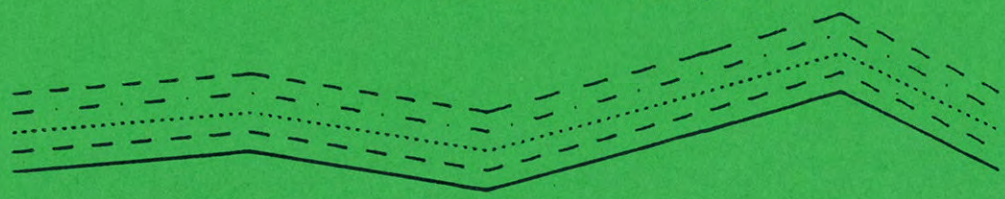
FEEDBACK

HEFT 14 1985

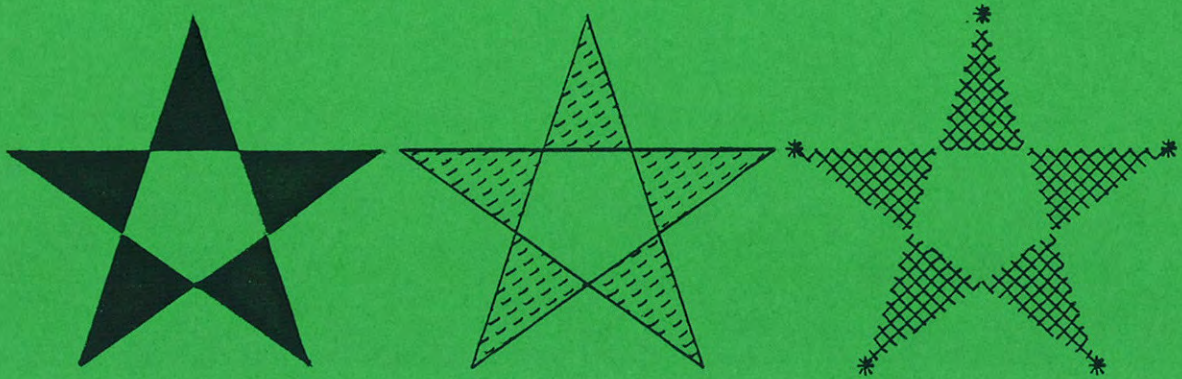
Das Graphische Kernsystem

GKS* an der VAX:
ГКС ау дер VAX
ГКС ан дер ВАКС
СУСТЕМ

Texte, Linienzuege,



Fuellgebiete & Polymarken



Herausgegeben von der Abt. Prozessrechenanlage des
EDV-Zentrums der Technischen Universität Wien,
1040 Wien, Gusshausstrasse 25

FEEDBACK

IMPRESSUM Feedback Nr. 14

Redaktion: Dr. G. Wehrberger, I. Poremba
Herausgeber, Verleger, Hersteller:
EDV-Zentrum der Technischen Universität Wien,
Abt. Prozeßrechenanlage,
Leitung: Dr. M. Paul
alle: A-1040 Wien, Gußhausstraße 25

INHALTSVERZEICHNIS

TAGUNGEN, BEITRÄGE & BERICHTE

SEITE der REDAKTION

GKS - drei Buchstaben, die dem Graphikanwender heute auf Schritt und Tritt begegnen, sind diesmal auch am Titelblatt des "Feedback" zu finden. Da das international genormte Graphische Kernsystem (GKS) in der Graphikwelt ständig an Bedeutung zunimmt, freuen wir uns, für unsere Kunden an der VAX-11/780 nun auch eine GKS-Implementation zur Verfügung stellen zu können.

Neben einer Einführung in das Graphische Kernsystem ist ein großer Teil dieser Nummer des "Feedback" Spezialgebieten wie Rechnernetzwerken und Real Time FORTRAN gewidmet. Im Rahmen verschiedener Tagungsberichte werden einerseits Beiträge von PRA-Mitarbeitern zu diesen Themenkreisen wiedergegeben, andererseits wird versucht, einen Einblick in die von uns besuchten internationalen Veranstaltungen zu geben. Sollte Ihr Interesse über die gebotenen Berichte hinausgehen, können Sie gerne bei den Autoren in die Tagungsbände bzw. in das mitgebrachte Informationsmaterial Einsicht nehmen.

G.W.

DAS GRAPHISCHE KERNSYSTEM GKS AN DER VAX

R. Sedlaczek

Inhalt:

1. PROBLEMSTELLUNG

Wenn man die heute verfügbaren Programme, Programmpakete und Systeme für Computer-Graphik betrachtet, stellt man fest, daß zwar alle eine Reihe von Grundfähigkeiten enthalten (praktisch jedes Computer-Graphik-Programm kann eine Linie erzeugen oder Text ausgeben), daß aber darüber hinaus mehr oder weniger Fähigkeiten enthalten sind, die für die jeweilige Anwendung wichtig sind (z.B. Diagramme und Koordinatenachsen für die Darstellung von Meß- und Rechenwerten oder Maßpfeile für technische Zeichnungen). Eine der ersten grundlegenden Entscheidungen auf dem Weg zu einer Systematik der Computer-Graphik und damit Grundlage einer möglichen Standardisierung war es daher, eine Trennung zu ziehen zwischen Kernfunktionen eines graphischen Systems und den zur jeweiligen Anwendung gehörenden Fähigkeiten. Diese Trennung in "modelling" und "core capabilities" wurde 1976 als Grundsatz für die Entwicklung eines graphischen Kernsystems formuliert und danach von den Gruppen, die sich die Standardisierung der Basisfunktionen der Computer-Graphik zur Aufgabe gemacht haben, sowohl in den USA als auch in Europa als Grundlage ihrer Arbeit betrachtet.

Das daraus entstandene graphische Kernsystem GKS ist die erste internationale Norm für das Gebiet der Computergraphik. Die Norm wurde von einem Unterausschuß des deutschen Normungsinstitutes DIN entworfen und seit 1979 durch eine internationale Überarbeitung im Rahmen einer ISO-Arbeitsgruppe zu einer internationalen Norm weiterentwickelt (ISO 7942, DIN 66252). Das GKS ist nicht nur die funktionale Beschreibung eines Computergraphik-Systems, es bildet darüber hinaus die Grundlage einer einheitlichen Methodologie für die graphische Datenverarbeitung.

2 Graphisches Kernsystem (GKS) Anwendungsbereich und Zweck

Diese Norm spezifiziert eine Menge von Funktionen für die Programmierung in der graphischen Datenverarbeitung. Sie bildet ein graphisches Grundsystem für Anwendungen, die in der Rechenanlage erzeugte, zweidimensionale Bilder auf Ausgabegeräten für Liniengraphik oder Rastergraphik erstellen. Sie unterstützt Bedienereingabe und Interaktion durch Grundfunktionen für die graphische Eingabe und Bildsegmentierung. Sie erlaubt die Speicherung und dynamische Veränderung von Bildern. Ein grundlegendes Prinzip im GKS ist die Zusammenfassung von graphischen Geräten zu einem graphischen Arbeitsplatz, der aus einer Anzahl von Eingabegeräten und einem einzigen Ausgabegerät besteht. Mehrere graphische Arbeitsplätze können gleichzeitig bedient werden. Das Anwendungsprogramm kann sein Verhalten an den graphischen Arbeitsplatz anpassen, um dessen Fähigkeiten besonders gut zu nutzen. Diese Norm enthält auch Funktionen zur Speicherung von Bildern in einer externen Bilddatei und zu ihrer Verarbeitung. Schließlich sind die Funktionen in aufwärtskompatiblen Leistungsstufen organisiert.

2.1 Einleitung

Das Graphische Kernsystem (GKS) liefert eine funktionale Schnittstelle zwischen einem Anwendungsprogramm und einer Anordnung graphischer Eingabe- und Ausgabegeräte. Die funktionale Schnittstelle enthält alle Grundfunktionen für die interaktive und passive Graphik auf einer breiten Vielfalt graphischer Geräte.

Die Schnittstelle hat eine so hohe Stufe der Abstraktion, daß Hardwareeigentümlichkeiten vom Anwendungsprogramm ferngehalten werden. Als ein Ergebnis ist eine übersichtliche Schnittstelle entstanden mit einheitlichen Darstellungselementen (POLYGON, POLYMARKE, TEXT, FÜLLGEBIET, VERALLGEMEINERTES DARSTELLUNGSELEMENT, ZELLMATRIX) und einheitlichen Eingabeklassen (LOKALISIERER, LINIENGEBER, WERTGEBER, AUSWÄHLER, PICKER, TEXTGEBER).

Die Konzeption mehrerer graphischer Arbeitsplätze erlaubt die gleichzeitige Ausgabe zu und Eingabe von verschiedenen graphischen Systemen. Die Fähigkeiten zur internen und externen Speicherung werden durch besondere graphische Arbeitsplätze bereitgestellt mit der Möglichkeit, graphische Elemente vom speziellen graphischen Arbeitsplatz für die interne Speicherung zu anderen graphischen Arbeitsplätzen zu übertragen.

Nicht jede GKS-Implementierung muß alle Funktionen unterstützen. Neun Leistungsstufen werden festgelegt, um die verschiedenen Anforderungen der graphischen Systeme zu berücksichtigen. Jede GKS-Implementierung stellt genau die Funktionen einer Leistungsstufe zur Verfügung. Die Leistungsstufen sind aufwärts verträglich.

GKS legt nur den sprachunabhängigen Kern eines graphischen Systems fest. Zur Integration in eine Sprache ist GKS in eine sprachabhängige Schicht eingebettet, die die Sprachkonventionen enthält, wie beispielsweise Parameter- und Namenszuweisung.

Das in Bild 1 gezeigte Schichtenmodell zeigt die Anwendung eines GKS in einem graphischen System. Jede Schicht kann die Funktionen der angrenzenden niedrigeren Schicht aufrufen. Im allgemeinen benutzt das Anwendungsprogramm die anwendungsorientierte Schicht, die sprachabhängige Schicht, andere von der Anwendung abhängige Schichten und Hilfsmittel des Be-

triebssystems. Alle Fähigkeiten des graphischen Arbeitsplatzes, die von GKS-Funktionen angesprochen werden können, werden nur mittels GKS benutzt.

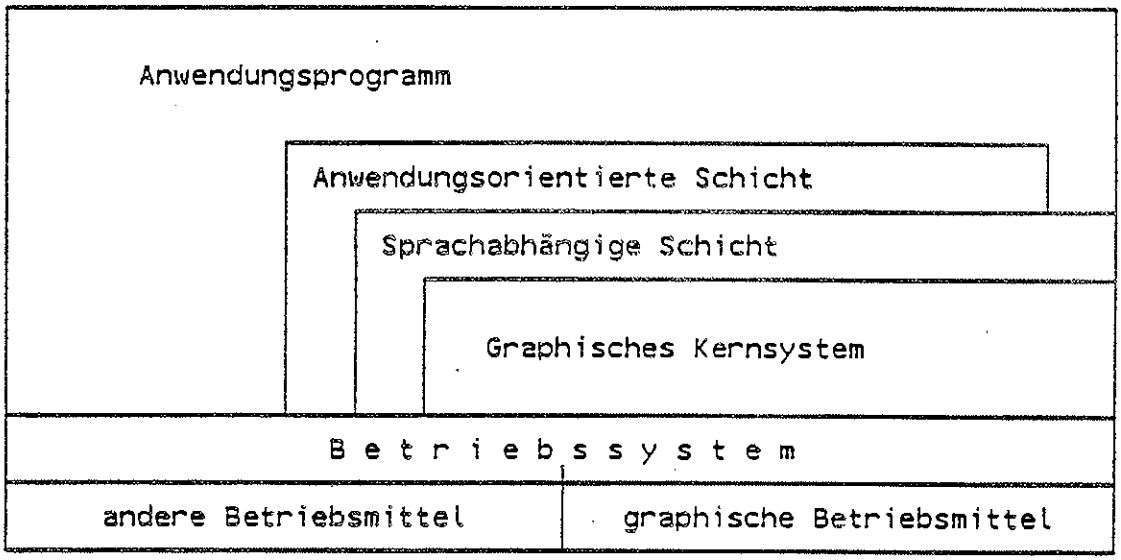


Bild 1. Schichtenmodell des GKS

2.2 Konzeption

Die graphische Ausgabe, die mit GKS erzeugt wird, ist aus zwei Gruppen von Grundelementen zusammengesetzt, die Darstellungselemente und Darstellungsattribute heißen. Die Darstellungselemente sind Abstraktionen der Grundfunktionen, die ein Gerät ausführen kann, wie das Zeichnen von Linien und das Drucken von Zeichenfolgen. Die Attribute steuern die Aspekte der Darstellungselemente an einem Gerät, wie zum Beispiel Linientyp, Farbe, Zeichenhöhe und Pickerkennzeichnung. Nichtgeometrische Aspekte wie zum Beispiel Farbe (im Gegensatz zu geometrischen Aspekten wie zum Beispiel Zeichenhöhe) können individuell für jeden graphischen Arbeitsplatz gesetzt werden, um dessen Fähigkeiten bestens zu nutzen.

Die graphische Information, die durch Bedieneraktionen an einem Gerät eingegeben wird, wird durch GKS in sechs Eingabeklassen abgebildet, wobei für jede Klasse ein Datentyp steht, der als logischer Eingabewert bezeichnet wird. Die Einrichtung einer solchen Gerätebeschreibung wird als logisches Eingabegerät bezeichnet. Die Wirkung der Eingabetätigkeit auf die Darstellungsfläche, wie Auffordern und Echoen, wird durch GKS individuell für jedes logische Eingabegerät gesteuert.

Zwei abstrakte Begriffe (abstrakte Ausgabe und abstrakte Eingabe) sind die Bausteine des sogenannten, abstrakten graphischen Arbeitsplatzes. Ein graphischer Arbeitsplatz des GKS ist eine Anordnung von Geräten, die eine oder keine Darstellungsfläche und keine oder mehrere Eingabegeräte, wie Tastatur, Tablett und Lichtstift, haben. Der graphische Arbeitsplatz bietet dem Anwendungsprogramm diese Geräte als eine Anordnung abstrakter Geräte an, so daß die Eigentümlichkeiten der Hardware ausgeblendet sind.

Die geometrische Information (Koordinaten) in den Darstellungselementen, Attributen und logischen Eingabegeräten (LINIENGEBER und LOKALISIERER) kann Transformationen unterworfen werden. Diese Transformationen führen

Abbildungen zwischen drei Koordinatensystemen durch :

a) Weltkoordinaten (WK), die der Anwendungsprogrammierer verwendet.

b) Normalisierte Koordinaten (NK), die zur Definition eines einheitlichen Koordinatensystems für alle graphischen Arbeitsplätze verwendet werden.

c) Gerätekoordinaten (GK), ein Koordinatensystem je graphischen Arbeitsplatz, das die Koordinaten seines Darstellungsbereichs beschreibt.

Darstellungselemente und Attribute werden von WK nach NK durch Normalisierungstransformationen, von NK nach NK durch Segmenttransformationen (siehe unten) und von NK nach GK durch Gerätetransformationen abgebildet. Eingaben vom LOKALISIERER werden durch eine inverse Gerätetransformation von GK nach NK und durch eine der inversen Normalisierungstransformationen von NK nach WK abgebildet.

Darstellungselemente und Darstellungsattribute können in einem Segment angeordnet werden. Segmente bilden die Einheit für die Manipulation und Änderung. Manipulation beinhaltet Erzeugen, Löschen und Umbenennen. Änderung beinhaltet Transformieren eines Segments, Unsichtbarmachen eines sichtbaren Segments und Hervorheben eines Segments. Segmente bilden auch die Grundlage für die geräteunabhängige Speicherung von Bildern zur Laufzeit. Über diesen Speicher, der als besonderer graphischer Arbeitsplatz mit der Bezeichnung arbeitsplatzunabhängiger Segmentspeicher angelegt ist, können Segmente eingesetzt und zu anderen graphischen Arbeitsplätzen übertragen werden.

Die Attribute, die das Aussehen von Teilen des Bildes (Darstellungselemente, Segmente, Aufforderungs- und Echoarten von Eingabegeräten) auf der Darstellungsfläche steuern, sind auf einheitliche Art und Weise festgelegt. Zwei Gruppen von Attributen beeinflussen die Darstellung eines jeden Darstellungselements: Darstellungs- und Arbeitsplatzattribute. Die Darstellungsattribute werden modal spezifiziert und bei der Erzeugung mit dem Darstellungselement verbunden. Die Darstellungsattribute beinhalten alle geometrischen Aspekte der Darstellungselemente, wie zum Beispiel Zeichenhöhe für Text und Mustergröße für Füllgebiet. Zusätzlich werden die nichtgeometrischen Aspekte durch Darstellungsattribute in einer von zwei Arten gesteuert. Entweder wird ein einziges Attribut benutzt, um alle nichtgeometrischen Aspekte des Darstellungselements mit einem Index zu spezifizieren, der auf eine geräteabhängige Beschreibung (Mengen von Werten) zeigt. Oder es wird für jeden der nichtgeometrischen Aspekte des Darstellungselements ein eigenes Attribut benutzt, um diesen Aspekt in einer geräteunabhängigen Art und Weise zu spezifizieren. Die erste wird als Bündelspezifikation und die letzte als individuelle Spezifikation bezeichnet.

Zu den Arbeitsplatzattributen gehören die konkreten Beschreibungen an einem graphischen Arbeitsplatz. Sie werden über Indizes angesprochen, wenn die Bündelspezifikation für nichtgeometrischen Aspekte benutzt wird. Beispielsweise enthalten die Beschreibungen (oder Bündel) für das Polygon jeweils Linientyp, Linienbreitefaktor und Farbindex. Sie spezifizieren auch die Farb- und Mustertabellen und die Steuerung der Aktualisierung von Bildänderungen. Arbeitsplatzattribute können dynamisch gesetzt werden.

Die Darstellung der Segmente wird über Segmentattribute gesteuert, die Segmenttransformation, Sichtbarkeit, Hervorheben, Segmentpriorität und

Ansprechbarkeit sind. Diese können auch dynamisch gesetzt werden. Segmentattribute können die Grundlage für Rückkopplungen während der Manipulation (z.B. Hervorheben) sein.

Die Attribute, die die Bedienung der logischen Eingabegeräte steuern, können entweder bei der Initialisierung oder als Teil der Einstellung von Eingabegeräten, abhängig von den Attributen, angegeben werden. Während der Initialisierung kann ein Anfangswert, Aufforderungs- und Echoverfahren und ein Bereich auf dem Bildschirm für Echos bezeichnet werden. Ein Datensatz kann zusätzlich die gerätespezifischen Attribute liefern. Während der Einstellung der Eingabegeräte kann der Betriebsmodus gewählt und das Echo ein- oder ausgeschaltet werden. Der Betriebsmodus eines logischen Eingabegerätes spezifiziert, wer (Bediener oder Anwendungsprogramm) die Initiative hat: Abfrageeingabe wird direkt vom Anwendungsprogramm erhalten; Anforderungseingabe wird vom Bediener als direkte Antwort auf das Anwendungsprogramm erzeugt; Ereigniseingabe wird asynchron durch den Bediener erzeugt und in einer Warteschlange für das Anwendungsprogramm gesammelt.

GKS kann zur Laufzeit einen von fünf verschiedenen Betriebszustände einnehmen. Jedem Zustand wird ein Satz von GKS-Funktionen, die in diesem Zustand erlaubt sind, und ein Satz von Zustandsvariablen zugeordnet. Die Konzeption der Betriebszustände und der Zustandsvariablen erlaubt eine korrekte Beschreibung der Initialisierung (z.B. bei ÖFFNE ARBEITSPLATZ) und der Auswirkung verschiedener Funktionen hinsichtlich der Erhaltung der Geräteunabhängigkeit. Ein besonderer Satz von Funktionen mit der Bezeichnung Erfragefunktionen ist in allen Zuständen erlaubt. Sie erlauben nur einen Lesezugriff zu den Zustandslisten. Auf diese Weise kann nützliche Information beim Auftreten von Fehlern geliefert werden. Andere Erfragefunktionen erlauben nur einen Lesezugriff zu den Arbeitsplatzbeschreibungen, damit das Anwendungsprogramm den besonderen Fähigkeiten des graphischen Arbeitsplatzes angepaßt werden kann. Erfragefunktionen verursachen niemals Fehler. Dafür geben sie Information zurück, die spezifiziert, ob das Erfragen gültig war.

GKS liefert eine Schnittstelle zu einem System zum Erstellen graphischer Information mit dem Zweck der externen Langzeitspeicherung und des Datenaustausches. Diese Schnittstelle bilden der Ausgabearbeitsplatz GKS-Bilddatei, der die Bilddatei beschreibt (die sequentiell ist) und der Eingabearbeitsplatz GKS-Bilddatei, der von der Bilddatei liest. Zusätzlich zu den normalen Funktionen für die Ausgabe zu graphischen Arbeitsplätzen akzeptiert der Ausgabearbeitsplatz GKS-Bilddatei Datensätze mit nichtgraphischer Information. Eingabe von der Bilddatei wird durch Lese- und Interpretierfunktionen gesteuert, die dieselbe Wirkung haben, als ob die entsprechenden Funktionen direkt vom Anwendungsprogramm aufgerufen worden wären.

2.3 GKS-Leistungsstufen

GKS wurde für den Einsatz in einem breiten Anwendungsspektrum, von der Zeichnung statischer Bilder bis zu Darstellungen dynamischer Vorgänge und Echtzeitinteraktionen, entwickelt. Außerdem fehlen bei vielen graphischen Geräten Fähigkeiten (wie z.B. Picken), deren Simulation mit Hilfe von Software einen beträchtlichen Aufwand erfordern würde. Es ist daher wünschenswert, GKS-Implementierungen zuzulassen, die nicht alle Funktionsfähigkeiten enthalten, die in dieser Norm festgelegt werden.

Die funktionalen Fähigkeiten vom GKS können in folgende Hauptgebiete eingeteilt werden:

- Ausgabe (minimale Leistung, vollständige Leistung);
- Eingabe (keine Eingabe, Anforderungseingabe, vollständige Eingabe);
- Anzahl der graphischen Arbeitsplätze (ein Arbeitsplatz, mehrere Arbeitsplätze);
- Attribute (nur vordefinierte Bündel möglich, vollständiges Bündelkonzept);
- Segmentierung (keine, einfache Segmentierung (ohne den arbeitsplatzunabhängigen Segmentspeicher), vollständige Segmentierung).

Würden beliebige Kombinationen von Fähigkeiten als gültige GKS-Implementierungen zugelassen, so würde sich eine fast unbegrenzte Anzahl von unterschiedlichen Dialekten der Norm ergeben und die Übertragbarkeit von Programmen, eines der Hauptziele dieser Norm, würde nicht erreicht werden. Deshalb sind neun gültige Leistungsstufen im GKS definiert, um die gebräuchlichsten Gerätetypen und Anwendungen anzusprechen. Jede GKS-Implementierung stellt genau die Funktionen einer Leistungsstufe zur Verfügung. Eine GKS-Implementierung ist ungültig, wenn sie zwischen zwei definierten Leistungsstufen liegt.

Die Struktur der Leistungsstufen enthält zwei unabhängige Achsen: Eingabe und 'alle weiteren Funktionen', zusammengefaßt als Ausgabe.

Die Achse der Leistungsstufe Ausgabe besitzt die drei Möglichkeiten:

0: Minimale Ausgabe,
1: einfache Segmentierung mit vollständiger Ausgabe,
2: arbeitsplatzunabhängiger Segmentspeicher.

Die Achse der Leistungsstufe Eingabe besitzt die drei Möglichkeiten:

a: keine Eingabe,
b: Anforderungseingabe,
c: vollständige Eingabe.

Die Fähigkeiten, die es in den Leistungsstufen gibt, sind folgende:

Ausgabeleistungsstufe 0: Minimale Ausgabe

- einfache Steuerungsfunktionen,
- alle Darstellungselemente sind mindestens mit minimaler Ausführung verfügbar,

- nur vordefinierte Bündel (keine Modifikation von Bündeln),
- Modifikation der Farbbeschreibung möglich,
- nur ein graphischer Arbeitsplatz mit Ausgabefähigkeiten steht gleichzeitig zur Verfügung,
- Bilddateien müssen nicht unterstützt werden. Werden sie unterstützt, so muß sowohl Eingabe als auch Ausgabe verfügbar sein. Werden sie nicht unterstützt, so werden entsprechende Fehlermeldungen generiert.
- mehrere Normalisierungstransformationen, wobei ein System mit Normalisierungstransformation 0 und nur einer setzbaren Normalisierungstransformation zulässig ist,
- geeignete einfache Erfragefunktionen,
- Rücklesen von Pixeln vorgesehen (Geräte ohne Pixelspeicher können 'Keine Verarbeitung' melden.)

Ausgabeleistungsstufe 1: Einfache Segmentierung mit voller Ausgabe

- alle Fähigkeiten der Ausgabeleistungsstufe 0,
- vollständige Arbeitsplatzsteuerung,
- vollständige Ausgabefähigkeiten,
- vollständiges Bündelkonzept,
- Konzept der mehrfachen graphischen Arbeitsplätze,
- Bilddateifähigkeiten werden verlangt,
- mehrere setzbare Normalisierungstransformationen,
- einfache Segmentierung (kein arbeitsplatzunabhängiger Segmentspeicher),
- geeignete Erfragefunktion.

Ausgabeleistungsstufe 2: Arbeitsplatzunabhängiger Segmentspeicher

- alle Fähigkeiten der Ausgabeleistungsstufe 1
- arbeitsplatzunabhängiger Segmentspeicher

Eingabeleistungsstufe a: keine Eingabe

- keine Fähigkeiten

Eingabeleistungsstufe b : Anforderungseingabe

- Funktionen zur Initialisierung von Eingabegeräten und zum Setzen der Modi,
- ANFORDERUNGS-Funktionen an allen entsprechenden Geräten,
- PICKER wird dann und nur dann unterstützt, wenn diese Leistungsstufe mit Fähigkeiten der Ausgabeleistungsstufe 1 kombiniert wird,
- Funktion zum Setzen der Eingabepriorität des Darstellungsfeldes.

Eingabeleistungsstufe c : vollständige Eingabe

- vollständige Fähigkeiten der Eingabeleistungsstufe b
- ABFRAGE- und EREIGNIS-Modus für die Eingabe.

Tabelle 1 gibt eine kurze Übersicht über die Funktionalität jeder gültigen

GKS-Leistungsstufe. Jedes Feld enthält nur die Funktionen, die zu denen der vorhergehenden Felder in der gleichen Reihe und Spalte hinzukommen.

Tabelle 1. Konzeption der GKS-Leistungsstufen

Ausgabe-leistungs-stufe	Eingabeleistungsstufe		
	a	b	c
0	Keine Eingabe, einfache Steuerungsfunktionen, nur vordefinierte Bündel, mehrere Normalisierungstransformationen möglich, aber nur eine braucht setzbar zu sein, alle Darstellungselemente, Bilddatei wahlweise	Anforderungseingabe, Setzen der Modi und Initialisierung der log. Eingabegeräte, kein PICKER und keine Eingabepriorität des Darstellungsfeldes	Abfrage- und Ereignis-eingabe, kein PICKER
1	Vollständige Ausgabefähigkeiten, vollständiges Bündelkonzept, mehrere Arbeitsplätze, einfache Segmentierung (alles außer dem arbeitsplatz-unabhängigen Segmentspeicher), Bilddatei gefordert	Anforderungseingabe, Setzen der Modi und Initialisieren des PICKER	Abfrage- und Ereignis-eingabe für PICKER
2	Arbeitsplatzunabhängiger Segmentspeicher		

Unterschiede in der Zahl von Möglichkeiten, die aus der Menge ausdrücklich definierter und geforderter Fähigkeiten gefordert werden, sind in der obigen Zusammenfassung der Leistungsstufen nicht enthalten.

3. Zusammenstellung der GKS-Funktionen

3.1 Steuerungsfunktionen

OEFFNE GKS (OPEN GKS)
    Beginne Arbeiten mit GKS.
SCHLIESSE GKS (CLOSE GKS)
    Beende Arbeiten mit GKS.
OEFFNE ARBEITSPLATZ (OPEN WORKSTATION)
    Stelle eine Verbindung zwischen einem graphischen Arbeitsplatz und GKS her.
SCHLIESSE ARBEITSPLATZ (CLOSE WORKSTATION)
    Hebe die Verbindung zwischen GKS und einem graphischen Arbeitsplatz auf.
AKTIVIERE ARBEITSPLATZ (ACTIVATE WORKSTATION)
    Leite die Ausgabe ab sofort zu diesem graphischen Arbeitsplatz.
DEAKTIVIERE ARBEITSPLATZ (DEACTIVATE WORKSTATION)
    Leite die Ausgabe nicht laenger zu diesem graphischen Arbeitsplatz.
LOESCHE ARBEITSPLATZ (CLEAR WORKSTATION)
    Fuehre alle aufgeschobenen Aktionen aus und loesche den Darstellungsbereich des graphischen Arbeitsplatzes.
ZEICHNE ALLE SEGMENTE AM ARBEITSPLATZ NEU
(REDRAW ALL SEGMENTS ON WORKSTATION)
    Zeichne alle sichtbaren Segmente eines graphischen Arbeitsplatzes neu.
AKTUALISIERE ARBEITSPLATZ (UPDATE WORKSTATION)
    Fuehre alle aufgeschobenen Aktionen aus und zeichne alle sichtbaren fuer den graphischen Arbeitsplatz gespeicherten Segmente neu.
SETZE AKTUALISIERUNGSZUSTAND (SET DEFERRAL STATE)
    Setze den Aktualisierungszustand fuer einen graphischen Arbeitsplatz.
NACHRICHT (MESSAGE)
    Sende eine Nachricht an einen graphischen Arbeitsplatz.
FLUCHTFUNKTION (ESCAPE)
    Eine Funktion zum genormten Aufruf nicht genormter Faehigkeiten.

3.2 Ausgabefunktionen

POLYGON (POLYLINE)
Erzeuge ein Polygon, das durch Punkte in WK definiert ist.

POLYMARKE (POLYMARKER)
Erzeuge Marken der angegebenen Art an Positionen in WK.

TEXT (TEXT)
Erzeuge eine Zeichenfolge an der angegebenen Position in WK.

FUELLGEBIET (FILL AREA)
Erzeuge ein Polygon, das mit einer Farbe, einer Schraffur oder einem Muster ausgefuellt oder leer ist.

ZELLMATRIX (CELL ARRAY)
Bilde eine angegebene Matrix von Farbindizes auf die Darstellungsflaeche ab.

VERALLGEMEINERTES DARSTELLUNGSELEMENT (GENERALIZED DRAWING PRIMITIVE)
Erzeuge ein verallgemeinertes Darstellungselement, das durch eine Folge von Punkten in WK und einen Datensatz definiert ist.

3.3 Ausgabeattribute

3.3.1 Arbeitsplatzunabhaengige Darstellungsattribute

SETZE POLYGONINDEX (SET POLYLINE INDEX)
Waehle einen Buendelindex fuer Polygone aus.

SETZE LINIENTYP (SET LINETYPE)
Setze den Linientyp, der benutzt wird, wenn der zugehoerige Aspektanzeiger auf INDIVIDUELL gesetzt ist.

SETZE LINIENBREITEFAKTOR (SET LINEWIDTH SCALE FACTOR)
Setze den Linienbreitefaktor, der benutzt wird, wenn der zugehoerige Aspektanzeiger auf INDIVIDUELL gesetzt ist.

SETZE POLYGONFARBINDEX (SET POLYLINE COLOUR INDEX)
Setze den Polygonfarbindex, der benutzt wird, wenn der zugehoerige Aspektanzeiger auf INDIVIDUELL gesetzt ist.

SETZE POLYMARKENINDEX (SET POLYMARKER INDEX)
Waehle einen Buendelindex fuer Polymarken aus.

SETZE MARKENTYP (SET MARKER TYPE)
Setze den Markentyp, der benutzt wird, wenn der zugehoerige Aspektanzeiger auf INDIVIDUELL gesetzt ist.

SETZE MARKENVERGROESSERUNGSFAKTOR (SET MARKER SIZE SCALE FACTOR)
Setze den Markenvergroesserungsfaktor, der benutzt wird, wenn der zugehoerige Aspektanzeiger auf INDIVIDUELL gesetzt ist.

SETZE POLYMARKENFARBINDEX (SET POLYMARKER COLOUR INDEX)
Setze den Polymarkenfarbindex, der benutzt wird, wenn der zugehoerige Aspektanzeiger auf INDIVIDUELL gesetzt ist.

SETZE TEXTINDEX (SET TEXT INDEX)
Waehle einen Buendelindex fuer Texte aus.

SETZE SCHRIFTART UND -QUALITAET (SET TEXT FONT AND PRECISION)
Setze die Schriftart und die Schriftqualitaet, die benutzt werden, wenn der zugehoerige Aspektanzeiger auf INDIVIDUELL gesetzt ist.

SETZE ZEICHENBREITEFAKTOR (SET CHARACTER EXPANSION FACTOR)
Setze den Zeichenbreitefaktor, der benutzt wird, wenn der zugehoerige Aspektanzeiger auf INDIVIDUELL gesetzt ist.

SETZE ZEICHENABSTAND (SET CHARACTER SPACING)
Setze den Zeichenabstand, der benutzt wird, wenn der zugehoerige Aspektanzeiger auf INDIVIDUELL gesetzt ist.

SETZE TEXTFARBINDEX (SET TEXT COLOUR INDEX)
Setze den Textfarbindex, der benutzt wird, wenn der zugehoerige Aspektanzeiger auf INDIVIDUELL gesetzt ist.

SETZE ZEICHENHOEHE (SET CHARACTER HEIGHT)
Setze die Zeichenhoehe.

SETZE ZEICHENAUFWAERTSVEKTOR (SET CHARACTER UP VECTOR)
Setze den Zeichenaufwaertsvektor.

SETZE SCHREIBRICHTUNG (SET TEXT PATH)
Setze die Schreibrichtung LINKS, RECHTS, OBEN oder UNTEN.

SETZE TEXTAUSRICHTUNG (SET TEXT ALIGNMENT)
Setze die horizontale und vertikale Ausrichtung der Zeichenfolge.

SETZE FUELLGEBIETSINDEX (SET FILL AREA INDEX)
Waehle einen Buendelindex fuer Fuellgebiete aus.

SETZE FUELLGEBIETSAUSFUELLUNG (SET FILL AREA INTERIOR STYLE)
Setze die Fuellgebietsausfuellung, die benutzt wird, wenn der zugehoerige Aspektanzeiger auf INDIVIDUELL gesetzt ist.

SETZE FUELLGEBIETSAUSFUELLUNGSINDEX (SET FILL AREA STYLE INDEX)
Setze den Fuellgebietsausfuellungsindex, der benutzt wird, wenn der zugehoerige Aspektanzeiger auf INDIVIDUELL gesetzt ist.

SETZE FUELLGEBIETSFARBINDEX (SET FILL AREA COLOUR INDEX)
Setze den Fuellgebietsfarbindex, der benutzt wird, wenn der zugehoerige Aspektanzeiger auf INDIVIDUELL gesetzt ist.

SETZE MUSTERGROESSE (SET PATTERN SIZE)
Setze die Mustergroesse, die fuer die Darstellung der Fuellgebiete mit der Ausfuellung MUSTER benutzt wird.

SETZE MUSTERREFERENZPUNKT (SET PATTERN REFERENCE POINT)
Setze den Musterreferenzpunkt, der fuer die Darstellung der Fuellgebiete mit der Ausfuellung MUSTER benutzt wird.

SETZE ASPEKTANZEIGER (SET ASPECT SOURCE FLAGS)
Lege fuer jeden nichtgeometrischen Aspekt fest, ob sein Wert aus dem zugehoerigen individuellen Attribut oder aus dem zugehoerigen Attributsbuendel des graphischen Arbeitsplatzes entnommen wird.

SETZE PICKERKENNZEICHNUNG (SET PICK IDENTIFIER)
Setze die Pickerkennzeichnung.

3.3.2 Arbeitsplatzattribute (Beschreibungen)

SETZE POLYGONBESCHREIBUNG (SET POLYLINE REPRESENTATION)
Definiere die Beschreibung von Polygonen an einem graphischen Arbeitsplatz.

SETZE POLYMARKENBESCHREIBUNG (SET POLYMARKER REPRESENTATION)
Definiere die Beschreibung von Polymarken an einem graphischen Arbeitsplatz.

SETZE TEXTBESCHREIBUNG (SET TEXT REPRESENTATION)
Definiere die Beschreibung von Texten an einem graphischen Arbeitsplatz.

SETZE FUELLGEBIETSBESCHREIBUNG (SET FILL AREA REPRESENTATION)
Definiere die Beschreibung von Fuellgebieten an einem graphischen Arbeitsplatz.

SETZE MUSTERBESCHREIBUNG (SET PATTERN REPRESENTATION)
Definiere das Muster, das zu einem Musterindex (d.h. einem Fuellgebietsausfuellungsindex) an einem graphischen Arbeitsplatz gehoert.

SETZE FARBBESCHREIBUNG (SET COLOUR REPRESENTATION)
Definiere einen Eintrag (Farbindex) in der Farbtabelle eines graphischen Arbeitsplatzes, dem eine Farbe zugeordnet wird.

3.4 Transformationsfunktionen

3.4.1 Normalisierungstransformation

SETZE FENSTER (SET WINDOW)
Setze das Fenster einer Normalisierungstransformation in WK.

SETZE DARSTELLUNGSFELD (SET VIEWPORT)
Setze das Darstellungsfeld einer Normalisierungstransformation in NK.

SETZE DARSTELLUNGSFELD-EINGABEPRIORITAET (SET VIEWPORT INPUT PRIORITY)
Setze die Eingabeprioritaet des Darstellungsfeldes einer Normalisierungstransformation fuer die Lokalisiereingabe.

WAEHLE NORMALISIERUNGSTRANSFORMATION (SELECT NORMALIZATION TRANSFORMATION)
Waehle eine Normalisierungstransformation fuer die Ausgabe aus.

SETZE KLIPPANZEIGER (SET CLIPPING INDICATOR)
Schalte das Klippen fuer die Normalisierungstransformation an oder aus.

3.4.2 Geraetetransformation

SETZE GERAETEFENSTER (SET WORKSTATION WINDOW)
Setze Geraetefenster in NK.

SETZE GERAETEDARSTELLUNGSFELD (SET WORKSTATION VIEWPORT)
Setze Geraetedarstellungsfeld in GK.

3.5 Segmentfunktionen

3.5.1 Funktionen zur Segmentmanipulation

ERZEUGE SEGMENT (CREATE SEGMENT)
Erzeuge ein Segment und beginne mit dem Aufbau.
SCHLIESSE SEGMENT (CLOSE SEGMENT)
Der Segmentaufbau ist beendet.
BENENNE SEGMENT UM (RENAME SEGMENT)
Aendere den Namen eines Segments.
LOESCHE SEGMENT (DELETE SEGMENT)
Loesche ein Segment.
LOESCHE SEGMENT AM ARBEITSPLATZ (DELETE SEGMENT FROM WORKSTATION)
Loesche ein Segment an einem bestimmten graphischen Arbeitsplatz.
ORDNE SEGMENT DEM ARBEITSPLATZ ZU (ASSOCIATE SEGMENT WITH WORKSTATION)
Ordne ein Segment, das im arbeitsplatzunabhaengigen Segmentspeicher ist, einem offenen graphischen Arbeitsplatz zu.
KOPIERE SEGMENT ZUM ARBEITSPLATZ (COPY SEGMENT TO WORKSTATION)
Kopiere die Darstellungselemente eines Segments, das im arbeitsplatzunabhaengigen Segmentspeicher ist, und sende sie an einen graphischen Arbeitsplatz.
FUEGE SEGMENT EIN (INSERT SEGMENT)
Kopiere ein Segment, das im arbeitsplatzunabhaengigen Segmentspeicher ist, und fuege es (nachdem Segmenttransformation und Einfuegetransformation ausgefuehrt wurden) in das offene Segment oder in den Strom von Darstellungselementen ausserhalb von Segmenten ein.

3.5.2 Segmentattribute

SETZE SEGMENTTRANSFORMATION (SET SEGMENT TRANSFORMATION)
Setze die Segmenttransformation fuer ein Segment.
SETZE SICHTBARKEIT (SET VISIBILITY)
Markiere ein Segment als sichtbar oder unsichtbar.
SETZE HERVORHEBEN (SET HIGHLIGHTING)
Markiere ein Segment als normal oder hervorgehoben.
SETZE SEGMENTPRIORITAET (SET SEGMENT PRIORITY)
Setze die Segmentprioritaet.
SETZE ANSPRECHBARKEIT (SET DETECTABILITY)
Markiere ein Segment als nicht ansprechbar oder ansprechbar.

3.6 Eingabefunktionen

3.6.1 Initialisieren von Eingabegeraeten

INITIALISIERE LOKALISIERER (INITIALISE LOCATOR)
    Initialisiere einen Lokalisierer.
INITIALISIERE LINIENGEBER (INITIALISE STROKE)
    Initialisiere einen Liniengeber.
INITIALISIERE WERTGEBER (INITIALISE VALUATOR)
    Initialisiere einen Wertgeber.
INITIALISIERE AUSWAEHLER (INITIALISE CHOICE)
    Initialisiere einen Auswaehler.
INITIALISIERE PICKER (INITIALISE PICK)
    Initialisiere einen Picker.
INITIALISIERE TEXTGEBER (INITIALISE STRING)
    Initialisiere einen Textgeber .

3.6.2 Setzen des Betriebsmodus von Eingabegeraeten

SETZE LOKALISIERERMODUS (SET LOCATOR MODE)
    Setze den Betriebsmodus eines Lokalisierers.
SETZE LINIENGEBERMODUS (SET STROKE MODE)
    Setze den Betriebsmodus eines Liniengebers.
SETZE WERTGEBERMODUS (SET VALUATOR MODE)
    Setze den Betriebsmodus eines Wertgebers.
SETZE AUSWAEHLERMODUS (SET CHOICE MODE)
    Setze den Betriebsmodus eines Auswaehlers.
SETZE PICKERMODUS (SET PICK MODE)
    Setze den Betriebsmodus eines Pickers.
SETZE TEXTGEBERMODUS (SET STRING MODE)
    Setze den Betriebsmodus eines Textgebers .

3.6.3 Anforderungseingabefunktionen

FORDERE LOKALISIERER AN (REQUEST LOCATOR)
    Fordere eine Position in WK und die Nummer einer
    Normalisierungstransformation von einem Lokalisierer an.
FORDERE LINIENGEBER AN (REQUEST STROKE)
    Fordere eine Punktfolge in WK und die Nummer einer
    Normalisierungstransformation von einem Liniengeber an.
FORDERE WERTGEBER AN (REQUEST VALUATOR)
    Fordere einen reellen Wert von einem Wertgeber an.
FORDERE AUSWAEHLER AN (REQUEST CHOICE)
    Fordere einen positiven ganzzahligen Wert, der eine Auswahl aus
    einer Reihe von Alternativen beschreibt, von einem Auswaehler an.
FORDERE PICKER AN (REQUEST PICK)
    Fordere einen Segmentnamen, eine Pickerkennzeichnung und einen
    Pickerzustand von einem Picker an.
FORDERE TEXTGEBER AN (REQUEST STRING)
    Fordere eine Zeichenfolge von einem Textgeber an.

3.6.4 Abfrageeingabefunktionen

Die aktuelle Einstellung eines logischen Eingabegeraetes wird, ohne auf eine Bedieneraktion zu warten, abgefragt und zurueckgegeben.

FRAGE LOKALISIERER AB (SAMPLE LOCATOR)
Frage einen Lokalisierer ab und liefere eine Position in WK und die Nummer einer Normalisierungstransformation zurueck.

FRAGE LINIENGEBER AB (SAMPLE STROKE)
Frage einen Liniengeber ab und liefere eine Punktfolge in WK und die Nummer einer Normalisierungstransformation zurueck.

FRAGE WERTGEBER AB (SAMPLE VALUATOR)
Frage einen Wertgeber ab und liefere einen reellen Wert zurueck.

FRAGE AUSWAEHLER AB (SAMPLE CHOICE)
Frage einen Auswaehler ab und liefere einen positiven ganzzahligen Wert, der eine Auswahl aus einer Reihe von Alternativen beschreibt, zurueck.

FRAGE PICKER AB (SAMPLE PICK)
Frage einen Picker ab und liefere einen Segmentnamen, eine Pickerkennzeichnung und einen Pickerstatus zurueck.

FRAGE TEXTGEBER AB (SAMPLE STRING)
Frage einen Textgeber ab und liefere eine Zeichenfolge zurueck.

3.6.5 Ereigniseingabefunktionen

Eingabesaetze werden in eine vom GKS verwaltete Eingabewarteschlange eingereiht und koennen von dort durch das Anwendungsprogramm erhalten werden.

ERWARTE ERGEIGNIS (AWAIT EVENT)
Falls die Warteschlange leer ist, dann warte auf einen Eingabesatz bis die angegebene Wartezeit verstrichen ist. Lies die Arbeitsplatzkennzeichnung, die Eingabeklasse und die Geraetenummer des aeltesten Eintrages in der Warteschlange und mache die Werte fuer die Funktionen HOLE <Eingabeklasse> verfuegbar.

STREICHE GERAETEEREIGNISSE (FLUSH DEVICE EVENTS)
Loesche alle Ereignisse des angegebenen logischen Eingabegeraetes in der Eingabewarteschlange.

HOLE LOKALISIERER (GET LOCATOR)
Uebertrage eine Position in WK und die Nummer einer Normalisierungstransformation von dem aktuellen Ereignisbericht zum Anwendungsprogramm.

HOLE LINIENGEBER (GET STROKE)
Uebertrage eine Punktefolge in WK und die Nummer einer Normalisierungstransformation von dem aktuellen Ereignisbericht zum Anwendungsprogramm.

HOLE WERTGEBER (GET VALUATOR)
Uebertrage einen reellen Wert von dem aktuellen Ereignisbericht zum Anwendungsprogramm.

HOLE AUSWAEHLER (GET CHOICE)
Uebertrage einen positiven ganzzahligen Wert, der eine Auswahl aus einer Reihe von Alternativen beschreibt, vom aktuellen Ereignisbericht zum Anwendungsprogramm.

HOLE PICKER (GET PICK)
Uebertrage einen Segmentnamen, eine Pickerkennzeichnung und einen Pickerzustand vom aktuellen Ereignisbericht zum Anwendungsprogramm.

HOLE TEXTGEBER (GET STRING)
Uebertrage eine Zeichenfolge vom aktuellen Ereignisbericht zum Anwendungsprogramm.

3.7 Bilddateifunktionen

SCHREIBE SATZ IN DIE BILDDATEI (WRITE ITEM TO GKSM)
Schreibe nichtgraphische Daten vom Anwendungsprogramm in die GKS-Bilddatei.

HOLE SATZART VON DER BILDDATEI (GET ITEM TYPE FROM GKSM)
Liefere die Art und die Laenge des naechsten Satzes der GKS-Bilddatei an das Anwendungsprogramm zurueck.

LIES SATZ VON DER BILDDATEI (READ ITEM FROM GKSM)
Liefere den naechsten Satz der GKS-Bilddatei an das Anwendungsprogramm zurueck (graphischer Satz oder Benutzersatz).

INTERPRETIERE SATZ (INTERPRET ITEM)
Interpretiere den von der Funktion LIES SATZ VON DER BILDDATEI eingelesenen Einsatzpuffer. Die Interpretation fuehrt zu entsprechenden Aenderungen des GKS-Zustandes und erzeugt zugehoerige graphische Ausgabe, so wie es durch die Spezifikation der Bilddatei festgelegt ist.

3.8 Erfragefunktionen

Im GKS sind ungefaehr 60 verschiedene Erfragefunktionen definiert. Alle Variablen, die in einer der vorhandenen Zustandslisten oder Beschreibungstabellen enthalten sind, koennen jederzeit erfragt werden, solange GKS offen ist.

Um das Verbinden von Zeichenfolgen zu erlauben, steht eine Erfragefunktion fuer die Textausdehnung zur Verfuegung.
Von Rastergeraeten kann die Groesse und Farbe der Pixel erfragt werden.

Der Betriebszustand des GKS kann auch dann abgefragt werden, wenn das GKS geschlossen ist.

3.9 Hilfsfunktionen

BILDE TRANSFORMATIONSMATRIX (EVALUATE TRANSFORMATION MATRIX)
Die durch Skalierungs-, Rotations- und Translationsparameter angegebene Transformation wird ausgewertet und das Ergebnis in der Transformationsmatrix fuer die Ausgabe bereitgestellt.

AKKUMULIERE TRANSFORMATIONSMATRIX (ACCUMULATE TRANSFORMATION MATRIX)
Die durch Skalierungs-, Rotations- und Translationsparameter angegebene Transformation wird ausgewertet. Die Transformationsmatrix fuer die Ausgabe wird durch Multiplikation der vorhandenen Transformationsmatrix mit der ausgewerteten Matrix berechnet.

3.10 Fehlerbehandlung

NOTABSCHLUSS GKS (EMERGENCY CLOSE GKS)
Diese Funktion versucht GKS im Falle eines Fehlers zu schliessen und dabei soviel Information wie moeglich zu retten.
FEHLERBEHANDLUNG (ERROR HANDLING PROCEDURE)
Eine Prozedur, die von GKS im Falle eines Fehlers aufgerufen wird. Sie kann auch vom Anwender bereitgestellt werden.
FEHLERPROTOKOLL (ERROR LOGGING PROCEDURE)
Eine Prozedur, die durch die GKS-Fehlerbehandlungsprozedur aufgerufen wird. Sie gibt eine Fehlermeldung und eine Kennzeichnung der GKS-Funktion auf eine Fehlerdatei aus.

4. GKSYS - FORTRAN 77 IMPLEMENTIERUNG

GKS definiert den sprachunabhängigen Kern eines graphischen Systems. Zur Integration in eine Programmiersprache muß GKS in eine sprachabhängige Schicht, die den besonderen Eigenheiten dieser Sprache gehorcht, eingebettet werden.

GKSYS ist die Produktbezeichnung der GKS Implementierung der Firma SYSGRAPH, Wien. Diese Implementierung ist vollständig in FORTRAN 77 und, um ein Höchstmaß an Portabilität zu erreichen, mit wenigen Ausnahmen maschinenunabhängig programmiert.

Diese Implementation steht ab sofort unseren Kunden an der VAX 780 zur Verfügung. Derzeit ist die Leistungsstufe 0.b mit Bilddatei-Unterstützung installiert. Leistungsstufe 2.b ist erhältlich, 2.c wird in nächster Zeit realisiert.

4.1 Benützungshinweise

- Linken mit GKS:

  durch Verwendung des Commandfiles GLINK

  $@APPL$GKS:GLINK "main,sub1,..."

Um auf ein Graphikgerät ausgeben zu können, muß der logische Name GKSDEVnn auf die zugehörige Terminal-Leitung zugewiesen sein. Die beim Öffnen eines Arbeitsplatzes anzugebende Verbindungsnummer ist dann die zweistellige Zahl nn.

zB: $ assign TT GKSDEV01 → Verbindungsnummer: 1

- Verfügbare Arbeitsplatztreiber:

  - Tektronix 4010 Graphikterminal
    Arbeitsplatz Typ: 4010

  - Tektronix 4014 Graphikterminal
    Arbeitsplatz Typ: 4014

  - Calcomp 960 Plotter
    Arbeitsplatz Typ: -10

  - Bilddatei Output und -Input
    Arbeitsplatz Typ: 0 und 1
    Verbindungs-Nummer: 1,2,3,...

    Diese Nummern sind dann bis zum Schließen des Bilddatei-Arbeitsplatzes nicht mehr als File-Units verfügbar. Die resultierenden Filenamen heißen GKSM001.DAT, GKSM002.DAT, usw.

    Gerade Nummern ergeben formatierte Dateien, ungerade Nummern unformatierte Dateien.

  - In Kürze wird auch ein Treiber für den HP 7221 Plotter zur Verfügung stehen.
    Arbeitsplatz Typ: 7221

- Es steht ein interaktiver GKS-Interpreter zur Verfügung, mit dem im Dialog jede der GKS Funktionen der installierten Leistungsstufe aufgerufen und die Effekte studiert werden können, ohne ein ganzes Testprogramm erstellen, compilieren und linken zu müssen.

  $ RUN APPL$GKS:GKSDEM

  Nach Eingabe des gewünschten GKS-Namens werden die Eingabeparameter der entsprechenden Routinen interaktiv erfragt und nach Durchführung des Aufrufs gegebenenfalls die Ausgabeparameter am aufrufenden Terminal ausgegeben.

- Demonstrationsprogramm:
  Das File APPL$GKS:DEM1.FOR enthält ein Demonstrationsprogramm, das einige Möglichkeiten von GKSYS darstellt. Der Programmtext kann als Beispiel genommen werden, wie ein einfaches GKS-Programm aussieht.

4.2 Quellangaben

Das 'GKSYS Referenz-Manual' und eine 'Einführung in GKS' liegen im Rechnervorraum zur Einsicht auf. Kopien davon können zum Preis von S 140,- bzw. S 100,- im Sekretariat erworben werden.

Für weitere Fragen stehen Ihnen gerne Rudolf Sedlaczek (Karlsplatz 13, Stiege 7, Kl.3443) oder Jadzia Rogl (Gusshausstr. 27, Kl.3612) zur Verfügung.

VAX-11/730 FÜR DIE NACHRICHTENTECHNIK

M. Paul

Im letzten "Feedback" konnte berichtet werden, daß dem BMWF mit Schreiben vom 2. Mai 1984 die Anbotsbewertung und -reihung übermittelt wurde. Die Vergabevorschlagskommission des BMWF schloß sich unserem Reihungsvorschlag an und am 2. bzw. 17. Juli 1984 wurde der Kaufvertrag mit der Fa. Digital Equipment Corp. Ges.m.b.H. abgeschlossen. Eine Installation und Betriebsaufnahme vor Semesterbeginn war damit leider nicht mehr möglich, da die Lieferzeit von 3 Monaten letzten Endes doch nicht unterschritten werden konnte. Der zunächst sehr optimistisch mit 23. Okt. 1984 angesetzte Abnahmetag konnte wegen Lieferschwierigkeiten bei Teilgeräten der VAX nicht gehalten werden und mußte schließlich auf 20. Nov. verschoben werden.

Nach Anlieferung aller Geräte und Behebung einiger kleinerer Liefermängel stand dann die Rechenanlage gebrauchsfertig aufgestellt ab 12. Nov. 1984 zur Vorbereitung des Abnahmetests zur Verfügung. Der Funktionstest wie auch der Leistungstest konnten am 20. Nov. 1984 problemlos abgewickelt werden. Der Leistungstest zeigte sogar ein um 5 % besseres Ergebnis als erwartet. In der Zeit von 12. Nov. bis 12. Dez. 1984 wurde weiters noch ein 30-tägiger Dauertest abgewickelt, der ebenfalls positiv verlief.

Die Rechenanlage VAX-11/730 des Institutes für Nachrichtentechnik umfaßt nunmehr:

1 VAX-11/730 mit 2 MByte Arbeitsspeicher
1 RUA81-Festplattenlaufwerk mit 456 MByte
1 TU80-Magnetbandstation, 1600 bpi, 25/100 ips
2 Magnetband-Kassettenlaufwerke, a 256 KByte
1 LA100-Matrixdrucker
2 VT101-Bildschirmterminals
1 VAX-11 FORTRAN-Compiler.

Damit steht dem Institut für Nachrichtentechnik ein sehr leistungsfähiges Arbeitsmittel zur Bewältigung der ursprünglich definierten Aufgabenstellung - eindimensionale digitale Signalverarbeitung sowie Zeit-Frequenzanalysen ("Feedback" Nr. 12, S.8) - zur Verfügung. Die Rechenanlage wird hinsichtlich Systemsoftware und Operating-Betriebsmittel vom EDV-Zentrum bedient, hinsichtlich Anwenderprogramme agiert das Institut für Nachrichtentechnik autonom.

DECUS INDEX

L. Tauer

DECUS Programme und Programmsammlungen gibt es wie Sand in der Wüste. Wir haben Ihnen daher die Möglichkeit geschaffen, Einblick in die an der PRA vorhandenen Symposien-Bänder zu nehmen.

Im Directory SYS$DECUS: auf VAX-11/780 finden Sie Informationen über die vorhandenen SIG-Tapes und ihre Inhalte. Es gliedert sich in 4 Teile:

1.) RSX SIG Bänder
2.) VMS SIG Bänder
3.) RT-11 SIG Bänder
4.) "Ready To Use" Programme, die an der 11/780 direkt verwendbar sind.

Diese Teile befinden sich auf eigenen Subdirectories. Die Kommandoprozedur SYS$DECUS:SETDECUS.COM setzt für Sie 4 lokale logische Namen (RSX$DECUS, VMS$DECUS, RT11$DECUS und EXE$DECUS) entsprechend der obigen Einteilung. In den jeweiligen Directories befinden sich Kurzinformationen über die jeweiligen Programmsammlungen bzw. Dokumentationen von installierter Software, soweit diese vorhanden sind.

Mit "HELP DECUS" bekommen Sie Informationen über die Gliederung der Inhaltsverzeichnisse und über die installierten Programme.

Die Bänder befinden sich im Rechenraum und werden auf Wunsch vom Operator zur Verfügung gestellt.

Wie kopiert man sich nun Files von den Bändern?

Die RSX und VMS Sammlungen sind mit BACKUP geschrieben und können daher auch wieder mit diesem Utility eingelesen werden. Z.B.:

```
$MOUNT MTA0:/FOREIGN
$BACKUP MTA0:savesetname/SELECT=[uic]*.* *
```

Hierbei ist "savesetname" entweder der Aufschrift auf dem Band oder dem entsprechenden *.INH File zu entnehmen. "uic" ist der UIC des/der gewünschten Programme(s).

Die Programme der RT-11 Sammlungen werden mit COPY eingelesen:

```
$MOUNT MTA0:tapelabel
$COPY MTA0:file *
```

wobei "tapelabel" die Programmsammlung darstellt und "file" ein Disk Image des Programmes darstellt. Dieses Image ist eine virtuelle Floppy Disk und kann unter dem RT-11 Emulator mittels JOAT attached werden.

"WORD" PROGRAMMBESCHREIBUNG

L. Tauer

Was ist WORD ?

Mit WORD können englische Texte auf korrekte Schreibweise überprüft werden. (Tippfehlersuche !)

Wie benutzt man WORD ?

Sie rufen die Kommandoprozedur SYS$DECUS:SETDECUS auf und starten das Programm mit "RUN EXE$DECUS:WORD".

Was kann WORD ?

- Sie können mit WORD ein File auf fehlerhafte Worte untersuchen

  Word? <infile> outfile

  Die spitzen Klammern "<" und ">" sind Teil der Syntax und daher anzugeben. (UNIX Notation !)

  Infile ist jenes File das untersucht werden soll,
  Outfile wird mit der Liste jener Worte beschrieben, die WORD in seinem Wörterbuch nicht findet.

- Sie können ein einzelnes Wort auf Richtigkeit überprüfen, indem Sie einfach dieses Wort eingeben.

  Word? Lunch

  Wird das Wort im Wörterbuch gefunden, so gibt WORD "Ok" aus, wenn nicht, wird "*NO*" ausgegeben.

- Sie können durch Eingabe eines Wortes, das "Wildcard Characters" enthält, eine Wortgruppe abfragen.

  Word? Li?e oder
  Word? Li...e

  Sie erhalten am Terminal eine Liste aller bekannten Worte, die Ihrer Spezifikation entsprechen, wobei das Fragezeichen für genau ein beliebiges Zeichen steht, die drei Punkte hingegen stehen für eine Zeichenfolge von 0 bis n Zeichen.
  Ein Wort darf NICHT mit einem Wildcard Character beginnen!

Wie beendet man Word ?

Entweder durch control/Z oder durch EXIT

  Word? ^Z
  Word? EXIT

Was kann WORD nicht ?

WORD ersetzt nicht das Korrekturlesen durch einen Kollegen, da es weder grammatikalische noch Punktuationsfehler findet. Ebenso bleibt verfehlte Wortwahl unentdeckt, solange die Schreibweise korrekt ist.

"SL 2000"-SOFTWARE FÜR MIKROELEKTRONIKSCHALTUNGEN

M. Höller
Institut für Allgemeine Elektrotechnik
und Elektronik der TU-Wien

Im Rahmen des auf der VAX-11/780 neu installierten Softwarepaketes "SL 2000" der Fa. Silvar Lisco, welches zum Design von Mikroelektronikschaltungen (Standardzellen und Gate-Arrays) gedacht ist, können unter anderem auch auf einfache Weise Logikpläne und Schaltpläne gezeichnet werden. Durch die Bedienung mit Menuefeldern und einem hierarchischen Aufbau der Datenbasis ist das Programm ebenso einfach zu bedienen wie zu erlernen und vielseitig einsetzbar. Neben der Ausgabe auf dem CALCOMP-Plotter ist auch das Extrahieren von Netzlisten zur weiteren Verarbeitung möglich. Auch Kunden, welche ihre Schaltpläne bisher auf der CALMA-Anlage gezeichnet haben, sollten von dieser Möglichkeit Gebrauch machen, da die SL 2000-Software speziell für diese Anwendungen entwickelt wurde und sehr einfach und schnell zu handhaben ist.

WER ist ZUSTÄNDIG FÜR.....?

Diese Frage haben Sie sich sicher im Zusammenhang mit der PRA auch schon gestellt. Da einerseits durch Fluktation der Mitarbeiter, andererseits aber auch durch neue bzw. anders gewichtete Spezialgebiete die Zuordnung zwischen einzelnen Aufgabengebieten und zuständigen Spezialisten immer nur eine gewisse Zeit lang stabil bleibt, wollen wir Ihnen hier wieder einmal den derzeit aktuellen Stand bekanntgeben.

		Mitarbeiter	Klappe
BETRIEBLICHES		Niedermayer	3611
Magnetbänder, Druckerpapier, Disketten und Kassetten			
Bestellungen		Niedermayer	3611
Verkauf	GH:	Operator	3613
	KP:	Operator	3440
	GM:	Operator	4821
Leitungsdurchschaltung		Kaineder	3611
Herstellung und Einsatz von Terminal-Umsetzern		Weiss	3440
Rechnerstörungen, Zentraleinheit, Peripheriegeräte		Niedermayer	3611
	GH:	Unterköfler, Amlacher Schöber	3613
	KP:	Gisch	3440
	GM:	Benovic	4821
Photoplotterfolienerstellung		Houdek	3609

SYSTEM-SOFTWARE

Betriebssysteme (ausgenommen CALMA)	Koblitz	3610
Kommunikations-Software	Wöber	3443
Sprachprozessoren	Tauer	3614
Tools	Selos	3443
DECUS-Bibliothek	Gollmann	3610
	Sprinzl	3612

ANWENDER-SOFTWARE	Wehrberger	3617
NAG- und CERN-Library	Sprinzl	3612
LINPACK	Sprinzl Sedlaczek	3612 3443
GRAPHIK (VAX, PDP)	Wehrberger	3617
GKS	Sedlaczek Wehrberger	3443
GOSY	Sedlaczek Wehrberger	3443
EUCLID	Oeschlmüller Wehrberger	3616
GRAPHIK (Calma)	Oeschlmüller Rogl	3616 3612
Software f. Entwicklung von Printplatten und Schichtschaltungen:		
GDSI	Houdek	3609
GDSII	Houdek	3609
3D-Software:		
DDM	Oeschlmüller	3616
Photoplotter	Houdek Rogl	3609 3612
DATENNETZE	Kunft	3608
Datex-P Datex-L X.25 Open Systems Interconnection von ISO ACONET	Kunft Plank Panigl	3608 3608 3614
ORGANISATORISCHES	Wehrberger	3617
Kundenbetreuung, Account-Nr, Projektberatung		
Koordination:	Wöber	3443
GH:	Wöber	3443
KP:	Wöber	3443
GM:	Sprinzl	3612
Graphik:	Oeschlmüller	3616
Kurse:	Sprinzl	3612
Feedback-Redaktion:	Poremba Wehrberger	3607
TU-PRA-Info:	Koblitz	3610

VERFÜGBARE SOFTWARE AN DEN ANLAGEN VAX-11/750 UND VAX-11/780

A. Sprinzl

Stand : 1985-03-31
Laufendes Betriebssystem : VAX/VMS, Version V3.6

Sprachpro- cessoren	Vers.	vorhanden 750	vorhanden 780	last update	Bemerkungen
CC	V1.5	j	j	85-01-03	
DECUS-C	V1	j	j		sys$sysroot: decusc1
DECUS-C	V2PL9	j	j		sys$sysroot: decusc2
FOR	V2.50	j	j		RSX, compat. mode
FORTRAN	V3.5	j	j	84-08-23	
F4P	V2.51	j	j		RSX, compat. mode
F77	V4.1	j	j		RSX, compat. mode
LISP	V1.0		j		Versuchsstadium, nicht gewartet
MACSYMA	V305	j			unter sys$sysroot: macsyma...
MODULA2	V3.1	j	j		
PASCAL	V2.5	j	j	84-11-08	
PROLOG	V1.0		j		Versuchsstadium, nicht gewartet

EMULATOREN

RTEM	V1.0	j	j		RT-11 Emulator

ANWENDUNGSBIBLIOTHEKEN

GOSY	V1.0	j	j		Universelle 2D-Plotter SW
GKSYS			J		GKS-Implementation
IMSL3	V1.0	j	j	84-12-06	
NAGLIB	V11	j	j	84-11-29	Vollständige Doku bei VAX-11/780
SL2000	V2.0		j	85-01-15	Simulation von IC's
EUCLID	V3.1B		j		3D-CAD/CAM Software

Nähere Informationen :
VAX-11/750 ... Wöber (Mail-Wilfried)
VAX-11/780 ... Sprinzl(Mail-K020103)

TAGUNGEN

BEITRÄGE & BERICHTE

CAMP '84

G. Wehrberger

"Computer Graphics - Anwendungen für Management und Produktivität", lautet die offizielle Bezeichnung der größten, ausschließlich auf Computer Graphik beschränkte Kongreß-Messe in Europa. Obwohl erst zum 2. Mal veranstaltet, kamen mehr als 6000 Besucher Ende September 84 nach Berlin, um sich über das Angebot von 85 Ausstellern zu informieren. Ungefähr die Hälfte aller Besucher konnten als Kongreßteilnehmer auch aus einem breit gefächerten Programm von Fachvorträgen die interessantesten für sich auswählen. Auch ich nützte beide Gelegenheiten und war einmal im Vortragssaal und gleich darauf wieder in der Ausstellungshalle anzutreffen.

Da praktisch alle namhaften Hersteller vertreten waren, und auf die Präsentation der Systeme von den Ausstellern besonderer Wert gelegt wurde, konnte man wirklich einen ausgezeichneten Überblick über den derzeitigen CAD-Markt erhalten. Dies war für mich im Interesse unserer zukünftigen CAD/CAM-Benützer äußerst wertvoll, da die PRA beabsichtigt hatte, bis Ende 1984 ein für die TU-Wien geeignetes CAD/CAM-System auszuwählen. Es war mir möglich, im persönlichen Gespräch mit den Vertretern der bekanntesten CAD/CAM-Anbieter auf die besonderen Probleme der TU-Wien einzugehen. CAD/CAM-Systeme wie:

PROREN (Fa. Isykon)
DDM u. GEOMOD (Fa. General Electric-Calma)
UNIGRAPHICS u. UNISOLIDS (Fa. McAuto)
MEDUSA (Fa. Computervision)
BRAVO (Fa. Applicon)
CDM300 (Fa. Kongsberg)
EUCLID (Fa. Matra Datavision)

wurden alle (nach eigenen Wünschen) demonstriert. Ebenfalls beeindruckend war die Vielfalt der ausgestellen Graphikterminals und der deutliche Trend zur Farbe, was bei den billigeren Geräten auf den Vormarsch der Präsentationsgraphik, und bei den teureren Geräten auf die Fähigkeiten moderner CAD-Software zurückzuführen sein dürfte.

Als Kongreßteilnehmer glaube ich feststellen zu können, daß das Vortragsniveau ein sehr hohes war. Ich habe mich zwar (vor allem aus Zeitmangel) nur auf Beiträge aus den Gebieten Computergraphik und Standardisierung, CAD-Aus- und Weiterbildung sowie Forschung im CAD/CAM-Bereich beschränkt, konnte aber doch einige interessante Aussagen aus Berlin mit nach Wien nehmen.

- GKS hat sich als Graphik-Standard endgültig durchgesetzt.
- GKS erhält eine 3D-Erweiterung
- GKS-Einsatz in Rechnernetzen wird vorbereitet
- In fast allen Industriestaaten laufen derzeit intensive CAD/CAM-Forschungsförderungsprogramme (auch in Österreich)

- Forschungsschwerpunkte amerikanischer Universitäten:
  Freiform Modelle, Modelldarstellung (Super Quadtrees), schnelle Verfahren für Flächenschnitte, realistische Bilddarstellung

Daß an anderer Stelle in diesem Heft bereits über eine Implementierung des graphischen Kernsystems GKS an der PRA berichtet wird, und im Februar das CAD/CAM-System EUCLID an der VAX-11/780 installiert werden konnte, ist zumindest zu einem kleinen Teil als Erfolg meiner Teilnahme an der CAMP'84 in Berlin zu sehen, und rechtfertigt wieder einmal den Besuch solcher Veranstaltungen.

DECUS-EUROPE SYMPOSIUM 1984 IN AMSTERDAM

W. Kunft

Im Herbst vergangenen Jahres fand wieder das traditionelle DECUS - EUROPE Symposium statt, und zwar diesmal im stimmungsvollen Amsterdam. Gemeinsam mit zwei Kollegen der PRA hatte ich die Gelegenheit, dieses Meeting zu besuchen und dort einem großen internationalen Kreis über unsere Arbeiten zur Schaffung eines experimentellen Universitätsrechnernetzes (ACONET) in Österreich zu berichten. Näheres über diese Arbeiten können Sie an anderer Stelle dieser "Feedback"-Ausgabe finden, wo das Manuskript meines Vortrages abgedruckt ist. In diesem Beitrag möchte ich kurz meine Eindrücke und Erfahrungen über das DECUS - EUROPE Symposium wiedergeben.

Mein spezielles Interesse galt dem Sachgebiet Datennetze und allem, was damit zusammenhängt. So besuchte ich ein Trainingsseminar, das am Vortag der Eröffnung abgehalten wurde, und das den Titel BASEBAND and BROADBAND LOCAL AREA NETWORKS hatte. Es wurde von Ron Davis, Managing Director der Firma Ungermann Bass Ltd., abgehalten. Herr Davis gab einen sehr guten, praxisbezogenen Überblick über die verschiedenen Techniken, mit denen Local Area Networks realisiert werden können. Besonderes Schwergewicht lag dabei auf den Vor- und Nachteilen die jede einzelne Technik mit sich bringt und die ausführlich diskutiert wurden. Auf diese Weise wurden den Seminarbesuchern Auswahlkriterien vorgestellt, die bei der Installation neuer Local Area Networks zu beachten sind. Ich habe eine kurze schriftliche Zusammenstellung (25 Seiten) über einzelne Technologien und Einsatzerfahrungen mitgebracht. Wenn Sie daran interessiert sind, rufen Sie mich bitte an (Klappe 3608).

DEC selbst präsentierte im Rahmen der ausgezeichnet organisierten Ausstellung DEKCHESS, bei der man praktisch alle Systeme und nahezu alle Softwareprodukte, die DEC 1984 anbot, im Einsatz sehen konnte, auch ein Local Area Network, nämlich ETHERNET. Dieses wurde verwendet, um die verschiedensten DEC - Systeme und Clusters (vom Professional 350 über PDP-11/xx, DEC-20 bis zur VAX-11/785) zusammenzuschließen. Im Einsatz zu sehen waren auch der Ethernet - Terminal - Server (PDP-11/24 mit speziellem Betriebssystem LAT-11 und speziellen Interfaces) und SNA- bzw. X.25-Gateways, die Systeme, die am Ethernet-Kabel hängen, an IBM- oder Datenpaketvermittlungsnetze heranführen. Der Besucher der Ausstellung hatte Gelegenheit, von mehreren Terminals aus über einen Terminal - Server und Ethernet mit den verschiedenen Hosts im Dialogbetrieb zu arbeiten. Die dadurch gebotene Flexibilität war beeindruckend, die Antwortzeit akzeptabel. Der Aufwand für einen Terminal - Server ist jedoch enorm und nur dann zu vertreten, wenn, wie eben bei dieser Ausstellung, sehr viele Terminals an viele verschiedene Wirtsrechner geführt werden sollen.

Die Vorträge während des Symposiums, die sich mit Datennetzen beschäftigten, zeigten, daß etliche Institutionen in Europa mit dem Problem kämpfen, Rechner verschiedener Hersteller über öffentliche Datennetze zu einem Verbund zusammenzuschließen. Die meisten scheuen jedoch davor zurück, den Weg zu gehen, den wir mit ACONET beschritten haben, nämlich an der konsequenten Realisierung der Dienste und Protokolle der Open Systems Interconnection von ISO zu arbeiten. Vielfach werden "Quasi - Standards" benützt, die in den letzten Jahren entwickelt wurden und die auf der von DEC erhältlichen X.25 - Software (PSI) aufbauen. Als Beispiel sei hier die "Coloured Books Software" genannt, die vom Institute of Science and Technology der University of Wales und an der St. Andrews University in Schottland entwickelt wurde. Sie baut auf VAX/PSI auf und umfaßt folgende Dienstleistungen: Transport Service (gelb), Character Terminals (grün) File Transfer (blau), Mail (grau) und Job Transfer (rot). Die Institutionen, die mit dieser Software bereits zur vollsten Zufriedenheit arbeiten, wollen verständlicherweise mit einer Umstellung auf ISO-Software solange warten, bis diese von DEC beziehbar ist. Von seiten der DEC gab es dazu eine interessante Stellungnahme zu hören: DEC wird schon in der nächsten DECnet - Phase (Phase V) Kommunikationssoftware nach den Regeln und Protokollen der Open Systems Interconnection - Software anbieten. Die Phase V von DECnet ist nach Auskunft von DEC-Experten 1986/87 zu erwarten.

Ein zweites Gebiet, über das ich hier kurz berichten möchte, war Bildschirmtext. Herr Ahlen (DEC Schweiz) stellte in einem sehr übersichtlichen und gekonnt präsentierten Vortrag die Videotex - Software der DEC vor. Mit der Software VAX VTX können auf einem Wirtsrechner Informationssysteme erstellt und verwaltet werden, die von intelligenten Terminals (z.B. PCs, VT100, VT200 oder Bildschirmtext - Terminals) abgerufen werden können und zwar auch über Netze mit DECnet hinweg. Sie bietet damit nicht nur die vom Bildschirmtext bekannten, sondern darüber hinaus noch weitere Dienstleistungen im eigenen Haus oder über Privatnetze. An einer Erweiterung der Software, die es ermöglicht, daß ein solches Informationssytem zu einem externen BTX-Rechner in einem öffentlichen Bildschirmtextdienst wird, wird gearbeitet.

PILOTPROJEKT ZUR ERSTELLUNG EINES EXPERIMENTELLEN UNIVERSITÄTSRECHNERVERBUNDES (ACONET)

W. Kunft

Im vergangenen Jahr konnte die erste Stufe eines Pilotprojektes abgeschlossen werden, das die Errichtung eines experimentellen Universitätsrechnernetzes in Österreich zum Ziel hat. Das Projekt wurde in Form von drei, vom Bundesministerium für Wissenschaft und Forschung geförderten Forschungsaufträgen von Projektgruppen der Universität Linz, der Technischen Universität Graz und der Technischen Universität Wien durchgeführt. Die Koordination der Projektgruppen erfolgte nach den Richtlinien, die von der beim Bundesministerium für Wissenschaft und Forschung eingerichteten Arbeitsgruppe Datenfernübertragung der Vorständekonferenz der österreichischen Universitätsrechenzentren beschlossen wurden.

Am 13. Dezember vergangenen Jahres wurden die Ergebnisse der Arbeiten von den drei beteiligten Projektgruppen an einem ACONET-Demonstrationstag im Kontaktraum der Technischen Universität Wien präsentiert. Dabei wurden auch Kommunikationsversuche (Dialogbetrieb und Interprozeßkommunikation) zwischen der VAX-11/780 der Prozeßrechenabteilung und dem PACER 600 der Hybridrechenabteilung der TU-Wien über das öffentliche Datenpaketvermittlungsnetz DATEX-P hinweg vorgeführt.

Im folgenden wird ein kurzer Überblick über das Projekt und über die Arbeiten, die die Abteilung Prozeßrechenanlage im Rahmen dieses Projektes durchgeführt hat, gegeben. Es wird dazu das Manuskript zu einem Vortrag wiedergegeben, der von mir beim DECUS EUROPE SYMPOSIUM 1984 in Amsterdam gehalten wurde, um die an der TU-Wien geleisteten Arbeiten einem breiten internationalen Interessentenkreis vorzustellen.

A PILOT PROJECT FOR AN ACADEMIC COMPUTER NETWORK IN AUSTRIA ACCORDING TO THE RULES OF ISO'S OPEN SYSTEMS INTERCONNECTION *)

W. Kunft
EDP-Centre of the Technical University of Vienna
Vienna, Austria

ABSTRACT

This contribution describes the concept of an experimental connection between subnetworks and single computers of EDP-Centres and institutes of Austrian universities. The Austrian packet-switching data network DATEX-P has been used and the existing rules and protocols of the Open Systems Interconnection have been applied to establish the experimental interconnection. This paper gives a general view of the methods and techniques which have been used to implement the necessary communication software in VAX- and PDP-11 systems under the operating systems VMS and RSX-11M/S.

INTRODUCTION

The Austrian Ministry for Science and Research established a special work group to give consultative assistance in the coordination of planning and of research activities in the field of data communications at Austrian universities. Especially the planning and all preparatory work concerning the establishment of an Academic Computer Network (ACONET) in Austria is supervised by this work group.

Within the framework of the Pilot Project ACONET Vienna, which is promoted by the Ministry for Science and Research, our department, the Department for Realtime Data Processing (DRDP) of the EDP-centre of the Technical University of Vienna, participated in establishing an experimental computer network connecting universities of Graz, Linz and Vienna. For this purpose a project group of the DRDP collaborated with project groups of the other universities concerned.

This paper describes the basic concept of the experimental interconnection. Further it presents the method and the implementation-techniques to integrate the subnetwork at the TU-Vienna into the pilot-computernetwork.

CONCEPT OF THE EXPERIMENTAL INTERCONNECTION

The following single computer systems or subnetworks were to be interconnected in the framework of the pilot project (see figure 1):

1) A Local Area Network (Ethernet) at the Technical University in Graz,
2) A Hybrid-Computer (EAI-PACER 600) at the Technical University in

*) erschienen auch in: Proceedings of the Digital Equipment Computer Users Society, 1984 DECUS Europe Symposium, RAI Congress Centre Amsterdam, The Netherlands, Sept. 24-28, 1984.

Vienna,
3) A subnetwork consisting of VAX- and PDP-11 computers operated with DECnet at the DRDP of the Technical University in Vienna and
4) A system IBM 370/155 at the University in Linz.

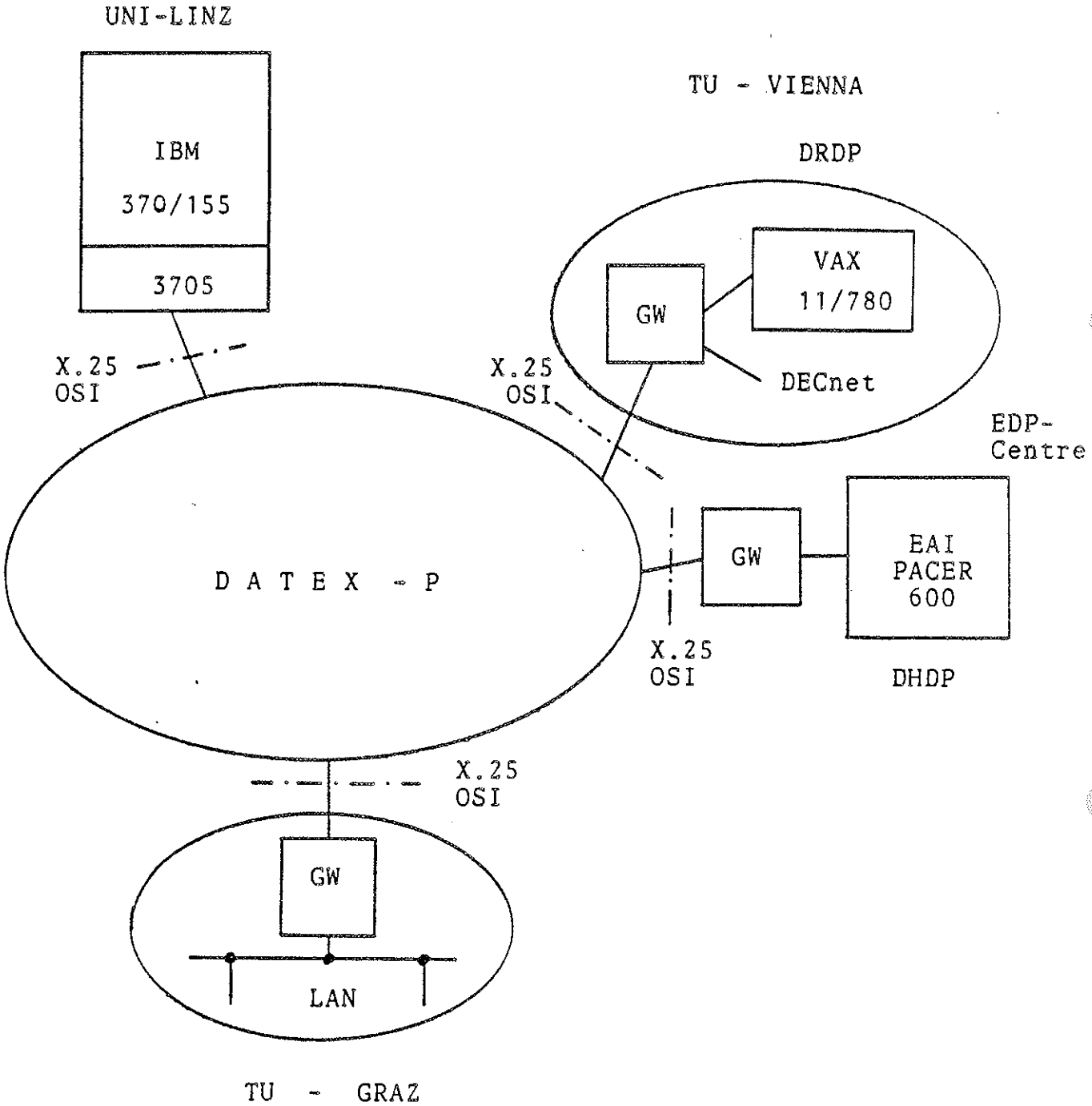


Fig.1 Construction of the experimental interconnection

GW ..... Gateway
OSI ..... Open Systems Interconnection
LAN ..... Local Area Network
DRDP ..... Department for Real Time Data Processing
DHDP .....Department for Hybrid Data Processing

The Austrian packet-switching data network DATEX-P was chosen to connect the computers and subnetworks. However very heterogenous systems had to be connected. Therefore the coordinating work group fundamentally decided that the interconnection should take place according to the rules of the Open Systems Interconnection (OSI) worked out by ISO [1]. Unfortunately, at the time we started our work two years ago, there existed only one stable set of OSI-rules, namely in the Transport Layer. So we decided to implement as a first step a Transport-Layer-protocol on top of X.25. Layers 5 and 6 should be realized in a second step in the future.

The second fundamental decision made by the project groups was to integrate not only the subnetworks but also single computers via gateway-systems into the experimental interconnection. There are two ways basically to establish such gateways:

1) The gateway contains only the functions of the first 3 layers. A Relay-function in the network layer is responsible to route information from the local region to remote regions and vice versa. Taking this solution a subnetwork can only be integrated into an OSI-conformable interconnection if the higher layers of OSI (layers 4 to 7) and their protocols are implemented in all end-systems of the subnetwork.
2) The gateway contains all seven layers. The Relay-function is situated in layer 7. This solution allows the integration of single systems or subnetworks which use any communications-architecture and -software.

Relying on the gateways according to the second variant the project groups integrated the subnetworks at the TU-Graz and at the TU-Vienna and the single hybrid-computer at the TU-Vienna. The IBM-system at the University in Linz was connected directly to DATEX-P via the frontend-system 3705. A PDP-11/34- and two PDP-11/23-Plus-systems have been used as gateways with the operating systems RSX-11M and RSX-11S. The PDP-11/34 is one of the computers of the DRDP and is equipped with disks, all other gateways have no disks or floppies.

SOFTWARE IN THE GATEWAY FOR THE SUBNETWORK AT THE TU-VIENNA

Figure 2 shows the software of the gateway used. On the one side, the "OSI-side", it contains X.25 software [2]. This software was developed several years ago at the DRDP and has been adapted to meet our requirements. In the Transport Layer protocolclass 0 was implemented [5]. This simple class is sufficient because DATEX-P provides acceptable error detection and error recovery in the first 3 layers. As mentioned above layers 5 and 6 remain empty in the first version of our gateways. This means that the application program in layer 7 in the gateway performing the relay-function directly uses the services of the Transport Layer.

On the second side, the "local-side", the gateway contains the communication software for the subnetwork, in this case DECnet.

Implementation-Method for the X.25- and the Transport-Layer-Software in the Gateways

Figure 3 shows the structure of the whole communication software on the OSI-side of a gateway.

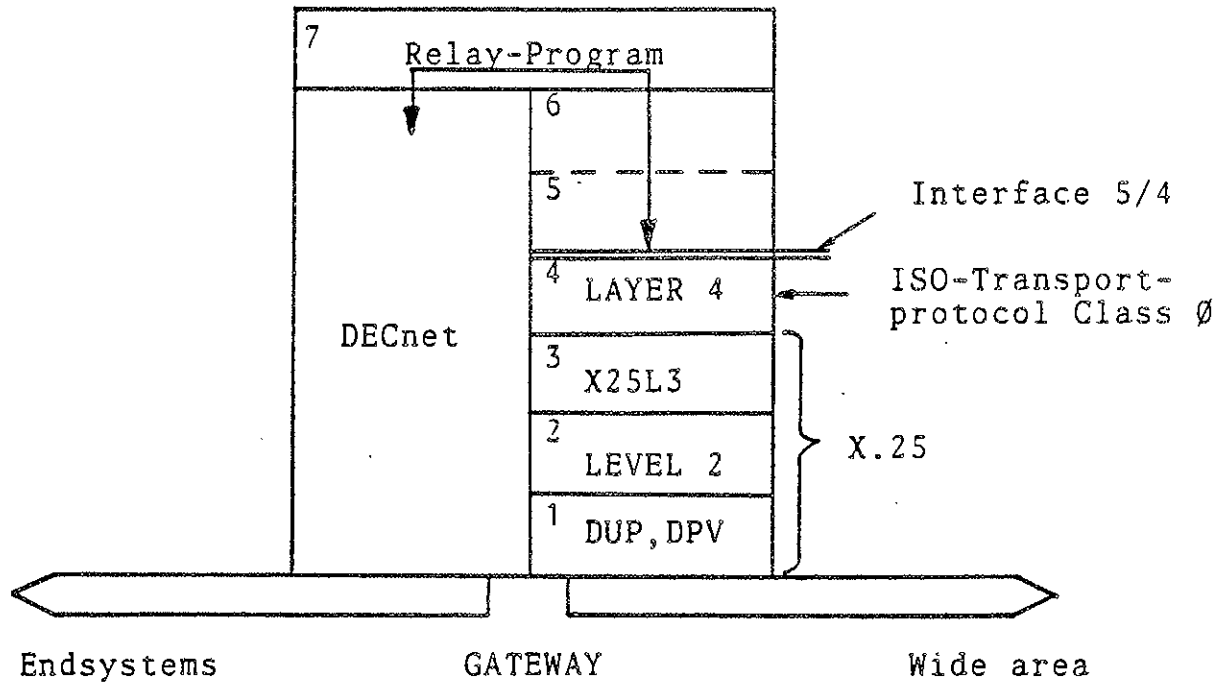


Fig.2 Structure for the Gateway-Software

The priviledged task LEVEL2 performs the functions of Level 2 of X.25 and has been implemented as a Finite State Machine. The task is able to communicate with a network node according to LAP or LAPB (LAP = Link Access Procedure). The latter procedure is applied in DATEX-P. LEVEL2 contains proper interrupt-service-routines for DUP-11- and DPV-11-interfaces. The services of LEVEL2 can be consumed only by one entity of Layer 3, namely the priviledged task X25L3. The events to be processed by the state-modules of LEVEL2 are orders from X25L3, indications from the interrupt-service-routines that a Frame has been transmitted or received and timeouts indicated by the operating system. Proper actions are carried out by the current state module according to the rules of the procedures of X.25 level 2 and, if necessary, a transition to a new state takes place. Possible actions are indications or confirmations to the task X25L3 in the higher layer, the start of a timer, the initiation of the transmission of a frame, etc. The service primitives of X25L3 are sent to LEVEL2 using the system-directives SDAT$ and RSUM$ of RSX-11M/S. Service primitives in the opposite direction (indications and confirmations) are sent from LEVEL2 to X25L3 by queuing asynchronous system traps to X25L3.

The services of the network layer can be used in the gateway by several user tasks. For this purpose we designed the interface between the layers 3 and 4 as a standard driver interface according to the conventions of RSX-11M/S. The driver NPDRV is called by the operating system in the course of the execution of a QIO$-directive issued by the user program. NPDRV performs only the management of Network-Service-Accesspoints (NSAPs) and sends all other I/O-packets to X25L3. The priviledged task X25L3 functions as an Ancillary Control Processor (ACP) and performs the real work according to the rules of the packet-level of X.25. To establish the cooperation between NPDRV and X25L3 we did not follow the conventional way to write ACPs, but we chose a more effective approach. No ACP-function-codes are used; I/O-packets which are not immediately processed by the driver are sent to X25L3 using SDAT$- and RSUM$-directives. Of course, NPDRV does not contain an interrupt-service-routine,

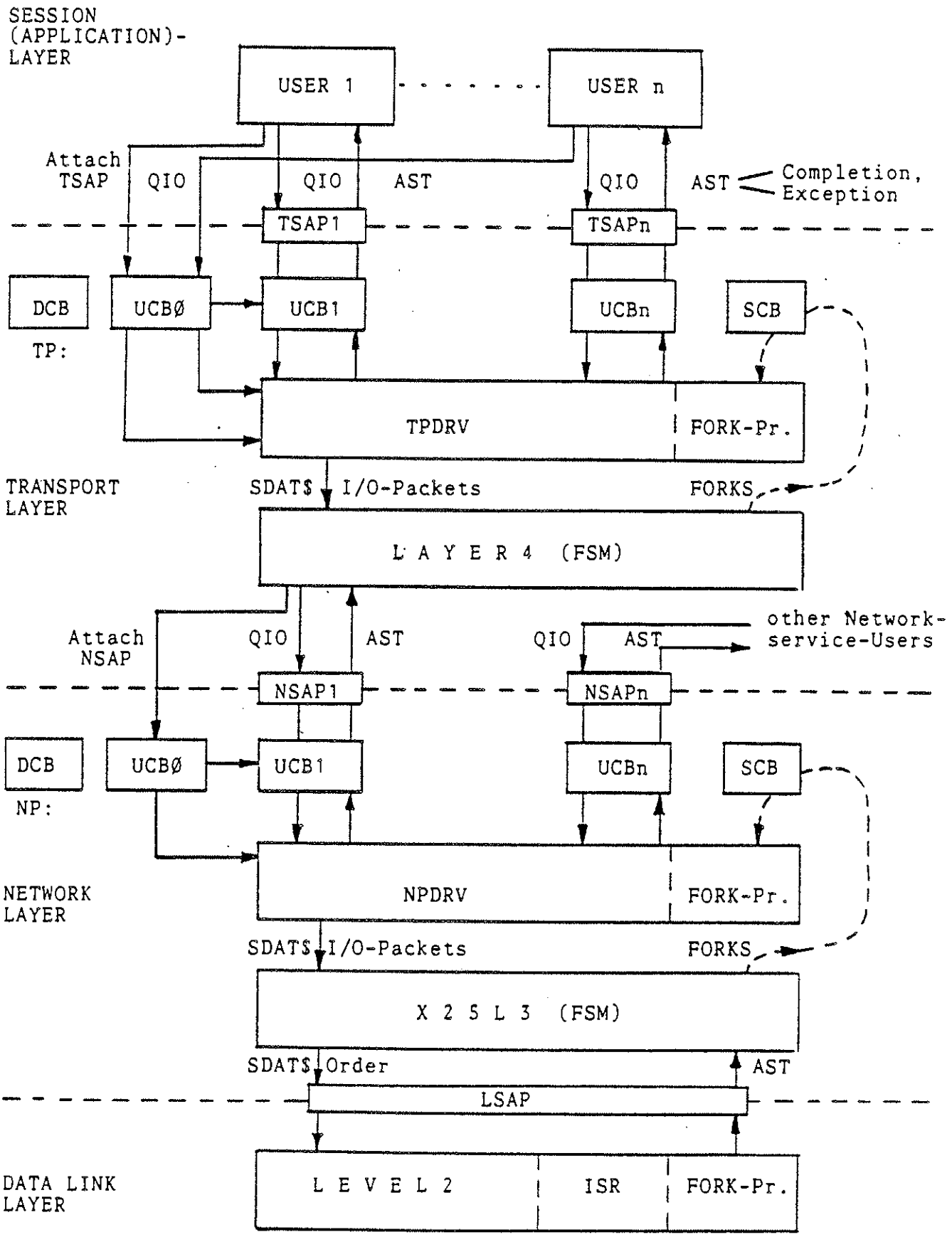


Fig.3 Implementation-method for the Gateway-Software

but rather a fork-process with the necessary code to process responses and indications of X25L3. If X25L3 wants to send any response or indication to NPDRV it fills several parameters in R4, R5 and in unused fields of the Status Control Block (SCB) of NPDRV's data base, and it creates a fork-process using the fork block in the SCB of NPDRV. There are two different kinds of "forks" generated by X25L3 and processed by NPDRV:

1) Completion-forks, which are used to indicate the completion of an operation. The address of the concerning I/O-packet and some status information are passed back to the driver as fork-parameters. The driver indicates the completion to the user by applying the normal procedures of RSX-11M/S.
2) Exception-forks, which indicate unexpected events, for instance the reception of an Incoming-Call-Packet or the reception of a Clear-Indication, and so on. As a reaction upon such forks NPDRV queues an asynchronous system trap (= Exception Ast) to the user program if this mechanism has been specified by the user program at the time when it has requested a Network-Service-Accesspoint from NPDRV.

Figure 3 shows the data base of NPDRV. There is one DCB for the Pseudodevice NP:, (n+1) UCBs and one SCB. N is the number of user programs which can be serviced by NPDRV at the same time. If a user program wants to apply the network service offered by NPDRV it has to make itself known to the service provider. For this purpose it has to attach a Network-Service-Accesspoint using NP0: and UCB0. The associated service primitive (QIO$) is processed directly by NPDRV. The driver looks for a free UCB which is further used as an NSAP for the user program concerned. Its address is put into the Logical Unit Table and there it replaces the address of UCB0 of the device NP:. All Logical Unit Numbers which are used by the user program to identify network connections must be assigned to NP0:. This means that the UCBn defining the NSAP of the user program concerned is always applied to process service primitives of this program. The SCB is used to provide the basis for fork-processing as described above.

X25L3 has been implemented as a finite state machine using the same technique as applied to implement LEVEL2. The events to be processed are Asts from LEVEL2 (for instance to indicate the reception of a packet on a logical channel), I/O-packets passed to X25L3 by NPDRV and timouts. Actions to react on occuring events are orders for LEVEL2 (for instance, to send a packet or to disconnect the link to the network node), the start of different timers and the start of the fork-process in NPDRV.

The QIOs, their function codes and parameters and all associated mechanisms such as Exception-Asts have been defined in an interface-specification according to CCITT X.25, Version 1980 and the Network Service Definition ISO DIS 8348 [3]. The current version of our X.25 software doesn't support the datagram-service (not applied in DATEX-P), the D-Bit mechanism and the negotiation of parameters during connection establishment. Further this version supports only switched virtual circuits.

The same methods and techniques as for the X.25 software apply to the implementation of the transport protocol class 0 in the gateways. The pseudodriver TPDRV provides a standard user interface and cooperates with its ACP, the priviledged task LAYER4, in exactly the same manner as described above. LAYER4 is also a Finite State Machine and uses the services of the network layer by issuing QIO$-directives which are processed by NPDRV and X25L3. A universal interface specification according to the Transport Service Definition ISO DIS

8072 [4] has been worked out to define the interface between layers 4 and 5. This specification considers the application of all protocol-classes.

The DRDP's project group was responsible for the implementation of the X.25- and the OSI-software in the gateways.

INTEGRATION OF THE SUBNETWORK OF THE TU-VIENNA IN THE EXPERIMENTAL INTERCONNECTION

The second part of our work in the pilot-project was the integration of the DRDP's subnetwork into the experimental interconnection. To do this the services offered by the Transport Layer in the gateway had to be made available to user programs running in the endsystems. This is performed by the application program DNCTP in the gateway in cooperation with a set of subroutines in the endsystems, named the TPES-Routines, which are linked to the user programs (see figure 4). For each sevice primitive defined in the specification of the Transport-Service in the gateway there is one corresponding TPES-Routine which can be called by the user in the endsystem to consume the service concerned. The parameters for each TPES-Call are transported to DNCTP via DECnet. DNCTP uses them to prepare the associate service primitives and issues the proper QIO$-requests which are processed by TPDRV in the gateway. Indications and confirmations coming up from layer 4 to DNCTP in the gateway are also reported to the proper TPES-module via DECnet. Unexpected indications are made known to the TPES-Routines and the user program in the endsystem by using the Exception-Ast mechanism of DECnet. Our work group developed a special application-protocol using DECnet for the cooperation of DNCTP and the TPES-Routines.

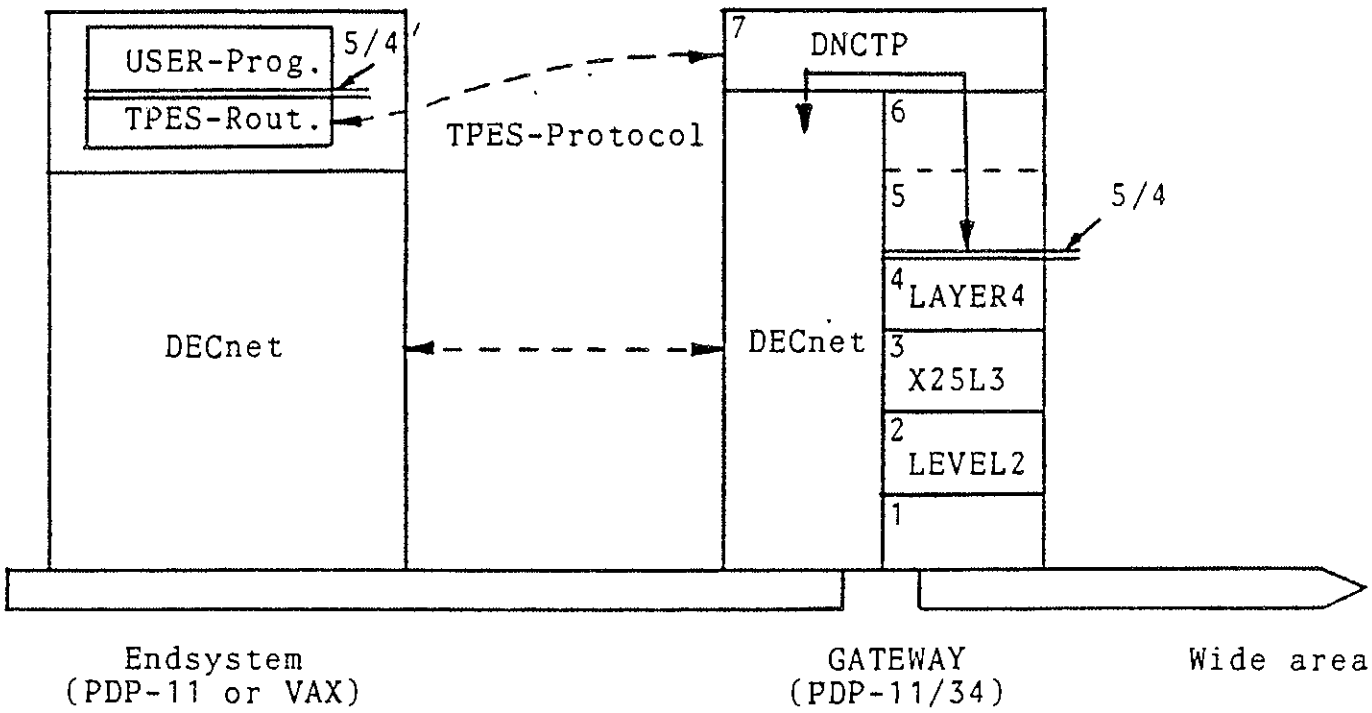


Fig.4 Integration of the DRDP's subnetwork

ADDITIONAL MECHANISMS TO SUPPORT GATEWAYS

As mentioned above only the gateway of the DRDP is equipped with disks. Two additional mechanisms have been established to support gateways which are not equipped with any mass storage:

1) Downline-loading via DATEX-P
2) A simple program-development-facility for gateways on the VAX-11/780 of the DRDP

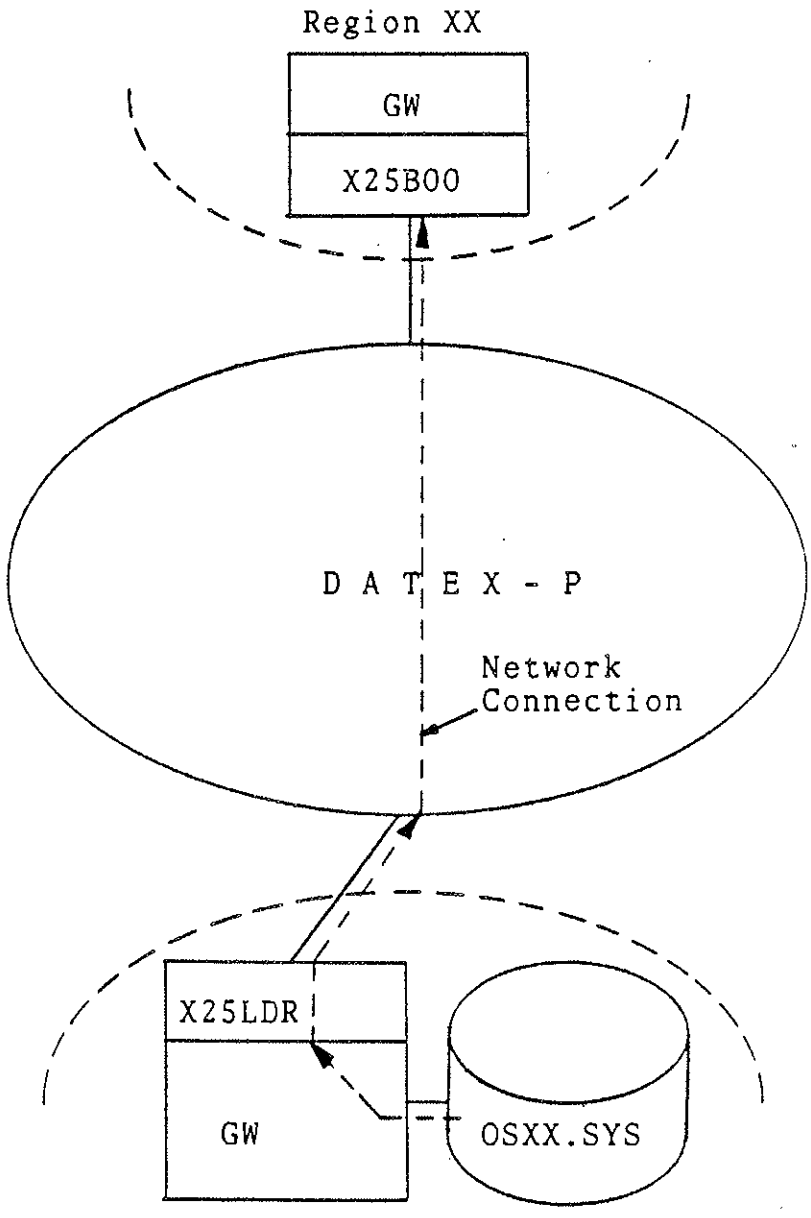


Fig.5 Downline-loading of a Gateway

OSXX.SYS .... Operating System for Gateway of region XX

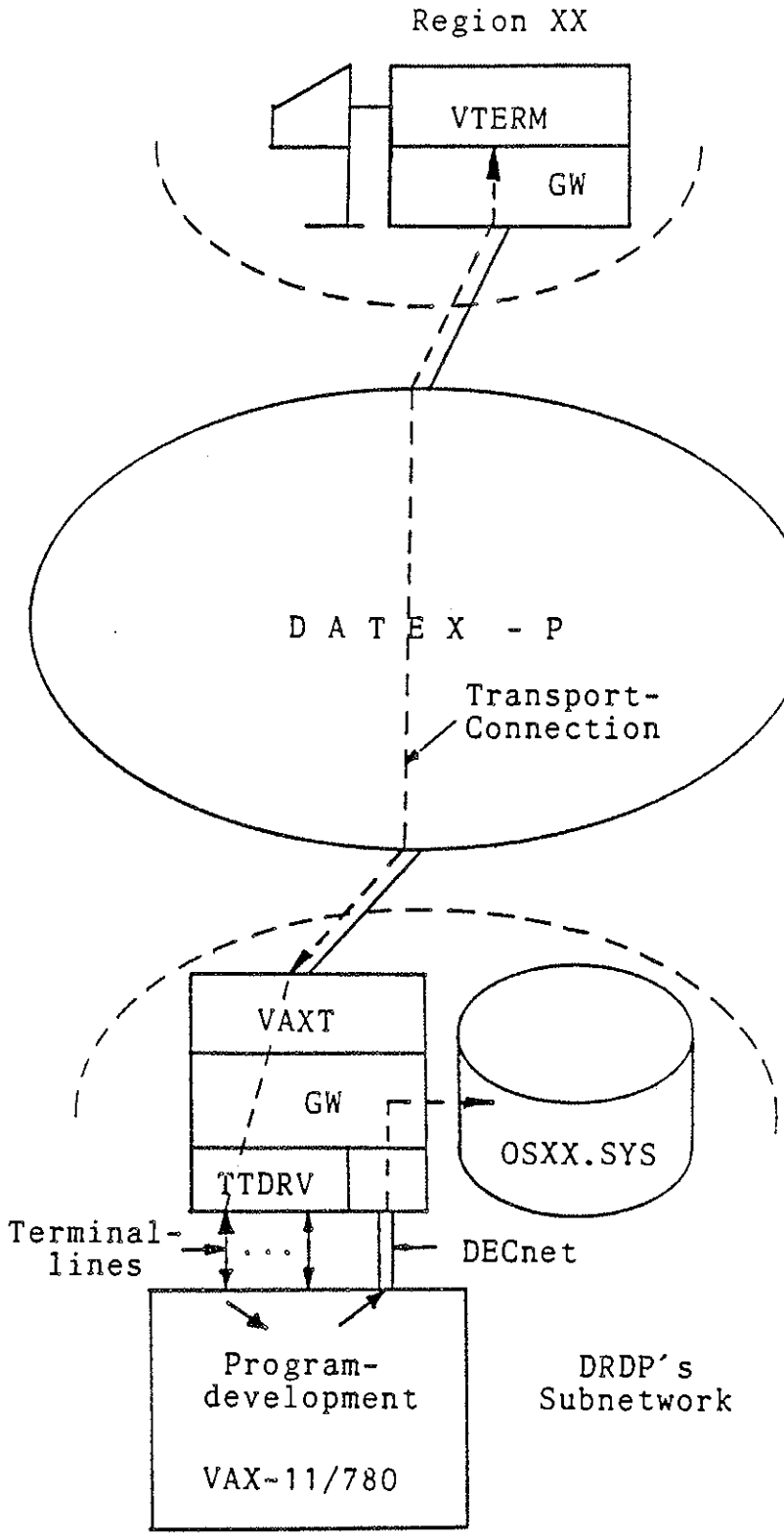


Fig.6 Simple Programdevelopment-Support
for Gateways

Loading a Gateway via DATEX-P

To load a gateway with its operating system RSX-11S and all necessary tasks we have developed two programs which cooperate via DATEX-P to provide this service (see figure 5).

We established the bootstrap-program X25BOO in two 2716-EPROMs which replace the DEC-EPROMs on the CPU-board of the PDP-11/23-Plus gateways. The bootstrap contains the basic functions of CCITT X.25 to communicate with a network node of DATEX-P. It establishes a Level 2 link to the network node and after initialization of the packet-level by a Restart a network connection is established to the program X25LDR which runs in the gateway of the DRDP.

X25LDR fetches the proper operating system of the calling gateway and all fixed tasks from the disk of the DRDP's gateway and sends the whole image to X25BOO. The last message is a load-complete-message which is indicated by Q=1 and which contains the transfer-address of the operating system. X25BOO answers with a load-okay-message (Q=1) and starts the new operating system.

A Simple Support to Develop Software for Gateways

The application program in a gateway depends on local requirements and communication software. Therefore it is generally developed by a programmer of the university concerned. If there is no DEC-environment it is necessary to provide a service which allows the programmer to develop his gateway-application on the VAX-11/780 host of the DRDP using the console-terminal of his gateway. This support is provided by two programs, which establish a very simple virtual terminal, namely VTERM in the remote gateway concerned and VAXT in the gateway of the DRDP (see figure 6).

VTERM establishes a transport connection to VAXT and sends all characters entered by the user to this program. VAXT passes the characters to the VAX via a terminal-port of the gateway which is connected to a terminal-port of the VAX. That means VAXT seems to the VAX to be a real local terminal. All characters sent by the VAX to VAXT are passed to VTERM via the transport connection and VTERM types them out to the user.

The method described above seems to be very simple, but the actual implementation showed that there were very complicated problems to be solved. For example, the VAX sent no terminating characters, flow-control-problems arose, logout had to be performed in the VAX automatically if an unrecoverable error occurred and so on. This method leads to a high load for the gateway of the DRDP especially if the terminaldriver is used to operate the connection between VAXT and the VAX. Therefore we plan to work out a better solution in the near future to provide this service.

ACKNOWLEDGEMENTS

We appreciate the support of our work by the Austrian Ministry for Science and Research. We also express our thanks to the Austrian Postal Authority for its support.

References

1. ISO TC97, Information Processing Systems - Open Systems Interconnection - Basic Reference Model. ISO DIS/7498, 1982
2. CCITT, Recommendation X.25. In: CCITT, Yellow Book, Vol. VIII, Fascicle VIII.2, Geneva, 1981
3. ISO TC97, Information Processing Systems - Open Systems Interconnection - Connection-oriented Network Service Definition. ISO DIS/8348, 1983
4. ISO TC97, Information Processing Systems - Open Systems Interconnection - Transport Service Definition. ISO DIS/8072, 1983
5. ISO TC97, Information Processing Systems - Open Systems Interconnection - Connection-oriented Transport Protocol Specification. ISO DIS/8073, 1983

PROBLEME UND ERFAHRUNGEN BEI DER REALISIERUNG EINES "VIRTUAL-FILE-STORE" FÜR EIN EXPERIMENTELLES UNIVERSITÄTSNETZWERK

Statusbericht und Zusammenfassung eines Vortrages präsentiert im Rahmen der Herbsttagung von DECUS-Europe 1984

W. Wöber

DAS PROJEKT

Im Rahmen der Aktivitäten der PRA-Arbeitsgruppe Netzwerke auf dem Gebiet ISO-OSI (International Standards Organisation - Open Systems Interconnection) wurde ein informelles Projekt gestartet, das folgende Ziele verfolgen soll:

- Studie über die Realisierbarkeit einer VFS-Funktionalität (VFS steht für Virtual-File-Store) entsprechend den ISO-OSI Protokollen auf momentan verfügbaren Minicomputern.
- Angebot eines VFS-Dienstes im Rahmen von ACONET auf zwei Ebenen, nämlich
    - o als ein Grundservice für Datenaustausch und Informationsabfrage,
    - o als eine experimentelle Umgebung, um die zukünftigen Anforderungen im Rahmen von Hochschulnetzen studieren zu können.

DIE ENTWURFSKRITERIEN

Folgende Forderungen waren wichtige Ausgangspunkte bei den Überlegungen:

- Implementierung der vollen (von ISO beschriebenen) Funktionalität.

- Integration der Software in bestehende Betriebssystemumgebungen, und zwar

    o primär in VAX/VMS,

    o allerdings sollte PDP-11/RSX als Umgebung denkbar bleiben.

Daraus ergab sich als Konsequenz, daß die neu entwickelte Software entsprechend den etablierten Software- und Systemstrukturen aufgebaut werden soll, damit ein möglichst hoher Grad von Kompatibiltät erreicht werden kann.

Weitere Randbedingungen für diese Entwicklung:

- Auf der PDP-11 ist bereits ein Teil der Software für die unteren Ebenen der ISO-OSI Struktur, entsprechend dem 7-Schichten-Modell, realisiert.

- Das Entwicklungsteam ist sehr klein und hat nur geringe Personalkapazität.

- Die Erfahrungen aus diesem Projekt sollen so weit wie möglich auf andere Entwicklungsprojekte anwendbar sein.

DIE SOFTWARESTRUKTUR

- Die Implementierung soll durch die Erstellung von Softwaremodulen erfolgen, die jeweils eine Protokollschicht behandeln und aus einem Pseudo-Device-Driver mit zugeordnetem ACP (Ancilliary Control Processor) bestehen.

- Die im Rahmen dieses Projektes für VAX/VMS erstellte Software soll auf die 16-Bit PDP-11 Architektur übertragbar sein.

- Die Realisierung muss in einer Hochschulumgebung erfolgen, das heißt, daß die Arbeitsgruppe während der Realisierungsphase unterschiedlich stark sein und die Zusammensetzung immer wieder wechseln wird. Darauf ist besonders bei Organisation, Dokumentation und Softwaretechnologie zu achten.

- Als Implementierungssprachen wurden 'C' und 'MACRO-Assembler' gewählt.

VORAUSSETZUNGEN FÜR DEN ERFOLGREICHEN ABSCHLUSS DES PROJEKTES

Eine Reihe von Maßnahmen mußten gesetzt, bzw. Vorarbeiten geleistet werden, um das Projekt durchführbar zu machen:

- Sichtung der vorhandenen und projektbezogenen Software, das sind besonders

    o Beispiele und Softwaregerüste für Device-Driver und ACPs für RSX-11,

    o Beispiele und Softwaregerüste für Devive-Driver und ACPs für VAX/VMS.

- Aufbau von Kontakten zu Arbeitsgruppen mit ähnlicher Zielsetzung.

- Sammlung von praktischer Erfahrung durch Anwendung der geplanten Methoden auf kleinere Softwareprojekte, zB. RK05-Driver für VAX/VMS.

- Erarbeitung und Verabschiedung von projektspezifischen Konventionen für die Schnittstellen zwischen den einzelnen Funktionsbausteinen.

- Durcharbeitung der für das Projekt relevanten ISO-Veröffentlichungen (siehe Kapitel Dokumentation), um die Anforderungen und die Entwicklung der Funktionalität wirklich zu verstehen.

PROBLEME

Verständlicherweise geht eine Projektgruppe in der Planungsphase von sehr optimistischen Einschätzungen der Umgebung aus, die Realität sieht meist etwas anders aus.
Im Rahmen dieses Projektes sind bisher folgende Schwierigkeiten aufgetreten:

- Der Zeitplan konnte nicht eingehalten werden, weil
  - o der Aufwand für die Beschaffung und das Verständnis der notwendigen ISO-Veröffentlichungen nicht vorhersehbar war,
  - o die funktionelle Beschreibung der VFS-Dienstleistungen nicht in einer stabilen Form verfügbar ist,
  - o und die Personalkapazität natürlicherweise oft für andere Aufgaben eingesetzt werden muß.

- Organisatorische und betriebliche Hindernisse ergaben sich im Bereich Finanzierung und Verfügbarkeit einer Testumgebung.

- Softwarebezogene Probleme sind
  - o Mangel an Dokumentation für die Systemschnittstellen,
  - o Unsicherheiten im Entwurf durch die bevorstehende Einführung von VAX/VMS Version 4,
  - o die existierende Netzwerkstruktur.

ERFAHRUNGEN

Die bisherigen Erfahrungen betreffen vor allem den Bereich Entwurf und Implementierungsmethoden. Dazu ist zu sagen, daß VAX/VMS eine sehr benützerfreundliche und robuste Entwicklungsumgebung für Systemsoftware anbietet, ganz besonders durch die Verfügbarkeit von SDA, XDELTA und DEBUG.

Der VAX-11 C Compiler scheint ein sehr brauchbares Entwicklungswerkzeug für diese Art von Software zu sein. Die Belastung von Computerressourcen ist durchaus überschaubar. Weiters ist die Programmiersprache 'C' ein sehr anpassungsfähiges Werkzeug, um auf bestehende Strukturen Rücksicht zu nehmen.

Darüber hinaus zeigt sich, daß eine sehr reguläre Softwarestruktur das Testen von Funktionsblöcken im Usermode gut unterstützt, womit die notwendigen Testzeiten im Kernelmode gering gehalten werden können.

UNTERSTÜTZENDE DOKUMENTATION

Im Rahmen dieses Projektes wurden folgende Informationsquellen benutzt:

- Standard Systemdokumentation
    - o Guide to Writing I/O-Drivers
    - o I/O Operations User's Guide
    - o System-Listings
- über DECUS verfügbare Informationen
    - o Programmbausteine und Beispiele für Device-Driver und ACPs
    - o ACP-Notes
- empfehlenswert ist weiters das Buch
    - o VAX/VMS System Internals and Data Structures
- Die ISO-OSI-Funktionalitäten, Protokolle und Konventionen sind beschrieben in den ISO-Veröffentlichungen
    - o ISO Standards
    - o ISO Draft Standards
    - o ISO Working Papers

ZUSAMMENFASSUNG

Die Projektgruppe hat bisher eine Reihe von wichtigen Erfahrungen gemacht:

- Der Personalaufwand für die gesamte Abwicklung des Projektes wurde wahrscheinlich erheblich unterschätzt.
- Die Projektgruppe war gezwungen, sich mit einer ganzen Reihe von verwandten und abhängigen Sachfragen zu beschäftigen.
- Die bisherigen Ergebnisse im Sinne einer experimentellen Testimplementierung haben den Aufwand sicher gerechtfertigt.

IMPLEMENTATION OF ISO/DIS 7846 INDUSTRIAL REAL-TIME FORTRAN IN RSX-11M

W. Koblitz
Technical University of Vienna
Austria

ABSTRACT

Industrial Real-time Fortran is a set of standardized subroutine calls to allow for task scheduling, synchronization, bithandling, process I/O and file access. It became an ISO DIS (Draft International Standard) in January 1984. The standard is based on Fortran 77 and does not require any extension to the compiler itself. The implementation in RSX11M is built on an ACP, responsible for maintaining event queues, semaphores etc, a pseudo device driver and a subroutine library.

The contribution describes the features of Industrial Real-time Fortran and outlines the solution in the RSX11M implementation.

INTRODUCTION

You will find an easy to understand introduction to ISO/DIS 7846 [1] in section 1. Section 2 explains our motivation for carrying out the project to implement IRTF (Industrial real-time Fortran) in RSX-11M. How we do this is shown in section 3. There you may find some useful hints for writing your own ACP.

1. ISO/DIS 7846

ISO/DIS 7846 'Industrial real-time FORTRAN - Application for the control of industrial processes' (IRTF) defines a set of subroutines and functions based on Fortran 77 (ISO 1539) [2]. These subroutines and functions allow for

- obtaining date and time information
- scheduling and synchronization of tasks
- bit manipulation
- performing process I/O
- synchronization of file access

IRTF may be used with any standard Fortran 77 compiler; no modifications to the compiler are necessary.

IRTF was published by ISO/TC97 (ISO = International Organization for Standardization, TC = technical committee, 97 = information processing) on 1984-01-26 as DIS (draft international standard) and was unanimously approved by the national member organizations [3]. Deadline for voting was 1984-07-26.

IRTF is divided into four sections preceded by a list of definitions and terminated with an appendix containing tutorial explanations. We will use this segmentation for the following introduction to its contents.

The only definition we need at the moment to avoid confusion is the definition of 'task':

> Task (in the context of IRTF) is a computation which can be scheduled. The operations of this computation are performed in a strict sequential order of operations.

With this definition we can examine what is in section 1 of IRTF.

1.1 IRTF section 1 'Multiprogramming and real-time features'

The essence of this section is the description of a state model (see figure 1) which allows a task to be in one and only one of five states:

nonexistent, dormant, pending, running and suspended.

'Nonexistent' means that the task is not known by the IRTF system and cannot be scheduled for execution. The first step to bring a task into the system is a call of subroutine CREATE. CREATE has two parameters: one is an integer array containing all information to identify the task and its associated program, the other parameter is an integer variable indicating the success of the call. CREATE makes the task known to the system, the task transits to state 'dormant'.

A task in state 'dormant' may be scheduled for execution by one of several possible subroutine calls allowing for

- immediate start
- start dependent on time (delay or absolute)
- repeated execution
- start dependent on occurrence of an event

To take an example:

```
call CYCLAF(i,t1,t2,m)
```

will schedule the task identified with the parameter i for repeated execution, starting after the delay time t1 (calculated from now). Argument t2 specifies the time interval between two starts, m indicates the success of the call.

We can get rid of the thing with:

```
call CANCEL(i,m)
```

with i identifying the task and m indicating the success. Task i will return to state dormant.

But - as we hope - we do not want to immediately cancel our scheduling but wait for execution. Taking our example again: as soon as t1 is reached task i will transit to state 'running'. 'Running' in the context of IRTF means that from the programmers point of view all conditions are satisfied that task i may run. In the context of the real system it means that task i is either waiting for a physical processor or it is really executed (as soon as it gets hold of a processor).

Obviously the transition of task i from 'dormant' to 'pending' or 'running' can only be caused by another task j. Within task j we may call CYCLAF to schedule task i. But as soon as task i is running it may cause a state transition for itself. It may delay its execution for a certain time or until a specified

event occurs. It transits to state 'suspended'. Lets have an example again:

call DELAY(t,m)

would suspend execution for a time delay as specified by t. Argument m indicates the success of the call.

The states and transitions are best shown with a little graph:

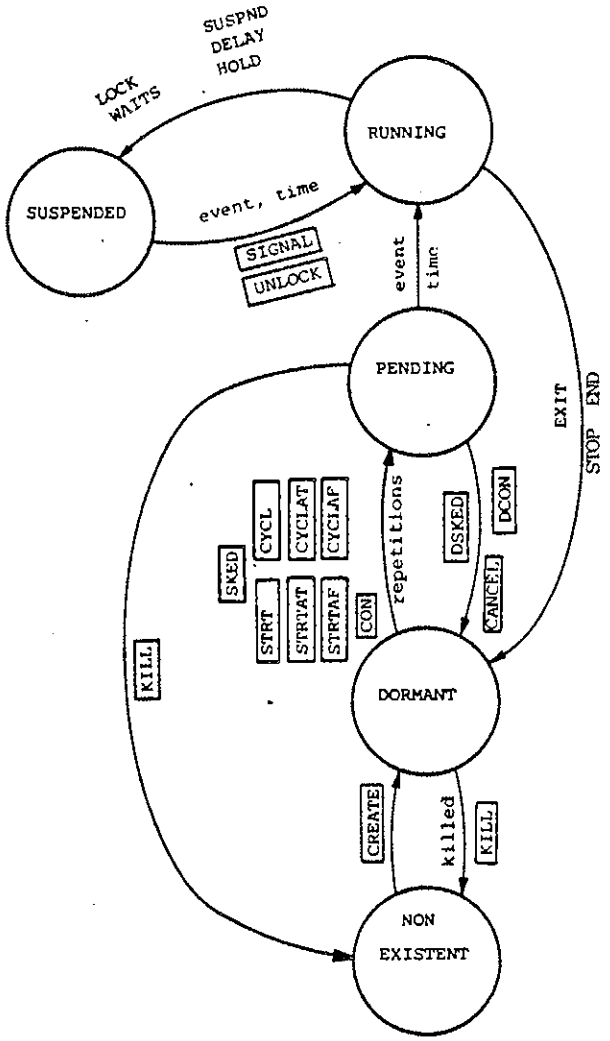


Figure 1. STATE MODEL AND TRANSITION DIAGRAM

Synchronization and scheduling is based on three concepts: eventmarks, resourcemarks and semaphores.

Eventmarks are entities indicating the occurrence of an external or internal event. Eventmarks are selected by reference to a numeric selector in the range of 1 to n, where n is implementation dependent. They have only two states

ON
OFF

You ever heard about eventflags ? Sounds familiar ? You are not far from the road.

Resourcemarks may be something new for you, but it is very simple. As eventmarks are bound to events resourcemarks are bound to resources. Imagine you want exclusive access to a line printer. If you make resourcemark k the 'line printer' the only thing the print task has to do is

```
call LOCK(k,m)
```

and the printer is yours. Of course IRTF explains what shall happen when the resource is already locked by another task etc but this is not important for our introduction here. As eventmarks are allowed the two states ON and OFF resourcemarks have the states

LOCKED
UNLOCKED

In addition to these easy to understand mechanisms IRTF defines subroutines working on counting semaphores to provide a more advanced means for task synchronization. If you would like to have a task waiting for a number of conditions until it may run, semaphores will be the right thing for you to handle it.

1.2 IRTF section 2 'Binary pattern and bit processing'

While IRTF section 1 is a real challenge, section 2 is nearly trivial. The subroutines defined in this section provide mechanisms for operations on bit patterns as well as individual bits of the internal representation of integer variables.

Lets illustrate with an example:

```
r = IOR(j,k)
```

is calculated according to

```
j 0101
k 0011
------
r 0111
```

There are subroutines defined for OR, AND, IOR and NOT, for logical, arithmetical and circular shift and for bit test, set, change and clear.

1.3 IRTF section 3 'Process input/output'

What do we need to do process I/O ? We must be able to specify the I/O port and we need some storage where we find the output values or where we place the input values. Lets see what IRTF has defined.

The IRTF procedures for process I/O normally take four parameters to specify:

the number of values to be transferred

the name of an integer array or array element containing the necessary information for the description of the I/O ports, i.e. address and data conversion information

the name of an integer array or array element that contains the input

or output values

a status indicator to characterize the success of the call (all data transferred, error conditions)

Here an example:

call AISQW(i,j,k,m)

will read a sequence of i analog input ports specified by information found in j to the integer array k. Argument m indicates whether all ports have been read successfully or if an error has occurred.

ISO 1539 [2] specifies that one statement must be completed before processing of the next statement begins. IRTF adheres to this rule. The calling task will wait for completion.

Subroutines are defined for

sequential analog input
analog input in random sequence
analog output
digital input
digital pulse output
latched digital output

1.4 IRTF section 4 'File handling'

The procedures described in this section provide means for controlling the access to files, and also provide means for resolving problems of file access contention in a multitasking/multiprocessing environment. The standard defines the meaning of 'file' in the context of IRTF and what is within the scope of IRTF. Two main attributes are: the contents of files are considered to be data; files are external to a concurrent task.

Files may be created, deleted, opened and closed and the access mode may be modified. The procedures for performing these operations are noninterruptable (from the IRTF programmers point of view), that is the processor will only execute one such procedure at a time.

Lets take 'open' for an example:

call OPENW(i,j,k,m)

i specifies the unit by which the file, named by the argument j, is referenced in the task; k specifies the access mode desired by the task.

Defined access mode are

1 Unlocked - read/write or write-access is requested by the calling task; other tasks are allowed the same access.

2 Protected read - read access is requested by the calling task and allowed to other tasks.

3 Locked - read/write or write access is requested by the calling task. The calling task excludes any file access by other tasks.

m indicates the disposition of the request - successfully opened or file not opened to the calling task.

2. MOTIVATION FOR IMPLEMENTATION

A number of mini and micro computers are used in our university labs to collect data and to control experiments, many of them are PDP 11's and LSI 11's running RT-11 or RSX-11M. Assisting our colleagues in solving their process control problems we saw the wheel being reinvented every day and therefore thought about general packages to make life easier in this area.

On some funny coincidences I happened to fall into the committee evaluating IRTF and participated for the last few years in the preparation of this standard. By that we had access to the most recent information and it was just natural to start the implementation project.

The decision to first try the implementation in RSX-11M was caused by the fact that Fortran 77 is available for RSX-11M but not for RT 11. The multitasking capability of RSX was no valid argument because we do not make use of it.

In addition we wanted to learn about writing ACP's and we were interested in methods for hooking an 'operating system' (set of services) upon an existing operating system. A second project in this area is running at our university aiming at network services. Of course the two projects are quite different from the applications point of view but they are very similar from the system programmers point of view. (see contributions from Walter Kunft and Wilfried Woeber in these proceedings)

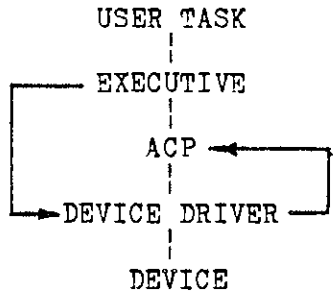
3. IMPLEMENTATION OF IRTF IN RSX-11M

How can IRTF be implemented in RSX-11M ? Reading chapter one of this paper, certainly some ideas came to your mind and I assume we agree that the implementation of 'multiprogramming and real-time features' is the most interesting task to do, followed by the section on file handling. 'Binary pattern and bit processing' is nearly trivial and 'process I/O' is not a big challenge, although it involves much more time for coding, testing and debugging. To get a clean RSX-11M solution for the I/O problem we need device drivers for all the fancy I/O interfaces integrated in our lab systems combined with an IRTF subroutine library. That's it. You have not written a device driver yet ? Start with reading [4] and look for examples either in the RSX-11M source or on the SIG Tapes.

It's not that easy with section one. We need some part in the system, which takes care of our eventmarks, resourcemarks, semaphores and queues. More generally formulated we must find a way to supply a set of services to several concurrent tasks in our RSX-11M system. But we are lucky. We are not the first to need something of that kind. If we look around there is the filesystem, the network services of DECnet and maybe some more. How are these services implemented in RSX-11M ? The solution is called ACP (Ancillary Control Processor) and is nothing magic. It is an independent privileged task which cooperates with a device driver and is accessed via the QIO directive. The 'device' may be physical or virtual.

With a little graph we can illustrate the flow of information from a user task

to a device when a QIO request is issued:



If we have a standard ACP (e.g. Files-11) the executive QIO processing, after passing several checks, first queues the package to the ACP. In our case of a nonstandard ACP the executive QIO processing directly calls the driver, where parameter checking takes place and the driver inserts the request package in the ACP's receive data queue. The ACP is unstopped or requested to run.

The 'device' in our case is the pseudo device 'IRTF', which very well could be a hardware timer. (A programmable hardware clock would allow better time resolution for the IRTF scheduling).

Using the QIO mechanism we make sure that all IRTF service requests end up at the same server (the IRTF ACP) sequentially and synchronization becomes easy.

Writing an ACP is not that difficult that an experienced programmer could not dare to do so. You will find helpful documentation on the RSX SIG Tapes [6] and a good introduction in Appendix D of [4]. You even find a skeleton driver and a dummy ACP on the SIG tapes containing all code that makes the RSX frame around your algorithm. And there is nothing that prevents you from writing your ACP algorithm in a high level language subroutine called from the ACP main program written in Macro 11. Of course we will write our IRTF scheduling in Fortran 77 (for portability reasons) as soon as we succeeded with our first try coded in C.

SUMMARY

ISO/DIS 7846 defines a set of services needed for multitasking in control applications. Such a set of services can be implemented in RSX-11M via an ACP and a (pseudo) device driver. The service algorithm itself may be coded in a high level language (e.g for portability reasons) and linked as subroutine(s) to the ACP frame written in Macro 11. Much help can be found on the SIG Tapes in form of excellent documentation, a dummy ACP and driver examples. Further information on the IRTF project may be obtained from the author by writing to Technical University of Viennna, Gusshausstrasse 25, A-1040 Vienna, Austria.

ACKNOWLEDGEMENT

Last not least I would like to mention that projects of this kind will only prosper if they are generously sponsored and promoted by one's boss, in my case Dr Manfred Paul and if you have at least one never tired colleague, with whom you can discuss the problems arising from misunderstanding of literature, undocumented system behavior etc. If the colleagues in addition take a big chunk from the work, like mine, Walter Selos and Ludwig Tauer, you have got a fine job. I would also like to thank Walter Kunft and Wilfried Woeber (the guys involved in the other ACP project at our site) for their valuable hints.

BIBLIOGRAPHY

[1] ISO/DIS 7846 Industrial real-time FORTRAN - Application for the control of industrial processes

[2] ISO 1539, programming languages - FORTRAN

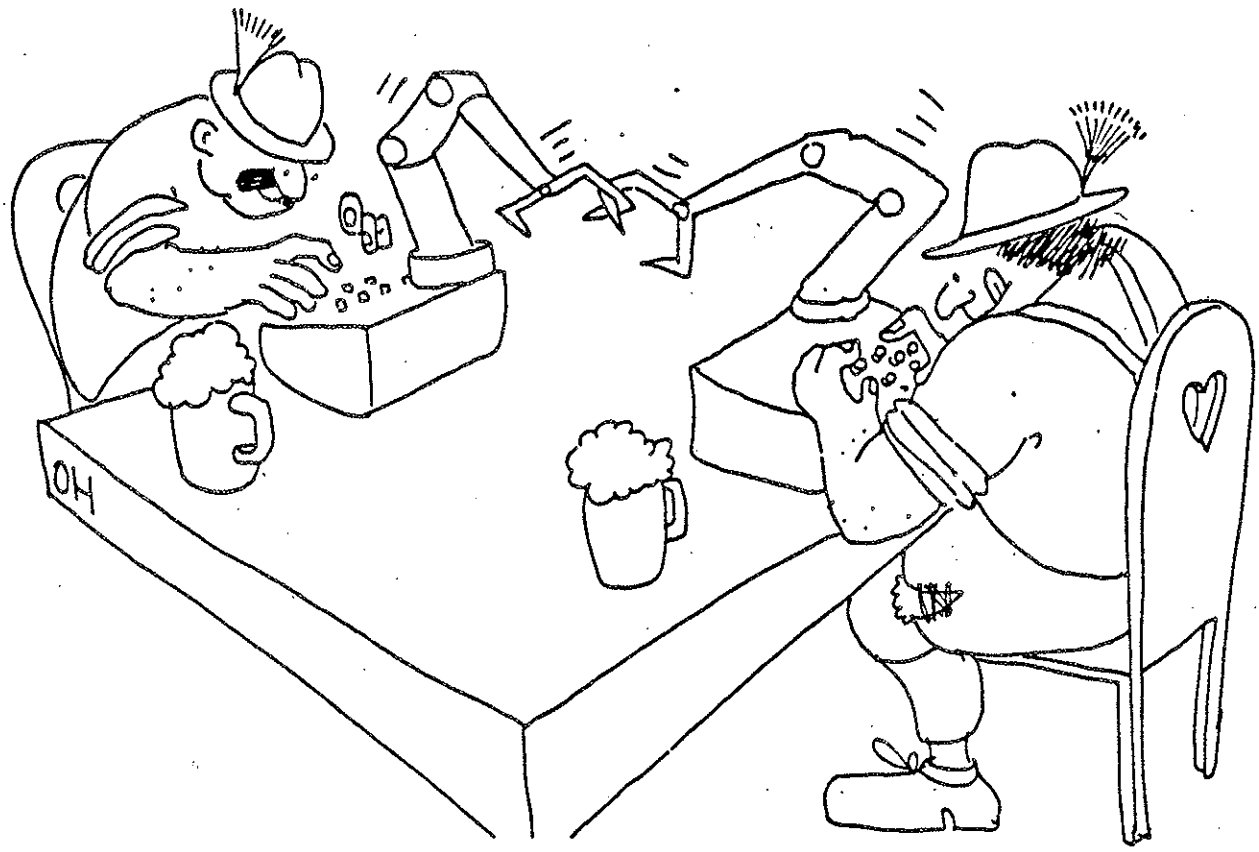
[3] ISO/TC 97/SC 5 N 893 Table of replies and comments submitted on DIS 7846

[4] RSX-11M Guide to Writing an I/O Driver, Order No. AA-2600E-TC

[5] RSX-11M/M-PLUS I/O Drivers Reference Manual, O.No AA-L677A-TC

[6] Ralph W. Stamerjohn, Up Your ACP (can be found on several SIG Tapes)

(The ISO standards can be obtained from the national standards organization, e.g. DIN, BSI, AFNOR, NNI etc)



INDUSTRIEROBOTERSTEUERUNG - eine "typische" IRTF-Anwendung

Bilderwitz aus dem Watzmann, Österreichs einziger satirischer Zeitschrift

DECUS-MÜNCHEN SIG-TREFFEN, HERBST 1984

Kurzbericht über die Herbst-Treffen der
deutschsprachigen DECUS-Mitglieder in München

W. Wöber mit Unterstützung von M. Oeschlmüller

WAS IST DECUS UND WAS SIND SIGs ?

Für alle Leser, die mit den Abkürzungen und Begriffen noch nicht so vertraut sind wie die 'alten Hasen', möchte ich mit einer kurzen Erklärung beginnen.

DECUS-München ist eine auf Vereinsbasis organisierte Gruppe der Benützer von Digital Equipment Corp. (DEC) Anlagen im deutschsprachigen Raum. Im Rahmen dieser Vereinigung werden regelmäßig Kurzveranstaltungen organisiert, wo der ungezwungene Informationsaustausch zwischen den Anwendern, aber auch zwischen Anwendern und Vertretern der Herstellerfirma, im Vordergrund des Interesses steht. Diese kurzen Treffen, von den einzelnen Special-Interest-Groups (SIGs) organisiert, wurden im Herbst 1984 zum ersten Mal als Blockveranstaltung an einem Ort abgehalten, um sowohl Reisekosten als auch Zeit zu sparen und den Informationsaustausch, auch zwischen Teilnehmern mit sehr unterschiedlichen Arbeitsgebieten, zu fördern.

Im Rahmen dieser Veranstaltung wurde eine Unmenge von Grundsatz- und auch Detailinformation umgesetzt und an den Mann gebracht. Deshalb möchte ich hier in diesem Beitrag nur auf die wichtigsten Themen eingehen, die vielleicht mittelfristig interessant sind.

PASCAL-SIG:

Diese Veranstaltung konzentrierte sich auf drei Hauptthemen: Der Einleitungsvortrag beschäftigte sich mit dem Thema 'Funktionale Programmierung' und brachte sowohl eine Begriffsbestimmung als auch eine sehr gelungene Überleitung zu Präsentationen und Diskussionen über Trends und Anwendungen von Künstlicher Intelligenz (Artificial Intelligence - AI). Ganz besonders wurde dabei auf die Möglichkeiten von Programmiersprachen für Artificial Intelligence (AI) wie zum Beispiel LISP eingegangen. Die Möglichkeiten sind faszinierend, allerdings ist momentan der Aufwand doch noch recht beträchtlich. Aber - haben wir dasselbe Gefühl nicht auch gehabt, als die ersten interaktiven Bildschirmprogramme oder symbolische Testhilfen eingesetzt wurden ?

Der nächste Schwerpunkt der Diskussion lag dann beim Thema 'Hilfsmittel für Entwurf und Abwicklung von Software-Projekten'. Besonders interessant war dabei die Vielfalt von Ideen und Standpunkten; allen Präsentationen gemeinsam war jedoch der Hinweis auf die Bedeutung von sauberen und durchdachten Lösungsansätzen. Es wurde eine Reihe von Produkten vorgestellt, die in der Entwurfsphase von Programmen und bei

der Abwicklung von Projekten helfen können, allerdings sind diese Dinge heute noch (wegen geringer Installationsbasis ?) recht kostenintensiv.

Der dritte Schwerpunkt der Präsentationen lag auf dem Einsatz von SW-Tools, also Programmbausteinen zur Arbeitserleichterung bei EDV-Aufgaben. Interessant war hier die Gegenüberstellung von Laufzeitsystemen für Echtzeitanwendungen auf PASCAL-Basis für PDP-11 Anlagen und für (Micro-)VAX Systeme. Der eigentliche Höhepunkt der Veranstaltung war aber dann doch die Präsentation der Probleme und Erfahrungen in einem immens großen Softwareprojekt durch Frau K. Hornbach von Lear-Siegler/Instrument-Division. Frau Hornbach ist Leiterin der amerikanischen Special-Interest-Group 'Languages and Tools' und bei Lear-Siegler verantwortlich für die Auswahl und den Einsatz von 'Software Werkzeugen'.

Den Abschluß der Veranstaltung bildeten dann noch zwei wichtige organisatorische Punkte. Zum ersten wurde der Name der SIG von PASCAL-SIG auf 'Special-Interest-Group für Sprachen und Werkzeuge' geändert, um dem sachlich doch über PASCAL hinausgehenden Einzugsbereich der Thematik gerecht zu werden. Weiters wurde ich selbst zum neuen Koordinator für diese SW-SIG gewählt, um die Nachfolge von Herrn Dr. Gliss vom Max-Planck-Institut Stuttgart anzutreten. Ich möchte im Namen der SIG auch hier Herrn Dr. Gliss für seine Aufbauarbeit danken; ohne seinen persönlichen Einsatz bis zuletzt hätte diese Gruppe sicher nicht so gut funktioniert.

GRAPHIK-SIG:

Zu dieser Veranstaltung ist grundsätzlich zu bemerken, daß die Qualität der Vorträge sehr unterschiedlich war. Es entstand der Eindruck, daß die Anwender die Probleme beim Einsatz von Graphik-Software im weitesten Sinne besser im Griff haben als die klassischen Anbieter in diesem Marktbereich.

Weiters wurde einstimmig festgestellt, daß 'weniger mehr gewesen' wäre, das heißt, eine Konzentration der Diskussion auf einige wenige Themen hätte sicher realere Ergebnisse gebracht. Trotzdem - alleine wegen der Präsentation des momentanen Standes der Normung auf dem Graphiksektor hat sich die Teilnahme schon gelohnt. Diese Informationen sind gerade in einer Phase des Aufbaues von Graphikdienstleistungen an den Anlagen der PRA sehr wichtig.

Im Bereich der Vorstellung neuer Graphikendgeräte hat besonders der Graphikschirm von Tektronix (4115B) gezeigt, welche Funktionalitäten in absehbarer Zeit - wenn auch sehr teuer - zur Verfügung stehen werden.

VAX-SIG:

Hauptthemen während der VAX-bezogenen Diskussionen waren 1) ein Überblick über momentane UNIX-Aktivitäten und 2) die Präsentation von neuen Hardware- und Software-Funktionalitäten.

UNIX, jeder hat davon gehört, aber man weiß dann halt doch nicht so ganz genau, ob das eigene Problem auch unter UNIX lösbar wäre. Daher war die Information über verschiedene Versionen, verschiedene Quellen und verschiedene Anwendungen sehr interessant. Ich hoffe, bald genauere technische Informationen für etwaige Interessenten parat zu haben.

Auf dem Gebiet neuer Geräte sind besonders überarbeitete Terminalmultiplexer zu erwähnen (DMZ-32, DHU-11), sowie die Verfügbarkeit von Disk-Controller Versionen für kleine bis kleinste MSCP-Platten, also z.B. für den Anschluß von 5 1/4 Zoll Floppies (RX-50). Der Softwaresektor war beherrscht von sehr wichtigen Vorinformationen bezüglich VMS Version 4. Im Mittelpunkt stand dabei die Präsentation von Erweiterungen in der Filestruktur (bis zu 38 Zeichen lange Filenames), Maßnahmen zur weiteren Erhöhung der Betriebssicherheit (neue, besser abgestimmte Benützerberechtigungen; 'Checkpoint-Facility', also die Möglichkeit, einen Prozeß an bestimmten Punkten wiederanlauffähig zu machen; erweiterte und komfortable Kommandos zur Druckersteuerung; u.s.w.). Weiters ist zu erwarten, daß mit VMS V4.0 die Kompatibilität zu RSX-11M V4.1, auch im Netzwerkbereich, wesentlich verbessert wird. Zusammenfassend kann gesagt werden, daß diese frühzeitigen Informationen ganz entscheidend dazu beitragen werden, den Übergang von VMS V3.6 zu VMS V4.0 für die Benützer recht problemlos vorzubereiten.

Den Abschluß dieser Veranstaltung bildete die nun schon traditionelle Diskussion mit der Geschäftsführung von DEC-Deutschland. Die während der Diskussion angesprochenen Probleme sind zwar für uns in Österreich nicht direkt relevant, es war aber recht nützlich zu sehen, daß die Probleme offensichtlich überall ziemlich ähnlich sind.

NETZWERK-SIG:

Diese Veranstaltung wurde in einem relativ neuen Umfeld abgehalten. Die Netzwerk-SIG hat sich nämlich entschlossen, keine hochspezialisierte Diskussion auszulösen, sondern gemeinsam mit den Teilnehmern der VAX-SIG und der RSX-SIG vor allem die täglich anfallenden Probleme zu besprechen und nach Möglichkeit auch einer Lösung näherzubringen.

Besonders interessant war die Präsentation von Erfahrungen bei der Installation und Inbetriebnahme eines relativ großen und heterogenen Netzwerkes auf der Basis von Ethernet. Solche sehr praxisorientierten Erfahrungsberichte sind sehr wertvoll, wenn es darum geht, bereits bekannte Fehler zu vermeiden.

Ein weiterer interessanter Tagesordnungspunkt war die Vorstellung von Terminalkonzentratoren für den Anschluß an Ethernet. Mit diesem Produkt (LAT-11) muß man sich offensichtlich noch näher beschäftigen.

RSX-SIG:

Bei den Vorstellungen der neuen Hardware und der neuen Softwaremöglichkeiten zeigte sich vor allem, daß die Zeit der explosiven Entwicklung vorüber ist. Der Schwerpunkt der Entwicklung liegt im Moment

ganz sicher im Bereich der VAX und ihrem Betriebssystem VMS. Doch sehe ich gerade diese Entwicklung sehr positiv, denn sowohl Systementwickler als auch Anwender von RSX-11M/S/MPlus gewinnen damit Zeit, sich auf die Erhöhung von Sicherheit und Benützerkomfort zu konzentrieren.

Eine ganze Reihe von neuen Produkten, anschließbar an den UNIBUS oder QBUS von PDP-11 Systemen wird, auch mit entsprechender Unterstützung des Betriebssystems, verfügbar sein:

LSI-11/73	Q-Bus CPU, basierend auf dem J-11 Chipset, Funktionalität der leistungstärksten jemals entwickelten PDP-11, nämlich der PDP-11/70. Leistung etwa bei PDP-11/44, FPP-Instructions im Grundmodell, Cachememory integriert auf der CPU-Platine. Diese CPU auf einem Dual-Board kann ohne andere Änderungen am System die LSI-11/23 ersetzen. Notwendig zum sauberen und problemlosen Betrieb ist RSX-Update-D.
LSI-11/73-Plus	Dieses CPU-Board ist fast identisch mit der LSI-11/73, allerdings wurden auf dem Quad-Board auch eine serielle Schnittstelle und Bootstrap ROMs vorgesehen. Diese Karte ist daher zum direkten Ersatz einer LSI-11/23-Plus CPU geeignet.
RD-52	Eine neue Version der 5 1/4 Zoll Winchesterplatten mit einer Kapazität von 31MByte, geeignet zum Anschluß an MSCP-Disk-Controller (RQDX1, RUDX1). Dieses Laufwerk kann an einem Kontroller mit RX-50, RD-50 und RD-51 Laufwerken gemischt werden.
RX-50	Die von der MicroPDP-11 bekannten 5 1/4 Zoll Floppies können jetzt auch an normalen Q-Bus und UNIBUS PDP-11 Systemen eingesetzt werden.
RC-25	Ein neues, altes Plattenkonzept: 2 Laufwerke zu je 26MByte auf einer gemeinsamen Spindel und mit gemeinsamen Schreib/Lese-Köpfen. Eine Kassette mit 26MByte kann wie eine normale Wechselplatte getauscht werden. Aber Vorsicht ist geboten: während des Plattenwechsels muß auch das zweite Laufwerk abgeschaltet werden.
TK-25	Mit diesem Gerät wurde erstmals ein konkurrenzfähiges Bandgerät für Kassetten-Backup vorgestellt. Verwendet werden die gut bekannten 1/4-Zoll Kassetten, der Kontroller arbeitet als Nachbildung des TS-11 Magnetbandes. Kapazität einer Kassette bis zu 60MByte.
DHU/DHV-11	Auf dem Kommunikationssektor wurden zwei neue Multiplexer für den Anschluß von seriellen Geräten präsentiert. Beide sind kompatibel mit

den früher oft eingesetzten DH-11 Multiplexern. der DHV-11 bringt 8 Leitungen mit DMA an den Q-Bus, der DHU-11 bringt dieselbe Leistung für den UNIBUS von PDP-11 oder VAX Systemen.

DEQNA — Nicht mehr ganz so neu ist der Ethernet Anschluß für Q-Bus Systeme. Allerdings wurde dieses Board in die Liste der Neuheiten mit aufgenommen, weil erst mit RSX-Update-D auch der notwendige Support von der Softwareseite existiert.

Weiters gaben einige RSX-Anwender ihre Erfahrungen bei der Leistungssteigerung von PDP-11 Systemen weiter. Vorgestellt wurden die Leistungsdaten der neuen 11/73 CPUs sowie die Möglichkeiten, ein RSX-11 System durch den Einsatz von 'Virtual Disks' oder 'Memory Disks' leistungsfähiger zu machen. Ebenso ausführlich diskutiert wurden die Unterschiede von UNIBUS, Q-18-Bus, Q-22-Bus und Einsetzbarkeit von Peripherigeräten von Fremdherstellern. Hier wurden einige sehr wesentliche Detailinformationen vermittelt, die viel Arbeit und Ärger ersparen werden.

Ein ebenso wichtiger wie interessanter Tagesordungspunkt war der Statusbericht für VAX RSX V1.0, also für das Nachfolgeprodukt des 'Compatibility-Mode' für VAX/VMS V4.0 (von manchen auch scherzhaft Incompatibility-Mode genannt). Dieses Produkt wird sicher in Zukunft eine noch bessere Integration der PDP-11-Umgebung mit der VAX-Funktionalität erlauben.

Während der 'Frage und Antwort Sitzung' wurden dann einige wichtige Fragen besprochen und zumeist auch beantwortet, sowie eine Liste von Verbesserungswünschen zusammengestellt. Diese Wunschlisten sind ein sehr wichtiges Instrument, um die Systementwickler über den Bedarf der Anwender gezielt zu informieren. Die Entwicklung von manchen heute in RSX-11 selbstverständlichen Dingen wurde auf diesem Weg angestoßen, deshalb ist hier Input von möglichst vielen Anwendern essentiell.

ZUSAMMENFASSUNG:

Abschließend kann gesagt werden, daß die Herbstveranstaltungen sowohl aus der Sicht der Veranstalter als auch aus meiner Sicht als SIG-Koordinator und Rechneranwender ein voller Erfolg waren. Alle Teilnehmer werden durch die neuen Informationen ihre Systeme effektiver einsetzen können.

Da verständlicherweise in so einer Zusammenfassung vieles nur sehr gekürzt weitergegeben werden kann, möchte ich an dieser Stelle die Leser daran erinnern, daß ich für genauere Auskünfte oder die Diskussion konkreter Fragen gerne zur Verfügung stehe.

HERRN DR.
W. KLEINERT
EDV-ZENTRUM
HYBRIDRECHENANLAGE

IM HAUSE