicklungs-PC der LHM im Verzeichnis `c:\lhmaxp`. Im Unterverzeichnis `c:\lhmaxp\istabf` findet man einige als Textdateien exportierte, zum Formular gehörende PLS/SQL Trigger. Gleichlautende Kopien sind auch auf dem LZH-Server (SUN) im Verzeichnis `/udd/lzhente/lhmaxp` abgelegt.

Der Aufruf des Formulars von der Windows NT Kommandozeile lautet wie folgt:

```
C:\orant\bin\f45run32.exe istabf <username/password> [objfile=<filename>]
```

ANHANG A

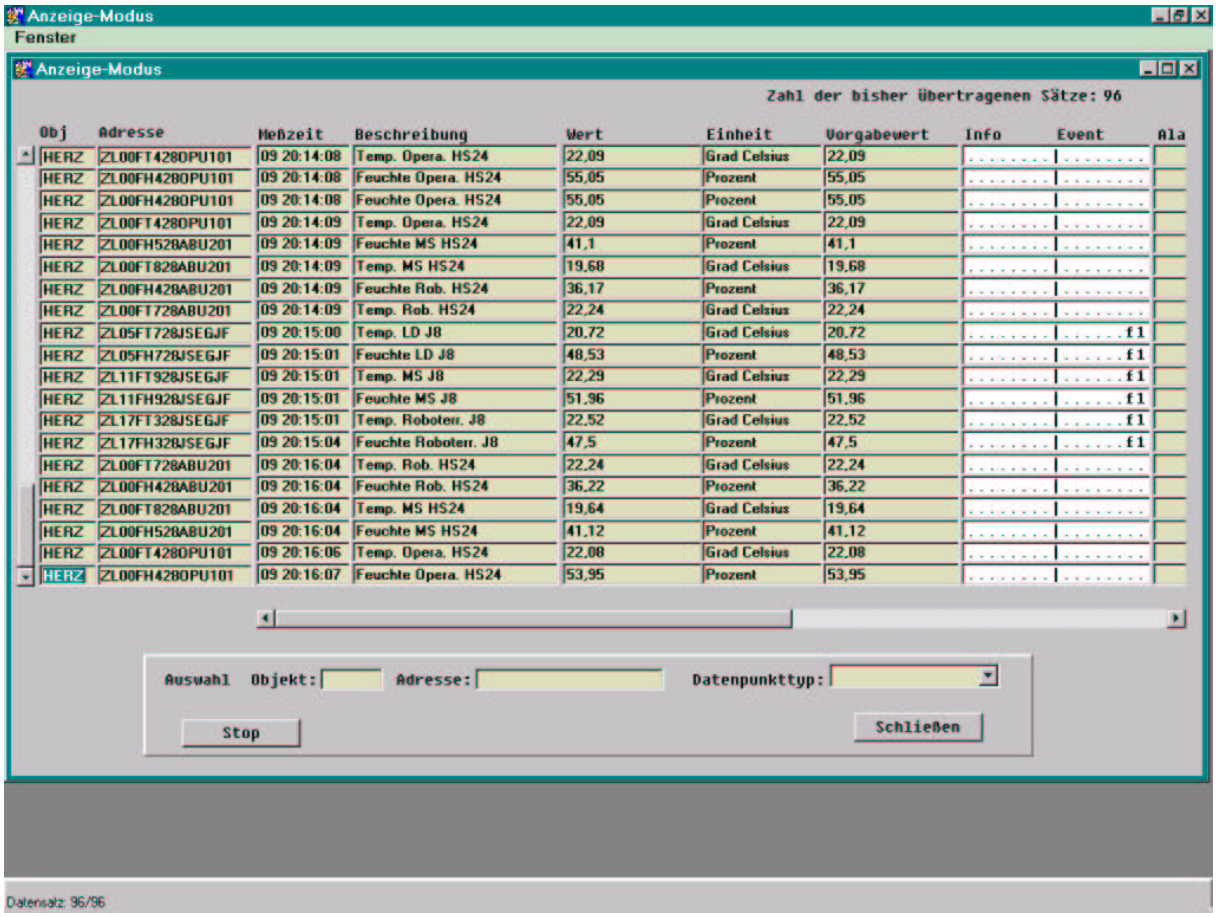

Abb.1 Anzeigemodus

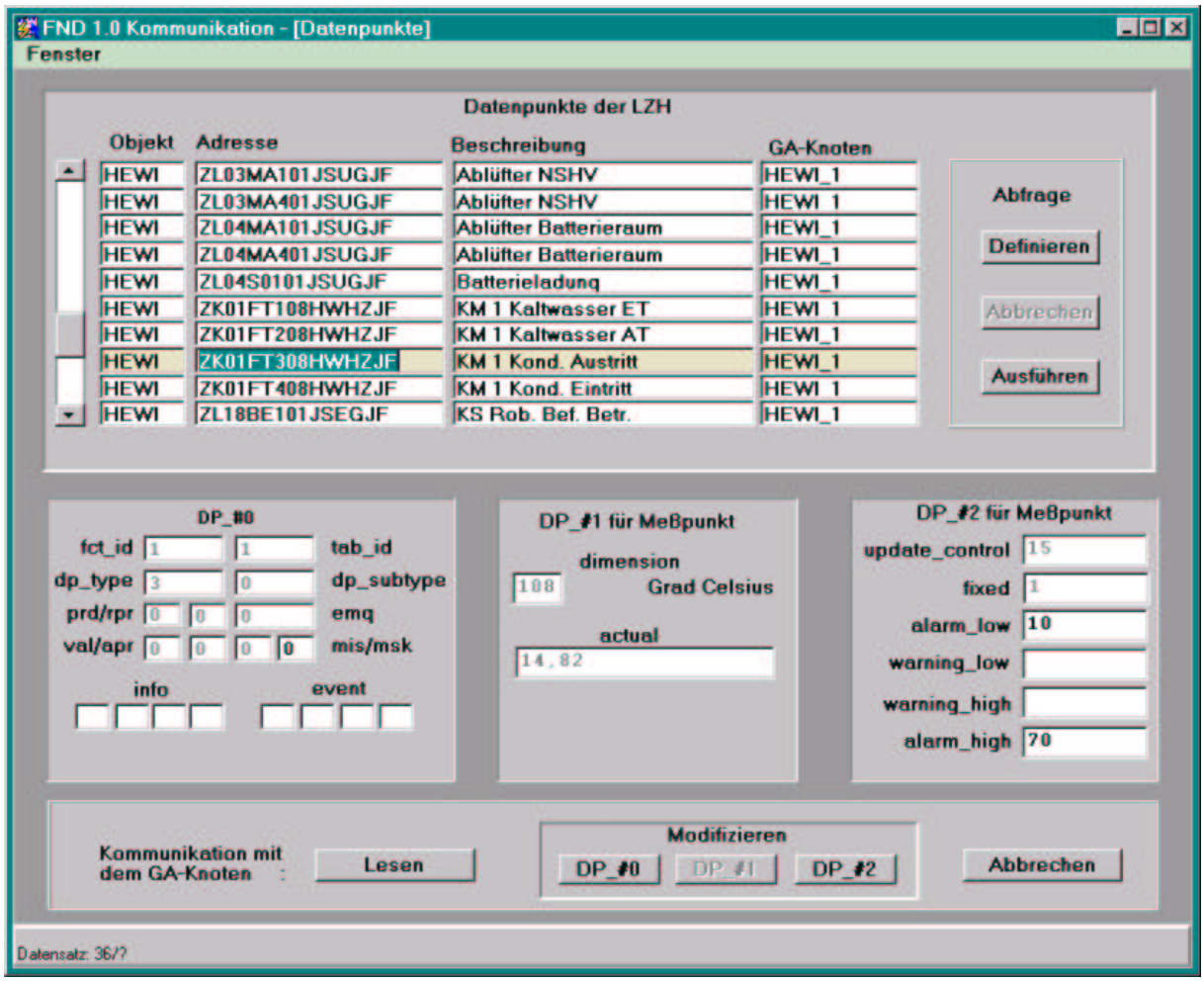

Abb.2 FND1.0 Kommunikation

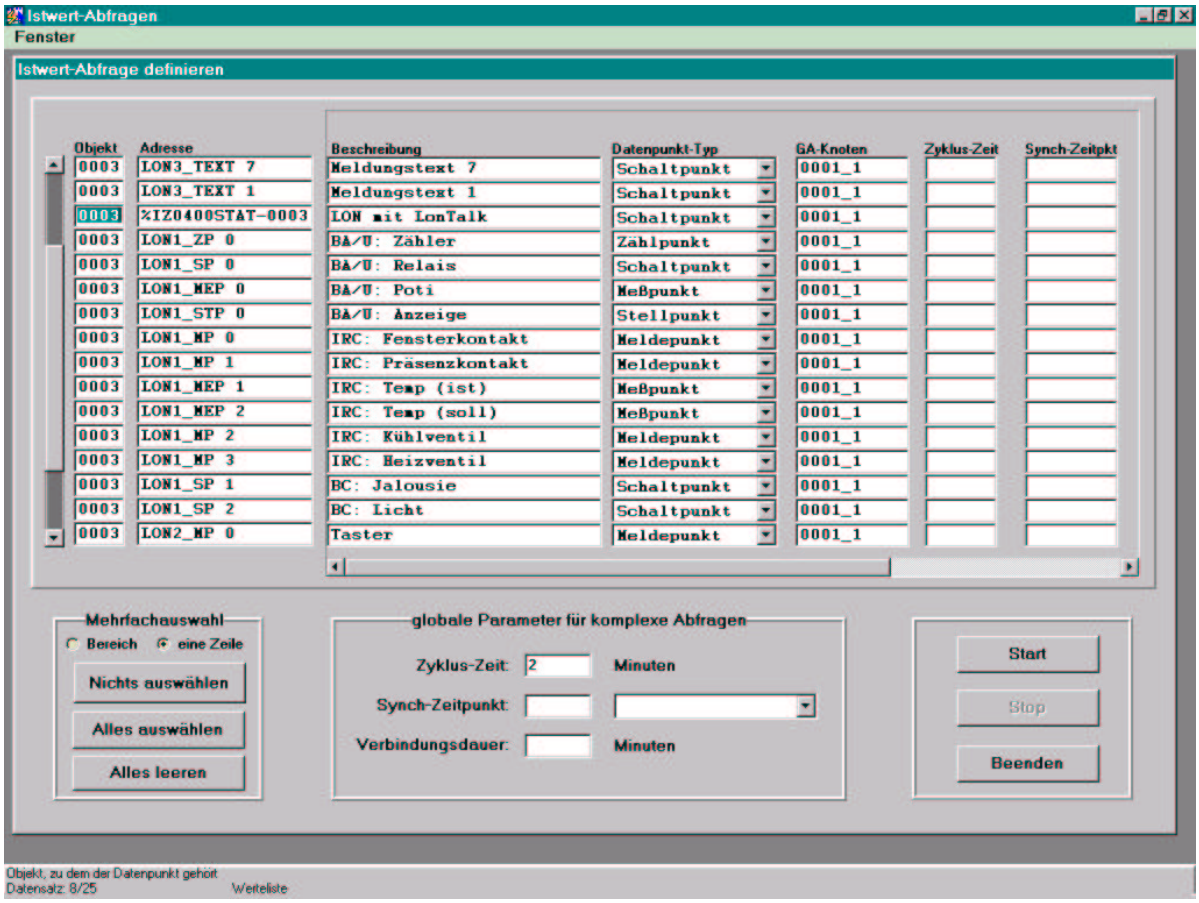

Abb. 3 Istwert-Abfrage: Definieren

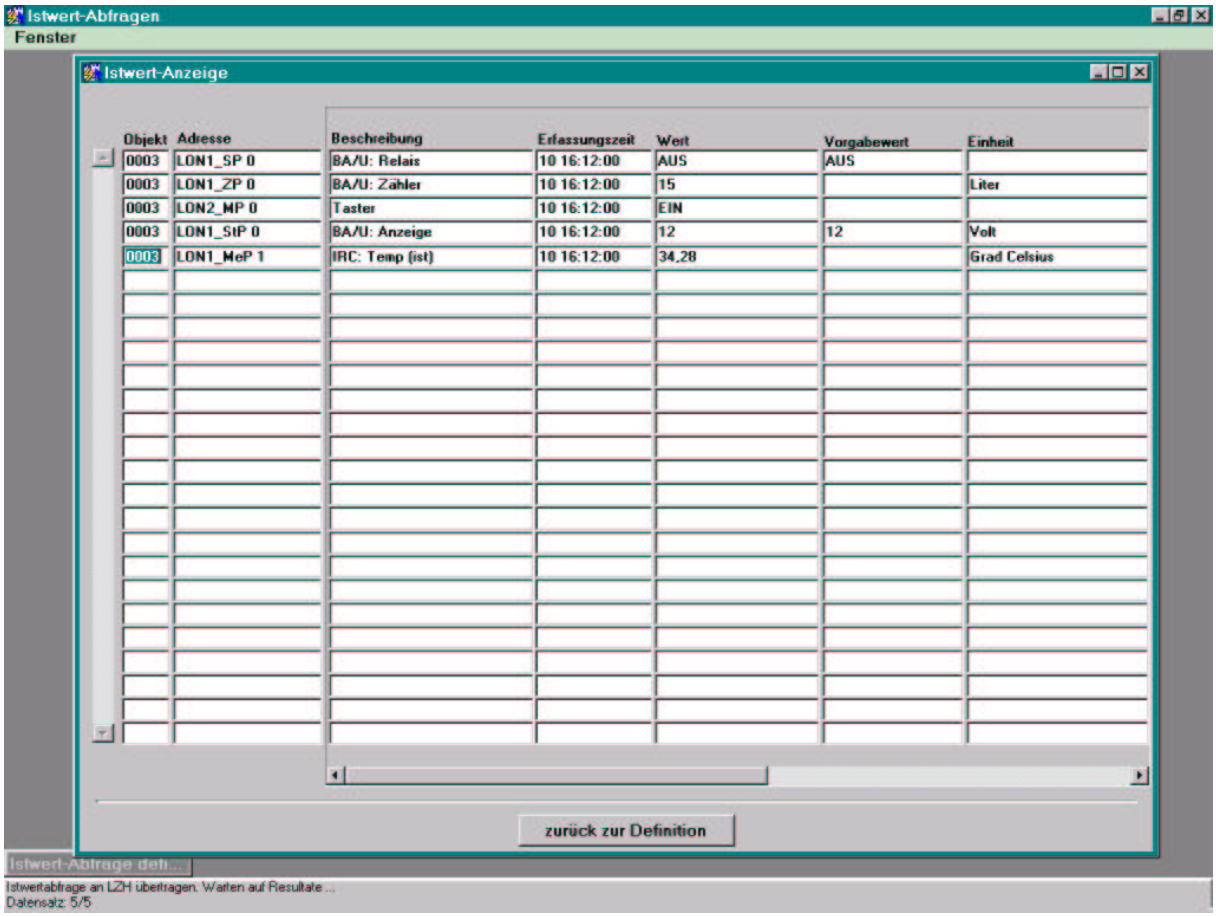

Abb.4 Istwert-Anzeige