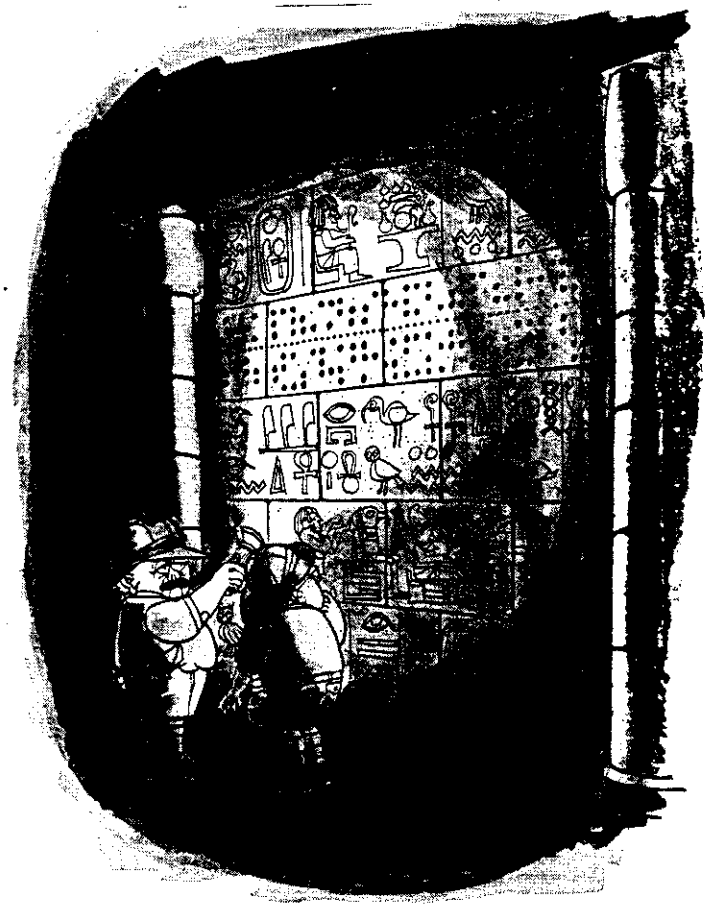
Handbuch 410

Version 1

Dokument Verarbeitung am MVS

410.10.25 - Text und Code



K. Daube

Oerlikon-Bührle
Rechenzentrum AG
Jungholzstr. 43
CH-8050 Zürich
Telefon 01/301 24 66

410.10 ALLGEMEINES ZUR DOKUMENT VERARBEITUNG

410.10.25 Text und Code

ABSICHTLICH LEER GELASSEN

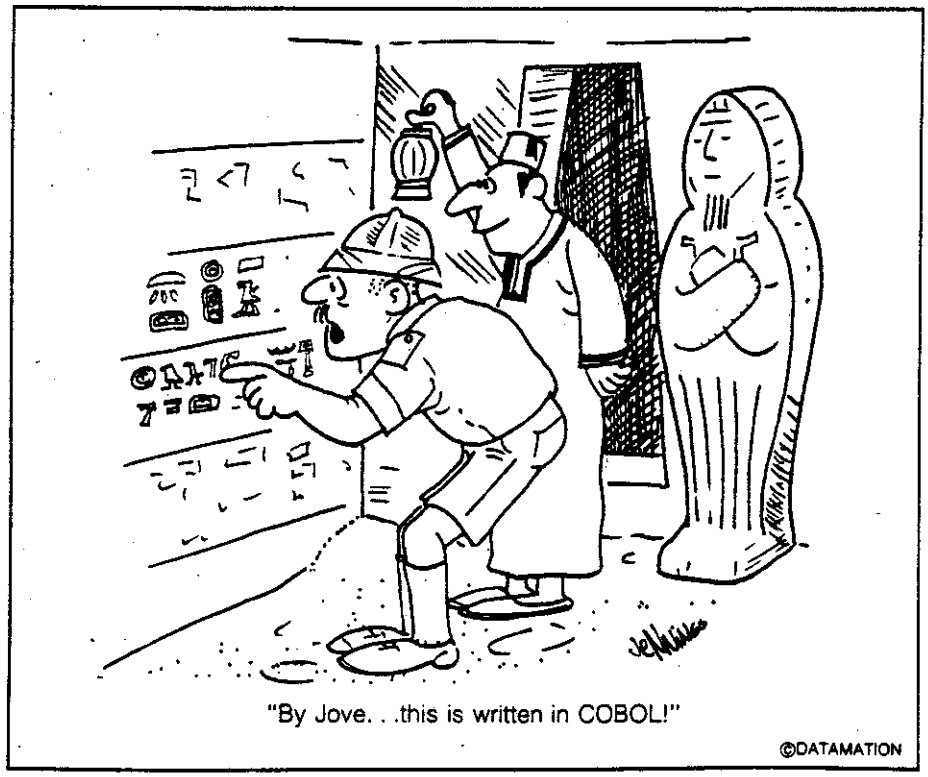
1 Einführung

1.1 Anlass

1986 wurden konzern-richtlinien erarbeitet, die regeln für den austausch von daten (insbesondere text und dokumente) festlegen. Darin wurde auch festgehalten, dass in der IBM umgebung die *code page 500* einzuführen ist. Die folgenden ausführungen sollen darlegen, dass

- richtlinien ein ziel vorgeben, über die länge des weges aber nichts gesagt werden kann
- auch nicht vorausgesagt werden kann, wie nahe das ziel erreicht werden kann
- der weg nicht nur von unseren eigenen anstrengungen bestimmt wird, sondern wir in einem sehr dynamischen umfeld leben
- wir uns weder warten auf die *ultima ratio* noch gewaltige daten konversionen leisten können
- wir also mit der unvolkommenheit leben lernen müssen.

Es muss heute auch festgehalten werden, dass bei der herausgabe der richtlinien die komplexität der probleme zwar erkannt, aber in den richtlinien nicht dargelegt wurden.



"By Jove. . .this is written in COBOL!"

©DATAMATION

National Language Support umfasst die folgenden eigenschaften von applikationen:

- Input und output-daten können alle zeichen umfassen, die für eine bestimmte sprache in einem bestimmten land notwendig sind. Zum beispiel erfordert Deutsch in Deutschland und Österreich das zeichen ß, in der Schweiz jedoch nicht.
- Der dialog mit dem benützer kann in einer von ihm gewählten sprache geschehen. Dies umfasst namen von anweisungen, abkürzungen, hilfe-texte, bildschirm-masken etc.
- Die applikation kann nationale usanzen berücksichtigen wie gruppierungszeichen in zahlen (CH: 17'300,-- USA: 17,300.--), form des datums, währungszeichen, sortierfolge etc.

In den folgenden ausführungen wird nur auf einen teilbereich der National Language Support problematik eingegangen, der **codierung von zeichen**.

1.2 Übersicht

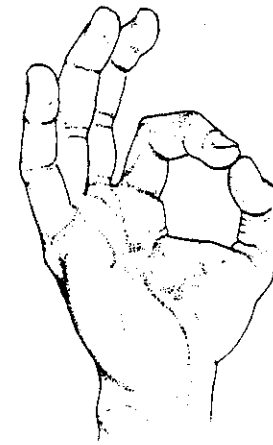
- Arten von codierungen
- Applikationen und code
- Missing links zur lösung der probleme
- IBM CH National Language Support cookbook
- IBM National Language Support design guide
- Weiteres vorgehen
- Code pages 500, 037, 850, ISO, /36 multinational und /36 German/Austria

2 Codierungen

2.1 Codes im täglichen Leben

Wir sind im täglichen leben von codes umgeben, ohne uns dessen bewusst zu sein:

- mimik und gesten
- pictogramme
- sprachregelungen
- schriftzeichen

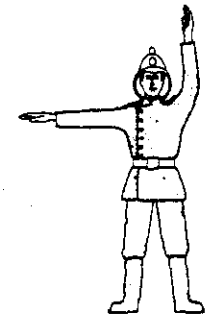


2.1.1 Gesten

Wir können mimik und gesten nur dann verstehen, wenn wir die situation kennen, aus der sie entstehen. Meist ist es sogar notwendig, den kulturkreis jenes menschen zu kennen, der uns mit einer geste gegenüber tritt [6]:

Die hier abgebildete geste hat beispielsweise sehr unterschiedliche bedeutungen:

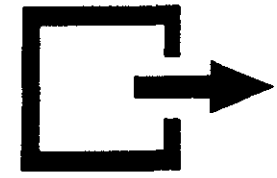
- prima, präzise (die geste zeigt das virtuelle festhalten von etwas sehr filigranem) - im deutschsprachigen raum
- geld - in Japan
- nichts, null, unbedeutend - in Frankreich
- obszöne andeutung - in Sardinien



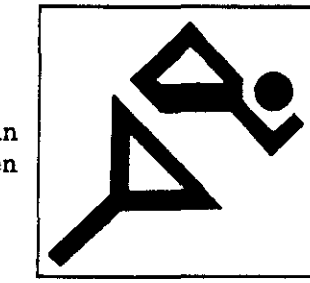
Andere gesten sind nur einer gruppe von spezialisten verständlich. Steht der feuerwehrmann auf einer kreuzung, werden verkersteilnehmern die geste als verkehrsregelung verstehen. Seine kameraden verstehen die geste aber genauer, sie bedeutet nämlich "druck senken".

2.1.2 Pictogramme

Pictogramme sind zwar abstraktionen von alltäglichen situationen, doch macht sich auch hier bemerkbar, dass die phantasie des erstellers sich nicht mit der phantasie des rezipienten deckt. Die phantasie ist vom erfahrungsschatz abhängig.



Das nebenstehende kann sicher immer als "ausgang" verstanden werden.



Dieses pictogramm ist ist schon etwas seltsam aufgebaut. Diese tafel an einer strasse durch den wald interpretierte ich lange zeit als: da kommen leute auf krücken... Gemeint sind aber gewöhnliche läufer.

2.1.3 Abkürzungen, Sprachregelung

Auch wenn wir in der informationsverarbeitung laufend mit abkürzungen zu tun haben: sie sind niemals eindeutig. Zudem wird verschiedenen worten mit der zeit ein anderer inhalt unterlegt. Zum teil werden sachverhalte bewusst verschleiert (siehe Newspeak von George Orwell).

AM meint auf einem radio: Amplituden Modulation. Auf einer uhr meint das aber: Ante Meridiam (vor dem mittag).

ACF Access Control Facility oder Advanced Communication Facility?

störfall meint meist einen unfall in einer (im allgemeinen grossen) technischen anlage.

verteidigunsministerium heisst es auch in jenen staaten, die sich recht aggressiv verhalten...

Auch für diese bereiche ist also unsere erfahrung, kenntnis der umgebung etc. wesentlich, wenn die "meldung" verstanden werden soll.

2.1.4 Schriftzeichen

Hier wird jedermann erkennen, dass es sich um codierungen von lauten oder bedeutungen handelt. Wer glaubt, eindeutigkeit sei hier trumpf, werfe einen blick auf die chinesischen zeichen, die alle drei žin = mensch bedeuten [9] oder die aussprache der amerikanischen bzw englischen wörter "lite", "light" und "enough" bzw "after". Das gehörte und auch das geschriebene kann oft nur im zusammenhang interpretiert bzw verstanden werden. Wie soll da ein automatischer prozess zurecht kommen?

2.2 Code und Datenverarbeitung

Die erläuterungen zu "codes im täglichen leben" machen deutlich, dass die lage in der datenverarbeitung sicher nicht einfacher sein kann, denn

> Das verständnis eines codes erfordert die kenntnis von abmachungen

und maschinen bzw programme sind heute noch bar jeden verständnisses über zusammenhänge. Zur definition des wortes "code" sei [7] zitiert. Dabei ist für uns interessant, dass die uns geläufige bedeutung erst am schluss erscheint...

Code [ko:d, frz.; kəʊd, engl.], 1) ⚖ Gesetzbuch. **Cinq Codes** [sẽk ko:d], frz. Gesetzessammlung unter NAPOLEON I.; dazu gehören der **C. civil (C. Napoléon)** [-siv'il, -napole'õ], frz. Zivilgesetzbuch, am 21. 3. 1804 veröffentlicht, das auch in Belgien und Luxemburg eingeführt wurde; in Baden galt der C. civil bis 1899 als **Badisches Landrecht; C. de commerce** [-dəkɔm'ɛrs], frz. HGB von 1807; **C. de procédure civile** [-dəprɔsed'yrsiv'il], frz. ZPO von 1806; **C. d'instruction criminelle** [-dẽstryksj'õ krimin'el], frz. StPO von 1808. Der C. de commerce ist durch zahlreiche Gesetzesnovellen in wesentl. Teilen aufgehoben und durch Einzelgesetze ersetzt worden. Der C. d'instruction criminelle ist durch den **C. de procédure pénale** [-pen'al] von 1957/58 abgelöst worden.

Uniform Commercial C. [j'u:nifɔ:m kəm'ə:ʃl-], das vereinheitlichte HGB der nordamerikan. Bundesstaaten mit Ausnahme des Staates Louisiana.

C. law system [-lɔ:s'istim], engl. Bez. für das kodifizierte kontinental-europ. Zivilrecht, i. Ggs. zum weitgehend ungeschriebenen anglo-amerikan. → Common Law.

2) Molekulargenetik: → genetischer Code.

3) 🔐 Verschlüsselungs-, Umsetzungsvorschrift, Chiffrierschlüssel; → Codierung.

Für die interne Zeichendarstellung bei Datenverarbeitungsgeräten werden unterschiedliche → Binärcodes (bit, → Byte) benutzt: z. B. EBCDIC (BCD), ASCII, Aiken, 3-Exzeß-C. Der BCD ist ein **Stellwert-C. Prüfbare C.** werden oft durch das Hinzufügen eines Parity-bits gebildet, wodurch ein Übertragungsfehler von 1 bit bemerkt werden kann, da die Zahl der ›1‹ in jedem Zeichen (un)gerade ist. Bei **reflektierendem (zyklischem, Gray-)C.** ändert sich nur ein bit beim Übergang von einem Zeichen zum benachbarten (→ Analog-Digital-Umsetzer). Beim Programmieren werden **Befehls-C.** als Schlüssel für → Befehle an den Computer gegeben (codieren).

In der datenverarbeitung sind codierungen an der tagesordnung und stark formalisiert. Wir verstehen unter code im allgemeinen die abbildung von einem zeichen auf die 8 bits eines bytes. Aber auch numerische werte werden codiert. Eine ganze zahl wird auf 16 oder 32 bit abgebildet etc.

Auch in diesem text werden besondere codes verwendet, um das nicht darstellbare sichtbar zu machen:

`<...>` stellt ein steuerzeichen dar. Eine applikation führt es aus, es wird in der ausgabe nicht dargestellt. Zum beispiel steht `<esc>` für das "escape" genannte steuerzeichen.

ABSICHTLICH LEER GELASSEN

3 Beschreibung eines Endzustandes

Als endzustand will ich hier das produkt einer aktion, einer handlung verstehen. Also etwa ein geschriebenes zeichen, als sichtbare auswirkung von strichen, die zu papier gebracht wurden. Oder ein am bildschirm gezeigtes zeichen, das hervorgehoben oder vergrössert wurde. Ein "endzustand" aus dem täglichen leben kann etwa eine speise sein.

Ein endzustand kann auf mindestens zwei arten beschrieben werden:

- Direkt durch einen namen
- Indirekt durch einen weg

In beiden fällen sind codes im spiel. Ich muss wissen, auf welchen territorium ich mich bewege! Sowohl benennungen als auch wegbeschreibungen erfordern voraussetzungen, grundkenntnisse und phantasie.

Um was geht es im folgenden?

- Châteauneuf-du-Pape
- Châteaubriand
- Château Chillon

Wer gerne einen guten wein trinkt, hält alles für weinsorten. Wer gerne gut isst, kann vielleicht sagen: Im Château Chillon habe ich letzthin ein Châteaubriand gegessen und dazu einen Châteauneuf-du-Pape getrunken.

Die wegbeschreibung zum Châteaubriand (das rezept) verwendet codes wie "marinieren", "saignant" oder "à point".

Chateaubriand

600-700 g Rindsfilet, z.B. Filetkopf Rindfleischmarinade	das Fleisch wenn möglich über Nacht marinieren. Vor dem Braten mit Haushaltpapier trocknen.
1 Teelöffel Salz 1 Esslöffel Sais Öl	das Fleisch mit Salz gut einreiben, mit Öl bepinseln, die Pfanne heiss werden lassen, das Fleisch rundum anbraten, Hitze reduzieren, das Fleisch soll jedoch immer noch braten. Von Zeit zu Zeit wenden und mit wenig übriggebliebener Marinade bestreichen. Bratzeit für Chateaubriand von ca. 5 cm Dicke: - bleu: etwa 7- 8 Minuten - saignant: etwa 12-14 Minuten - à point: etwa 17-18 Minuten - bien cuit: etwa 22-25 Minuten.

Tip
Das Fleisch am Schluss der Bratzeit aufstellen und die Enden leicht anbraten.

3.1 Endzustand benennen

- Der endzustand erhält einen namen (zb "kursives kleines zeichen PI" oder Châteaubriand)

IBM (und andere) wählten diese methode, weil für die meisten anwendungen grosse effizienz darin liegt, auf ein zeichen direkt zugreifen zu können, um es zu bearbeiten. Dies gilt aber **nur dann**, wenn folgende voraussetzungen gelten:

- Die codierung des files steht fest.
- Die applikation kennt die codierung (Was dann, wenn die applikation zwar text verarbeiten kann, π aber einfach als p behandelt?)

Es ist nicht damit getan, **anzunehmen**, dass alle daten im gleichen code zu einer applikation gebracht werden. Der von vielen als "standard EBCDIC" verstandene code ist nur der invariante teil der über 800 verschiedenen code pages!

Man glaubte auch, mit dieser methode rascher mit text umgehen zu können. So lässt sich die länge einer zeichenkette (angeblich) durch zählen der bytes bestimmen. Das gilt wohl für die verarbeitung im speicher, nicht aber zwangsläufig auch für den input bzw die darstellung an bildschirm und drucker (proportionale schrift, schrift-wechsel, grössen modifikationen, ...).

Hier kann vielleicht noch darauf hingewiesen werden, dass die proportionale schrift das "normale" ist (handschrift, buchdruck), die schriften fester teilung wegen der beschränkungen mechanischer mittel entstanden und somit "abnormal" sind.

In unserem zusammenhang wird der endzustand im allgemeinen **code page** genannt. Einer menge von 192 verschiedenen zeichen [1] werden codes zugewiesen. Diese code pages werden von IBM registriert und tragen daher eine identifizierende nummer (CPGID).

Eine teilmenge der in einer code page zusammengestellten zeichen wird coded graphic character set (identifiziert durch eine nummer - CGCSGID) genannt. Die teilmenge kann bis zur gesamtmenge auswachsen:

- Der zeichensatz der code page 500 ist identisch mit dem zeichensatz der code page 037 und hat den CGCSGID 00697. Dieser satz ist seit mitte 1987 auch vollkommen identisch mit dem von ISO 8859/1 (Latin alphabet number 1).
- Die code page 850 (mulitlingual code page von PS/2) enthält alle zeichen des CGCSGID 00697 als untermenge. Die volle code page 850 hat den CGCSGID 00980.

Die vielfalt an zeichensätzen und code pages stammt daher, dass man sich zunächst in einem 7-bit code rahmen bewegte (128 codes, davon 32 für steuerzeichen). Ausserdem gab es lange jahre hardware-beschränkungen, die diesen zustand zementierten.

In den meisten ländern wurde deshalb der versuchung nachgegeben, die codes "nicht gebrauchter" zeichen anders zu belegen:

USA	¢	|	!	$	`	#	@	{	}	\
Frankreich	°	!	$	$	µ	£	à	é	è	ç
Brasilien	È	!	$	Ç	ã	Õ	Ã	õ	é	\

1) Ein coderahmen umfasst 64 steuerzeichen und 190 → graphische zeichen. Dazu kommt das blank und ein zeichen, dessen code aus allen bits gebildet wird. Diesem ist keine ausdrucksform zugeordnet.

3.2 Weg zum Endzustand beschreiben

- Ausgangszustand muss bekannt sein (zb ASCII oder leerer küchentisch)
- Wegbeschreibung in form von codes (steuersequenzen bzw. rezept)

Die beste wegbeschreibung führt nicht zum ziel, wenn der falsche ausgangspunkt gewählt wird.

In der ANSI welt wird angenommen, dass jedes file mit ASCII beginnt. Die verschiedenen zeichensätze, hervorhebungen, grössenveränderungen etc werden mit codes (steuersequenzen) dargestellt. Diese sind alle genormt. Der inhalt der zeichensätze und ihre identifizierung ist festgelegt. Beispielsweise:

`<esc> ) E`	Als (erster) alternativer zeichensatz gilt ab jetzt NATS (NATS: Newspaper text transmission in Denmark and Norway). Der hex-code 5C stellt darin zum beispiel ein Ø dar. Im ASCII steht an dieser stelle ein \.
`<so>`	Umschalten auf den alternativen zeichensatz (NATS wird also aktiv).
`<si>`	Zurückschalten in den primären zeichensatz.
`<esc> [ 1 ; 3 m`	Die folgenden graphischen zeichen werden fett und kursiv dargestellt.

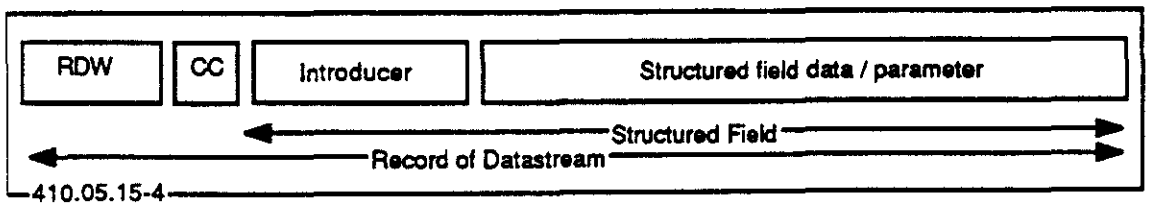
Nach dieser methode arbeiten praktisch alle applikationen der mini- und microcomputer. In OBRZ arbeitet VARIO damit.

Verwenden files diese methode, so müssen applikation dafür eingerichtet sein, alle genormten sequenzen zu akzeptieren (allenfalls einen input-fehler melden). Sie müssen sie aber nicht alle verstehen und verarbeiten können.

Eine applikation kann hier nicht direkt in ein file "eintauchen" und zb das 337. byte bearbeiten. Seine attribute (vergleichbar mit den koordinaten im gelände, höhenlage etc.) sind nicht bekannt, da der weg zu diesem zustand nicht bekannt ist. Solche daten müssen also immer als strom behandelt werden.

Während der bearbeitung dieses stromes müssen die relevanten stati festgehalten werden. Ein editor muss zb über die zeichenbreite und -höhe bescheid wissen. Auch über den zeichensatz muss er bescheid wissen, wenn zb forms drawing zeichen (zb ┌──┬──┐) durch besondere editor anweisung verarbeitet werden können (zb "zeichnen" mit dem cursor).

IBM greift diese methode neuerdings wieder auf mit den **structured fields**, die zb in SNA, DCA, IPDS etc verwendet werden.



Diese konstruktionen können verschachtelt werden. Im gegensatz zu ANSI steuersequenzen können **alle** bitmuster in den daten vorkommen, da bei der verarbeitung nicht nach einen *terminantor symbol* gesucht wird. Dies ist sehr relevant für applikationen mit rasterbildern.

Interessant ist in diesem zusammenhang, dass schriftzeichen sehr leicht automatisch erkannt werden können, wenn ihre bildung analysiert werden kann. Wenn etwa chinesische zeichen erfasst werden, während sie zu papier gebracht werden (da die reihenfolge der striche sehr formalisiert ist). Das erfassen eines musters während seiner entstehung ist offenbar einfacher als das vestehen des fertigen "endzustandes".

ABSICHTLICH LEER GELASSEN

4 Applikationen

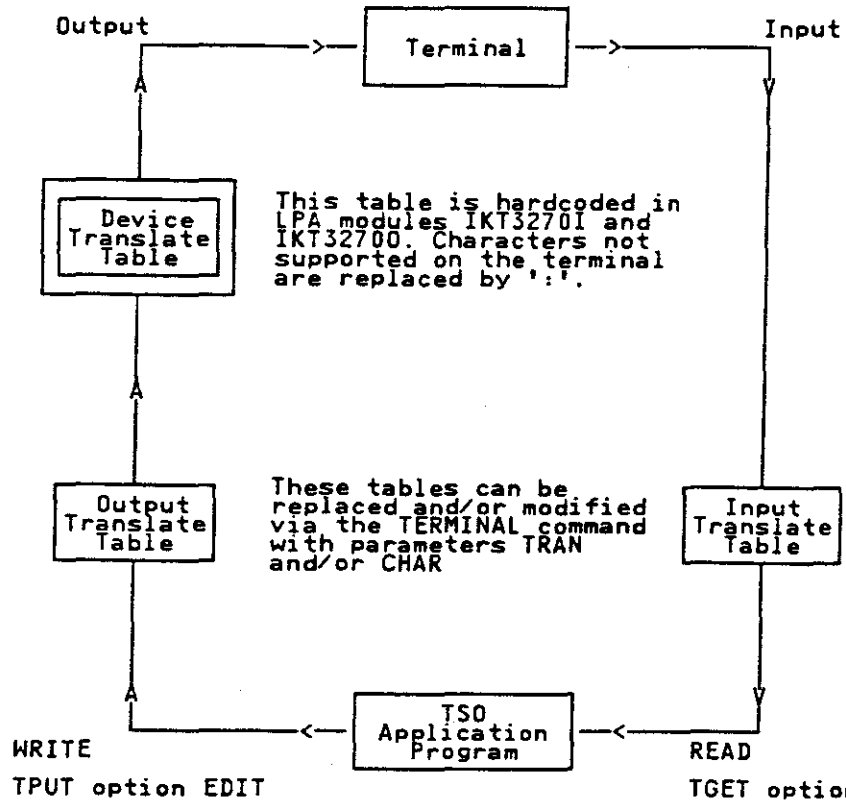
- Input aus der nicht system-welt
- Speicherung im system
- Output zur nicht system-welt

Jede applikation umfasst im allgemeinen diese drei aufgaben. Da die bits "wertneutral" sind, muss ihre bedeutung bekannt sein. Die 8-bit-folge B'110000011000010' kann den zahlenwert 171, die grossbuchtaben AB oder ein punktmuster oder sonst etwas bedeuten.

Im rahmen von datenbank applikationen wird man sich langsam klar, dass die attribute von daten nicht im programm festgelegt sein sollen, sondern in einer daten-beschreibung (dictionary). Diese erkenntnis muss bis in die tiefe der codierung angewandt werden:

> Der verwendet code muss den daten *eingeprägt* sein, eine applikation muss ihn erkennen können.

Mit dem aufkommen offener bzw vernetzter systeme sind auch die datenquellen vielfältig. Sie halten sich keinesfalls an den invarianten teil des EBCDIC. Die datenverwalter können wohl festlegen, welche codes auf ihren speichermedien vorkommen dürfen. Es kann aber nicht mehr vorgeschrieben werden, welcher code zb über die datenleitung aus Sidney kommt. Ich darf auch nicht einfach annehmen, dass alles, was aus Sidney kommt, einfach englisch ist und der "standard EBCDIC" (dem US-subset von code page 037) verwendet wird. Vielmehr können die daten von den Fidschi inseln stammen (wo offiziell französich gesprochen wird), und nur über Sidney transportiert worden sein.



Die vorstehende zeichnung verdeutlicht, welche code transformationen bei in- und output an einem trminal in der applikation TSO stattfinden. Dabei ist noch zu beachten, dass eventuelles anpassen der entsprechenden tabellen durch die TSO-anweisungen TERMINAL und CHAR beim aufruf des dialog systems ISPF verloren gehen!

Für die im OBK noch nicht vollständig eingeführte applikation "elektronische post" werden diese verhältnisse noch ausführlich dargelegt.

4.1 Eingabe von Daten

- Tastatur bestimmt zeichensatz
- Stellt einen ausschnitt aus einer code page dar
- In der Schweiz zunächst 107 zeichen, jetzt 131 zeichen
- Bildschirm kann oft mehr zeichen darstellen (107 → 116)

4.1.1 Zeichensatz

Unter zeichensatz versteht man eine menge von darstellbaren zeichen (bildschirm, drucker). Die menge dieser zeichen ist im allgemeinen kleiner als die gesamtmenge, die mit einem bestimmten code belegt werden kann.

Im EBDCIC kann eine code page 256 verschiedene codes enthalten (1 byte erlaubt 256 verschiedene bit-kombinationen). Davon sind 64 codes als steuerzeichen reserviert, 1 code (alle bit gesetzt) wird nicht mit einem graphischen zeichen besetzt. Das zeichen blank (leerstelle) ist ebenfalls nicht sichtbar. Somit verbleiben 190 sichtbare (graphische) zeichen in einer EBCDIC code page.

Verschiedene zeichensätze können sich derselben codierung bedienen. Dann liegt den entsprechenden zeichensätzen dieselbe code page zugrunde.

Da der verarbeitbare zeichensatz eines gerätes nicht 190 (192) zeichen umfassen muss, werden diese sätze von IBM numeriert (CGCSGID). Dies im hinblick auf die zukunft, wenn geräte sich identifizieren können. Dann könnte überprüft werden, ob das gerät die entsprechenden zeichen überhaupt darstellen kann...

4.1.2 Tastatur

Die tastatur an einem gerät (bildschirm) enthält im allgemeinen nicht genug tasten, um 190 verschiedene graphische zeichen zu produzieren. Eine bestimmte tastatur stellt einen bestimmten zeichensatz dar.

Die tastatur kann aber mit besonderen tasten susgerüstet sein (tot-tasten), mit deren hilfe zeichen kombiniert werden. Wenn zb die akzente aigu, circumflex und grave als tot-tasten definiert sind, können damit alle akzentuierten vokale generiert werden. Somit können mit 3 (akzente) + 5 (vokale) = 8 tasten deren 3 mal 5 = 15 zeichen bzw codes erzeugt werden.

In der 3270 terminal familie erzeugen tastatur und bildschirmcontroller codes, die an die applikation gesandt werden. Die beschränkungen auf weniger als 192 zeichen bei der 3270 terminal familie ist offenbar eine hardware beschränkung der älteren controller. In wirklichkeit kann die applikation nämlich nicht das terminal direkt ansprechen. Dazwischen liegt eine code-ebene begrenzter aussagekraft: der display code.

Die tastatur definition (erzeugter code) kann im allgemeinen an jedem gerät, das an einem 327x controller "hängt", anders sein. Die zuordnung von code und geräte-adresse ist fest in VTAM tabellen definiert. Ein "umstecken" von tastaturen bei bedarf ist also nich möglich.

4.1.3 Daten-input

Der zur applikation gelangende code ist in seiner art nicht identifiziert. Deshalb können an multi-user-applikationen wie zb CICS nur gleichartige tastaturen angeschlossen sein. Oder die applikation kann aufgrund einer benützer-kennung entscheiden, was für ein code daher kommt.

Im ISPF beispielsweise muss in einer tabelle eingetragen werden, an was für einem terminal gerade gearbeitet wird (menue 0.1). Dann können automatisch gewisse umformungen vorgenommen werden. Diese beteffen aber nur den input durch die tastatur und den output an den bildschirm. Was gespeicher wird, wird nach wie vor in keiner weise identifiziert.

Wechselt der benutzer aber das terminal, vergisst er nur zu gerne, diese definition zu ändern.

Viele applikationen sind nicht in der lage, code transformationen des inputs vorzunehmen. Dazu gehören etwa:

AS	! wird als command delimiter verwendet
Pascal	[und] haben zum glück die traditionelle ersatz-darstellung (. und .). Für { und } können ebenfalls ersatzdarstellungen verwendet: (* und *) werden.
PL/1	¬ und | sind operationszeichen
QMF	¬ wird in operationen verwendet
REXX	| und ¬ sind operationszeichen

Diese zeichen haben einen unterschiedlichen code, je nach der verwendeten tastatur und damit der angewandten code page.

Im rahmen einer applikation kann die identifizierung des codes schon heute durchgeführt werden, wie dies mit der applikation *SUSI* geschiet:

- Die tastatur wird via ISPF festgelegt. Dies legt den input code fest.
- Werden daten editiert, können sie entsprechend von cp500 in cp037 bzw in der anderen richtung umgesetzt werden (edit macro ADAPT).
- Die daten werden durch eine *SUSI* anweisung `.codepage xxx` identifiziert. Diese anweisung wird bei bedarf an den anfang der daten gestellt (edit macor ADAPT).

4.1.4 Systeme /36 und /38

Zur zeit der systemgenerierung kann festgelgt werden, mit welcher codierung gearbeitet wird:

- national character set
- multinational character set

Character set meint hier aber auch "code page". Die terminals sind von einem code in einen anderen umschaltbar. Doch werden auch hier die daten in keiner weise identifiziert. Die daten tragen kein attribut "code page".

Die für das betriebssystem global gemachte angabe über den verwendeten code kann auf dem /38 je terminal bzw drucker zusätzlich anders definiert werden. Dann erfolgen im system entsprechende code umformungen auf dem weg zwischen speicher und peripherie. Daten werden nur in der für das system festgelegten codierung abgelegt.

4.2 Speicherung von Daten

- Eine code page ist zu wenig (192 verschiedene zeichen)
- Identifizierung der daten notwendig
- Je nach applikation bis auf stufe *feld*

Wer zb einen technischen bericht verfasst, wird in den angebotenen code pages 037 bzw 500, ja sogar in 850 etliche zeichen vermissen. Im rahmen von applikationen kann auf meta-symbole zurück gegriffen werden, wie dies in *SUSI* geschieht. Das zeichen ∫ ist zb in keinem von IBM angebotenen zeichensatz enthalten. In *SUSI* wird dafür beispielsweise `$23$'$-23$` geschrieben (das zeichen ist im dritten "referenz" zeichensatz von *SUSI* definiert).

Es wäre nicht gut, wenn sich jede applikation eigene mechanismen zulegen müsste. Ein umschalten zwischen verschiedenen codes bzw zeichensätzen ist erst im rahmen des DCA definiert. Aber es gibt noch kaum applikationen, die dieses format unterstützen (DW/x). Ausserdem sind in diesen applikationen nur jene zeichen definiert, die in den PC-zeichensätzen vorkommen (zeichensatz 980).

Da sowohl zur input- wie zur output-zeit verschiedene codierungen vorkommen können bzw notwendig sind, muss der code von files erkennbar sein. Applikationen können dies nur innerhalb des files selber tun (siehe *SUSI*). Auf system-ebene gebotene mechanismen könnten dies in fileattributen tun (wie die DCB attribute).

4.2.1 Situation heute

Da die daten (auch auf den /3x systemen) kein attribut "code page" tragen, also nicht identifiziert sind, kann deren codierung nur in "nacht und nebel-aktionen" geändert werden. Dabei ist aber zu bedenken, dass ein file selten als ganzes aus text besteht. Vielmehr sind etliche felder als binär (numerische werte) in ihrem bitmuster zu belassen.

Utilities zur umformung können also nur dann angesetzt werden, wenn der aufbau von files bekannt und fest ist. "Hard coded" text in vielen "standard applikationen" entzieht sich der umformung, wenn kein quellencode vorhanden ist.

4.3 Ausgabe von Daten

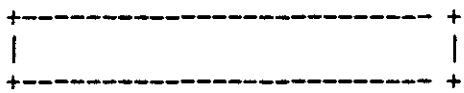
- Kann das gerät den code verarbeiten?
- Wird der code richtig dargestellt?

Kaum eines der heutigen IBM geräte kennt eine abfrage (query) funktion. Bei DEC reagieren die VT-terminals seit mehr als 10 jahren entsprechend. Deshalb konnten dort auch editoren geschrieben werden, die auf VT-100 nur 128 verschieden zeichen darstellen, auf VT-220 aber mit ladbaren zeichensätzen hantieren. Und jedes zeichen kann die unterschiedlichsten attribute (zb blinkend oder rot) und grössen (zb doppelt hoch) aufweisen.

Eine applikation kann also nicht feststellen, ob der vorliegende code auf dem gerät überhaupt verarbeitet werden kann. Aufgrund einer benutzer-aktion kann allenfalls eine tabelle zur filterung aktiviert werden. Aber hier gilt dasselbe, wie für ISPF unter "input" gesagt wurde: Die information kann falsch sein (bzw der benutzer weiss gar nicht, an was für einem gerät er arbeitet).

Sehr relevant ist die kenntnis der fähigkeit des output-gerätes, wenn nicht unterstützte zeichen durch substitute dargestellt werden könnten. Es wäre besser, statt nichts ein ae zu schreiben, wenn ein ä nicht zur verfügung steht. Oder

auf nicht intelligenten geräten als



darzustellen, statt das ganze weg zu lassen. Es erscheint mir wesentlich, dass die ausgabe möglichkeiten als erstes vervollkommnet werden, bevor neue datenquellen erschlossen werden.

Ein prinzip bei der datenausgabe sollte auch sein, dass undefinierte codes (code, für die auf einem bestimmten gerät kein zeichen definiert ist) erkennbar sind. Für solche codes sollte also nicht ein blank (leerstelle) erzeugt werden. Zum beispiel wird dem benützer bei der folgenden zeile sofort klar, dass undefinierte zeichen verwendet werden:

```
IF (A ■ B) CALL (xyz, abc, 'why ■', on)
```

Was aber die folgende zeile bedeuten soll, ist absolut unklar

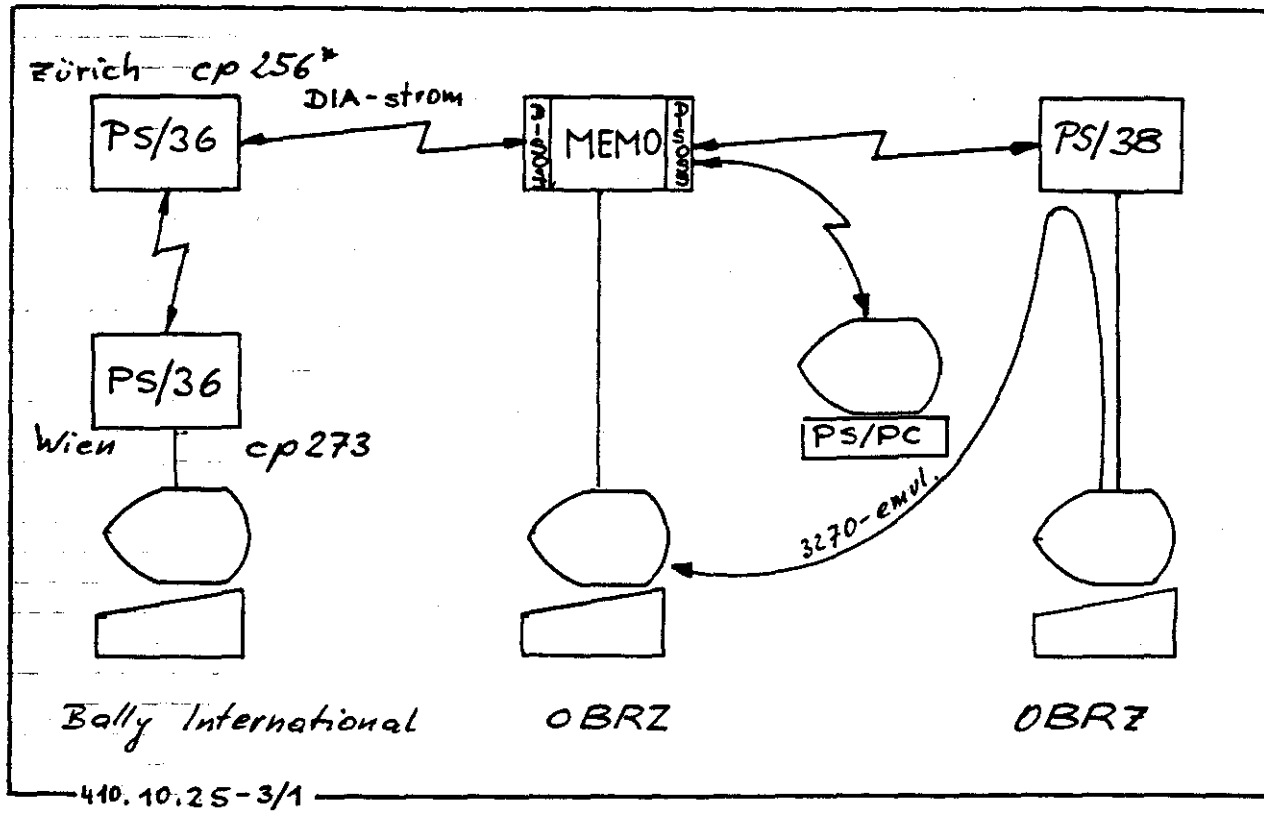
```
IF (A   B) CALL (xyz, abc, 'why  ', on)
```

Oder (noch schlimmer), was mit sogenannten IBM defaults passiert:

```
IF (A - B) CALL (xyz, abc, 'why -', on)
```

Die zentralen laserdrucker am OBRZ sind zb noch nicht in der lage, den gesamten zeichensatz "Swiss" darzustellen. Aus performance- oder anderen gründen war es notwendig, die zahl der verschiedenen zeichen unter 128 zu halten. Deshalb fehlen zb einige akzentuierte zeichen (á, ç, í, ñ, ó, ú, ÿ, É, Ñ). Der VSM zeichensatz umfasst 131 zeichen.

4.4 Die Applikation "Elektronische Post" mit MEMO



Bezüglich der verwendeten codierung in diesem netz gilt:

- Bally Wien arbeitet am S/36 mit dem Austrian/German character set (code page 273).
- Bally International in Zürich arbeitet am S/36 mit dem multinational character set. Dieses ist bis auf wenige (für uns) unwesentliche zeichen mit der code page 256 identisch.
- DISOSS arbeitet mit der default code page 256. Derzeit sind alle bekannten umformungstabellen (ca 40) installiert.
- MEMO arbeitet vollkommen transparent, das heisst, kein code wird irgendwie verändert.
- S/38 bei OBRZ arbeitet mit dem "multinational character set". Beim terminal durchgriff (3270 emulation) werden deshalb "falsche" codes verwendet. Büro /38 arbeitet nicht mit dem multinational character set.

Detailprobleme können hier (noch) nicht dargelegt werden. Zu vieles konnte noch nicht verifiziert werden. An der oberfläche ist folgendes sichtbar: Bally International in Zürich schicke eine meldung mit dem text "Grüße an André" an verschiedene empfänger. Diese werden folgendes bild an ihren schirmen sehen:

Grüße an André an allen bei Bally International in Zürich ans /36 angeschlossenen stationen.

Grüße an André an allen in Wien ans /36 angeschlossenen stationen.

Gr}¯e an Andr? an allen am OBRZ/MVS angeschlossenen bildschirmen. Die stelle mit dem ? wird an den ITT bildschirmen im OBRZ als é dargestellt, an den IBM-bildschirmen vermutlich als punkt, an anderen schirmen wieder anders. Dies hängt von der konfiguration des bildschirm-kontrollers ab.

Gr}¯e an Andr? an den S/38 bildschirmen des OBRZ, die sich als terminals am MVS verhalten.

Es muss hier daran erinnert werden, dass die anscheinend mit MEMO auftauchenden probleme nicht diesem produkt anzulasten sind. Auch wenn an DISOSS über PS/370 verschiedene terminals (verschiedene tastatur - und damit verschiedener code) angeschlossen werden, tritt dasselbe

ABSICHTLICH LEER GELASSEN

5 Missing Links

Die vorhergehenden abschnitte zeigen deutlich, dass zu einer umfassenden lösung mindestens zwei dinge gehören:

- Identifizierung der verwendeten codierung
- Fähigkeiten der peripherie sind abfragbar

Bis IBM "saubere" lösungen anbieten kann, stehen nur "krücken" in den applikationen zur verfügung. Diese krücken müssen aber auf die mutmassliche zukünftige entwicklung rücksicht nehmen. Deshalb muss diese entwicklung anhand von veröffentlichungen und implementierungen von

- SAA strategie der IBM
- X.400 dienste der öffentlichkeit
- etc.

beobachtet werden.

5.1 Identifizierung der Codierung

Die verwendete codierung kann nach folgenden methoden den files "eingeprägt" werden:

- *structured fields* ist die methode der wahl (zb DCA, IPDS)
- applikationsbezogenes arbeiten (beispiel *SUSI*)

Eine "saubere" lösung besteht nur für den DCA-bereich, also die DW/x applikationen. Das angebot an zeichensätzen ist aber noch mager, die zahl der unterstützten geräte (output) gering. Insbesondere fehlen verbindungen zur "klassischen datenverarbeitung".

Auf der outputseite macht IPDS gebrauch von identifizierungen. Ich hoffe sehr, dass im rahmen von SAA diese prinzipien eingang in die gesamte IBM software findet. Nur ist das natürlich ein jahrhundertwerk. Lösungen müssen immer auch die bestehenden situationen beachten - insbesondere, wenn sie von "zentraler" stelle kommen.

Für *SUSI* werden die files in sich identifiziert. Zusammen mit der applikation ISPF/PDF funktioniert das ganze recht gut. Die methode steht und fällt aber mit der identifizierung der eigenschaften eines input gerätes.

Eine generelle taktik wäre das verbot von zeichen, die ausserhalb des sogenannten **syntactic set** liegen. Dieser zeichensatz umfasst

Zeichentyp	Zeichen
Alphabetisch	ABCDEFGHIJKLMNOPQRSTUVWXYZ abcdefghijklmnopqrstuvwxyz
Numerisch	1 2 3 4 5 6 7 8 9 0
Sonderzeichen	. , : ; ? ( ) ' " - _ & + % * = < >

Allgemein kann dieses verfahren sicher nicht empfohlen werden. In Einzelfällen (zb weltweite elektronische post) werden wir sicherlich damit leben lernen müssen.

Um der problematik *identifizierung von code* nachdruck zu verschaffen, laufen zumindest im rahmen des SEAS [2)] folgende requirements (die alle von OBRZ lanciert wurden, aber breite unterstützung fanden):

- Identifizierung von files mit der code page in DOS bzw OS/2 (eintrag im directory)
- Identifizierung von files mit der code page im MVS (zb DCB-attribut)
- VS Fortran compiler muss alle zeichen einer code page im quellentext verarbeiten können.
- VS Pascal compiler muss alle zeichen einer code page im quellentext verarbeiten können.

IBM intern ist es vor allem eine grosse aufgabe des Toronto National Language Technical Centre, die 25'000 software entwickler mit der problematik vertraut zu machen [1, 2].

2) SEAS = SHARE European Association, eine vereinigung von benützern von MVS- und VM-system

5.2 Abfrage der peripheren Fähigkeiten

Da eine wirkliche anfrage bei der peripherie heute nicht möglich ist, bedienen sich bestehende applikationen verschiedener "krücken":

- ISPF kann vom benutzer "eingestellt" werden
- GDDM kennt einen rudimentären query
- *SUSI* übernimmt die angaben vom ISPF

Es besteht heute keine möglichkeit, von einem input-gerät der 3270 familie zu erfahren, mit welchem code es arbeitet. Diese angabe liegt zwar im controller vor, doch sie kann nicht abgefragt werden.

Deshalb wird diese information in applikations-bezogenen tabellen oder allenfalls im VTAM geführt, muss aber nicht mit der realität übereinstimmen. Denn die terminial situation ist recht dynamisch geworden.

Auch in bezug auf die ausgabe (drucker, bildschirme) ist die situation gleich. Über komplizierte definitionen, die absolut wartungs-unfreundlich sind, muss einer applikation mitgeteilt werden, welche möglichkeiten ihr für die output gestaltung gegeben sind.

Für die drucker hat sich IBM daher im rahmen des SAA auf IPDS festgelegt. SAA kann nicht auf "dumme" geräte wie zb zeilendrucker aufbauen. Für die bidldschirme deutet die weiterentwicklung der controller, insbesondere aber der übergang auf "workstations" (PC's) als terminals ebenfalls den trend zu mehr intelligenz an. Aber gerade dadurch wird eine vielfalt an möglichkeiten geboten. Solange die "potenz" nicht erfahrbar ist, muss also vom kleinsten gemeinsamen nenner ausgegangen werden!

ABSICHTLICH LEER GELASSEN

6 Empfehlungen von IBM

6.1 IBM Cookbook CH

> If national keyboards are to be introduced on an existing network, careful analysis and planning is required in order to avoid severe end user and data integrity problems.

[3,4] legen nahe, vorsichtig zu werke zu gehen. In [3] wird ausgeführt, wie eine installation für verschiedene applikationen auf den "Schweizer zeichensatz" 00908 (131 zeichen) der code page 500 umgestellt werden können:

DFSORT Vorschlag zu einer sortier-methode für code page 500.

GDDM mit ICU und QMF sowie PS/370 und APL2-applikationen: modifikationen an translate tables und vector symbol set.

ISPF und PDF Anpassung verschiedener tabellen bzw einführen zusätzlicher tabellen. Diese arbeit wurde von DTA bereits (ohne kenntnis der IBM unterlagen) früher vorgenommen.

JES328X JES3/328x Print Facility ist bei OBRZ nicht eingesetzt.

PL/1 Aufforderung an den benützer, andere zeichen zu verwenden.

QMF Aufforderung an den benützer, andere zeichen zu verwenden. ISPF und GDDM müssen auf Swiss umgestellt werden.

SDSF Spool Display and Search Facility: so viel ich weiss, ist dies bei OBRZ nicht eingesetzt.

TSO/E TSO/VTAM und TSO/session manager müssen umgestellt werden

Die darlegungen im FSC cookbook machen klar, dass ausser ISPF/PDF kein produkt in der lage ist, mit mehr als einer codierung (code page) zu hantieren! Beim sort kann von aussen durch eine procedur nachgeholfen werden. Eine umstellung auf den zeichensatz Swiss bzw die code page 500 muss deshalb sehr sorgfältig bedacht und geplant werden.

Nach wie vor scheint es mir nur möglich, applikationsbezogen vorzugehen, gegebenfalls die code page abhängigen programm-module in entsprechenden bibliothen mehrfach zu führen. Es bleibt aber immer noch das problem, wie den programmen bzw proceduren mitgeteilt wird, welche code page sie bearbeiten sollen - und dies möglichst ohne benutzer-intervention!

6.2 National Language Support Design Guide

Dieses dokument war bis mitte 1987 "IBM internal use only", ist nun aber frei verfügbar [1,2]. Es gibt richtlinien an die IBM entwicklungs-mannschaft, wie produkte zu gestalten sind, damit sie für mehrere mehrere sprachen und länder tauglich sind. Die anforderungen sind in form von regeln formuliert und umfassen (neben anderen) die bereiche:

- Character sets and code pages
- General design considerations
- Electronic character generation for displays and printers
- Keyboard and keystroke processing
- Machine readable information
- Panels and messages
- National usage considerations
- Terminology - grammatical and style sensitivity

Diese arbeit unter leitung von Denis Garneau vom IBM Toronto Lab wurde wesentlich beeinflusst durch das SEAS White Paper der National Character Task Force, an dem OBRZ sowohl inhaltlich wie in der ausführung grossen anteil hat.

7 Weiteres Vorgehen

Aus heutiger sicht muss eine strategie zur lösung der skizzierten probleme die verschiedensten bereiche berühren:

- Benützer (EDV kommitee etc) über die tragweite der code-problematik informieren.
- Output auf vollen zeichensatz 00697, eventuell 00980 ausbauen.
- Nur noch "multilingual" 3270-controller (bzw äquivalente) beschaffen.
- Werkzeuge (compiler etc) mehr-code-fähig machen.
- Neue applikationen bzw grosse änderungen richten sich nach [2]
- Druck auf IBM ausüben (SEAS, GUIDE, kauf-verhandlungen).
- "Schweizer Inseln" anstreben, volle umstellung unrealistisch.
- Ausrichten der massnahmen auf X.400

7.1 Allgemeines

Der vorliegende text sollte klar gemacht haben, dass die problematik um zeichensätze und codes vielschichtig ist. Saubere lösungen können nicht ohne den willen der hersteller (in unserem fall in erster linie IBM) gefunden werden.

Lange zeit werden keine echten lösungen möglich sein, sondern nur mittel zur vermeidung der ärgsten unzukömmlichkeiten. Diese müssen aber im hinblick auf das ziel gewählt werden, damit die zukunft nicht verbaut wird. Die wahl dieser massnahmen erfordert deshalb ein dauerndes beobachten der weiteren entwicklung auf diesem gebiet. Dies heisst auch, dass getroffene massnahmen eventuell wieder geändert werden müssen.

Für massnahmen können nur zielrichtung, nicht aber detaillierte anweisungen gegeben werden. Extreme forderungen, wie zb ersatz aller "dummen" terminals durch "intelligente" PCs (damit code umwandlungen transparent vorgenommen werden können), sind unrealistisch. Ebenso muss die forderung nach umformen aller bestehender datenbestände von der hand gewiesen werden. Allein der zeitbedarf einer solchen aktion (geschweige denn vom personellen aufwand) würde einen geordneten betrieb während dieser aktion verhindern.

IBM ist "sales driven" wie jede andere firma auch. Es muss deshalb über alle möglichen kanäle gewicht auf die forderungen des National Language Support gelegt werden. Die gängige haltung, dass "man" sich ja bisher auch arrangierte, ist im zeitalter der endbenutzer zynisch. Requirements werden gehört! Es bereitet natürlich auch aufwand, sie zu verfassen und einzubringen - der prozess bringt aber auch klarheit darüber, welche anforderungen wirklich vorliegen.

OBRZ ist abhängig von der "globalen" entwicklung. Es sollte daher kein aufwand in offenbar kurzfristige übergangslösungen gesteckt werden. Statt übersetzungstabellen dem momentanen gusto anzupassen, sollten sie gänzlich transparent gemacht werden (wie zb in MEMO). Nur so kann die vielzahl an derzeit vorhandenen "filtern" vermindert werden und später an geeigneter stelle ein automatismus eingeführt werden.

7.2 Empfehlungen

Als erstes sollten die output-möglichkeiten geschaffen werden, bevor neue datenquellen erschlossen werden. Denn sonst kann der vorgelagerte prozess nie kontrolliert werden.

Bildschirme mit ihren controllern sollten abfragbar sein - und sei das durch eine anweisung aus der applikation heraus! Dasselbe muss für output geräte gelten, die nicht einen definierten funktionsumfang haben (PostScript drucker haben alle einen vollen funktionsumfang; IPDS drucker weisen unterschiedliche gruppen von fähigkeiten auf).

Bei der beschaffung neuer geräte (insbesondere bildschirm controller) muss darauf geachtet werden, dass jedes angeschlossene terminal mit einem anderen code (und anderer tastatur) arbeiten

 ./.

kann. Denn wenn am gleichen controller programmierer und text-bearbeiter sitzen, laufen deren forderungen einander zuwieder.

Damit applikationen umgestellt werden können, müssen zuerst die werkzeuge umgestellt werden. Compiler müssen allenfalls mit precompilern ausgerüstet werden, damit files unterschiedlicher codierung verarbeitet werden können. Für DELTA sollte das eigentlich ein klacks sein!

Neue applikationen müssen sich unbedingt an die von IBM ausgearbeiten regeln [2] halten, um verschieden codierungen und sprachen verarbeiten zu können. Beachten dieser regeln gibtbestmögliche verträglichkeit mit dem von IBM verfolgten weg. Deshalb sollten sich auch grössere änderungen an diese regeln halten.

Ich halte es für unzweckmässig, alle DV aktivitäten auf die code page 500 umzustellen. Die gesamte systemarbeit sollte davon ausgenommen werden können. Daher meine vorstellung des applikationsbezogenen vorgehens. Dies heisst aber andererseits, dass mindestens zwei codierungen nebeneinander bestehen - nicht nur vorübergehend.

7.3 Vorgehensweise

Beim einsatz von produkten darf der zweck der beschaffung nicht aus den augen verloren werden.

MEMO wurde nur für "elektronische post" beschafft. Wenn also probleme beim filetransfer auftreten, ist es unsinnig, diese (mit unerhörtem aufwand) beheben zu wollen. Für den problemkreis "elektronische post" - der im allgemeinen zu weit gefasst wird, sind die anforderungen zu stufen:

- Geringste anforderungen an die übertragunsqualität für den austausch von meldungen über verschiedene netzwerk-knoten. In diesem fall ist eine Beschränkung auf den "syntaktischen zeichensatz" als zulässig erachten.
- Diese anforderungen können für "lokale" verbindungen - wenn sender und empfänger am gleichen netzwerk-knoten mit gleicher hardware (bildschirm) arbeiten - erhöht werden.
- Diese anforderungen können auch für verbindungen über verschiedene netzwerk-knoten erhöht werden, wenn dieselben "end-systeme" eingesetzt werden (zb verbindung von Bally Österreich mit Bally International in Zürich).

8 Appendizes

8.1 Code page 500/1

Im sommer 1987 wurde der zeichensatz der code page 500 in vier zeichen an den zeichensatz von ISO 8859/1 angepasst. Deshalb wird diese code page seit dann 500/1 genannt.

Der zeichensatz der vollen code page 500/1 trägt den CGCSGID 697

	4x	5x	6x	7x	8x	9x	Ax	Bx	Cx	Dx	Ex	Fx
x0	space	&	-	ø	Ø	°	µ	¢	{	}	\	0
x1	req. sp	é	/	É	a	j	~	£	A	J	÷	1
x2	â	ê	Â	Ê	b	k	s	¥	B	K	S	2
x3	ä	ë	Ä	Ë	c	l	t	•	C	L	T	3
x4	à	è	À	È	d	m	u	©	D	M	U	4
x5	á	í	Á	Í	e	n	v	§	E	N	V	5
x6	ã	î	Ã	Î	f	o	w	¶	F	O	W	6
x7	å	ï	Å	Ï	g	p	x	¼	G	P	X	7
x8	ç	ì	Ç	Ì	h	q	y	½	H	Q	Y	8
x9	ñ	ß	Ñ	`	i	r	z	¾	I	R	Z	9
xA	[	]	¦	:	«	ª	¡	¬	syl. hy	1	2	3
xB	.	$	,	#	»	º	¿	|	ô	û	Ô	Û
xC	<	*	%	@	ð	æ	Ð	¯	ö	ü	Ö	Ü
xD	(	)	_	'	ý	¸	Ý	¨	ò	ù	Ò	Ù
xE	+	;	>	=	þ	Æ	Þ	´	ó	ú	Ó	Ú
xF	!	^	?	"	±	¤	®	×	õ	ÿ	Õ	all bits

`space` space, blank, Leerstelle

`req. sp` required space, notwendige Leerstelle

`syl. hy` sylable hyphen, mögliche Trennstelle, dargestellt als -

`all bits` x'ff', alle Bits gesetzt

8.2 Code page 037

code page 037 ist das US-äquivaltent zu code page 500. Die in der datenverarbeitung üblichen zeichen

gross- und kleinbuchstaben, ziffern, sowie die sonderzeichen { } [ ] / \ ^ ~ ! ` ¢ : . $, # < * % @ ( ) _ ' + ; > = | ¬ ? "

sind so codiert, wie sie programmierer aus dem "Reference Summary" der System /370 architektur kennen. Alle compiler sind zb für diese codierung ausgelegt.

Der zeichensatz der vollen code page 037 trägt den CGCSGID 697

	4x	5x	6x	7x	8x	9x	Ax	Bx	Cx	Dx	Ex	Fx
x0	space	&	-	ø	Ø	°	µ	^	{	}	\	0
x1	req. sp	é	/	É	a	j	~	£	A	J	÷	1
x2	â	ê	Â	Ê	b	k	s	¥	B	K	S	2
x3	ä	ë	Ä	Ë	c	l	t	•	C	L	T	3
x4	à	è	À	È	d	m	u	©	D	M	U	4
x5	á	í	Á	Í	e	n	v	§	E	N	V	5
x6	ã	î	Ã	Î	f	o	w	¶	F	O	W	6
x7	å	ï	Å	Ï	g	p	x	¼	G	P	X	7
x8	ç	ì	Ç	Ì	h	q	y	½	H	Q	Y	8
x9	ñ	ß	Ñ	`	i	r	z	¾	I	R	Z	9
xA	¢	!	¦	:	«	ª	¡	[	syl. hy	1	2	3
xB	.	$	,	#	»	º	¿	]	ô	û	Ô	Û
xC	<	*	%	@	ð	æ	Ð	¯	ö	ü	Ö	Ü
xD	(	)	_	'	ý	¸	Ý	¨	ò	ù	Ò	Ù
xE	+	;	>	=	þ	Æ	Þ	´	ó	ú	Ó	Ú
xF	|	¬	?	"	±	¤	®	×	õ	ÿ	Õ	all bits

space space, blank, Leerstelle

req. sp required space, notwendige Leerstelle

syl. hy sylable hyphen, mögliche Trennstelle, dargestellt als -

all bits x'ff', alle Bits gesetzt

8.3 Code page 850

Die code page 850 ist eine von vieren, die mit den PS/2 definiert wurden. Gegenüber den früheren "PC-code pages" 860 (für Portugal), 863 (Canada-French), 865 (Norwegen) und 437 (für USA und den rest der welt) fehlen alle formular-zeichen, welche von doppelter linie in einfache linie übergehen.

Der zeichensatz der vollen code page 850 trägt den CGCSGID 980. Da der zeichensatz 697 vollständig im zeichensatz 980 enthalten ist, wird empfohlen, auf PS/2 diese code page zu verwenden.

Hex Digits 1st → 2nd ↓	0-	1-	2-	3-	4-	5-	6-	7-	8-	9-	A-	B-	C-	D-	E-	F-
-0		►		0	@	P	`	p	Ç	É	á	▒	└	ð	Ó	-
-1	☺	◄	!	1	A	Q	a	q	ü	æ	í	▒	┴	Ð	ß	±
-2	☻	↕	"	2	B	R	b	r	é	Æ	ó	▒	┬	Ê	Ô	=
-3	♥	‼	#	3	C	S	c	s	â	ô	ú	▒	├	Ë	Ò	¾
-4	♦	¶	$	4	D	T	d	t	ä	ö	ñ	┤	─	È	õ	¶
-5	♣	§	%	5	E	U	e	u	à	ò	Ñ	Á	┼	ı	Õ	§
-6	♠	▬	&	6	F	V	f	v	å	û	ª	Â	ã	Í	µ	÷
-7	•	↨	'	7	G	W	g	w	ç	ù	º	À	Ã	Î	þ	¸
-8	◘	↑	(	8	H	X	h	x	ê	ÿ	¿	©	╚	Ï	Þ	°
-9	○	↓	)	9	I	Y	i	y	ë	Ö	®	╣	╔	┘	Ú	¨
-A	◙	→	*	:	J	Z	j	z	è	Ü	¬	║	╩	┌	Û	·
-B	♂	←	+	;	K	[	k	{	ï	ø	½	╗	╦	▉	Ù	¹
-C	♀	∟	,	<	L	\	l	|	î	£	¼	╝	╠	▬	ý	³
-D	♪	↔	-	=	M	]	m	}	ì	Ø	¡	¢	═	¦	Ý	²
-E	♫	▲	.	>	N	^	n	~	Ä	×	«	¥	╬	Ì	´	■
-F	☼	▼	/	?	O	_	o	⌂	Å	ƒ	»	┐	¤	▬	'	

8.4 ISO 8859/1

Diese code page enthält den gleichen zeichensatz wie code page 500/1. Die volle bezeichnung lautet:

ISO 8859/1.2: Information processing - 8-bit single byte coded graphic character sets - Part 1: Latin alphabet No. 1.

	2x	3x	4x	5x	6x	7x	Ax	Bx	Cx	Dx	Ex	Fx
x0	space	0	@	P	`	p	num. sp	°	À	Ð	à	ð
x1	!	1	A	Q	a	q	¡	±	Á	Ñ	á	ñ
x2	"	2	B	R	b	r	¢	²	Â	Ò	â	ò
x3	#	3	C	S	c	s	£	³	Ã	Ó	ã	ó
x4	$	4	D	T	d	t	¤	´	Ä	Ô	ä	ô
x5	%	5	E	U	e	u	¥	µ	Å	Õ	å	õ
x6	&	6	F	V	f	v	¦	¶	Æ	Ö	æ	ö
x7	'	7	G	W	g	w	§	·	Ç	×	ç	÷
x8	(	8	H	X	h	x	¨	¸	È	Ø	è	ø
x9	)	9	I	Y	i	y	©	¹	É	Ù	é	ù
xA	*	:	J	Z	j	z	ª	º	Ê	Ú	ê	ú
xB	+	;	K	[	k	{	«	»	Ë	Û	ë	û
xC	,	<	L	\	l	|	¬	¼	Ì	Ü	ì	ü
xD	-	=	M	]	m	}	syl. hy	½	Í	Ý	í	ý
xE	.	>	N	^	n	~	®	¾	Î	Þ	î	þ
xF	/	?	O	_	o	delete	¯	¿	Ï	ß	ï	ÿ

Die Kolonnen 0x bis 7x sind identisch zu ASCII bzw ISO 646. Die Kolonnen 0x, 1x, 8x und 9x sind reserviert für Steuerzeichen

`space` space, blank, Leerstelle

`num. sp` numeric space, Leerstelle in Zahlen

`syl. hy` sylable hyphen, mögliche Trennstelle, wird als - dargestellt

`delete` Steuerzeichen delete

8.5 /36 Multinational Character Set

Diese code page stimmt im grossen und ganzen mit dem vorläufer von code page 500, der code page 256 überein.

	4x	5x	6x	7x	8x	9x	Ax	Bx	Cx	Dx	Ex	Fx
x0	space	&	-	ø	Ø	°	µ	¢	{	}	\	0
x1	req. sp	é	/	É	a	j	~	£	A	J	÷	1
x2	â	ê	Â	Ê	b	k	s	¥	B	K	S	2
x3	ä	ë	Ä	Ë	c	l	t	Pt	C	L	T	3
x4	à	è	À	È	d	m	u	©	D	M	U	4
x5	á	í	Á	Í	e	n	v	§	E	N	V	5
x6	ã	î	Ã	Î	f	o	w	¶	F	O	W	6
x7	å	ï	Å	Ï	g	p	x	¼	G	P	X	7
x8	ç	ì	Ç	Ì	h	q	y	½	H	Q	Y	8
x9	ñ	ß	Ñ	`	i	r	z	¾	I	R	Z	9
xA	[	]	¦	:	«	ª	¡	¬	syl. hy	¹	²	³
xB	.	$	,	#	»	º	¿	|	ô	û	Ô	Û
xC	<	*	%	@	ð	æ	Ð	¯	ö	ü	Ö	Ü
xD	(	)	_	'	≤	¸	Ý	¨	ò	ù	Ò	Ù
xE	+	;	>	=	þ	Æ	Þ	´	ó	ú	Ó	Ú
xF	!	^	?	"	±	¤	®	×	õ	ÿ	Õ	all bits

`space` space, blank, Leerstelle

`req. sp` required space, notwendige Leerstelle

`syl. hy` sylable hyphen, mögliche Trennstelle, dargestellt als -

`all bits` x'ff', alle Bits gesetzt

8.6 /36 Austrian/German Character Set

Gegenüber dem "Multinational Character Set" weisen die folgenden zeichen eine abweichende codierung auf:

{ } [ ] ß ä Ä ö Ö ü Ü ¨ @ § ¦ \

	4x	5x	6x	7x	8x	9x	Ax	Bx	Cx	Dx	Ex	Fx
x0	space	&	-	ø	Ø	°	µ	¢	ä	ü	Ö	0
x1	req. sp	é	/	É	a	j	ß	£	A	J	÷	1
x2	â	ê	Â	Ê	b	k	s	¥	B	K	S	2
x3	{	ë	[	Ë	c	l	t	Pt	C	L	T	3
x4	à	è	À	È	d	m	u	©	D	M	U	4
x5	á	í	Á	Í	e	n	v	@	E	N	V	5
x6	ã	î	Ã	Î	f	o	w	¶	F	O	W	6
x7	å	ï	Å	Ï	g	p	x	¼	G	P	X	7
x8	ç	ì	Ç	Ì	h	q	y	½	H	Q	Y	8
x9	ñ	¨	Ñ	`	i	r	z	¾	I	R	Z	9
xA	Ä	Ü	ö	:	«	a	¡	¬	syl. hy	1	2	3
xB	.	$	,	#	»	o	¿	|	ô	û	Ô	Û
xC	<	*	%	§	ð	æ	Ð	¯	¦	}	\	]
xD	(	)	_	'	≤	¸	Ý	¨	ò	ù	Ò	Ù
xE	+	;	>	=	þ	Æ	Þ	´	ó	ú	Ó	Ú
xF	!	^	?	"	±	¤	®	×	õ	ÿ	Õ	all bits

`space` space, blank, Leerstelle

`req. sp` required space, notwendige Leerstelle

`syl. hy` sylable hyphen, mögliche Trennstelle, dargestellt als -

`all bits` x'ff', alle Bits gesetzt

8.7 Glossarium

Derived from [1,3].

ANSI American National Standards Institute.

ASCII American National Standard Code for Information Interchange. A coded character set consisting of 128, 7-bit characters. There are 32 control characters, 94 graphic characters, the space character and the delete character.

CECP Country Extended Code Page. The extended code page of countries with code pages based on a western latin alphabet.

CGCSGID Coded Graphic Character Set Global IDentifier. A number given to a set of graphic characters like 00697 for the set of characters contained in code page 500.

coded character set A specific set of bit patterns to which specific graphic meanings and control meanings have been assigned. This is synonymous with *code*.

code page A specification of code points for each graphic character in a set or in a collection of graphic character sets. Within a code page, a code point can have only one specific meaning.

control character A specific bit pattern with an assigned control meaning. Contrast with *graphic character*.

CPGID Code Page Global IDentifier. The number of the code page given by IBM. → code page 500 or code page 037.

EBCDIC Extended Binary Coded Decimal Interchange Code. A coded character set consisting of 8-bit coded characters.

GCGID Graphic Character Global IDentifier. A name for a graphic like LA010000 for the letter a, or ND021000 for ^{2} (superscript 2).

graphic character ISO: A character, other than a control character, that is normally represented by a graphic.

invariant character set (1) A character set, such as the Syntactic Character Set, that does not change from code page to code page. (2) A minimum set of characters that is available as part of all character sets.

IPDS Intelligent Printer Data Stream. Datastream consisting of *structured fields* to control a printer. Functions are grouped into classes. A specific printer normally can perform only a set of function classes, not the total set of functions specified in IPDS.

language ISO: A set of characters, conventions, and rules, that is used for conveying information. The three aspects of language are pragmatics, semantics, and syntax.

NLS National Language Support. The ability for a user to communicate with applications in a language other than US English.

8.8 Literatur

[1] National Language Information and Design Guide Volume 1: Designing enabled Products, Rules and Guidelines; IBM publication SE09-8001

[2] National Language Information and Design Guide Volume 2: Left-to-right Languages and double byte Character Set Languages; IBM publication SE09-8002

[3] IBM Switzerland FSC Cookbook: Swiss Terminal / Swiss Character Set Support; Ulrich Abderhalden. Available on request from SE at Zürich.

[4] Unterlagen zur tagung vom 28.10.87: National Language Support in IBM Hardware und -Software; W. Dormann / U. Abderhalden.

[5] ISO 8859/1: Information processing - 8-bit single byte coded graphic character sets - Part 1: Latin alphabet No. 1

[6] Desmond Morris: Der Mensch, mit dem wir leben [Droemer & Knaur 1978]

[7] Der Grosse Brockhaus, Jubläumsausgabe 1978

[8] Ernst Doblhofer: Zeichen und Wunder - Die Entzifferung verschollener Schriften und Sprachen [dtv 1964]

[9] Carl Faulmann: Das Buch der Schrift [Kaiserlich-Königliche Hof- und Staatsdruckerei Wien 1880] reprint 1985 durch Greno, Nördlingen 1985