SERIES OF TECHNICAL PRESENTATIONS RELATING TO THE
DIGITAL DOCUMENT INTERCHANGE FORMAT

BABBAGE AUDITORIUM, SPIT BROOK ROAD FACILITY

JULY 9, 10, 11

Wednesday July 9

9:30 Review of DDIF status
Review/update of agenda

10:00 Introduction to DDIF — Bob Travis

DDIF has been under development at Digital for over a year, and forms the basis for at two current document processing development efforts. This introductory presentation will describe DDIF's syntax and semantics.

11:30 lunch break

13:00 DDIF/DDIS VMS RTL Access — Bill Laurune

Two utility packages will be described. The first package provides low level access to DDIS TLV-encoded data. The subroutines will provide basic encoding functions (such as length field construction) and syntax checking based on dynamically loaded syntax tables.

The second routine package is DDIF specific, and provides the service of maintaining the current context of DDIF segment attributes.

The DDIS access routine package is scheduled for submission to the VAX/VMS Run-Time Library in V5.0. The DDIF access routines are not currently scheduled for RTL submission, but will be available for by DEC applications.

14:20 coffee break

14:40 WPSPLUS/DECPAGE logical access — Bruce Taylor

The Logical Access Layer (LAL) is planned as a functional interface to DDIF documents. It provides primitives for data editing, hierarchical navigation, searching and attribute resolution. For each open document, the LAL maintains a tree representing structure and content. If the document is large, not all of this data is represented explicitly at a given time; but from the caller's point of view, the entire document appears to be "in memory".

This talk will describe a high-level design for the LAL and its internal data structures, together with two related components: the Cache Manager, which controls paging of the data structure; and the Physical Access Layer, which converts between the internal format and DDIF.

16:00 DDIS Character Sets — Tom Hastings

DDIS is the basic data syntax upon which DDIF is based. It defines the encoding of all binary and character items. Fundamental to any DDIF application are the text character sets; DDIS defines both 8- and 16-bit character set encodings. This session will describe the on-going process to define these character sets, and the pending issues surrounding them.

17:00 end of day

Thursday July 10

9:30 Status of standards in computer graphics & documents — Dr. J. Schönhut et al.

Under the auspices of an external research grant, Fraunhofer-Gesellshaft and the Technical University at Darmstadt researchers have been doing some investigations into the application of DDIF to existing and future graphics applications. All of today's sessions are related to various aspects of that research, and will be presented by Dr. Jürgen Schönhut and some of his associates from the Fraunhofer-Gesellshaft.

10:30 Compatibility of graphics stds: PHIGS, GKS, etc. — Schönhut et al.

11:30 lunch break

13:00 PDC: A device independent graphics interface to DDIF/LAL — Paul Wong

14:00 Interfaces and data formats for transfer and communications in computer graphics systems — Schönhut et al.

15:00 coffee break

15:20 text models in computer graphics, Videotex and documents — Schönhut et al.

16:00 Comparison of DDIF, SGML, ODA/ODIF — Schönhut et al.

17:00 end of day

Friday July 11

9:30	Role of DDIF in the Compound Document Processing Strategy	Jim Kapadia
10:30	SLIDEX: an example for integration of text and graphics	Schönhut et al.
11:30	end of seminar	

CURRENT DDIF STATUS:

- "24-OCT" version used by WPSPLUS and SARAH prototypes
- "14-JAN" version now in use in SARAH field test
- Several improvements ade since then; targetted for Sum ıer release
- Formal Review Process being established in SSG and DS

Introduction to DDIF

Bob Travis

July 9, 1986

What is DDIF?

DDIF is a storage and interchage format for compound documents in revisable form. The primary purpose of DDIF is to serve as a medium of exchange for revisable documents between Digital compound document processors. But many documen creating applications will simply store use files in DDIF in order to greatly simplify the interchange process, and also because DDIF has been designed to be an efficient representation for revisable compound documents.

It will be a formal DEC standard after it has been successfully implemented and released in at least one product.

What is a Compound Document?

A compound document is a unified collection of data that may be edited, formatted, or otherwise processed as a document. A compound document is likely to contain a number of integrated components, including proportionally spaced text in various renditions and styles, positioning and formatting parameters, abstract synthetic graphics, and scanned images. In the future, other forms of data representation may be tightly integrated with these document components.

DDIF's Goals:

- Completeness

o high quality text and graphics
o structure
o suitability for supporting application integration

- Ease of Processing

o clear syntax
o usable on small systems

- Ease of Maintenance

o self-describing syntax
o provision for upward migration and version tracking

- Common Encoding for Interchange, Mail

- DDIS domain

Key Points:

- Based on DDIS
- Hierarchical Structure
- Generic Elements
- Text, Graphics, Image, 'Foreign' data
- Links to external content and attribute info

Review of DDIS:

- TLV encoding
- Formal meta-syntax for Domains
- Defines basic data ele ents, and constructors
- Based on international standards (X.409, ANS-1)

DDIF DOCUMENT :: =

Descriptor

identifies this as DDIF, also what version

Header

Title, author, dates, fo s, external files,
page layouts, print parameters, etc.

Content

Hierarchical segment structure, with content and attributes

Segment:

- BeginSegment element
  - Contains all segment attributes
- Series of content primitives and/or nested segments
  - Provides the actual co ent of the segment
- EndSegment element
  - No added information; just terminates the segment

Content Primitives:

Text strings — 8- and 16- bit character sets

Directives — Line, page, block, tab, layout

Graphics — Multipoint, arc, spline

Image — Bi-tonal, grayscale multi-spectral

Macro-reference — Includes standard content, transformed

Other — PLP, Domain, Private

Text:

LATIN1_STRING — the DEC-standard multi-national set

TEXT_8 — first byte of string selects 8-bit set

TEXT_16 — first byte-pair selects 16-bit set

Directives:

Hard/soft text — integer value, selects Page, Line, etc

Page Layout — integer valu , selects new page layout

Graphics:

Multipoint — sequence of points, draw pattern, draw/fill/mark/reg/close flags

Arc — center, radius, eccentricity, start/extent, rotation, draw/fill/pie/close flags

B_Spline — sequence of points, order, draw/fill/close flags

Image:

— sequence of planes, with coding attributes, pixels
— rich variety of coding schemes, component ordering
— compatible with CCITT FAX for bitonal

Other content:

Macro Reference — with transformation

PLP — included 'pictures' in existing protocols: ANSI, GKS, etc

Domain, Private — application data, not directly rendered

Coding Example:

```
ARC_DEF ::= [APPLICATION 11] IMPLICIT
        SEQUENCE {
    Arc_Center_X [0] IMPLICIT MEASURE,
    Arc_Center_Y [1] IMPLICIT MEASURE,
    Arc_Radius_X [2] IMPLICIT MEASURE,
    Arc_Radius_Delta_Y [3] IMPLICIT
        MEASURE DEFAULT 0,
    Arc_Start [4] IMPLICIT ANGLE DEFAULT
        0,
    Arc_End [5] IMPLICIT ANGLE DEFAULT
        21600,
    Arc_Rotation [6] IMPLICIT ANGLE
        DEFAULT 0,
    Arc_Flags [7] IMPLICIT INTEGER {
        Draw_Arc(1), Fill_Arc(2), Pie_Arc(4),
        Close_Arc(8) } DEFAULT 1 }
```

Begin Segment element:

ID — for reference from other segments

User Label — for application reference via UI

Generic Reference — for indirect attribute specification

Local Attributes — for direct attribute specification

Element Definitions — supply source for generic attrs and macros

Structure Description — supplies rules for segment construction

Generic Referencing:

[Parent segment
    Element definitions: X, Y

[...
[Child segment
    Generic reference = X]
...]
]

Attribute Inheritance:

[Parent segment
Local attributes: text font1, colors 1/0, other (initial)

[Child 1
Local attributes: font2]

[Child 2
nolocal attributes]

]

Attributes:

Semantic Tags — indicate paragraph, chapter, etc
Frame — clipping rectangle, relocation, transformation
Color Map — in RGB
Text — font, color, margins, tabs
Character — orientation, alignment
Line — thickness, color, end, pattern
Fill — pattern, color
Marker — selection, color
Conversion — format of section numbers etc.
Computed Content — numbers, cross-reference, external
Page Layout — selects page layout
Language — selects linguistic support
Image — type, orientation, etc

Page Layouts:

- in Header
- indexed from content and/or attributes
- sequence of BLOCKS
- each block either predefined content or poured from content
- filled in sequence
- each page layout can link to successor

Transformations:

- Sequence of Transformation_Primitive

Transformation_Primitives:

- translation, rotation, skew, scaling

Recently Added:

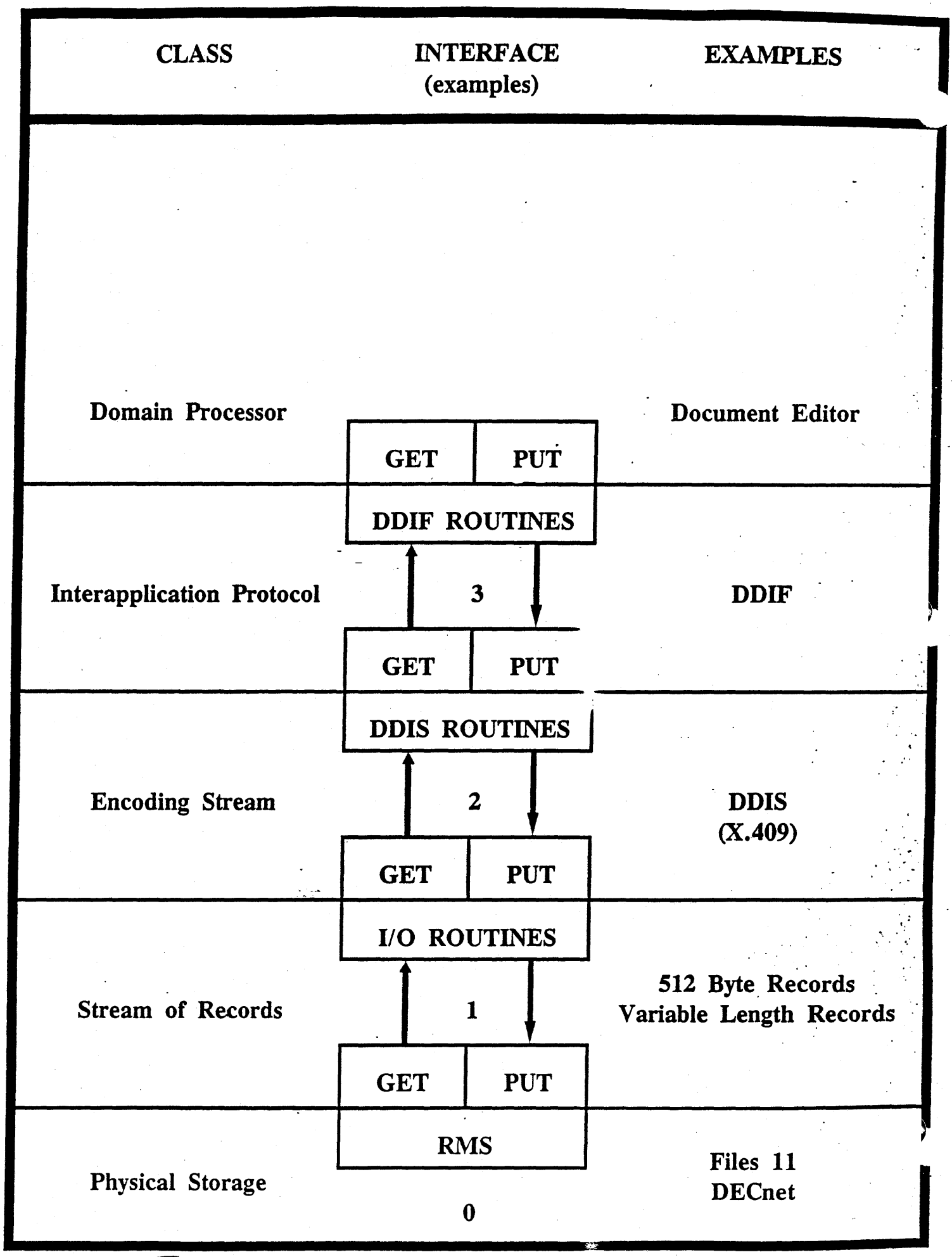
- Image content and attributes
- Hierarchical generic/content definitions

Work in Progress:

- Improved layout for annotation and footnotes
- Table specification, including link to data/spreadsheet
- Review for use in engineering drawing domain

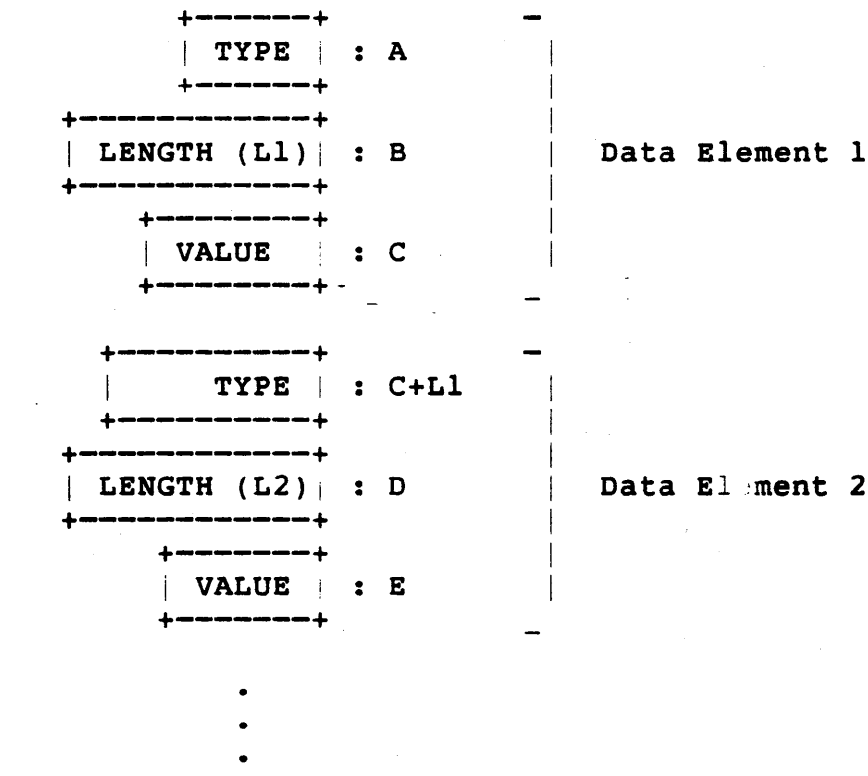
BILL LAURUNE

DDIS ACCESS ROUTINES


CLASS
INTERFACE
(examples)
EXAMPLES
Domain Processor
Document Editor
GET
PUT
DDIF ROUTINES
Interapplication Protocol
3
DDIF
GET
PUT
DDIS ROUTINES
Encoding Stream
2
DDIS
(X.409)
GET
PUT
I/O ROUTINES
Stream of Records
1
512 Byte Records
Variable Length Records
GET
PUT
RMS
Physical Storage
0
Files 11
DECnet

DEC DATA INTERCHANGE SYNTAX
(STREAM OF DATA ELEMENTS)

Figure Drawing Convention: Memory locations are shown with increasing addresses running right to left & top to bottom. If this data were to be transmitted over a communication line, the order of transmission is low-address bit/byte first.



TYPE Field - variable length (1 .. 4 bytes)

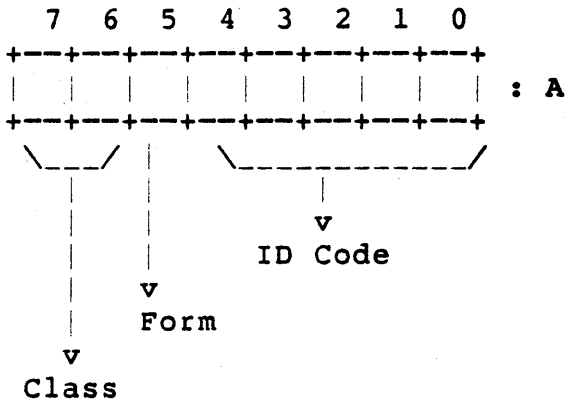
LENGTH Field - variable length (1 .. 5 bytes)

VALUE Field - variable length (0 .. 2**32-1 bytes for primitive & counted constructor,

unlimited for uncounted constructor)

FIGURE 1

1-BYTE TYPE FIELD



Class Bits - TYPE <7:6>

00 = Universal
01 = Application wide
10 = Context-specific
11 = Private (Customer/OEM)

Form Bit - TYPE <5>

0 = Primitive
1 = Constructor

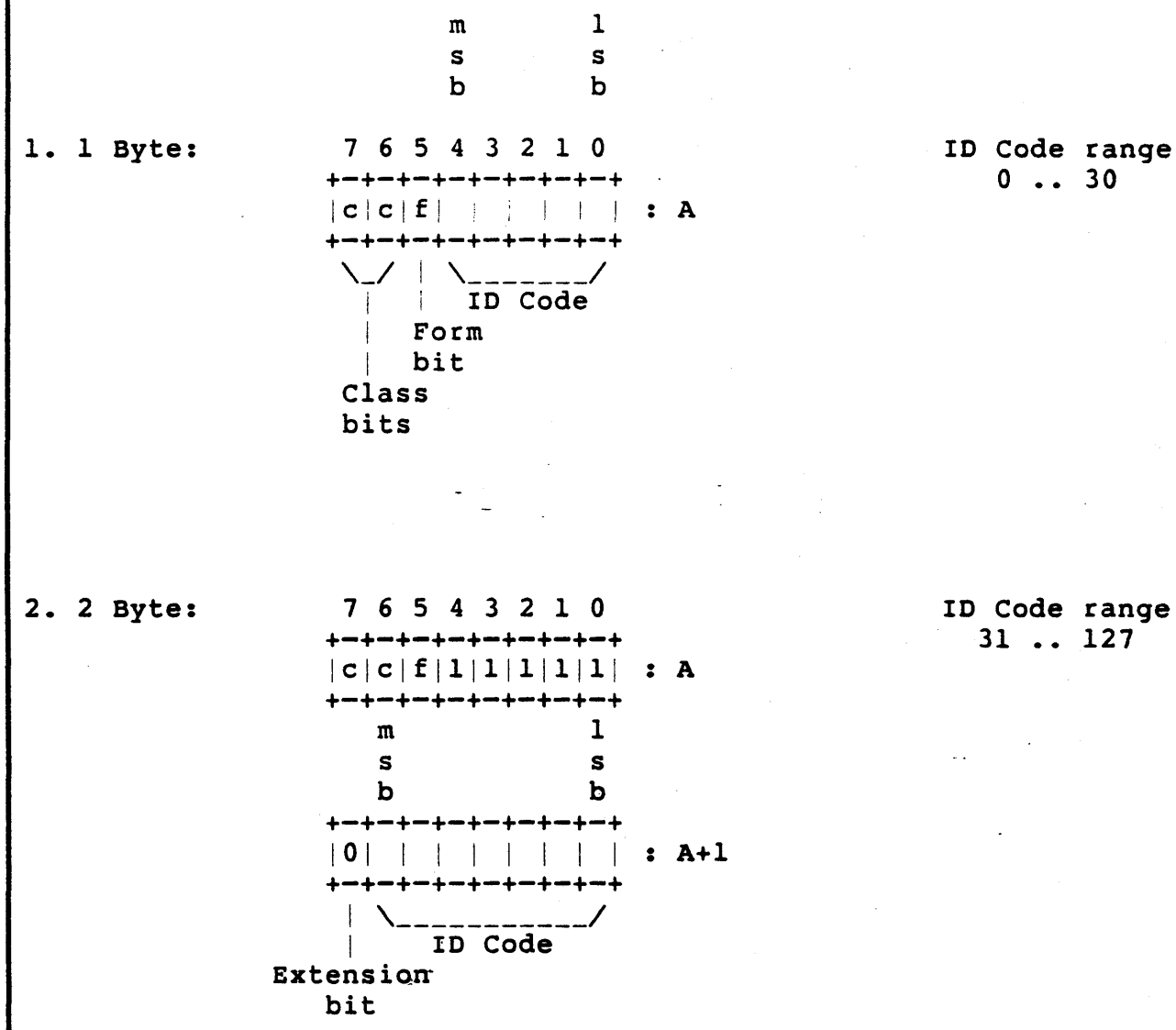
ID Code Bits - TYPE <4:0>

00000 .. 11110 = ID Code

11111 = ID Code is encoded in one or more extension bytes

FIGURE 2a

EXTENSION OF TYPE FIELD

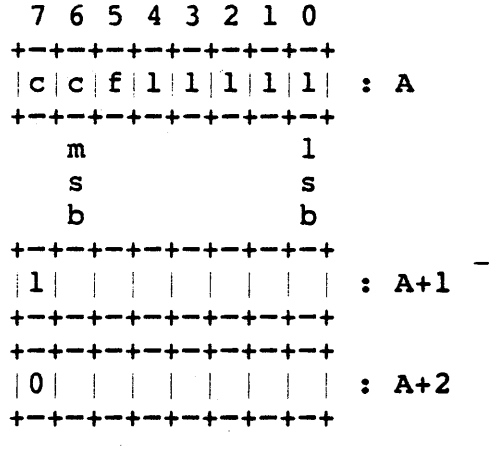


The ID Code shall be encoded in the fewest possible bytes.

FIGURE 2b

EXTENSION OF TYPE FIELD (continued)

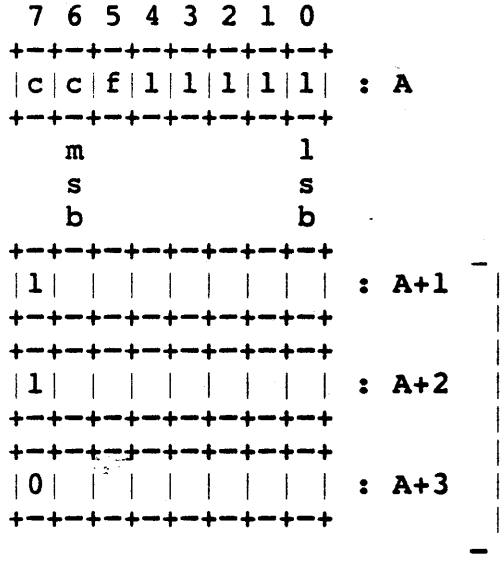
3. 3 Byte:



ID Code range
128 .. 2**14-1

ID Code (Note: bit 7 of extension bytes is not included in ID Code)

4. 4 Byte:



ID Code range
2**14 .. 2**21-1

ID Code (Note: bit 7 of extension bytes is not included in ID Code)

The ID Code shall be encoded in the fewest possible bytes.

FIGURE 2c

LENGTH FIELD

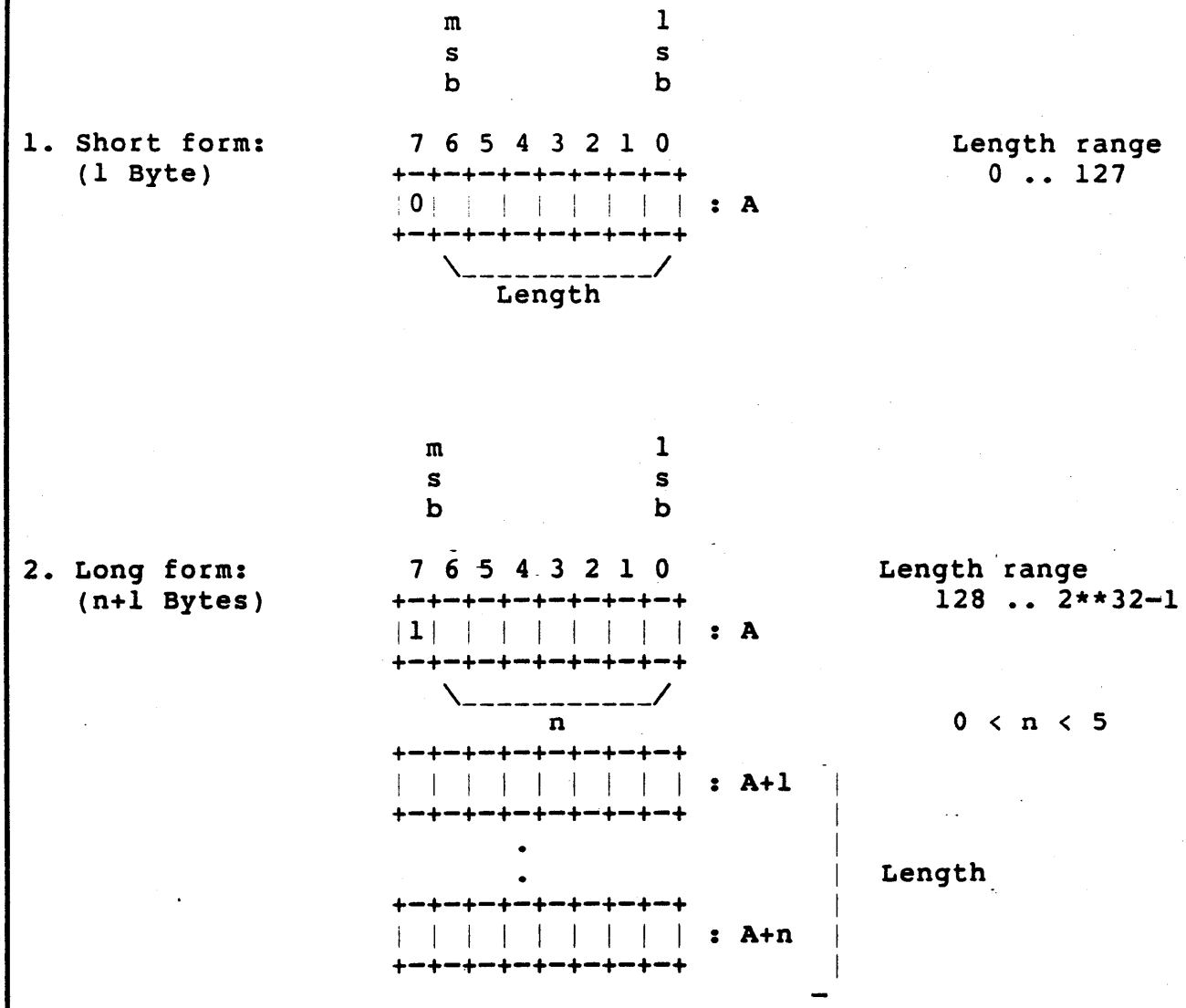
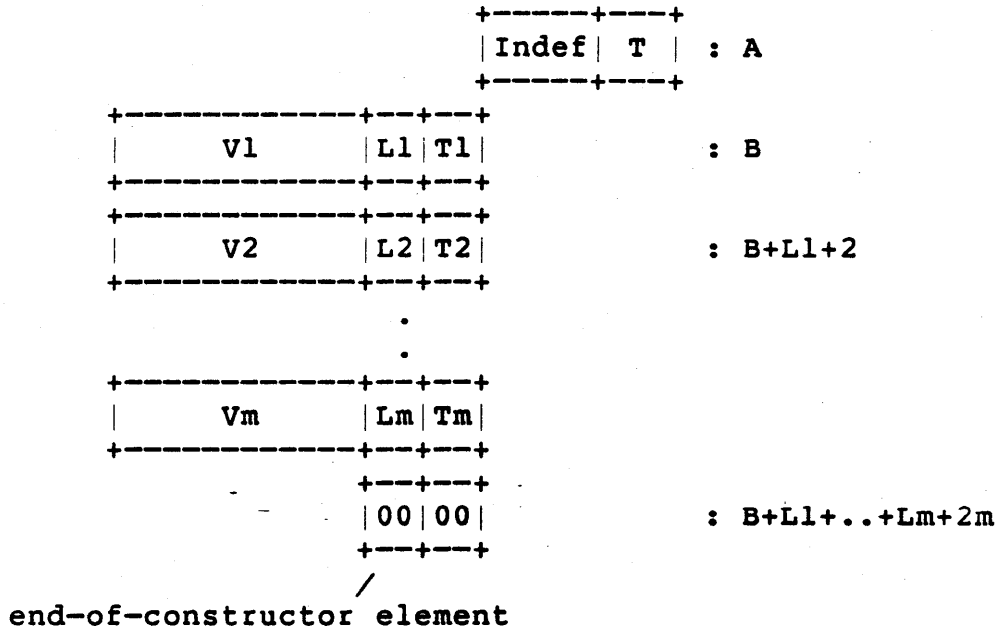


FIGURE 3

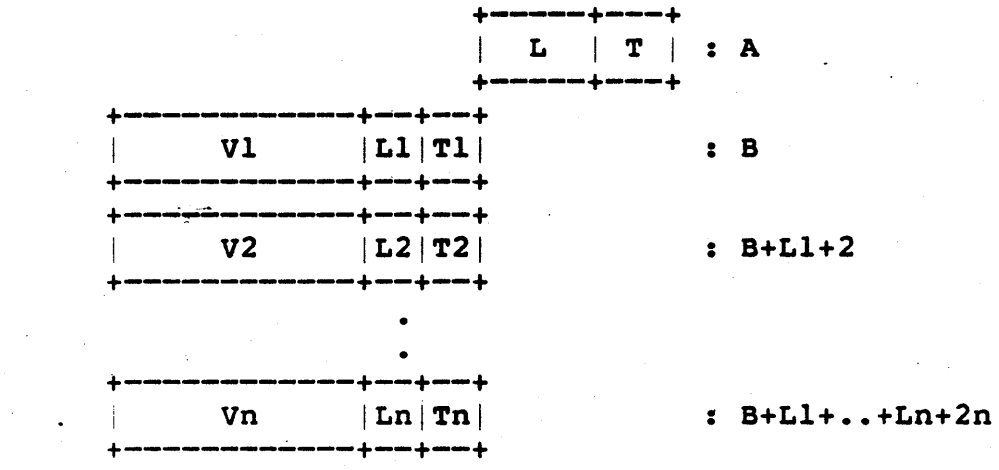
CONSTRUCTORS

(Example : assume Type & Length fields are 1 byte long)

1. Uncounted Constructor



2. Counted Constructor



where L = L1+L2+...+Ln+2n

FIGURE 4

UNIVERSAL ID CODES
(BUILT-IN TYPES)

	ID Code	Data Type
*	0	End_of_Constructor
	1	Boolean
	2	Integer
	3	Bit_String
	4	Octet_String
	5	Null
	6	F_Float
	7	G_Float
	8	D_Float
	9	H_Float
	10-15	RESERVED FOR FUTURE STANDARDIZATION
	16	Sequence
	17	Set

* Although not a data type, the End_of_Constructor data element consumes an ID code

FIGURE 5

DDIS ACCESS ROUTINES

1. Sponsered by Display Systems Software

2. Used in GPU (Sarah) file I/O system

3. Suitable for RTL usage

   A. Shareable Code
   B. I/O independent
   C. Syntax Independent
   D. Written in Bliss-32

4. Advantages of RTL support (VMS 5.0)

   A. Allows customers to read & write DDIS
   B. Reduces size of applications
   C. Provides extensive error checking
   D. Extended testing

5. Issues:

   How do we supply syntax to customer?
   DDIS must be very stable

DDIS FEATURES

1. Any constructor may be counted or indefinite
2. Full support of SET
3. Support of all DDIS syntax, including ANY
4. Strict syntax enforcement
5. Default Values are TBD

RESTRICTIONS

1. Tag Length: 4 Bytes or Less
2. Maximum value length: 2**32 - 1 bytes
3. Max. Stream Length: currently 2**32 - 1 bytes

CONTEXT BLOCK MANAGEMENT

DDIS$ALLOCATE_CONTEXT

1. Allocates context block, its buffers & stacks
2. Initializes internal queue headers
3. Associates Parse Tables
4. Parameters: Context block return
   - Get memory routine address
   - Free memory routine address
   - I/O routine address
   - Condition handler address (established)
   - Parse table addresses...

DDIS$DEALLOCATE_CONTEXT

0. Parameters: Context block address
1. Deallocates context block & substructures
2. Application should check parse state

DDIS Read Routines

1. DDIS$GET_TAG

A. Parameters: context block address
B. Skips value if pending
C. Tag is read into context block
D. Tag is located in the parse table (or error)
E. Length is decoded and placed in context block
F. Returns end of counted constructor as EOC tag

2. DDIS$GET_VALUE

A. Parameters: context block [,buffe
B. Uses context block buffer by de ult
C. Built-in buffer may be reallocate
D. Error if no value is available (e.g. EOC)

3. DDIS$SKIP_TO_EOC

A. Parameters: context block
B. Proceeds to end of current constructor
C. No parse table checking - can skip private data &c

WRITE ROUTINES

1. DDIS$PUT_TLV

A. Parameters: context block [,tag] [,length] [,value]
B. Tag is checked against syntax
C. Length is encoded
D. Value is written (integers are compressed)
E. Constructor is counted if length is passed
F. Counted constructors must end on a tag

2. DDIS$PUT_EOC

A. Parameters: context block, count
B. Writes a number of EOCs to the stream
C. A convenience routine, NYI

2. DDIS$GET_INTLEN

A. Parameters: Maximum length, address
B. Returns length of compressed integer

PARSING ABSTRACT SYNTAX

1. SEQUENCE

A. Listed tags must be in designated order
B. Tags must be present unless OPTIONAL
C. No other tags can be present

2. SET

A. Listed tags may be in any order
B. But no tag may be repeated
C. Tags must be present unless OPTIONAL

3. CHOICE

A. Only one may be selected
B. May be nested in SET or SEQUENCE
C. Forms a constructor if tagged

4. DDIS routines parse TLV stream

A. Use a dynamically loaded parse table
B. Maintains unique context in table (supports dispatch)
C. Discourages creation of invalid files, structures
D. Catches logic errors (during execution)

PARSE TABLES

1. Tree structure reflects syntax

2. Contains implicit types

3. Each tag has an unique entry

4. Flags for OPTIONAL, CHOICE, etc

5. Common definition, common nodes

6. Tree structure is PIC & shareable

```
.MACRO TAG tag
        .LONG       tag
        .ENDM TAG

.MACRO DESCENDANTS location
        .WORD       location
        .ENDM DESCENDANTS

.MACRO TYPE type
        .BYTE type
        .ENDM TYPE

.MACRO FLAGS flags
        .BYTE flags
        .ENDM FLAGS
```

PART OF DDIF TABLE

```
DDIF_DDIS_TABLE::
LABEL1 :        TAG             DDIF$C_DDIF     ; 0
                DESCENDANTS     LABEL2-LABEL1
                TYPE            ddis$c_sequence
                FLAGS           0

            END_OF_CONSTRUCTOR          ; 1

LABEL2 :        TAG             DDIF$C_DOCUMENT_DESCRIPTO
                DESCENDANTS     LABEL30-LABEL2
                TYPE            ddis$c_sequence
                FLAGS           0

LABEL3 :        TAG             DDIF$C_DOCUMENT_PROFILE
                DESCENDANTS     LABEL35-LABEL3
                TYPE            ddis$c_sequence
                FLAGS           0

LABEL4 :        TAG             D  IF$C_DOCUMENT_HEADER
                DESCENDANTS     L-BEL83-LABEL4
                TYPE            ddis$c_sequence
                FLAGS           0

LABEL5 :        TAG             DDIF$C_DOCUMENT_CONTENT
                DESCENDANTS     LABEL103S-LABEL5
                TYPE            ddis$c_sequence
                FLAGS           0

            END_OF_CONSTRUCTOR          ; 6
```

DEVELOPMENT ENVIRONMENT

What's needed:

A. BNF Compiler, table builder
B. Symbols for DDIS tags: EOC, INTEGER, etc
C. Symbols for application tags: DDIF, etc
D. File Analyzer, Dump Facility
E. Run-time Debugging Tools

What we have so far:

A. PTU: Prototype BNF Com)iler, table builder
B. SDL file of Symbols for)DIS tags
C. PTU creates Symbols for application tags
D. TAN: File Analyzer, Dump Facility
E. DDIS module with audit trail
F. ..and the debugger works with this stuff

may need a complete compiler...

PARSE TABLE UTILITY

1. PTU compiles BNF into Macro-32 declarations
2. Human readable output, for development (.mar)
3. Tables can be linked to application (.obj)
3. Or tables can be loaded dynamically (.exe)
4. PTU provides symbol declarations (require files)

TLV ANALYZER (TAN)

1. Dumps a DDIS file in MACRO-32
2. Output can be re-assembled, linked
3. Displays tags by name (PTU-generated)
4. Parse tables can be loaded
5. DDIF Parse table is built in

```
; This TAN X3.7 output was generated on 27-MAR-1986 08:11:04
; from the file DDIF$:[EXAMPLES]SIMPLE_PARAGRAPHS.DDIF;4

; 306 symbols read from DDIF$:[PRGSRC]DDIF_ENTRY_SYMBOLS.T

;Based on built-in parse table for DDIF
;
    .PSECT ddis,rd,nowrt,byte,shr
;

C0: .BYTE ^B00111111        ; Entry = 0 (DDIF$C_DDIF)
    .BYTE  ^B10000000       ; Continued tag
    .BYTE  ^B00000001       ; Continued tag
    .BYTE ^X80              ; Indefinite Length

C1: .BYTE ^B00110000        ; Entry = 2 (DDIF$C_DOCUMENT_DESC
    .BYTE ^X80              ; Indefinite Length
P0: .BYTE ^B10000000        ; Entry = 18 (DDIF$C_MAJOR_VERSION
    .BYTE ^X00              ; Length
                            ; Value (INTEGER) = virtual zero
```

Logical Access Layer

Introduction

- purpose of LAL
- related components
  - KODDIF
  - Physical Access Layer (PAL)
  - Cache Manager

Logical Access Layer

- internal data structure
- general purpose operations
- application specific operations

Physical Access Layer

- relevant KODDIF features
- action routines
- conversion between internal data structure and DDIF constructs

Cache Manager

- writing from internal data structure
- pruning the " " "
- reading into " " "

Current design vs. Gold prototype

Conclusions

presented by Bruce Taylor
M/S ZK02-1/N20
E-Net MAGIC::TAYLO

Internal data structure

A tree for each open DDIF document

Interior nodes (complex objects)

- correspond to DDIF segments
- have attributes, children
- format is application indep ndent

Leaf nodes (primitive objects)

- correspond to DDIF content elements or low-level segments
- may encode rendition information
- formats are application specific

General purpose operations

Object naming

Attribute resolution

Data editing

Search for attributes

Navigation

Secondary index creation

Document level operations

Object naming scheme

Object variables: used by an application to access objects

Similar to pointers

- repositioned during navigation

Unlike pointers

- retain connection with objects no longer in-memory
- don't allow direct manipulation of data structure

Managed by LAL functions:

- create_object_variable
- destroy_object_variable

Attribute resolution

Terms:

- specific attributes
- effective attributes
- attribute resolution

Computing effective attributes

- specific attributes
- generic references
- defaults
- inheritance

Problem: relative attributes

- relative margins
- transformations

General purpose data editing

Create complex object

Delete complex object

Cut, paste

Other operations, concerning

- segment id's
- external references
- data type boundaries
- access rights to objects

Navigation

Three kinds of navigation:

- tree navigation
- navigation by unique segment id
- navigation by secondary index

Secondary index:

- stores page breaks, line b aks, etc.
- a set of pointers into the stored document
- created, managed by application
- LAL provides two services:
  - make_primary_key
  - position_using_primary_key

Application specific operations

General categories:

- create primitive object with content
- search for primitive conter
- get copy of primitive conte t
- modify primitive content
- specialized operations on structure and content

Application specific operations: examples

Text Editor

- apply_rendition_change
- split, merge

Graphics Editors

- transform_2d_object

Attribute Editor

- specify_attributes

Untyped operations

KODDIF

Two major components:

Access method

- document-level services
- positioning of "streams" at content_elements
- reading, insertion, update, deletion at a stream

KODDIS with DDIF parse table

- input is driven by DDIF syntax
- output is controlled by PAL

KODDIS

Encodes, decodes DDIS TLV's (primitives and constructors)

Uses two tables:

- parse table
- action table

Tables are generated for an application by Koala DDIS Utility (KDU)

Input: KODDIS checks syntax, invokes appropriate action routine when an element is recognized

Output: KODDIS generates DDIS strings, while checking syntax. Provides calls to

- start a constructor
- end a constructor
- output a primitive value

Physical Access Layer

Input ("action") routines convert DDIF content elements to tree nodes

- invoked by KODDIS
- correspond to DDIF constructors or primitives
  - allocate a tree node
  - assign a value to a field
  - link completed node int tree structure

Output routines convert tree nodes to DDIF

- call each other
- at lowest level, call KODDIS
  - begin a constructor
  - output a primitive element
  - end a constructor

Cache Manager

Writing from internal data structure

- occurs before document is closed or checkpointed
- may occur before internal d.s. is pruned
- objects are written out using PAL
- only changed objects are written out

Pruning internal data structure

- occurs when the d.s. is getting too large
- least recently used tree nodes are written out, deallocated from tree
- some information is saved to facilitate later retrieval

Reading into internal data structure occurs

- when existing document is opened
- after roll-back to a previou document state
- when an application tries t access an object not in-memory

Current design vs. Gold prototype

In the protoype, "LAL" corresponded to Physical Access Layer

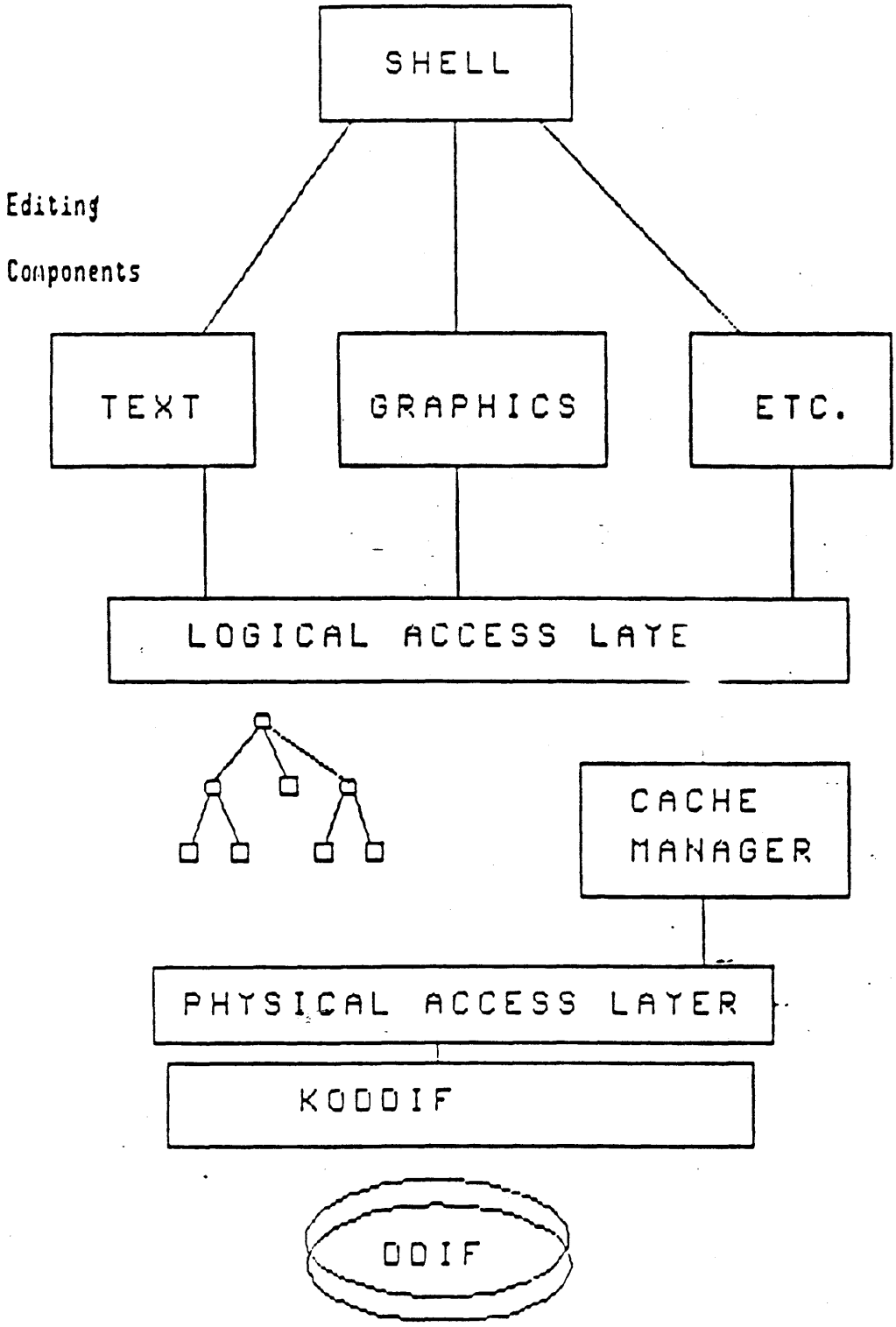
- each DDIF content element was read into a temporary d.s.
- LAL extracted useful information and added to editor's d.s.

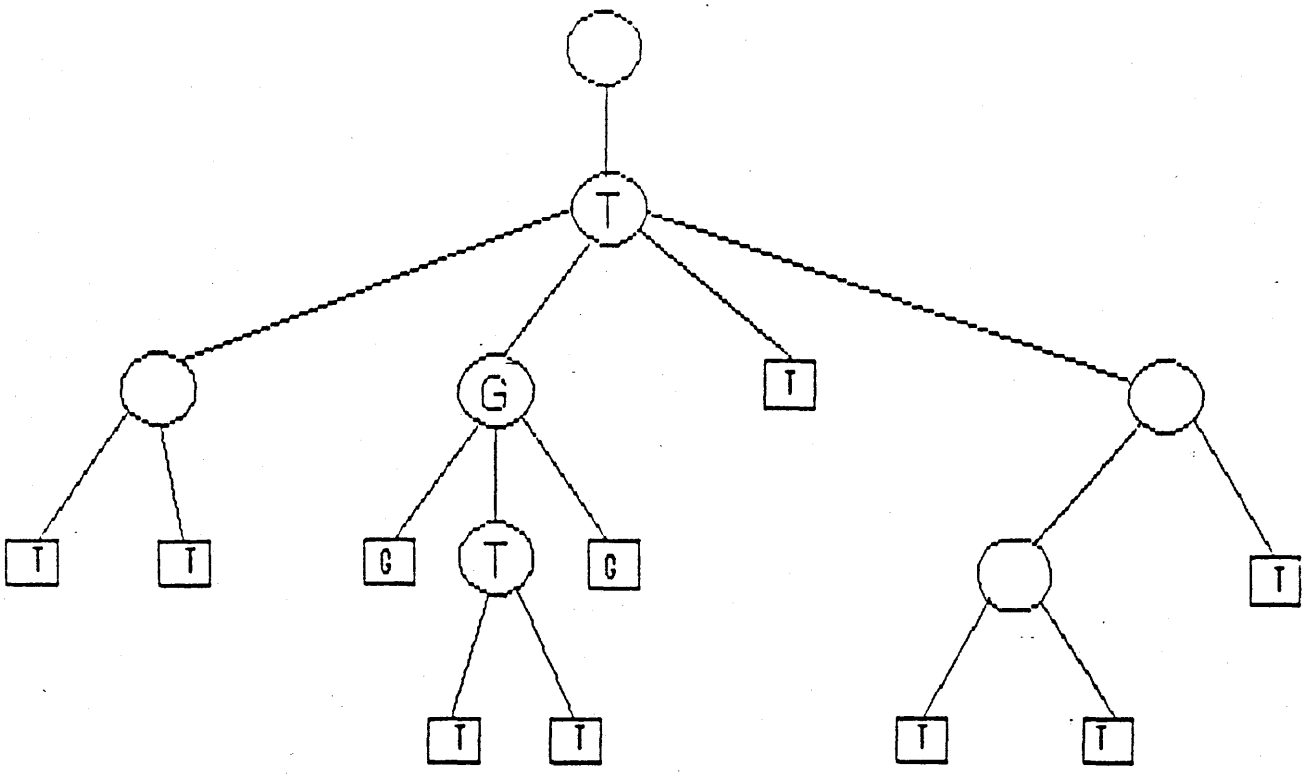
"Document Manager" corresponded to LAL

- provided a view of a section of the stored document
- document manager, editor shared this data structure
- editor had to distinguish between "in memory", "on disk" data and compute effective attributes

Current design

- avoids an extra conversion
- encapsulates application-independent functions
  - attribute handling
  - common data editing oper ions
- eliminates "in memory", "on sk" distinction
- provides a cleaner interface to document


SHELL
Editing
Components
TEXT
GRAPHICS
ETC.
LOGICAL ACCESS LAYE
CACHE
MANAGER
PHYSICAL ACCESS LAYER
KODDIF
ODIF


T
G
T
T
T
G
T
G
T
T
T
T
T

DDIS & DDIF

Character Sets

9-July-86

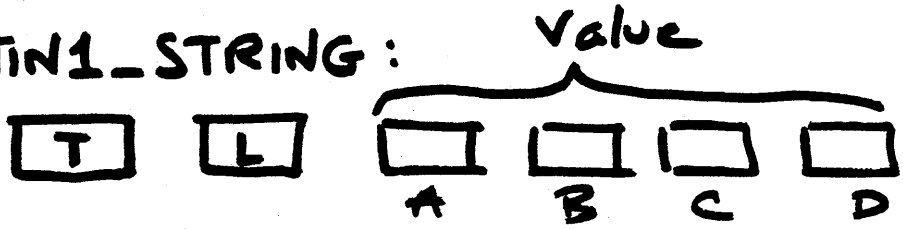
T. Hastings

CHARACTER SET CODES FOR 8-BIT TEXT

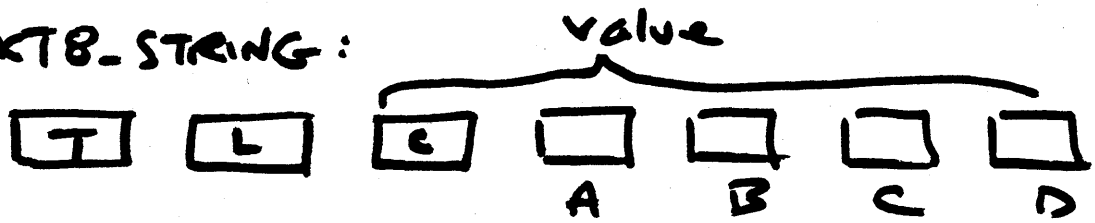
Code	Character Set
0	RESERVED
1	DEC MCS (has only 1 character)
2	Latin-2 (Eastern Europe)
3	Latin-3 (Southern Europe)
4	Latin-4 (Northern Europe)
5	Greek
6	Cyrillic
7	Hebrew
8	Arabic
9	Technical character set
10	Publishing character set
11	Output Rendering character set
12	Special Graphics (Line Drawing) char set
13-127	RESERVED FOR FUTURE STANDARDIZATION
128-254	Reserved for use by DEC customers/OEMs
255	RESERVED FOR FUTURE EXTENSIONS

FIGURE 6

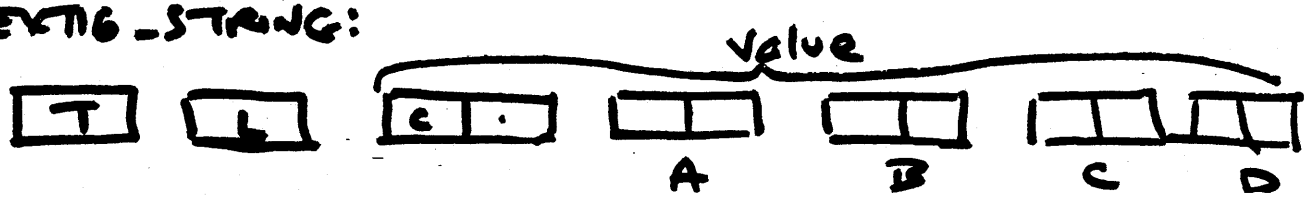
LATIN1_STRING:



TEXT8_STRING:



TEXT16_STRING:



To represent ten Ω resistor

tag	length	value			
	4	t	e	n	
Latin 1					
	2	5	Ω		
Greek					
	9		r	e	s	i	s	t	o	r
Latin 1										

Principle of Uniqueness

Each character appears in only one DDIS set.

If DDIS set was drawn from a standard, duplicate characters are omitted.

Example:
Latin-2 has both a left hand and many right hand characters duplicated. ∴ dropped.

TEXT16 has no latin characters.

ISSUE - some say applications have to convert from DDIS to internal in order to process. ∴ Could fold duplicates.

Table 1. DEC Multinational Character Set

b4 b3 b2 b1	b8	b7	b6	b5	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
0 0 0 0	0	0	0	0	NUL	DLE	SP	0	@	P	`	p		DCS		°	À		à	
0 0 0 1	0	0	0	1	SOH	DC1	!	1	A	Q	a	q		PU1	¡	±	Á	Ñ	á	ñ
0 0 1 0	0	0	1	0	STX	DC2	"	2	B	R	b	r		PU2	¢	²	Â	Ò	â	ò
0 0 1 1	0	0	1	1	ETX	DC3	#	3	C	S	c	s		STS	£	³	Ã	Ó	ã	ó
0 1 0 0	0	1	0	0	EOT	DC4	$	4	D	T	d	t	IND	CCH			Ä	Ô	ä	ô
0 1 0 1	0	1	0	1	ENQ	NAK	%	5	E	U	e	u	NEL	MW	¥	µ	Å	Õ	å	õ
0 1 1 0	0	1	1	0	ACK	SYN	&	6	F	V	f	v	SSA	SPA		¶	Æ	Ö	æ	ö
0 1 1 1	0	1	1	1	BEL	ETB	'	7	G	W	g	w	ESA	EPA	§	•	Ç	Œ	ç	œ
1 0 0 0	1	0	0	0	BS	CAN	(	8	H	X	h	x	HTS		¤		È	Ø	è	ø
1 0 0 1	1	0	0	1	HT	EM	)	9	I	Y	i	y	HTJ		©	¹	É	Ù	é	ù
1 0 1 0	1	0	1	0	LF	SUB	*	:	J	Z	j	z	VTS		ª	º	Ê	Ú	ê	ú
1 0 1 1	1	0	1	1	VT	ESC	+	;	K	[	k	{	PLD	CSI	«	»	Ë	Û	ë	û
1 1 0 0	1	1	0	0	FF	FS	,	<	L	\	l	|	PLU	ST		¼	Ì	Ü	ì	ü
1 1 0 1	1	1	0	1	CR	GS	-	=	M	]	m	}	RI	OSC		½	Í	Ÿ	í	ÿ
1 1 1 0	1	1	1	0	SO	RS	.	>	N	^	n	~	SS2	PM			Î		î	
1 1 1 1	1	1	1	1	SI	US	/	?	O	_	o	DEL	SS3	APC		¿	Ï	ß	ï	

ASCII Control Set	ASCII Graphic Character Set	Add'l Control Set	DEC Supplemental Graphic Set

DEC Multinational Character Set

Notes

Empty positions are reserved for future standardization.

Append ⑦
April 1986

CODE TABLE

ISO Latin Alphabet Nr 1
ISO 8859/1

b8	b7	b6	b5	b4	b3	b2	b1	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
0	0	0	0	0	0	0	0			SP	0	@	P	`	p			NBSP	°	À	Ð	à	ð
0	0	0	1	0	0	0	1			!	1	A	Q	a	q			¡	±	Á	Ñ	á	ñ
0	0	1	0	0	0	1	0			"	2	B	R	b	r			¢	²	Â	Ò	â	ò
0	0	1	1	0	0	1	1			#	3	C	S	c	s			£	³	Ã	Ó	ã	ó
0	1	0	0	0	1	0	0			$	4	D	T	d	t			¤	´	Ä	Ô	ä	ô
0	1	0	1	0	1	0	1			%	5	E	U	e	u			¥	µ	Å	Õ	å	ð
0	1	1	0	0	1	1	0			&	6	F	V	f	v			¦	¶	Æ	Ö	æ	ö
0	1	1	1	0	1	1	1			'	7	G	W	g	w			§	·	Ç	×	ç	÷
1	0	0	0	1	0	0	0			(	8	H	X	h	x			¨	¸	È	Ø	è	ø
1	0	0	1	1	0	0	1			)	9	I	Y	i	y			©	¹	É	Ù	é	ù
1	0	1	0	1	0	1	0			*	:	J	Z	j	z			ª	º	Ê	Ú	ê	ú
1	0	1	1	1	0	1	1			+	;	K	[	k	{			«	»	Ë	Û	ë	û
1	1	0	0	1	1	0	0			,	<	L	\	l	|			¬	¼	Ì	Ü	ì	ü
1	1	0	1	1	1	0	1			-	=	M	]	m	}			SHY	½	Í	Ý	í	ý
1	1	1	0	1	1	1	0			.	>	N	^	n	~			®	¾	Î	Þ	î	þ
1	1	1	1	1	1	1	1			/	?	O	_	o				¯	¿	Ï	ß	ï	ÿ

HYPHEN -

Table 3 — Primary and supplementary sets of graphic characters for text communication (coding when represented by b
combinations 2/1 to 7/14 and 10/1 to 15/14 of an 8-bit code)

ISO 6937/2 Text Communication

b4 b3 b2 b1	00	01	02	03	04	05	06	07	08	09	10	11	12 [1]	13	14	15
0 0 0 0				0	@	P	`	p				°	See 4.3.4	—	Ω	K
0 0 0 1			!	1	A	Q	a	q			¡	±	`	¹	Æ	æ
0 0 1 0			"	2	B	R	b	r			¢	²	´	®	Đ	đ
0 0 1 1			#	3	C	S	c	s			£	³	^	©	ª	ð
0 1 0 0			¤	4	D	T	d	t			$	×	~	™	Ħ	ħ
0 1 0 1			%	5	E	U	e	u			¥	µ	¯	♪	See 4.3.4	ı
0 1 1 0			&	6	F	V	f	v			See 4.3.3	¶	˘	See 4.3.4	IJ	ij
0 1 1 1			'	7	G	W	g	w			§	•	˚	See 4.3.4	L·	l·
1 0 0 0			(	8	H	X	h	x			See 4.3.3	÷	¸	See 4.3.4	Ł	ł
1 0 0 1			)	9	I	Y	i	y			‘	’	See 4.3.4	See 4.3.4	Ø	ø
1 0 1 0			*	:	J	Z	j	z			“	”	°	See 4.3.4	Œ	œ
1 0 1 1			+	;	K	[	k	{			«	»	¸	See 4.3.4	º	ß
1 1 0 0			,	<	L	\	l	|		←	¼	_	⅛	Þ	þ
1 1 0 1			-	=	M	]	m	}			↑	½	˝	⅜	Ŧ	ŧ
1 1 1 0			.	>	N	^	n	~			→	¾	˛	⅝	Ŋ	ŋ
1 1 1 1			/	?	O	_	o				↓	¿	ˇ	⅞	ʼn	

[1] See 4.3.1

non-spacing →

DDIS & DDIF

ISO LATIN1 instead of DEC MCS
as fundamental character set

LATIN1_STRING (universal code 20)

need video fonts - 15 more glyphs
need UIS - compose sequence input

March DDIS speaches ISO latin 1
but Search Field text still DEC MCS

ISO Latin 1 adds:

NBSP		NO BREAK SPACE
¡		BROKEN BAR
¨		DIAERESIS
-		HYPHEN
®		REGISTERED
¯		MACRON
´		ACUTE ACCENT
¸		CEDILLA
¾		THREE QUARTERS
Ð	ð	ETH
Þ	þ	THORN
Ý	ý	Y WITH ACUTE ACCENT
×		TIMES
÷		DIVIDE
¬		LOGICAL NOT

Icelandic

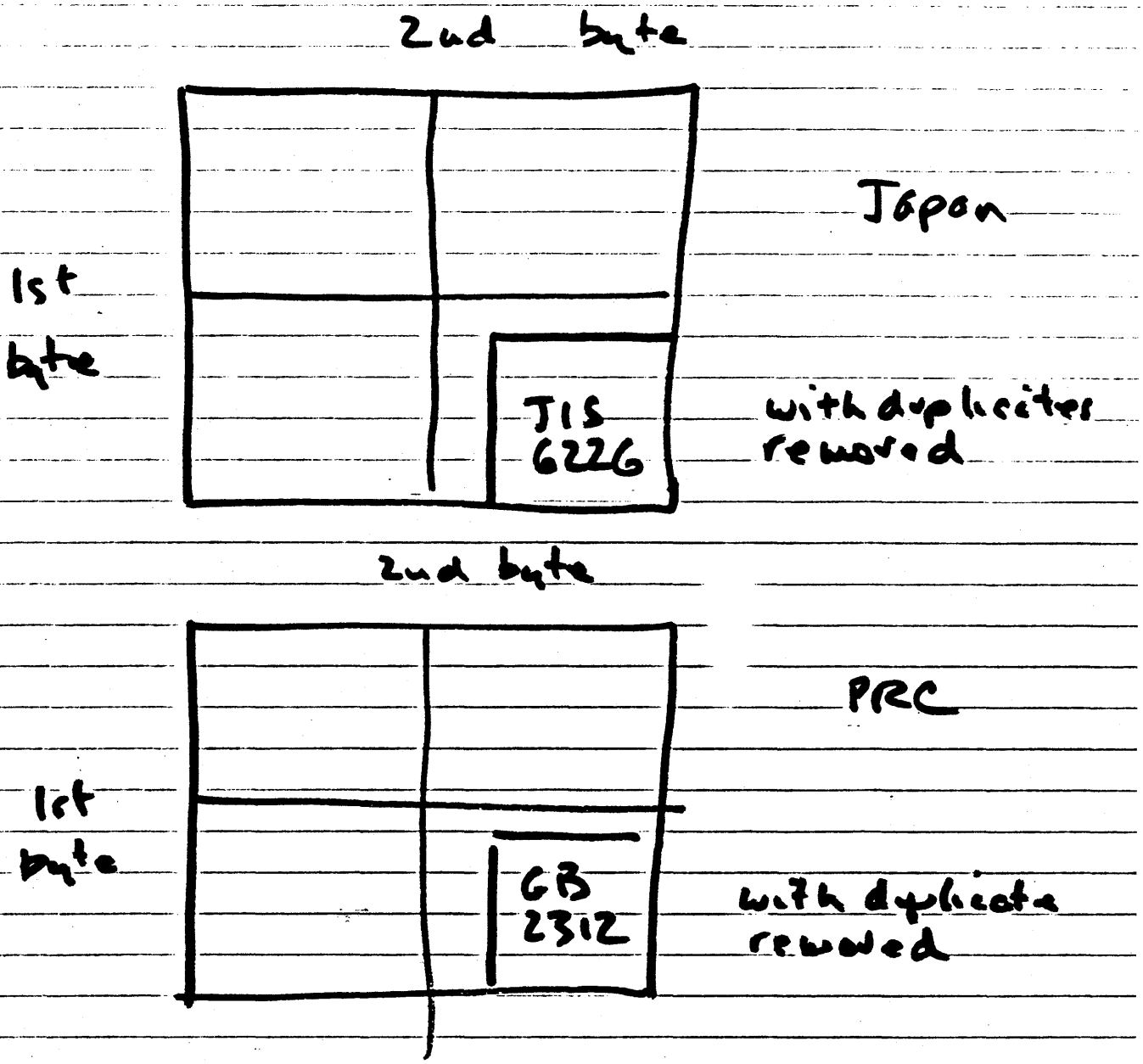
Iso latin 1 drops:

Œ
œ
Ÿ

Iso latin 1 changes:

¤
Y

DDIS Two Byte



International 2, > 2 byte standards

ISO TC97/SC2/WG2 committee

only two byte scope

DEC wants greater than two byte for China Japan

China > 50,000 :characters

Japan 16 000 — 50 000

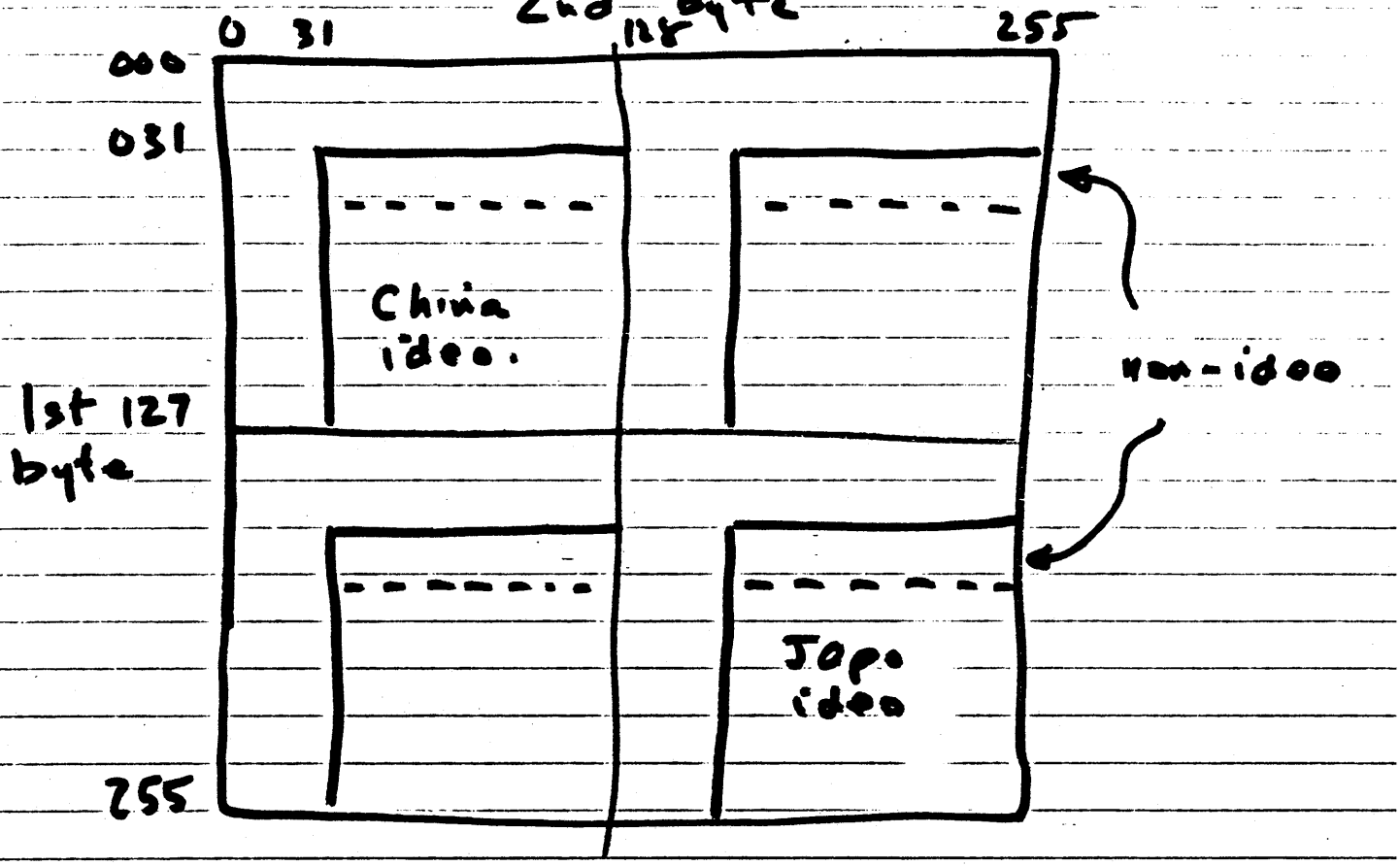
Taiwan

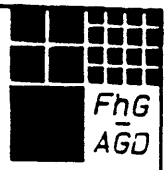
Hong Kong

Korea

etc

Current ISO Two Byte Thinking:





Status of Graphics and Document Standards

J. Schönhut

FhG-AGD

Disclaimer:

The information given does not represent any official position of ISO nor of its computer graphics working group ISO TC97/SC21/WG2. The author is chairing that working group; nevertheless this is purely the personal opinion and understanding of the author.

1. Overview and Introduction

Computer Graphics Industry + 20 percent/year

- reduction of hardware prices
- increased complexity of applications
- main cost factor: software

Demand for standardization

not "industry standards"

standards by standards making bodies

2. Standards Making Bodies
Bodies involved in Graphics Standards

ISO - International Standardization Organisation
ISO TC97/SC21/WG2 Compu... Graphics

Member bodies: national standardization organizations
other international organizations

DIN (Fed. Rep. of Germany)
ANSI (USA)
BSI (United Kingdom)
AFNOR (France)
NNI (Netherlands)
CSA (Canada)

IFIP (Interntl. Fed. for Information Processing)
ECMA (European Computer Manufactures Assoc.)

Example: GKS

first developed by DIN NI 5.9 (now NI 21.2)

brought to ISO by DIN

international review by ISO TC97/SC21/WG2

result: ISO 7942

Operation Principle:

Consensus rather than mere decisions by voting

requires time (plus time for voting procedures)

normally ensures good quality

Other groups (esp. videotex):

ECMA

CEPT

time frame shorter -> earlier results

work based on work of ISO TC97/SC21/WG2

cooperation

3. Status of Standards Documents

Refs. to standards often unqualified
status of document important
status shows relative : ' :lity of document

ISO technical work structured in 3 levels
- TC (Technical Committee)
- SC (Sub-Committee)
- WG (Working Group)

e.g.

TC97 Information Processing
TC97/SC21 Open Systems (short: SC21)
TC97/SC21/WG2 Computer Graphics (short: WG2)

Procedural Steps

- New Work Item (NWI)
  approved by TC letter ballot
  commitment necessary
- NWI assigned to SC or ...
  WG does technical work
- Initial draft developed
- Working Document (visible to SC for information)
- DP Registration (SC registration ballot/SC meeting
- DP ballot (SC ballot - technical comments)
- multiple DP's possible
- DIS Registration (out of DP ballot)
- DIS ballot (TC ballot - no technical changes)
- multiple DIS's possible
- Final Text for IS to ISO Central Secretariat Geneva
- Publication provided ISO Council accepts
- IS

4. Graphics Standards

Characteristics of standards
and standards under development

Status of projects

4.1 Graphical Kernel System

Graphical Kernel System (GKS) ISO 7942

- functional standard
- one or more workstat
- passive and interactive graphics
- two dimensional graphics
- device independent attribute setting
- one level of picture segments
- storing, retrieving and interpreting picture information via GKS Metafile (GKSM)
- inquire functions
- error handling concept

4.2 Graphical Kernel System - Extension to 3D

Extension of two dimensional GKS to three dimensions
- upward compatible to GKS
- GKS program runs unmodified under GKS-3D
- one level of segmentation

Status: DIS 8805

4.3 GKS Language Bindings

GKS as functional specification needs embedding into programming languages by language bindings.

- many languages
- not only ISO programming languages

GKS Language Bindings for

- FORTRAN
- Pascal
- Ada

by WG2 in close operation with WG of specific langua

If no extension of language (e.g. FORTRAN 3rd routines) lead is with WG2

In case of integration into language lead is with WG dealing with langua (e.g. BASIC)

Status: FORTRAN, Pascal, Ada DP 8651 parts 1, 2 and 3
FORTRAN and Pascal going into DIS processing
Ada point to 2nd DP 8651/3
BASIC currently dpANS
C outside ISO, progress of a C binding closely monitored by WG2

4.4 Programmer's Hierarchical Interactive Graphics Standard (PHIGS)

response to need for
hierarchically structured pictures

GKS: one level of segmentation
PHIGS: hierarchical segment structures

editing of such structures
upward compatible to GKS whereever possible
most critical and difficult issue in actual graphics standardization work

4.5 Computer Graphics Metafile

GKS already contains functionality of GKSM
- formal status ambiguous
- not integral part of standard in ISO
- integral part of standard in e.g. DIN

Formal discrepancy:
- Need for International Standard for
  - graphics metafile or
  - set of metafiles

Computer Graphics Metafile (CGM) formerly known as VDM

- four part standard
  - functional specification
  - three parts data encodings
- no segmentation in current version
- possible, however not easy, use of CGM as a GKSM
  - for level 0 GKS
- full GKS not met, but work is under way

Status: DIS 8632/1-4

4.6 Computer Graphics Interface

Interface to graphical devices

New Work Item Computer Graphics Interface (CGI)
formerly known as Virtual Device Interface

Major issues:

Compatibility with CGM and GKS
Full support for GKS workstations

4.7 Conformance Testing of Computer Graphics Standard

After feasibility study
request for New Work Item on
Conformity Testing

If New Work Item accepted
ISO Technical Report (TR)

4.8 Formal Specification of Computer Graphics Standards

Natural language descriptions:

- sometimes difficult with correct or intended interpretation
- wish to formally specifying standards
- one problem: poor readability of formally specified standards for non formal specs trained reader
- possible solution: natural language text along with a formal description, in dubio pro formal specification

After feasibility study request for New Work Item on Use of Formal Specification Techniques for Computer Graphics Standards

If NWI is accepted, ISO Technical Report (TR)

4.9 Terminal/Window Management Standard

Fast expanding field: use of bit mapped displays
overlapping window systems
Progress monitored by WG2
Possibility of standard Window Management
investigated by ANSC X3H3

OSI context: virtual terminals
virtual graphics terminals
terminal management issues
across all virtual terminals

Combination: window and terminal Management

4.10 Graphics in Documents

Graphics in Documents: growing importance
ISO TC97/SC18
WG2 experts participating
Proposals for integrating graphics in
- ODA/ODIF
- SGML

by WG2

Status of Document Standards:
- ODA/ODIF DIS 8613/1-6 (integration into CCITT T.73 expected)
- SGML DIS 8879

4.11 Graphical Data Syntax/Videotex (ECMA/CEPT)

ECMA GDS Standard

based on GKS functionality

multi workstation interface

CEPT Videotex Standard

subset for output only from GDS

to form geometrical parts of

Videotex Standard

5. Towards a Reference Model for Computer Graphics Standards

Relation of graphics standards projects
among each other
and to the outside world
(OSI, Documents, Codings, Programming Languages)
Issue of compatibility

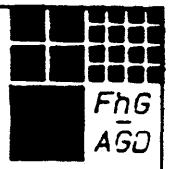
Reference Model (RM) for Computer Graphics:

Some questions:

- difference between Codings and Language Bindings
- Interfaces to be i ified in the RM
- relation of Functional Standards (GKS, GKS-3D, PHIGS) to CGI (and CGM/GKSM)
- Structuring Concepts to be used in Graphics Standards (level vs. option sets, dimensionality, etc.)
- concepts for Attribute Binding

Difficult, but badly needed

Current situation: everyone has a RM, but each one is a bit different



Compatibility of Graphics Standards

J. Schönhut

FhG-AGD

Compatibility Problems

- GKS vs. PHIGS

- GKS vs. CGM

FhG
AGD

PHI-GKS

may be seen as:

1. a shell on top of PHIGS

2. a superset of GKS-3D and PHIGS

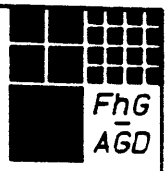
3. the current work item PHIGS
   (with some changes)

4. a GKS-3D extension with hierarchy
   and editing

FhG AGD

PHI-GKS Summary

- level structure
- workstation control
- state diagram
- transformation pipeline
- hierarchical data structures
- attribute model
- primitives outside segments
- archiving
- deferral mode
- state lists and description tables



PHI-GKS

Level Structure

- levels as in GKS (output, input)
- dimensionality (2D/3D)
- additional output levels for hierarchy and editing

PHI-GKS

State Diagram

The state list is described by triple

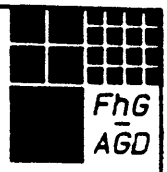
[WS-S, SEG-S, ARC-S]

with the following state values:

- workstation state
  (PHI-GKS_OPEN, WSOP, WSAC)

- segment state
  (SEG_CL, SEG_OP)

- archive state
  (ARC_CL, ARC_OP)



PHI-GKS

Workstation Control

- workstation independent segment structure vs. structure store
- WISS always active
- associate, insert, copy
- archival

FhG
AGD

PHI-GKS

Transformation Pipeline

- workstation dependent transformations
  - viewing transformation
  - workstation transformation

- workstation independent transformation
  - modelling transformation
    - global
    - local
  - normalization transformation
  - normalization clip
  - segment transformation

PHI-GKS

Hierarchy & Editing

- segments vs. structures
  - different data types
  - one data type
- segment attributes
  - segment header
  - structure elements
- 2D/3D - structure elements
- hierarchy of segments
  - execute segment
  - existence of (dummy) segments
- editing
-

PHI-GKS

Attribute Model

- individual/bundled & ASFs
- explicit: set modal attr. functions
- generation vs. traversal time binding
  - new attr. value: 'to be inherited' (tbi)
  - PHI-GKS state list
  - edit state list
  - at segment creation: st segment header from PHI-GKS state list
  - binding:
    outside segments: use PHI-GKS state list values
    segment open: use edit state list
    at segment creation:
    values of Edit State List and PHI-GKS State List identical
  - Traversal State List

PHI-GKS

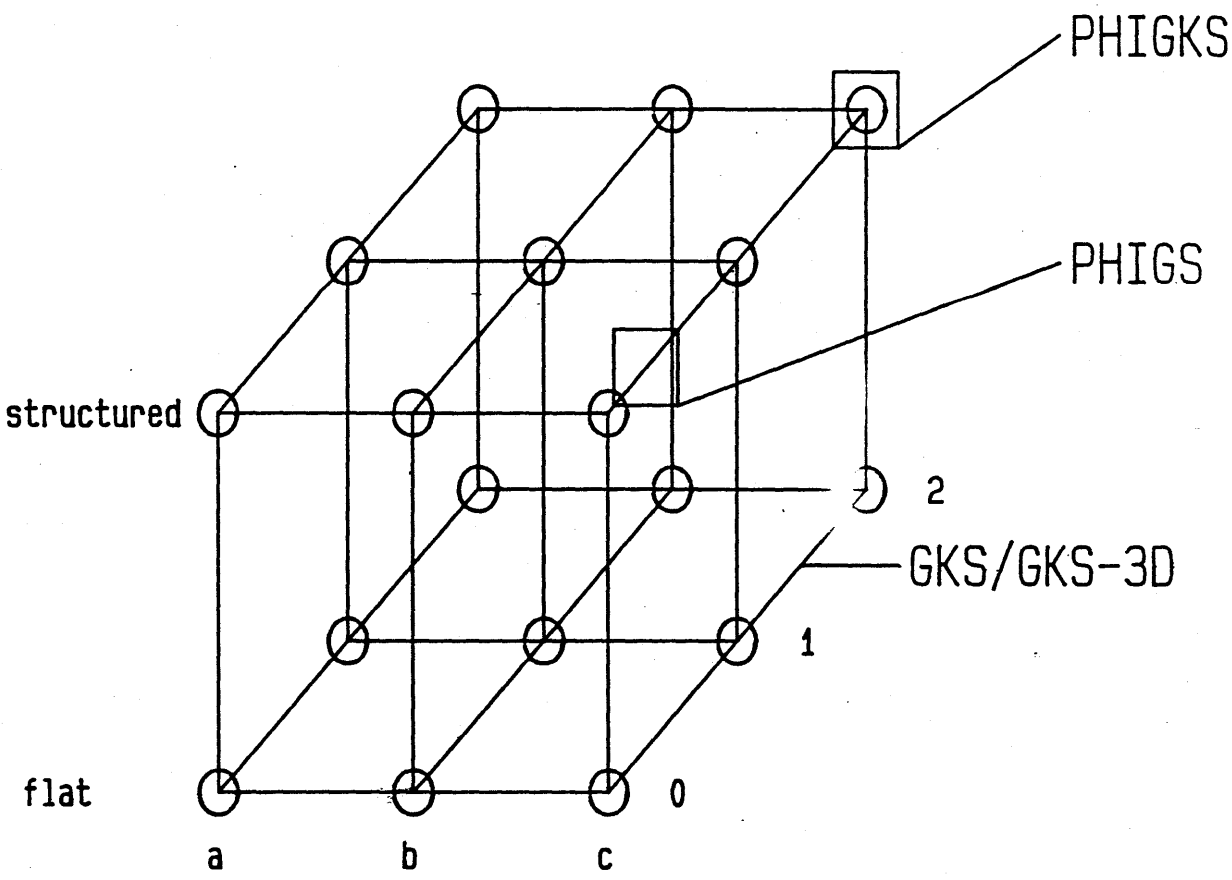
- primitives outside segments

- metafile / archiving

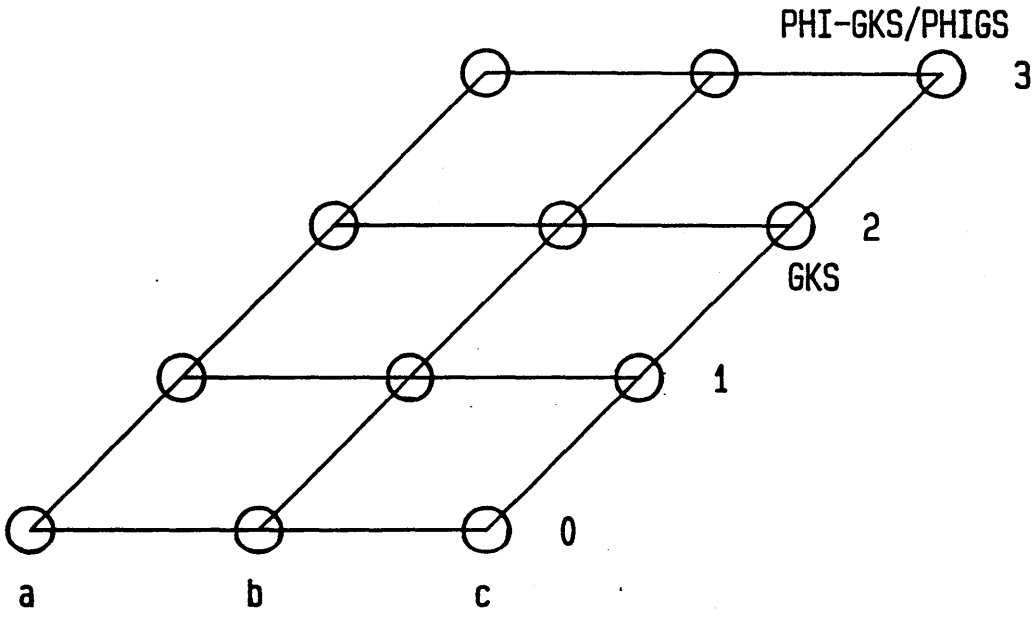
- deferral / update states
  - deferral mode
  - implicit regeneration

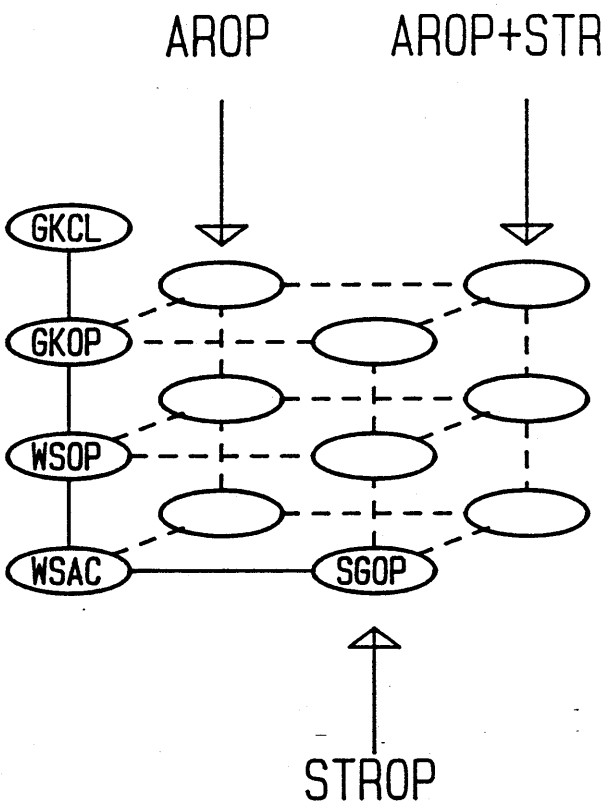
- state lists
  - PHI-GKS State List
  - Workstation State List
  - Edit State List, in addition
  - Traversal State List (internal)

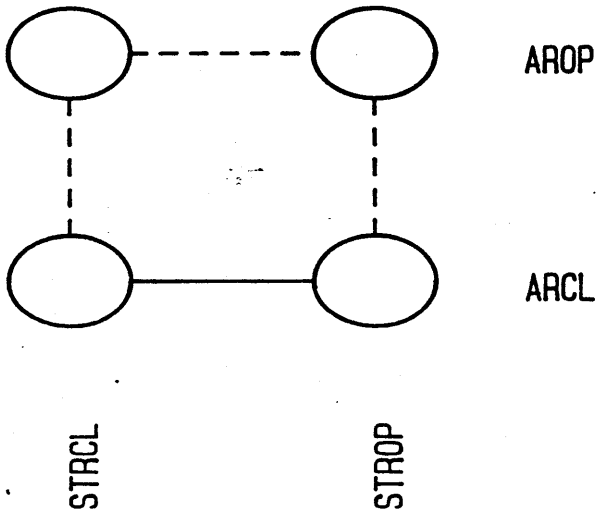
- description tables

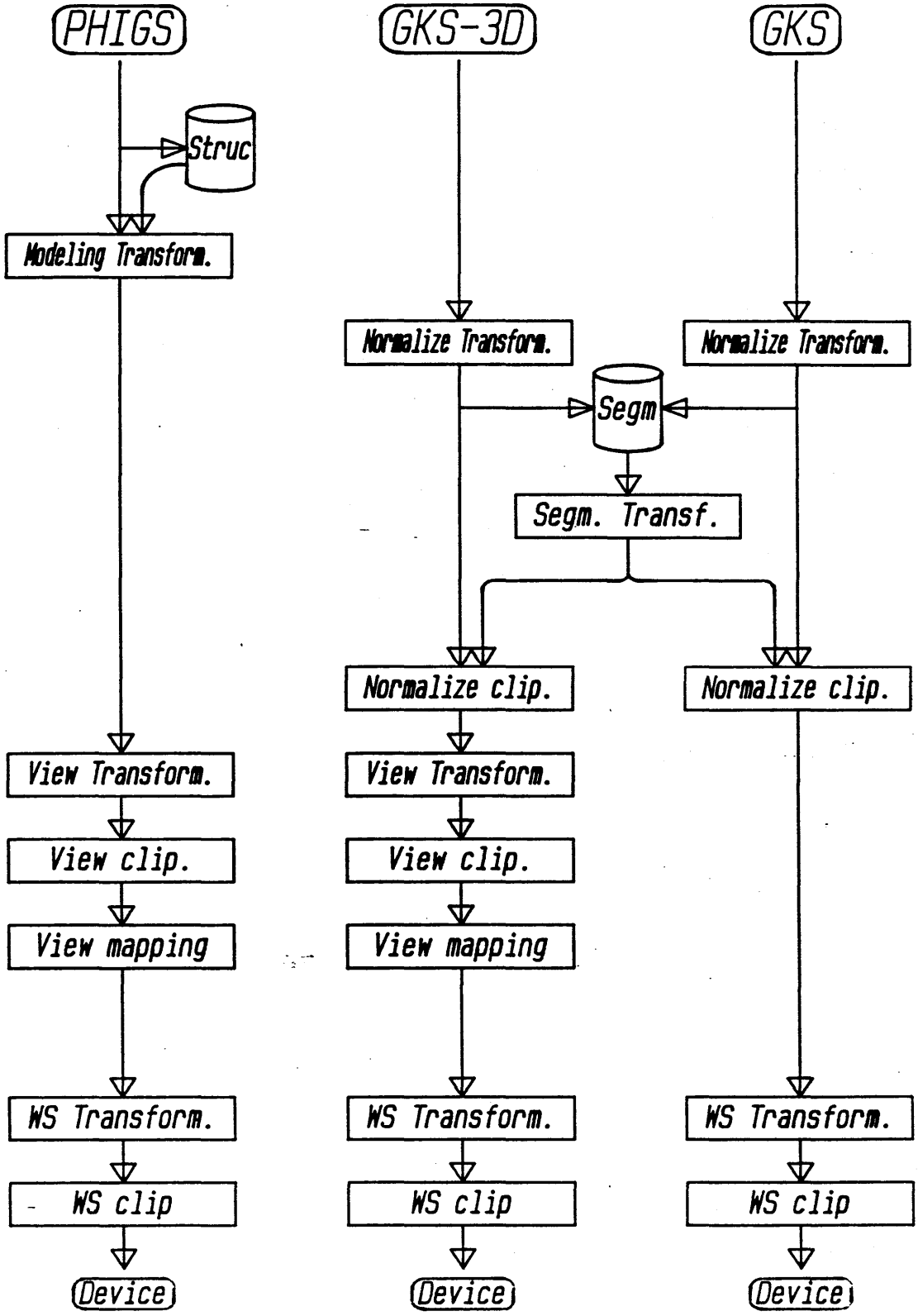


FhG AGD | PHIGKS Level Structure

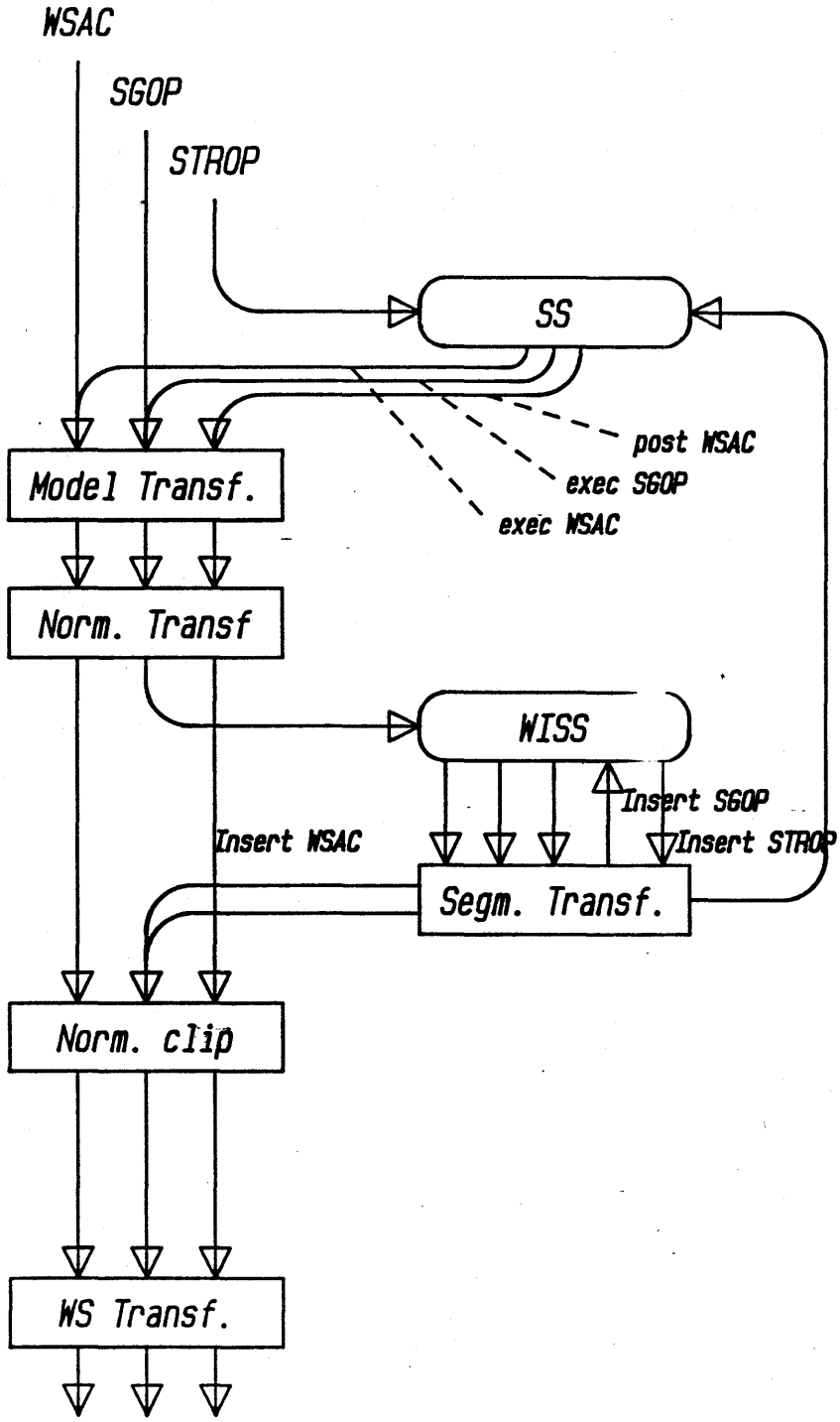

PHI-GKS/PHIGS
3
2
GKS
1
0
a
b
c

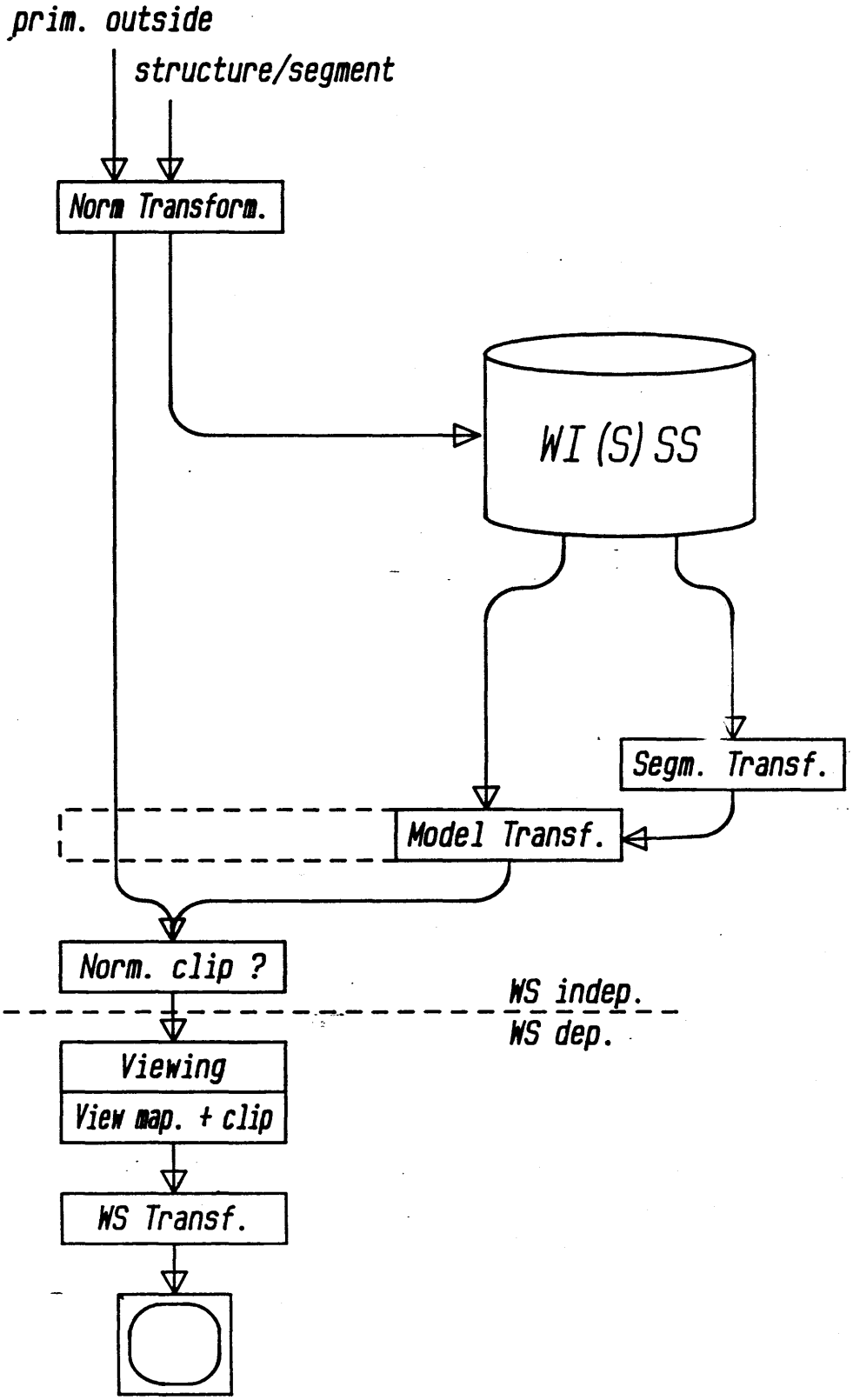

AROP
AROP+STROP
GKCL
GKOP
WSOP
WSAC
SGOP
STROP

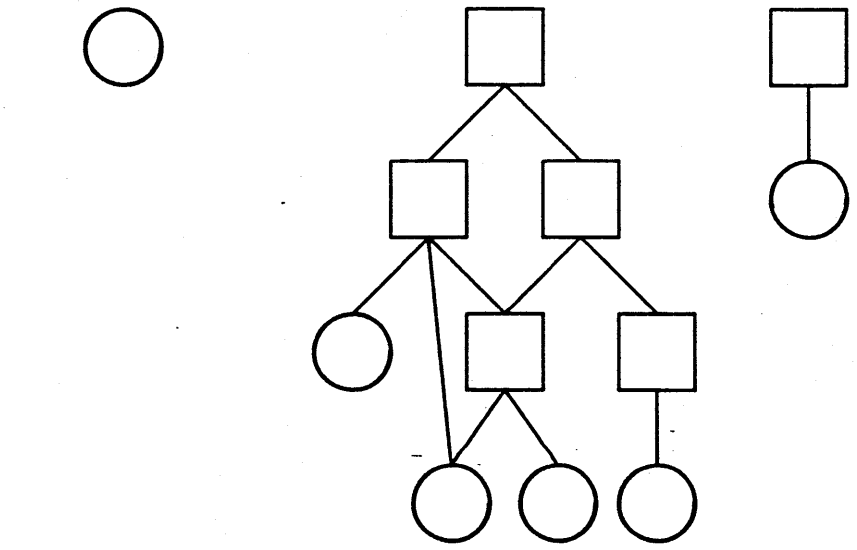

AROP
ARCL
STRCL
STROP



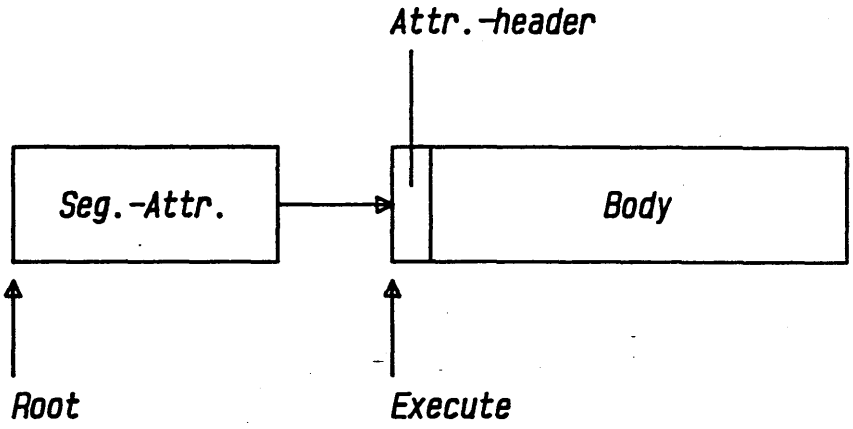
FhG-AGD | Funct. Graphics Pipeline


WSAC
SGOP
STROP
SS
post WSAC
exec SGOP
exec WSAC
Model Transf.
Norm. Transf
WISS
Insert SGOP
Insert STROP
Insert WSAC
Segm. Transf.
Norm. clip
WS Transf.


prim. outside
structure/segment
Norm Transform.
WI (S) SS
Segm. Transf.
Model Transf.
Norm. clip ?
WS indep.
WS dep.
Viewing
View map. + clip
WS Transf.



structure segment


Attr.-header
Seg.-Attr.
Body
Root
Execute

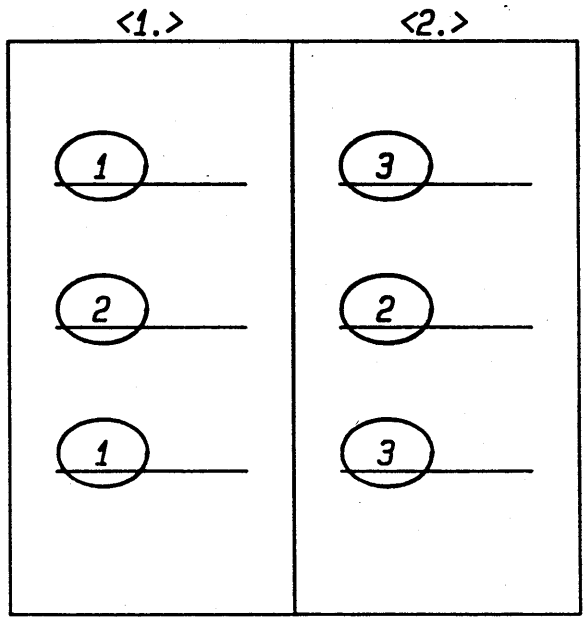
FhG AGD

< set polyline index (" to be inherited") >

```
create segment (a)
polyline
set polyline index 2
polyline
set polyline index (" to be inherited")
polyline
close segment

create segment (b)
set polyline index 1
execute (a)                    <1.>
set polyline index 3
execute (a)                    <2.>
close segment

post (b)
```



FhG AGD

function	Edit state list	PHI-GKS state list
open (create)	copied from PHI-GKS state list	
set attribute	set values	set values
re-open	set from segment header + scan to edit position (set values)	

PHI-GKS

Workstation Control

- workstation independent segment structure vs. structure store
- WISS always active
- associate, insert, copy
- archival

Problems between GKS and CGM

FhG AGD

GKS		CGM	
TEXT	- point - string	TEXT	- point - append flag - string
CELL ARRAY	- corner points - nx, ny - colour array	CELL ARRAY	- corner points - nx, ny - local colour precision - colour array
SET PATTERN REPRESENTATION	- WS identifier - pattern index - nx, ny - colour array	PATTERN TABLE	- pattern index - nx, ny - local colour precision - colour array

Problems between GKS and CGM

GKS		CGM	
SET TEXT FONT AND PRECISION		TEXT FONT INDEX TEXT PRECISION	
SET FILL AREA INTERIOR STYLE		HATCH INDEX PATTERN INDEX	
SET ASPECT SOURCE FLAGS	- 13 flags	ASPECT SOURCE FLAGS	- n (ASF type; ASF value)
GDP	systematic use of a GDP identifier	GDP	no GDP identifier for standardized GDPs

PDC

Picture Description Call Interface

- A device independent graphics interface to DDIF/LAL

Paul H. Wong
10th July, 1986

digital software

Picture Description Call Interface

A device independent graphics interface to DDIF/LAL

Introduction

(0)

PRESENTER	PAUL H. WONG PIXEL::PWONG
GROUP	Document Processing Systems / Graphics
FORMER PROJECTS	VAX DECgraph / VAX DEC de
CURRENT PROJECTS	PDC / CHARTER / XPRESS in WPSplus V3.0 (GOLD)

Picture Description Call Interface

A device independent graphics interface to DDIF/LAL

Agenda

(1)

(1) AGENDA

(2) WHY PDC?

(3) WHY TREE STRUCTURE?

(4) ORIGINAL PDC ARCHITECTURE

(5) CURRENT PDC/LAL ARCHITECTURE

(6) KEY FUNCTIONS OF PDC

(7) PDC FRONT END TO THE LAL

(8) PDC PRIMITIVES

(9) EXPERIENCE FROM THE GOLD PROTOTYPE

(10) QUESTIONS?

Picture Description Call Interface

A device independent graphics interface to DDIF/LAL

Why PDC?
(2)

XPRESS and CHARTER

Needed a device- and operating environment-independent graphics interface for the VAX and the PC (Rainbow at that time).

Wanted to produce the same output file format for interchange purposes.

Wanted a hierarchical data structure for picture description:

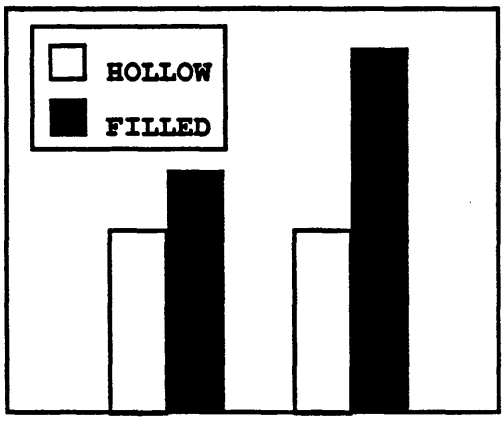
- to have full tree structure vs. DECslide's linked list structure
- to use the tree structure for inheritance and generic references.

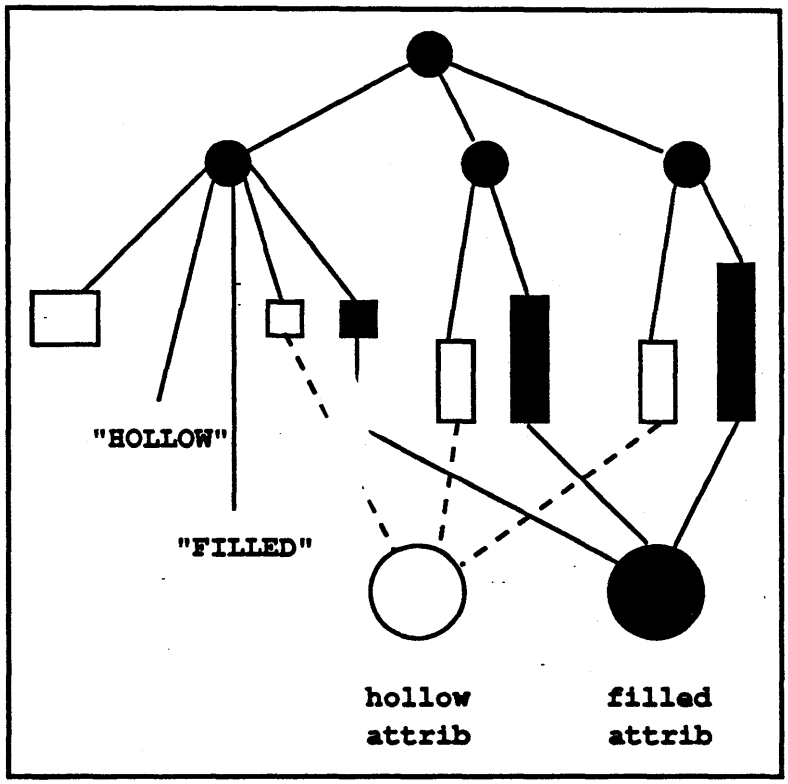
Picture Description Call Interface

A device independent graphics interface to DDIF/LAL

Why Tree Structure?

(3)





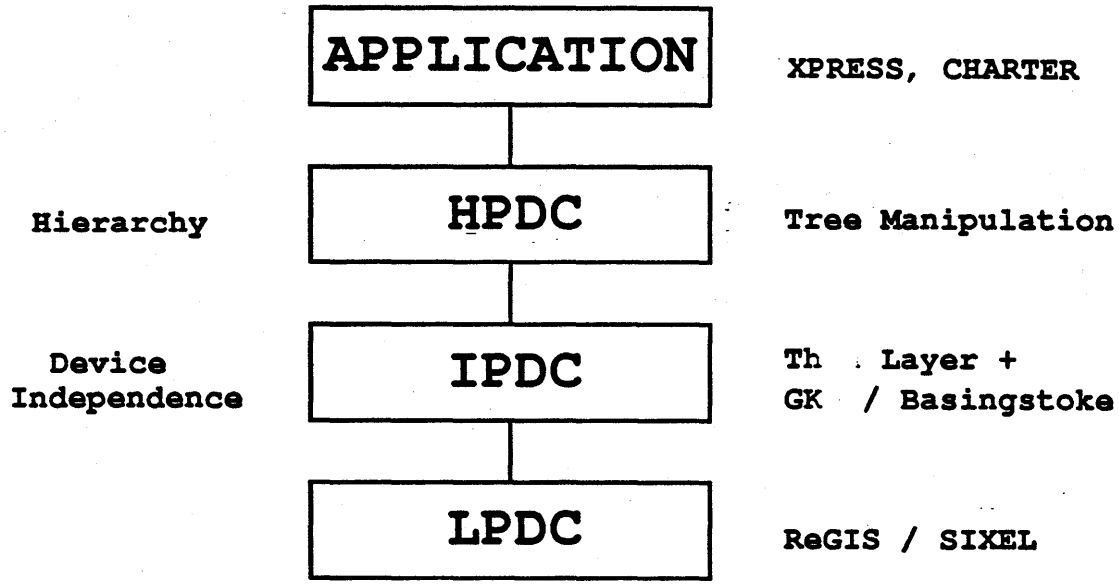
TYPICAL PDC TREE

Picture Description Call Interface

A device independent graphics interface to DDIF/LAL

Original PDC Architecture

(4)

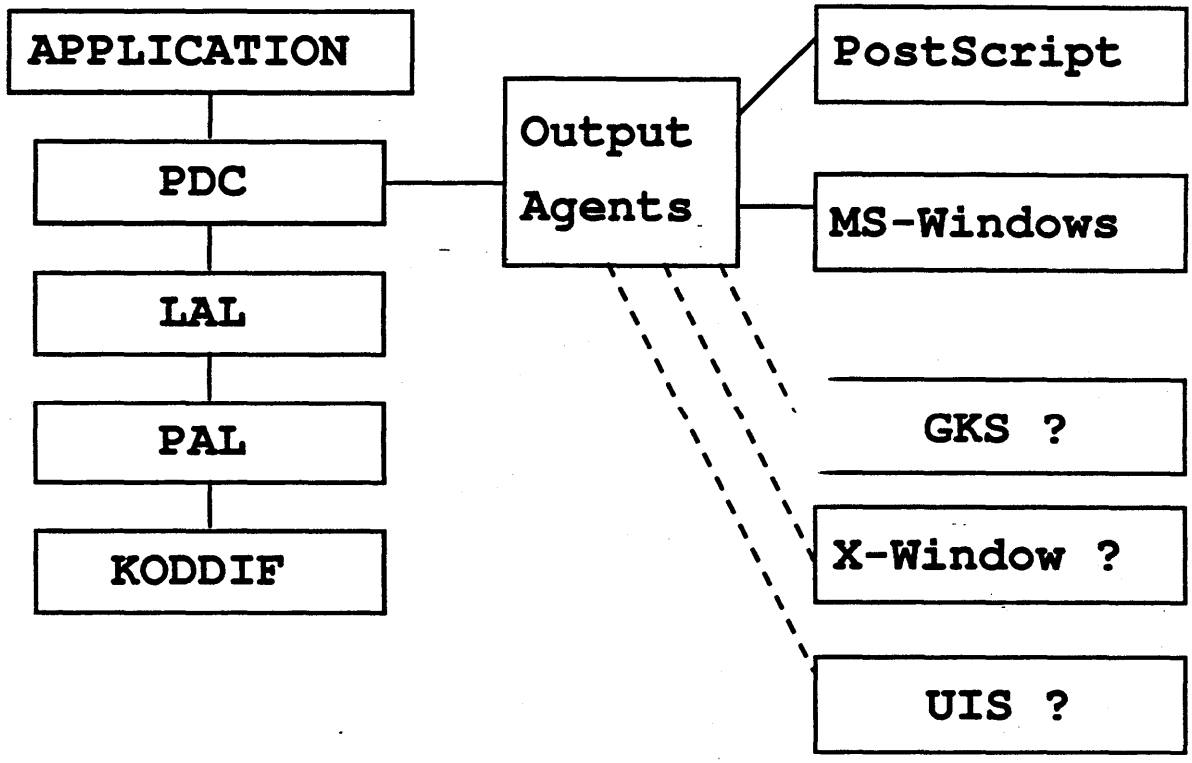


Picture Description Call Interface

A device independent graphics interface to DDIF/LAL

Current PDC/LAL Architecture

(5)



Picture Description Call Interface

A device independent graphics interface to DDIF/LAL

Key Functions of PDC
(6)

KEY FUNCTIONS

Inquire / Manipulate Primitives

Inquire / Manipulate Attributes

Traverse the LAL tree, passing data and attribute information to output agents

Issue MS-Windows calls

Issue PostScript calls

Picture Description Call Interface

A device independent graphics interface to DDIF/LAL

PDC Front End to the LAL

(7)

FRONT END TO LAL

PDC currently provides a front end to LAL to:

- Walk (traverse) a sub-tree
- Copy a sub-tree
- Delete a sub-tree
- Move a sub-tree

VOLATILE STATUS

These functions form a log al layer so we can develop code independent of the LAL sch dule. This layer might vanish. If that happens, the applica on will make calls directly to the LAL.

Picture Description Call Interface

A device independent graphics interface to DDIF/LAL

PDC Primitives
(8)

USAGE OF PRIMITIVES

PDC deals with Graphics Primitives in
in 4 ways:

- It processes them and puts them onto the LAL tree
- It edits them on the LAL tree
- It displays them in MS-Windows
- It translates them into PostScript code

LIST OF PRIMITIVES

- ARC
- CIRCLE
- ELLIPSE
- PIE
- ELLIPTICAL ARC
- ELLIPTICAL PIE
- MULTIPOINT
- SPLINE

Picture Description Call Interface

A device independent graphics interface to DDIF/LAL

Experience from the GOLD Prototype

(9)

IMPORT SARAH GRAPHICS

We successfully read in the Sarah "Helicopter" DDIF file with over 1000 line segments and were able to display it in PostScript and edit it with XPRESS.

Here are the steps we took:

- The Sarah prototype generated a DDIF file
- File format was translated to KODDIF format
- PDC used KODDIF to read the file into a PDC tree.

Note that the imported image was not exactly the same as the original Sarah one because the PDC prototype chose to ignore certain attributes in the DDIF file due to schedule constraints.

WRITING PORTABLE CODE

All the PDC, XPRESS and HARTER code was written in C in a portable manner. The prototype ran on the VAX and the Rainbow with no major difficulties.

FILE SIZES

We did some file size polling for the prototype. Here's a comparison of relative file sizes:

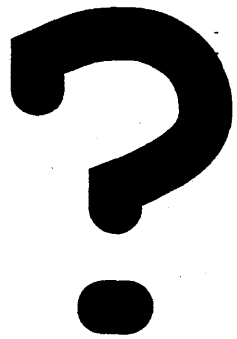
- DECslide file	- 100%
- PDC flat file	- <50%
- ASCII DDIF file	- >300%
- Binary DDIFfile	- <50%

Picture Description Call Interface

A device independent graphics interface to DDIF/LAL

Questions?

(10)



FhG
AGD

Interfaces and Data Formats for Transfer and Communication in Computer Graphics Systems

J. Encarnacao

J. Schönhut

FhG
AGD

Graphische und Geometrie-Normen für Graphische Systeme und ihre Anwendungen

FhG
AGD

Text Models

J. Schönhut

Text Models in Documents and Graphics

3 basic Models:

- Text & Graphics side by side
  "graphics" text & "text" text

- everything Text:
  Mosaic Graphics only

- everything Graphics:
  should simplify situation

B U T

Attributes	DDIF	GKS	CEPT
Backgroud (color)	*		
Color	*	*	
Font	*	*	
Rendition	*		
Tab Stops	*		
Wrap Format	*		
Quad Format	*		
Line Spacing	*		
Position (horiz./vert. * abs/rel)	*		
Precision	*	*	
Path	*	*	
Left Margin	*		
Right Margin	*		
Indent	*		
Top Margin	*		
Bottom Margin	*		
Vertical Alignment	*	*	
Horizontal Alignment	*	*	
Up Vector	*	*	
Spacing	*	*	
Expansion	*	*	

Attributes (continued)	DDIF	GKS	CEPT
Underline		*	
Size (normal, double height, double width, double)		*	
Flash		*	
Conceal		*	
Invert		*	
Marked (for further action)		*	
Protected		*	

Textprocessing:

string --> break into resp. catenate multiple lines, hyphenation, blocks

a. *b.*

1D *=====>* *2D*

processable

Graphics:

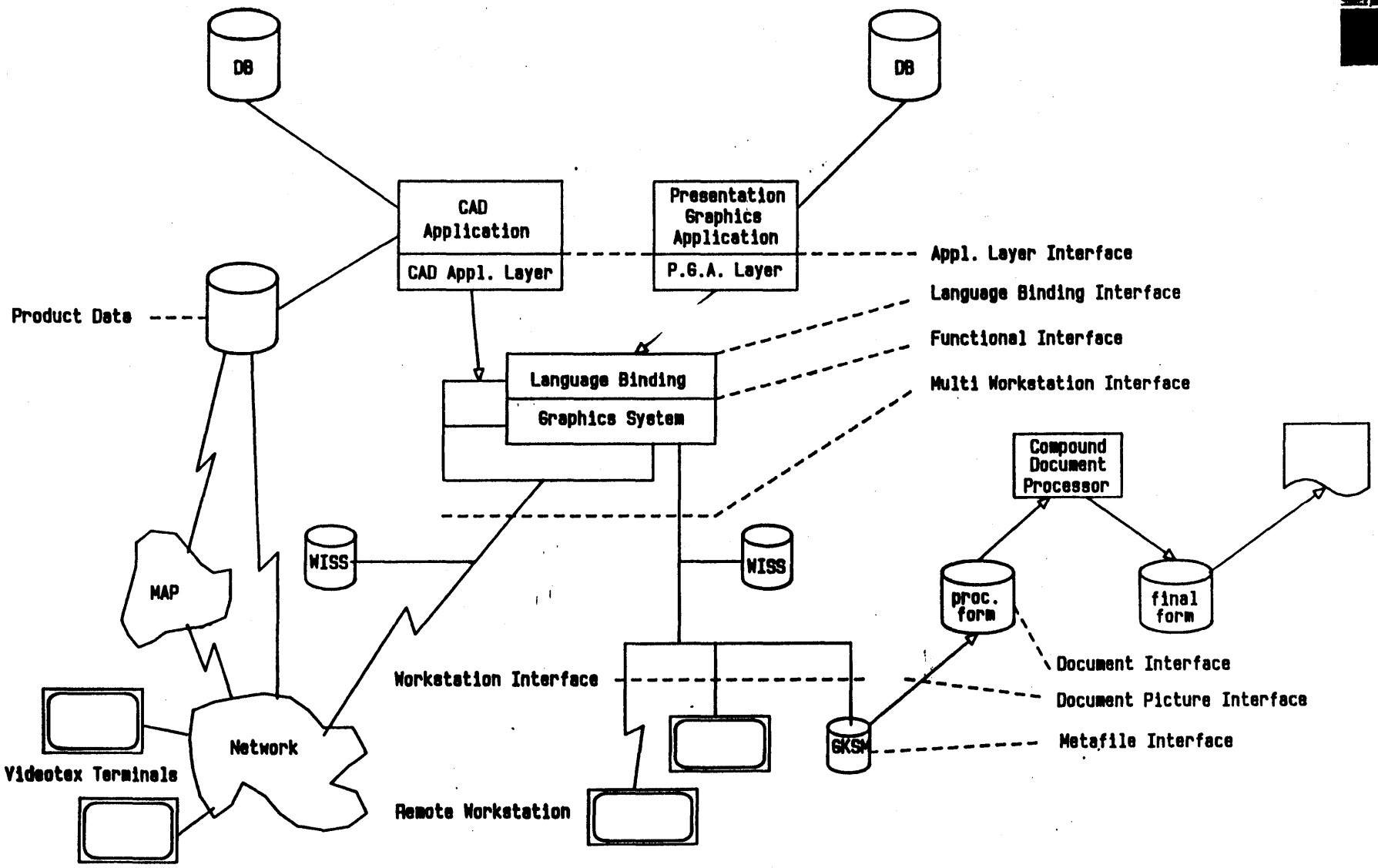
string --> exact po tioning of strings, no modific ion

c. *d.* *c.*

2D *or even* *3D* *=====>* *2D*

processable

Currently there is
NO simple way of mapping b. to c. in general case except if breaking it down to single characters.



FhG-AGD

Functional and Data Interfaces

FhG AGC

Presentation Graphics Layer on Top of GKS

Interface GKS - Presentation Graphics Package

- primitives
- data structures
- cluster/levels

Interface to the Environment

- operating system
- language binding
- interface to data handling utilities
- interface to methods handling utilities

Operator Interface

- passive/interactive
- dialog
- interaction techniques

FhG
AGD

GKS in a Network Environment

WSI of GKS opens possibility of using GKS in a network environment

GKS oriented communication protocol is based on services supplied by the T.70 transport protocol

DFN

FhG
AGC

Product Definition Data

- STEP
- PDES
- SET
- MAP Manufacturing Automation Protocol (General Motors)
- EDIF Electronic Design Interchange Format

FhG
AGD

Graphics in Documents

Two Standards:

- Standard General Markup Language SGML
- Office Document Architecture/Office Document Interchange Format ODA/ODIF

- logical and layout structure
- processible and image form
- integrate graphics in fo[illegible] of graphics metafiles
- compound documents

FhG
AGD

IGES

IGES includes three entity types:

- geometry (point, line, circle ...)
- dimensioning/annotation (label ...)
- structure (drawing, font and view relationships)

IGES history

- NBS standard
- ANSI standard (immature s age)

VDAFS

Geometric entities:

- point
- point set
- point vector set
- composite curve (include parametric splines)
- parametric spline surface

Videotex

- alpha-mosaic character graphics
- scanned image facsimile mode graphics
- geometrically encoded graphics

output subset of GDS as Videotex Standard Geometric Encoding

NAPLPS
- not efficient encoding
- poor functionality

IGES

IGES file contains:

- Prolog Section
- Global Section
- Directory Entry Section
- Parameter Data Section
- Terminator Section

IGES file concept is strongly directed to the "transfer of drawings"

CGM

- descriptor elements
- control elements
- picture descriptor elements
- graphical elements
- attribute elements
- escape elements
- external elements

used as GKS Level 0 Metafi

Device & Wkst Interfaces

Workstation Interface (WSI)

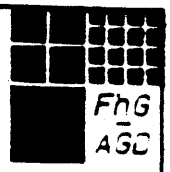
Separation of

- GKS Kernel
- GKS Workstation

CGI (ISO)

ECMA GDS

- GKS functionality
- adaption of the output subset by CEPT



Functional Standards

- GKS (ISO 7942)
- GKS-3D
- PHIGS
- PHI-GKS

Migration Issue

- from GKS to GKS-3D
- from GKS-3D to PHI-GKS
- from GKS-3D to PHIGS

Graphics Metafiles

Ideally: formats for storing transmitting pictures device application independent

GKSM

GKSM contains:

- *file/picture header*
- *end item*
- *control items*
- *output primitive items*
- *attribute items*
- *non graphical items (user items)*

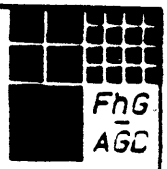
GKSM: input & output workstation

Data Interfaces

- GKS, PHIGS, PHI-GKS & Language Bindings
- CEPT Videotex (GDS subset)
- CGI (VDI)
- CGM (VDM)
- US Videotex Interface NAPLPS
- IGES, VDAFS, SET, STEP
- SGML & ODA/ODIF
- Presentation Graphics (based on GKS)

GKS Impact

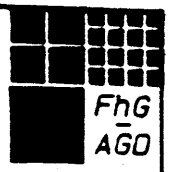
- VLSI support of GKS implementations
- GKS based applications for LANs and WANs



Computer Graphics Standards

Milestones

1974	GSPC by ACM SIGGRAPH
1975	DIN
1976	SEILLAC I (Methodology in Computer Graphics)



Graphics System

- Operator (User)
- Graphics Support System (Services)
- Other User Interface Support System (Services)
- Application Functions
- Generic Action Routines
- Data Base

Graphics System

- Application
  - Control
  - Heuristics

- Model
  - Model Metafiles (Design Data)
  - Model Data Management

- Graphics
  - Graphics Viewing
  - Request Processing
  - Graphics Metafiles

- User (Operator of the Computer-Aided Environment based on a Graphics Support)

FhG
AGD

Comparison

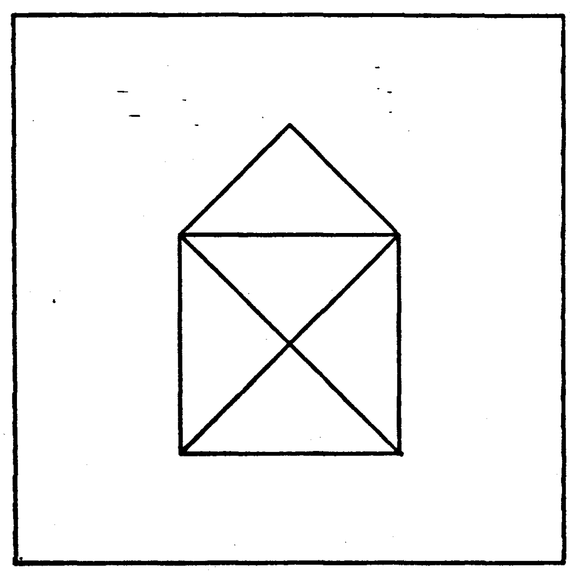
of

DDIF, SGML and ODA/ODIF

J. Schönhut

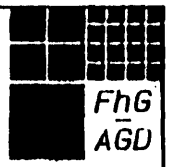
FhG-AGD

AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAA



The House of Nicolaus

BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
BBBBBB



cument_Descriptor

Major_version	0 (?)
Minor_version	11 (?)
Application_id	2**15 (not registered)
Product_name	"SlideX"

cument_Profile

Title	"DDIF Example"
Author	"FhG-AGD Daun/Puchtler/Schoenhut"
Version	"0.00"
Date	"22-May-1986 09:23:23.00"
Fonts	fontfile (?)

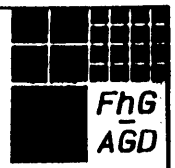
```
cument_Header
Page_Layout_Defs
  Page_Desc_Name "Testpage"
  Blocks
    Block_Desc
      Lower_Left      944, 944     (2cm, 2cm)
      Upper_Right     472, 9440    (1cm, 20cm)
      Block_Content
        Begin_Segment
          Segment_Id       "Fhg Logo"
        TEXT8_Content          "FhG-AGD"
        End_Segment
    Block_Desc
      Lower_Left      13700, 944     (29cm, 2cm)
      Upper_Right     1416, 9440   (3cm, 20cm)
```

```
:ument_Content
Begin_Segment
  Local_Attributes
    Segment_Bindings
      Registered_Tag        14    (Paragraph_tag)
TEXT8_Content   "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA...AA"
Begin_Segment
  Local_Attributes
    Segment_Bindings
      Registered_Tag        21    (Graphics_Object_tag)
    Frame_Attributes
      Frame_Height          3780  (8cm)
      Frame_Width           3780  (8cm)
```

```
  Frame_Position
    Frame_X_Position
      X_Rel      2      (frame_x_center)
    Frame_Y_Position
      Y_Rel      0      (Frame_y_centered)
  Top_Space       472  (1cm)
  Quadrant        1     (Origin lower left) OPTIONAL
  Frame_Transform
    Sx   378          (coordinates in range of)
    Sy   378          (0 to 10 scaled to full frame)
Color_Map
    1    0    0    0     (background black)
    1    1    1    1     (foreground white)
    1    1    0    0     (pure red for house)
Line_attributes
  Line_Color     2
```

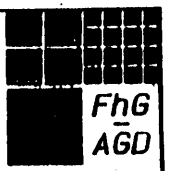
```
Multipoint
  Points     3, 2,  7, 6,  5, 8,  3, 6,  7, 2,  3, 2,  3, 6,  7, 6,  7, 2
End_Segment
Begin_Segment
  Local_Attributes
    Segment_Bindings
      Registered_Tag        19    (Single_Text_Line_tag)
    Frame_Attributes
      Frame_Height          472  (1cm)
      Frame_Width           3780  (8cm)
      Frame_Position
        Frame_X_Position
          X_Rel      2      (frame_x_center)
        Frame_Y_Position
          Y_Rel      0      (Frame_y_centered)
      Top_Space       472  (1cm)
      Bottom_Space    944  (2cm)
```

```
EXT8_Content   "The House of Nicolaus"
nd_Segment
EXT8_Content   "BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB....B"
nd_Segment
```




```
ELEMENT P          - -   (#RCDATA)                              >
ELEMENT
        GRAPHIC    - O   ((CGM | CGMREF), SUBTITLE?)
        SIZEX      NUMBER                          REQUIRED
        SIZEY      NUMBER                          REQUIRED
        POSX       (FLOAT | FIXED)                 FLOAT
        POSY       (FLOAT | FIXED)                 FLOAT
        X0         NAME                            UNKNOWN
        Y0         NAME                            UNKNOWN
        X1         NAME                            UNKNOWN
        Y1         NAME                            UNKNOWN
        XALIGN     (CENTER | LEFT | RIGHT)         CENTER
        YALIGN     (CENTER | TOP | BOTTOM)         CENTER
        SCMODE     (UNIFORM | NONUNIF)             UNIFORM
        ORIENT     ( 0 | 90 | 180 | 270 )          0
        PICNUM     NUMBER                          "1"
        FRAME      (FRAME | NOFRAME)               NOFRAME    >
```

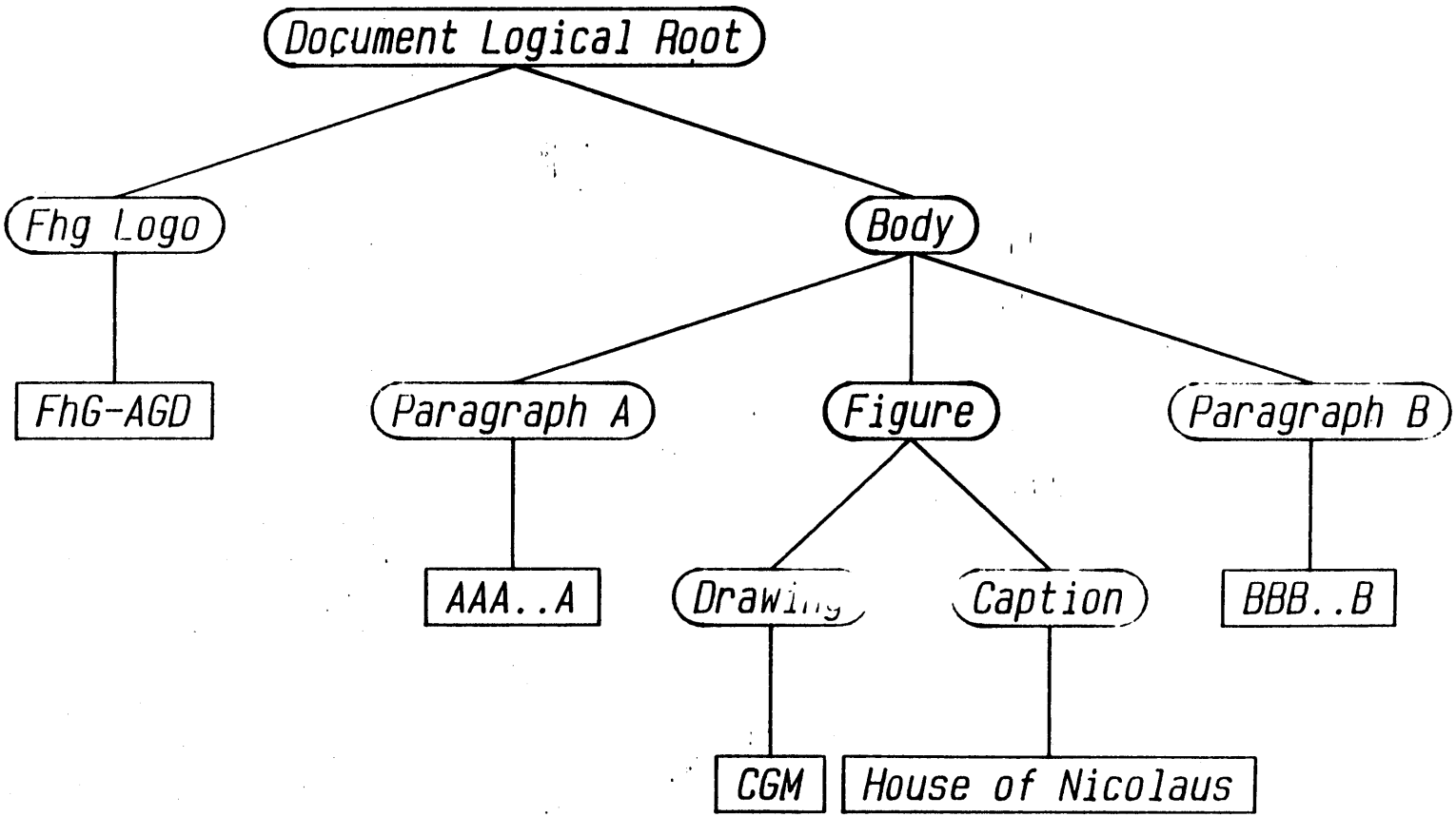
```
LEMENT CGM        - -       (#CDATA)
      CODING      (CLEAR |  CHAR)                CLEAR      >
LEMENT BANNER     - O       NULL
TEXT                        (#CDATA)                        >
LEMENT CGMREF     - O       NONE
      FILE        CONREF                         REQUIRED
      CODING      (CLEAR |  CHAR)                CLEAR      >
LEMENT SUBTITLE   - O       (#RCDATA)                       >
```




```
nner  text="FhG-AGD">
> AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA...A </p>
raphic sizex="80" sizey="80" >
m>
GMF "NICOLAUS";
RSION 1;
EMLIST "DRAWINGPLUS";
CTYPE REAL
GPIC OFF, 1, "BILD 1";
GPICBODY;
LOR TABLE 1, (1, 0, 0);
T LINE COLOR 1;
```

```
 (0.3000000E+00, 0.2000000E+00),
'000000E+00, 0.6000000E+00),
'000000E+00, 0.8000000E+00),
'000000E+00, 0.6000000E+00),
'000000E+00, 0.2000000E+00),
)000000E+00, 0.2000000E+00),
)000000E+00, 0.6000000E+00),
'000000E+00, 0.6000000E+00),
'000000E+00, 0.2000000E+00);
)IC;
IF;
im>
raphic>
)title> The House of Nicolaus </subtitle>
 BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB...B </p>
```

FhG AGD



G-AGD | ODA Specific Logical Structure

ct Type	DOCUMENT LOGICAL ROOT
ct Identifier	1
Visible Name	"Testpage"
rdinates	0, 1
ct Type	BASIC LOGICAL
ct Identifier	1 0
Visible Name	"FhG Logo"
ent Portion	0
ent Portion Identifier	1 0 0
ent	"FhG-AGD"

ect Type	COMPOSITE LOGICAL
ect Identifier	1 1
r Visible Name	"Body"
ordinates	0, 1, 2
ect Type	BASIC LOGICAL
ect Identifier	1 1 0
r Visible Name	"Paragraph A"
sentation Style	2
tent Portion	0
tent Portion Identifier	1 1 0 0
tent	"AAAAAAAAAAAAAAAAAAAAAAAAAAAAA...A"

ect Type	COMPOSITE LOGICAL
ect Identifier	1 1 1
r Visible Name	"Figure"
ordinates	0,1
ect Type	BASIC LOGICAL
ect Identifier	1 1 1 0
r Visible Name	"Drawing"
sentation Style	0
tent Portion	0
tent Portion Identifier	1 1 1 0 0
tent	/* sequence of geometric elements */

ect Type	BASIC LOGICAL
ect Identifier	1 1 1 1
r Visible Name	"Caption"
sentation Style	1
tent Portion	0
tent Portion Identifier	1 1 1 1 0
tent	"House of Nicolaus"

ect Type	BASIC LOGICAL
ect Identifier	1 1 2
r Visible Name	"Paragraph B"
sentation Style	2
tent Portion	0
tent Portion Identifier	1 1 2 0
tent	"BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB...B"

sentation Style	0
tent Type	Geometric Graphics
sentation Style	1
gnment	CENTRED
sentation Style	2
gnment	JUSTIFIED

COMPOUND DOCUMENT SYSTEMS

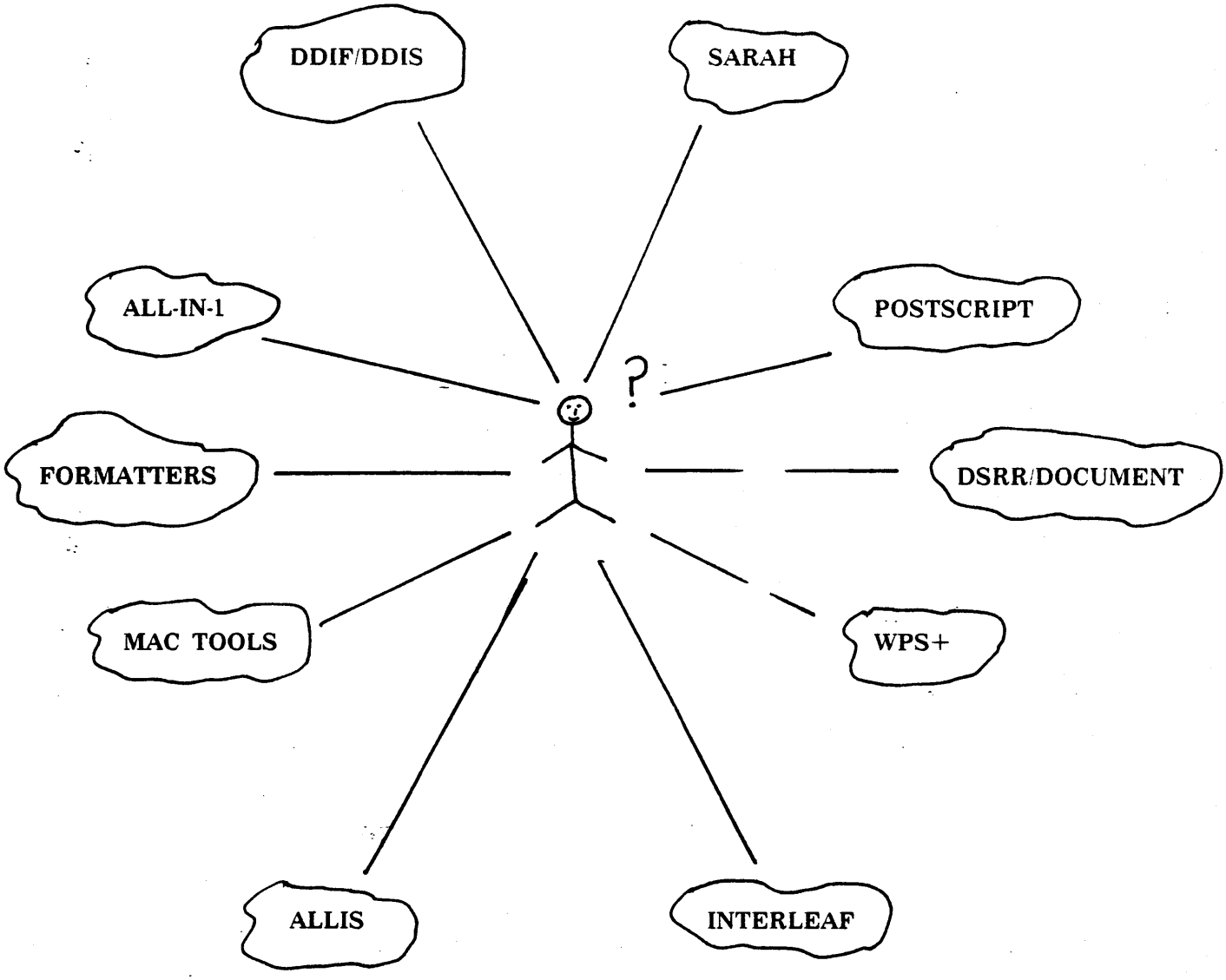
JULY 1986

Jim Kapadia

Workstation & Terminal SW Arch.
381-2326
ZKO2-3/Q08
ELUDOM::KAPADIA

digital

COMPOUND DOCUMENTS?



COMPOUND DOCUMENTS

USER'S VIEW OF COMPOUND DOCUMENT SYSTEM:

- COMPOSITE DOCUMENT PROCESSING FUNCTION

  .TEXT, LINE GRAPHICS...
  .IMAGES

- OPTIONAL FUNCTIONS LIKE:

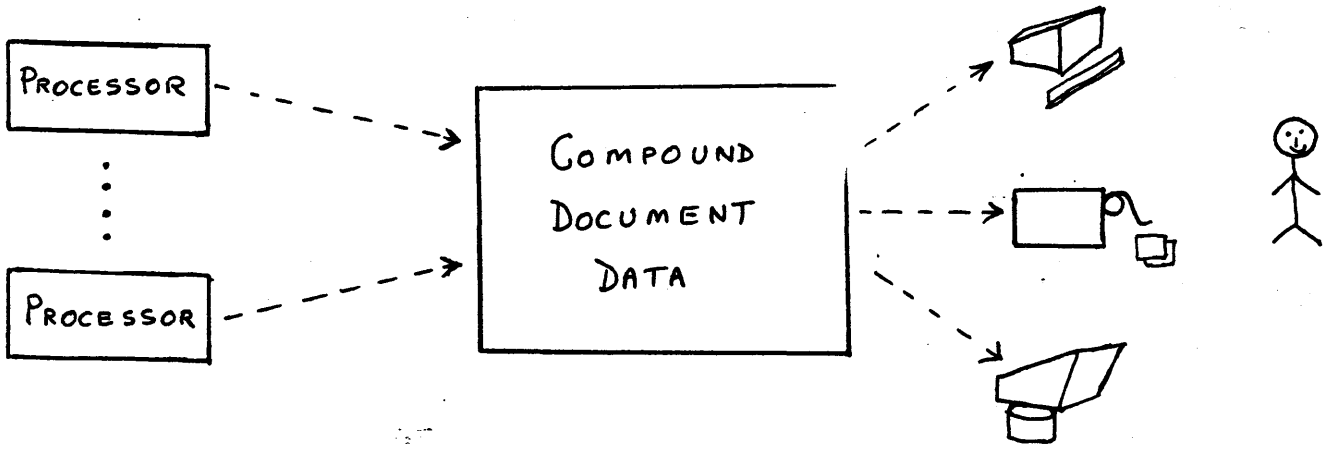
  .GRAPHING & CHARTING
  .SPREAD SHEETS
  .FORMS
  .DATA BASE QUERRY
  .....
  .....

IN ESSENCE, AN INTEGRATED SET OF FUNCTIONS (applications)

- OPERATING ON MULTI-MODE DATA (text, graphics,..)
- PERFORMING IN A UNIFIED MANNER
- APPEARING TO BE ONE SYSTEM TO THE USER

WHAT IS A COMPOUND DOCUMENT?

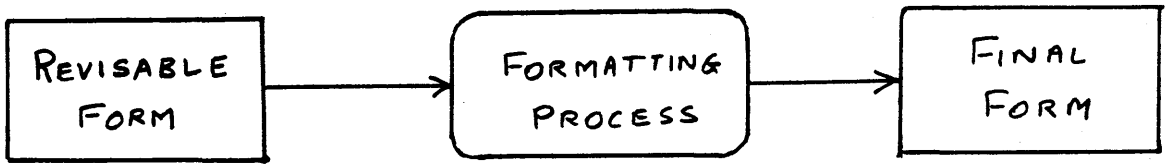
- ORGANIZED COLLECTION OF DATA
  - . TEXT, GRAPHICS, IMAGES,...
  - . PLUS OTHER STRUCTURED DATA LIKE
  - . CHARTS, GRAPHS, TABLES...
- FOR THE PURPOSE OF PRESENTATION TO USER
- CAPABLE OF BEING EFFICIENTLY PROCESSED



OPERATIONS ON COMPOUND DOCUMENTS

- RENDER · DISPLAY, PRINT, ...
- REVISE, EDIT, MODIFY, PROCESS, ...
- EXCHANGE, INTERCHANGE, ...
- TRANSMIT, RECIEVE, MAIL, COPY ...
- RETRIEVE & ACCESS
- ....

DEFINITIONS



- REVISABLE FORM:

  CD REPRESENTATION LENDING TO EFFICIENT MODIFICATION, REVISION, AND PROCESSING

  ABSTRACT RELATIONSHIPS NECESSARY FOR MODIFICATIONS ARE PRESERVED

- FINAL FORM:

  CD REPRESENTATION READY FOR ENDERING ON A DISPLAY DEVICE (VIDEO, PRINTER,..) FOR THE HUMAN

- FORMATTING:

  PROCESS BY WHICH COMPOUND DOCUMENT DATA IN A REVISABLE FORM IS TRANSFORMED INTO FINAL FORM SUITABLE FOR RENDERING FOR USER

- DATA INTERCHANGE FORMAT:

  REPRESENTATION OF DATA IN A FORM THAT LENDS ITSELF TO INTERCHANGE BETWEEN TWO PARTIES FOR THE PURPOSE OF INTERCHANGE AT A GIVEN LEVEL OF EXPECTATION

CHARACTERISTICS OF THE SYSTEM

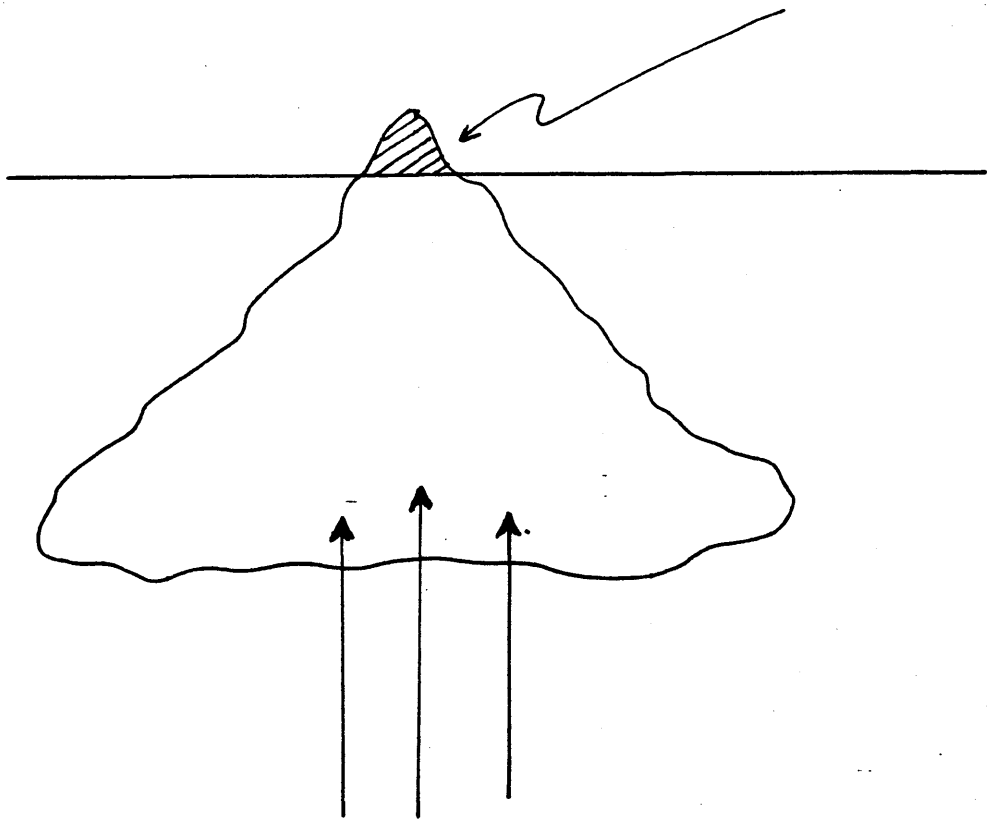
- TRANSPARENT DATA INTERCHANGE BETWEEN FUNCTIONS
- UNIFIED & COHERENT USER INTERFACE TO THE SYSTEM
- SMOOTH USER TRANSITION BETWEEN FUNCTIONS
- SEAMLESS FUNCTIONALITY OF SYSTEM
- ABIILITY TO ADD FUNCTIONS

ABOVE IMPLIES:

AN OPEN ENDED
COMPOUND DOCUMENT SYSTEM ARCHITECTURE

SCENARIO TODAY

PRODUCTS LIKE ALLIS, SYMPHONY... · TIP OF ICEBERG

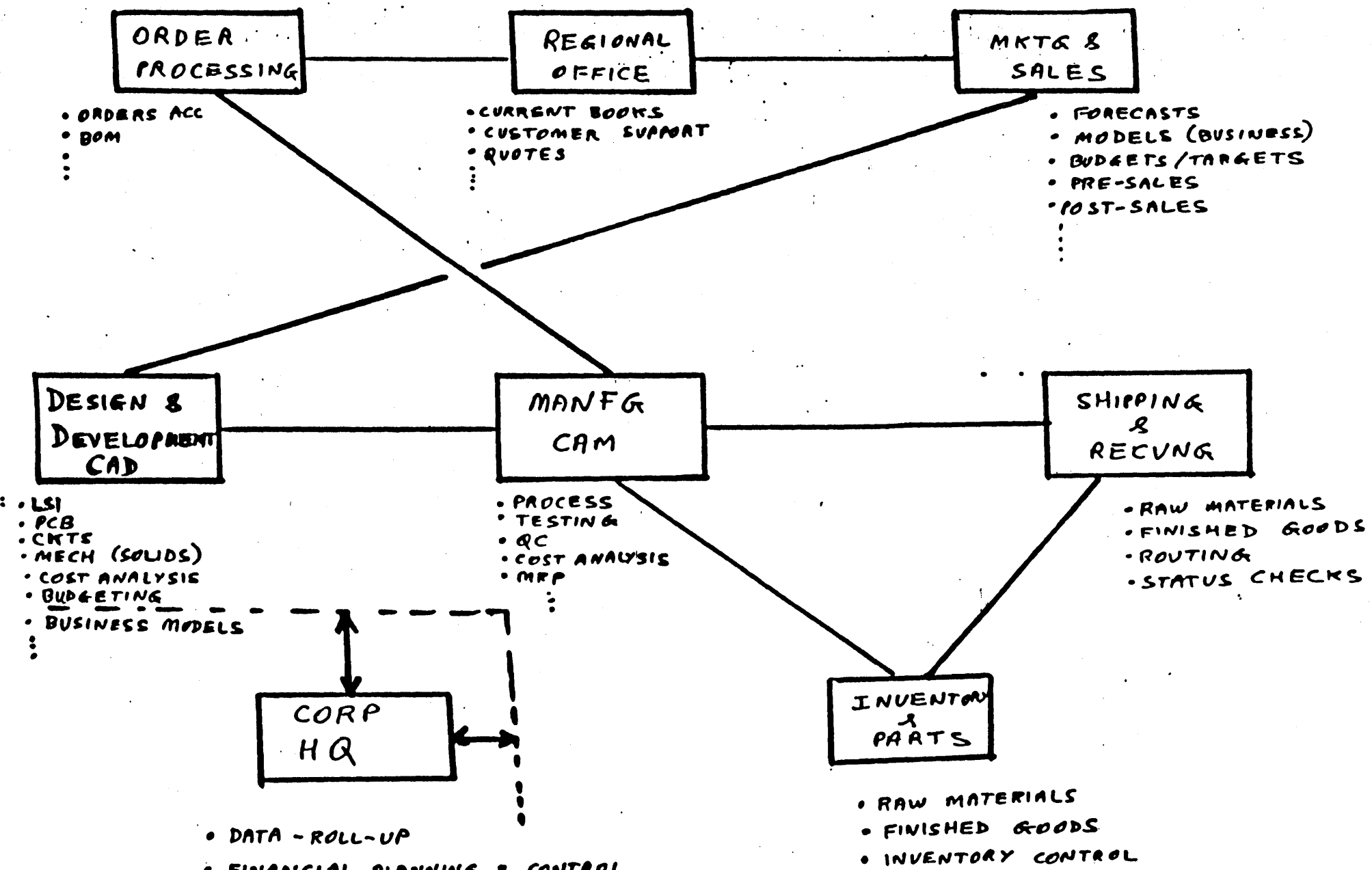


. INFORMATION SHARING AMONG VARIOUS PEOPLE

. MULTIPLE FUNCTIONS CORRESPONDING TO NEEDS

. PRESENTATION OF MULTIMODE DATA TO USERS

LIKELY FUTURE SCENARIO



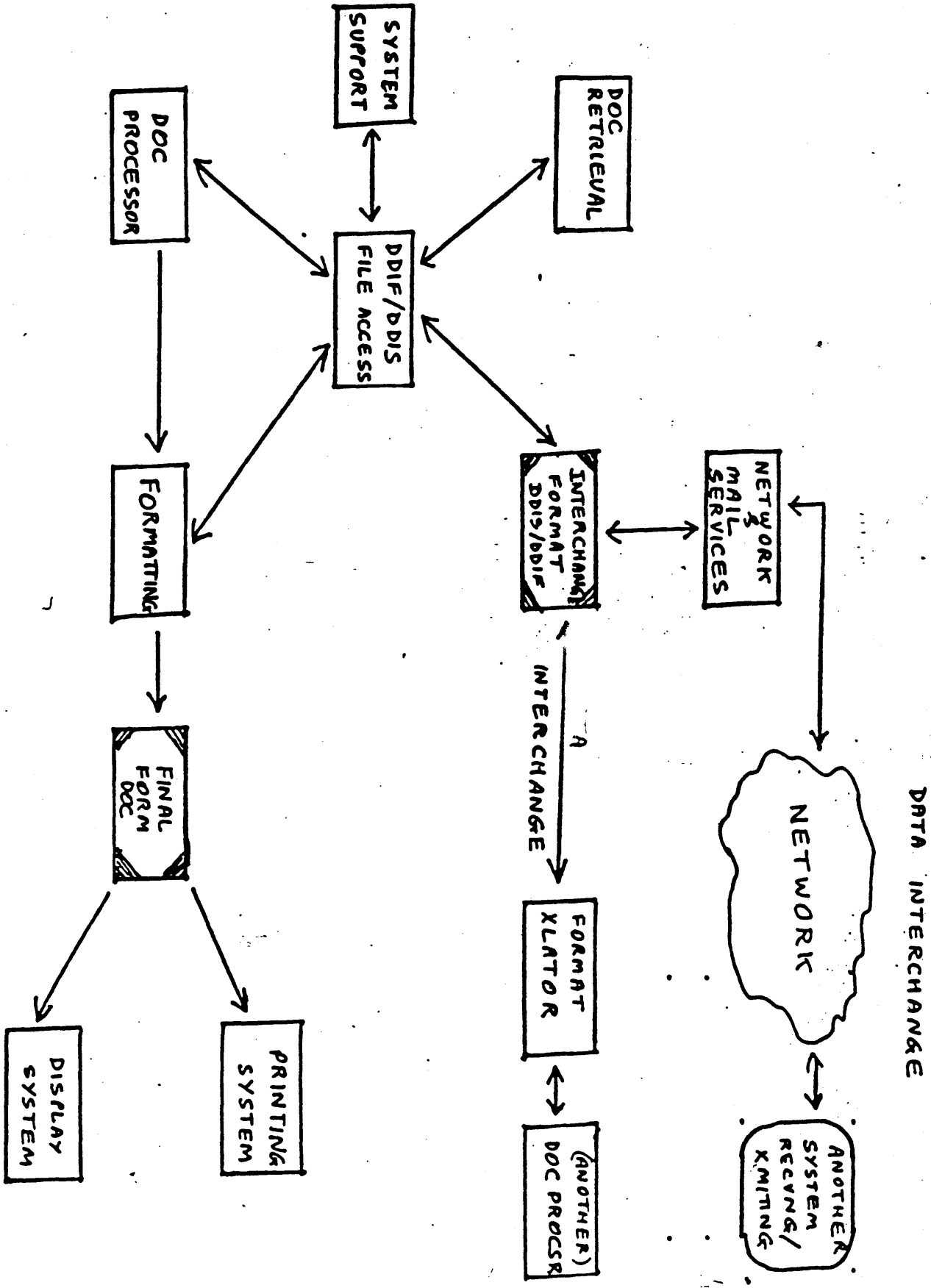
COMPOUND DOCUMENT SYSTEM ARCHITECTURE

FUNCTIONAL COMPONENT	SOFTWARE/ HARDWARE
• Document Processing	• Creator SW Editors - WYSIWYG, Batch...; Text, Graphics, • Spread Sheet Processors • Forms Processors • Chart Processors
• Formatting	• Print Formatters • Display Formatters
• Printing	• Printers - Laser, Impact, ..
• Display	• Video - WSs, PCs, Terminals • Display System SW . Graphics Subsystem . Drivers . PLP Translators

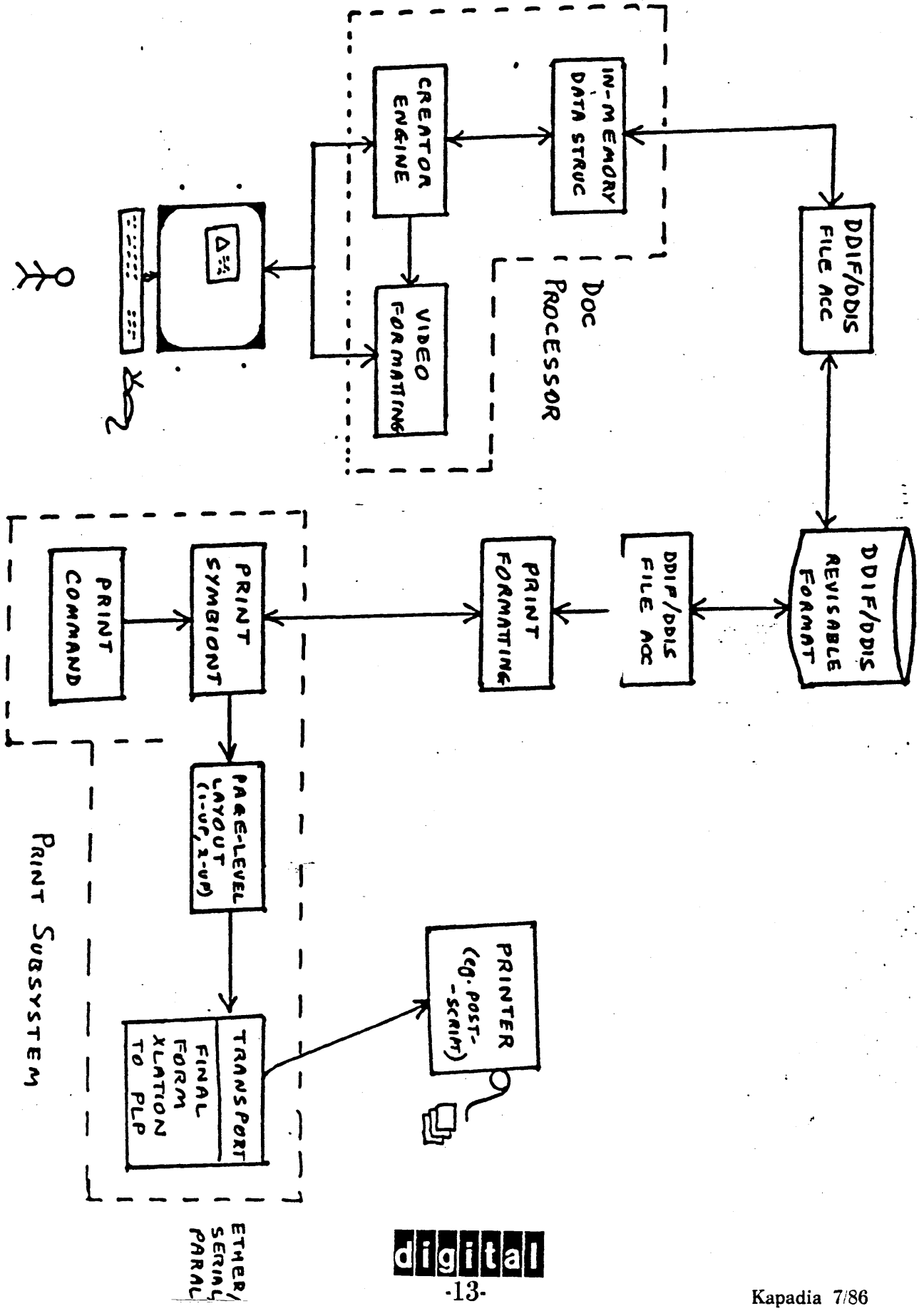
COMPOUND DOCUMENT SYSTEM ARCHITECTURE

FUNCTIONAL COMPONENT	SOFTWARE/ HARDWARE
• Data Interchange	• Intra-DEC: DDIF/DDIS Format Intra-vendor: IBM, Wang, .. ODIF, DCA,.. Gateways
• Document Distribution	• Networking Services • Mail Services • Intra-Cluster Services
• Document File Storage	• DDIF/DDIS File Storage & Access Services • DDIF/DDIS File Access • DDIF/DDIS File Storage
• User Document Retrieval	• Document Access & Retrieval System . File Cabinets,.. . File Folders,...
• System (OS) Support	• OS Utilities . Copy, Mail, Type, Print,....

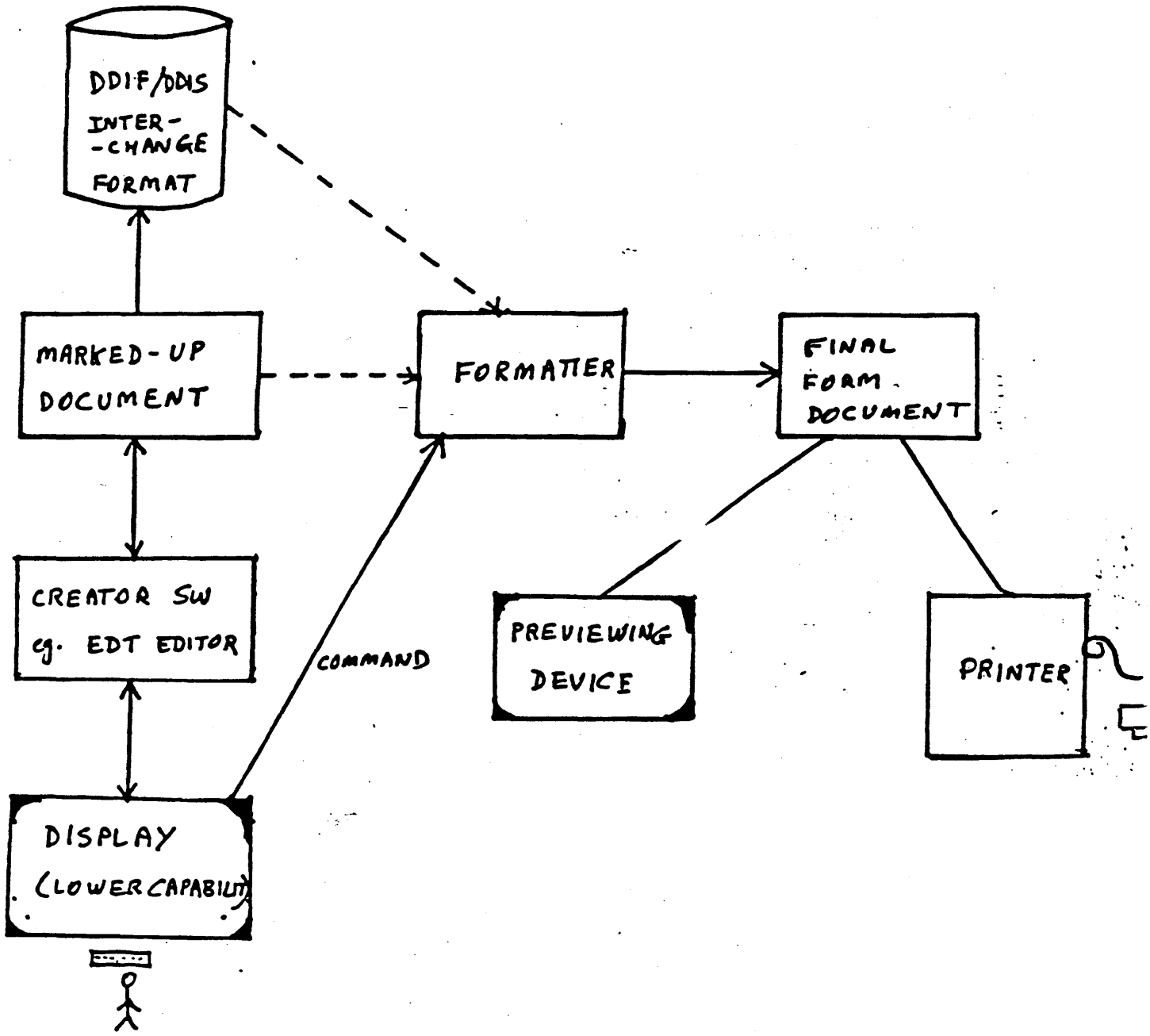
COMPOUND DOCUMENT SYSTEM ARCHITECTURE



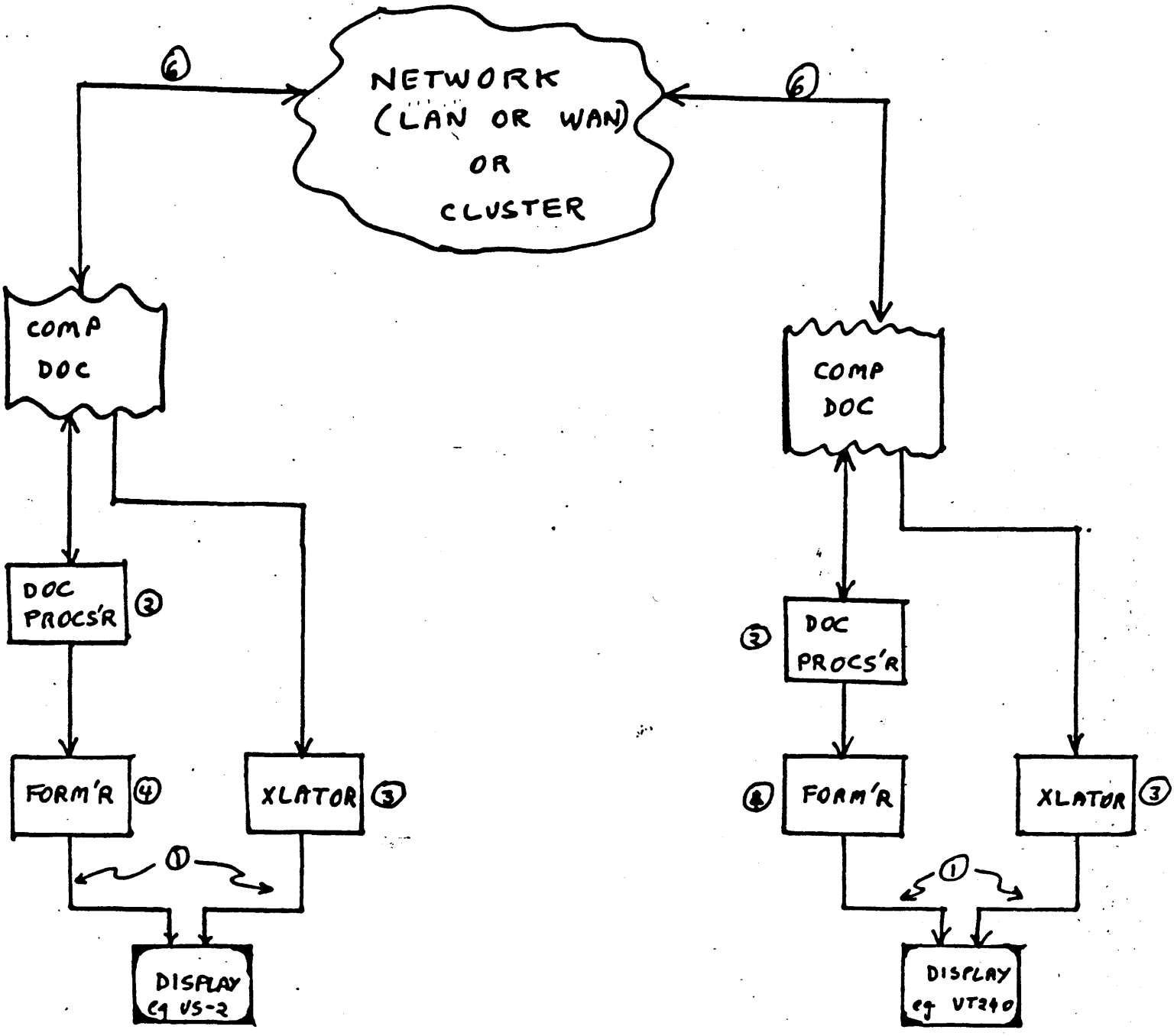
COMPOUND DOCUMENTS (WYSYIWYG) CREATION & PRINTING



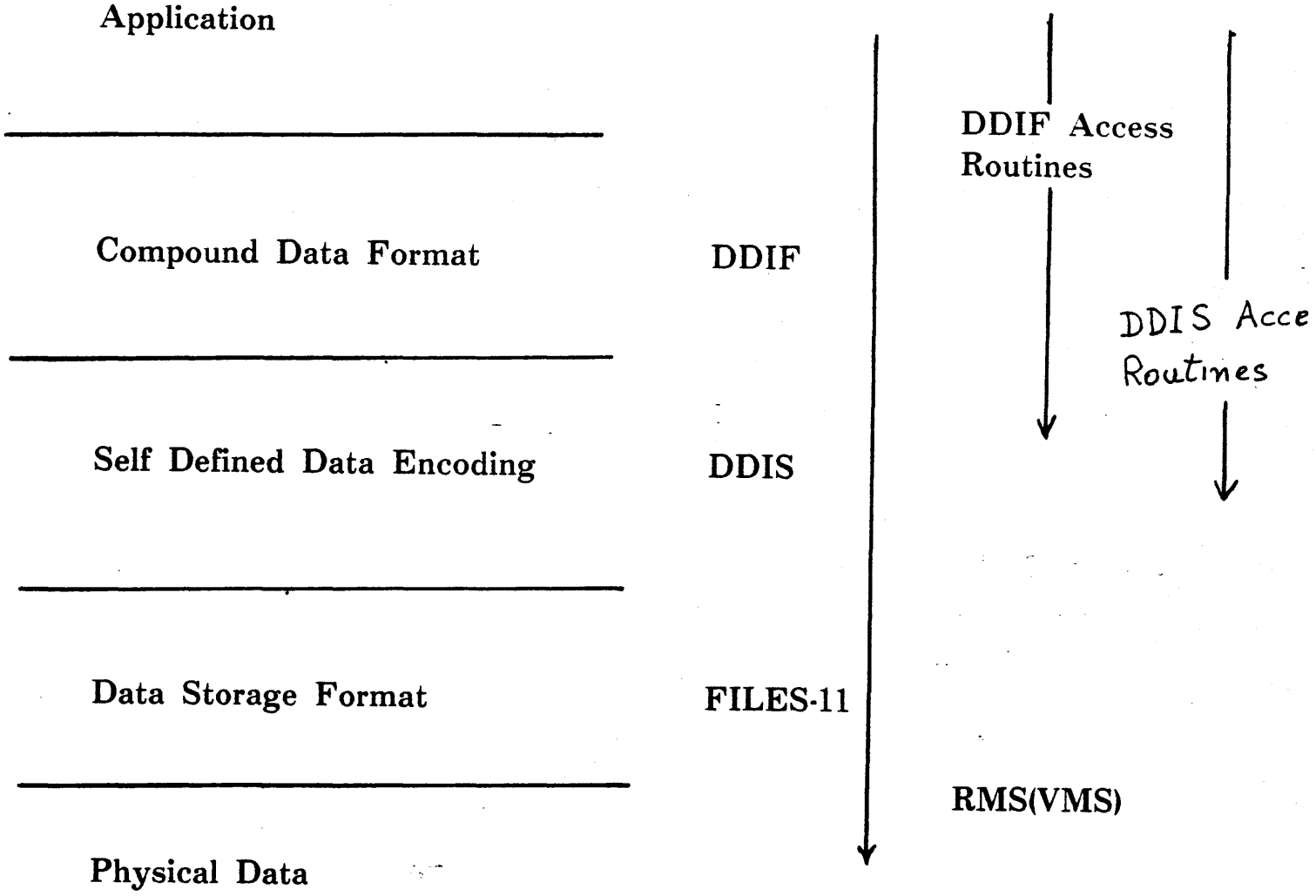
DOCUMENT PROCESSING/PRINTING (BATCH)
A LA RUNOFF, DOCUMENT



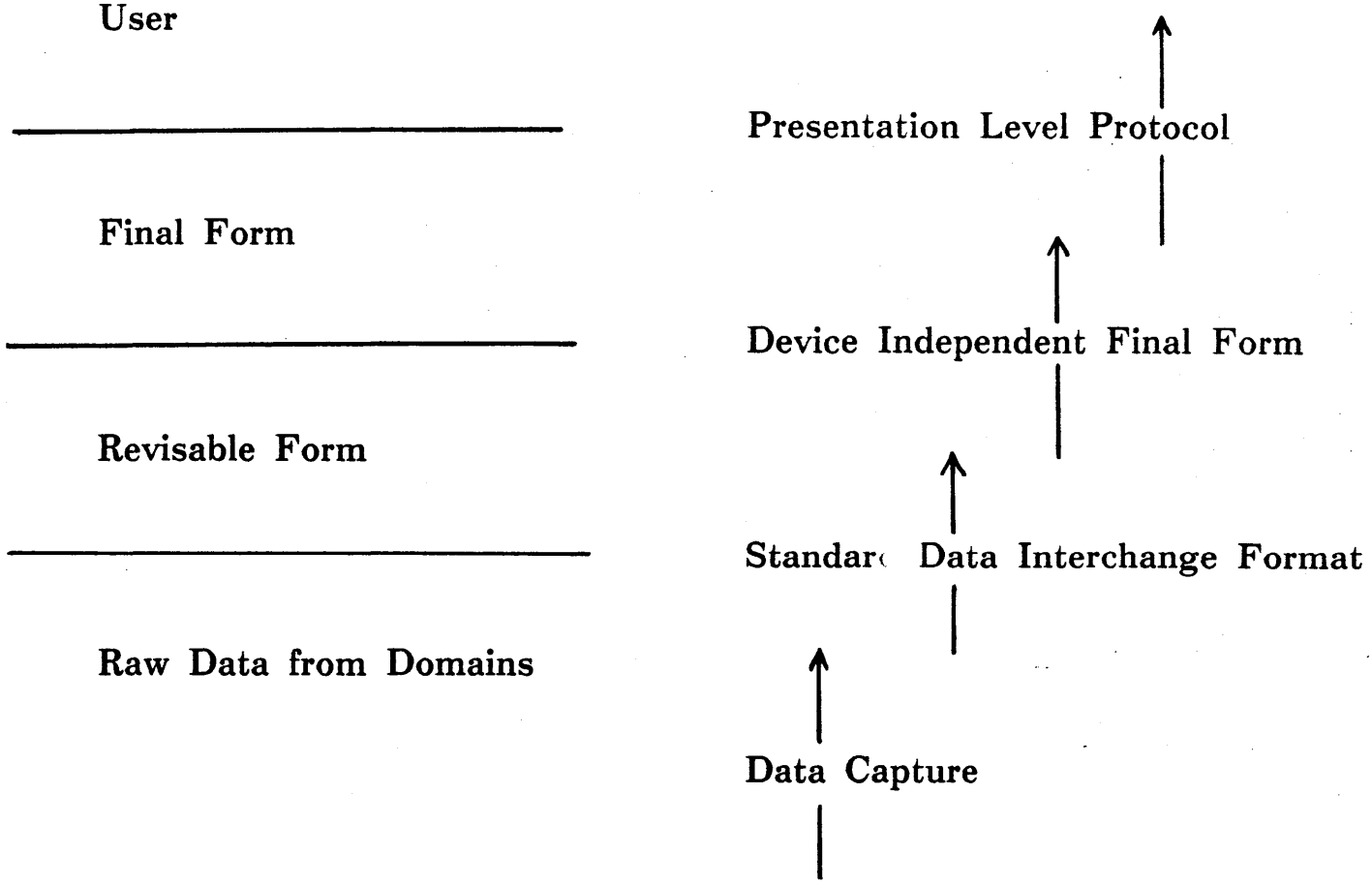
INTERCHANGE BETWEEN UNLIKE DEVICES - VIDEO & PRINTERS



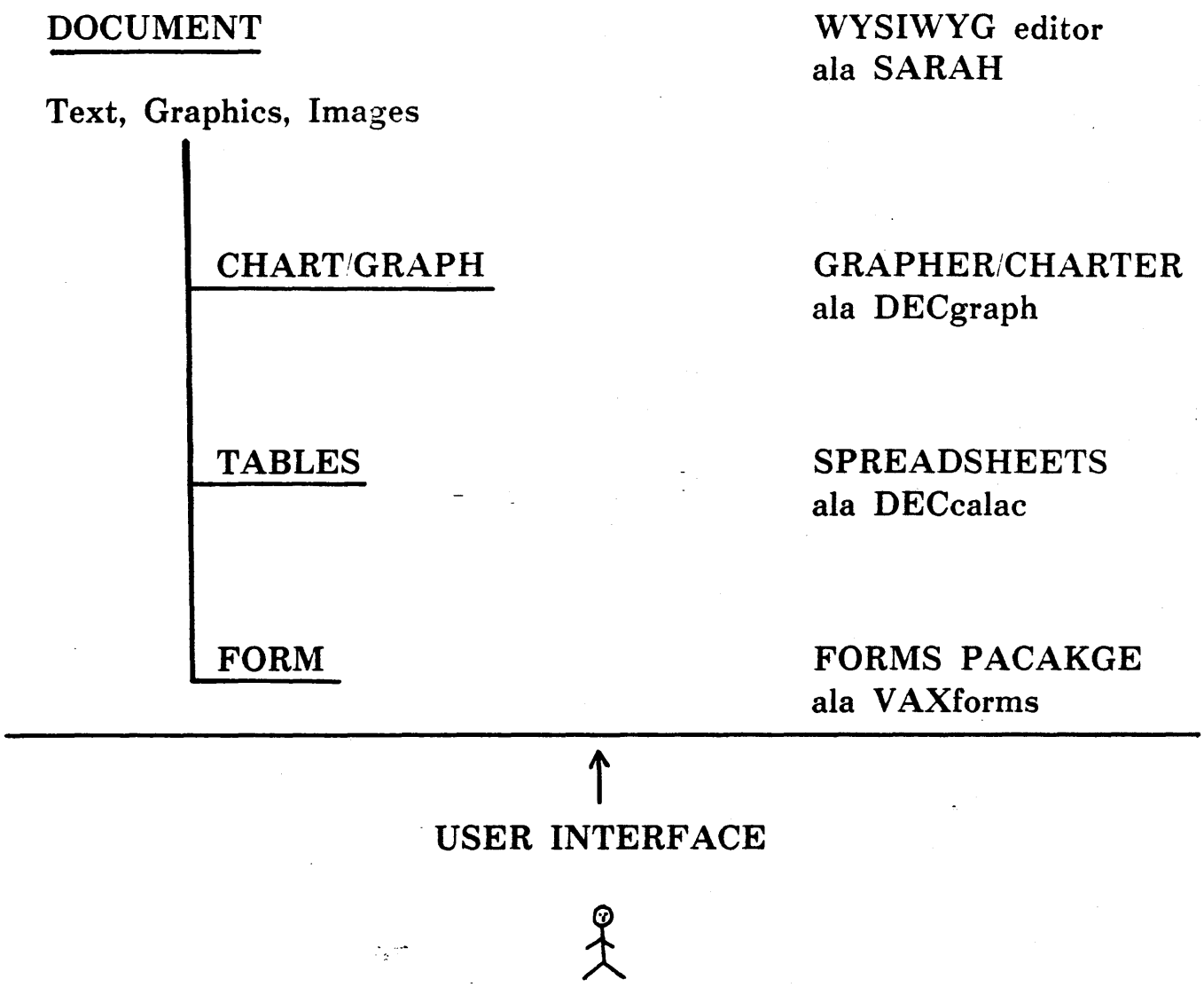
APPLICATION VIEW OF DATA



USER'S VIEW OF COMPOUND DATA



EXAMPLE - COMPOUND DOCUMENT DOMAINS



ISSUES

- ARCHITECTURE
- PRODUCT STRATEGY
- PRODUCT PLAN

ARCHITECTURE

- NEED AN OPEN SYSTEM ARCHITECTURE
- A MUST FOR THE LONG TERM
- FOUNDATION BLOCKS IN OR BEING PUT IN PLACE
- OPEN ARCHITECTURE - DEC'S EDGE OVER COMPETETION
- DIAF - ARCHITECTURE REVIEW PROCESS NEEDED
- RESOURCES - NEED PARTICIPATION & HELP

STATUS

- FORMING DIAF - MEMO/CHARTER OUT
- DATA INTERCHANGE ARCHITECTURE FORUM
  - . OVERALL REVIEW GROUP
  - . OVERVIEW, ARCHITECTURE & STRATEGY DEVELOPMENT GR
- OVERVIEW & TAXONOMY IN WORKS
- SYSTEM ARCHITECTURE OVERVIEW IN WORKS

PRODUCT STRATEGY

- WORKING ON A SSG PROPOSAL
- DEC PRODUCTS:
  - . SARAH *
  - . WPS V3 *
  - . DECGRAPH
  - . DECSLIDE
  - . VAXFORMS *
  - . DECPAGE
  - . RUNOFF
  - . DSRR *
  - . DOCUMENT *
  - . TEAM DATA/RALLY
  - . FMS
  - . ALL-IN-1
  - . MAIL (which one?)

PRODUCT STRATEGY

ISSUES

GOOD NEWS:

. SEVERAL FUNCTIONS AVAILABLE · POTENTIAL'S THERE!

BAD NEWS:

. EACH PRODUCT UNTO ITSELF
. LIMITED OR NO DATA INTERCHANGE BETWEEN FUNCTIONS
. DIFFERENT USER INTERFACES
. ROUGH SEAMS SHOWING BETWEEN FUNCTIONS
. FUNCTIONALITY MATCHING WITH USER EXPECTATIONS

. WHO IS RESPONSIBLE FOR THE SYS M PRODUCT?

In Summary

- DDIF/DDIS - Major role in compound doc systems
- Will evolve beyond current
- Need your help & co-operation in doing this

SlideX :
A Graphics Interpreter in Practical Use

Dr. Juergen Schoenhut

June 86, Schoe

produced b

SLIDEX
erlgraph

SlideX:

An Example for Integration of Text and Graphics

J. Schönhut

SlideX - ERLGRAPH - GKSMINT

Design - Principles

Portability of the System
- FORTRAN
- Machine Independence
  - (trade off with Efficiency)
- Modular Structure
- Use of Standards (GKS)
- Building Blocks

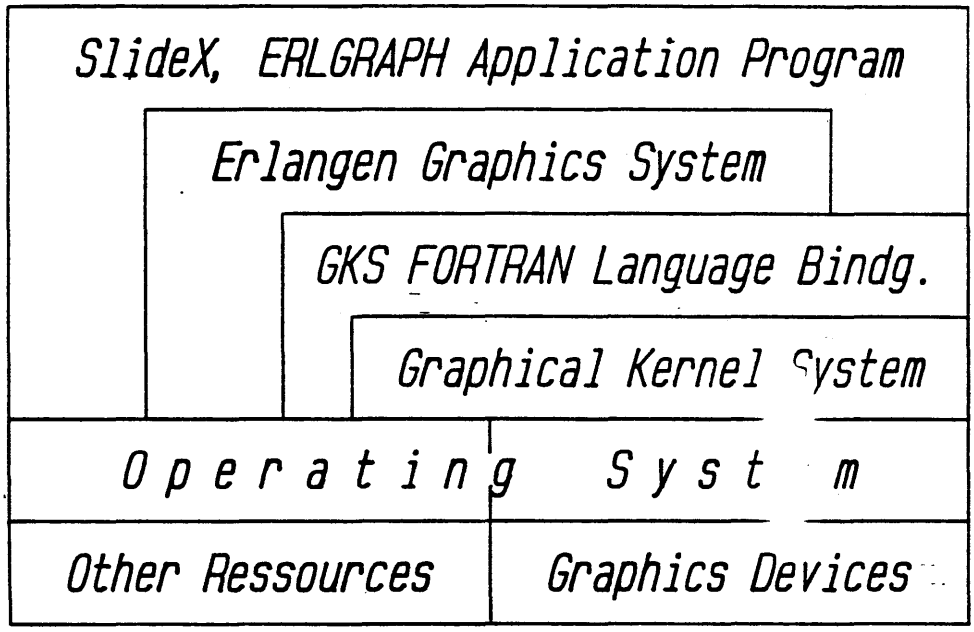
Machine Independence of Output

Device Independence of Output
- via GKSM

User Interface
- Simple
- more complex for complex Applications

SlideX - Structure

Layer Model with SlideX, ERLGRAPH and GKS



produced by

SLIDEX erlgraph

GKSMINT - Metafile Interpreter

Menue Control

Device Adaptation of Pictures

SlideX - ERLGRAPH

Tool Kit System for Picture Construction

Calls with few Parameters

Many simple Calls

Default Settings

All Defaults can be changed

Primitives of GKS available

Line, Marker, Text, Fill Area, Cell Array

Attributes of GKS available

Color, Line Width, Line Type, Fill Style

Clipping

User defined Rectangles

Drawing Sheet

Blanking

Convex Polygons (--> Fill Area)

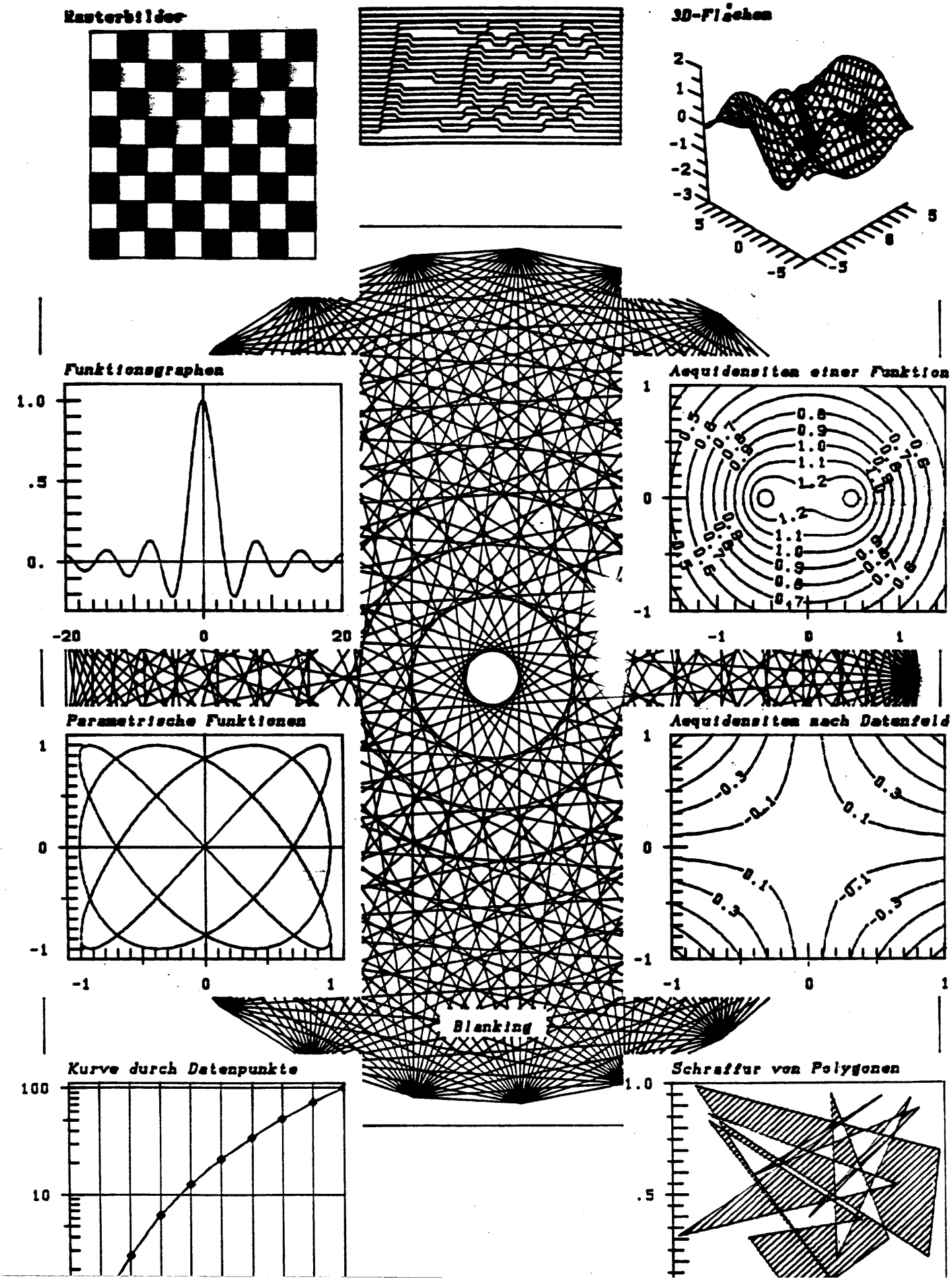
produced by

SLIDEX erlgraph

Coordinate Systems SlideX - ERLGRAPH

User Coordinates UC 2D
(cm, hard clip limits)
Problem Coordinates PC 2D
(Units defined according to Problem,
Position in UC 2D arbitrary,
Clipping possible)
User Coordinates UC 3D
(Unit of the User Volume,
Format Filling Projection by
Viewing Transformation)
Problem Coordinates PC 3D
(Units defined according to Problem,
Position in UC 3D arbitrary,
Page Concept in SlideX additional
Page - all 4 Coord. Systems
+ Line Positioning
Picture - all 4 Coord. Systems
within Rectangle within Page

produced by SLIDEX erlgraph


Rasterbilder
3D-Flächen
Funktionsgraphen
Aequidensiten einer Funktion
Parametrische Funktionen
Aequidensiten nach Datenfeld
Blanking
Kurve durch Datenpunkte
Schraffur von Polygonen

SlideX - ERLGRAPH Features

SlideX ERLGRAPH

2D Base Software

Drawing Sheet Administration
Definition, Excluded Areas
Lines + Attributes
Text + Attributes
Symbols + Attributes
Fill Areas + Attributes

Typwriter Mode

Geometric Figures

High Level 2D Software

Problem Coordinate Systems
Cartesian, Polar,
Linear, Logarithmic
Axes
Curves
Contour Lines — only from Arrays
Cell Arrays

SlideX Extensions

Business Graphics
Struktograms
Syntax Diagrams

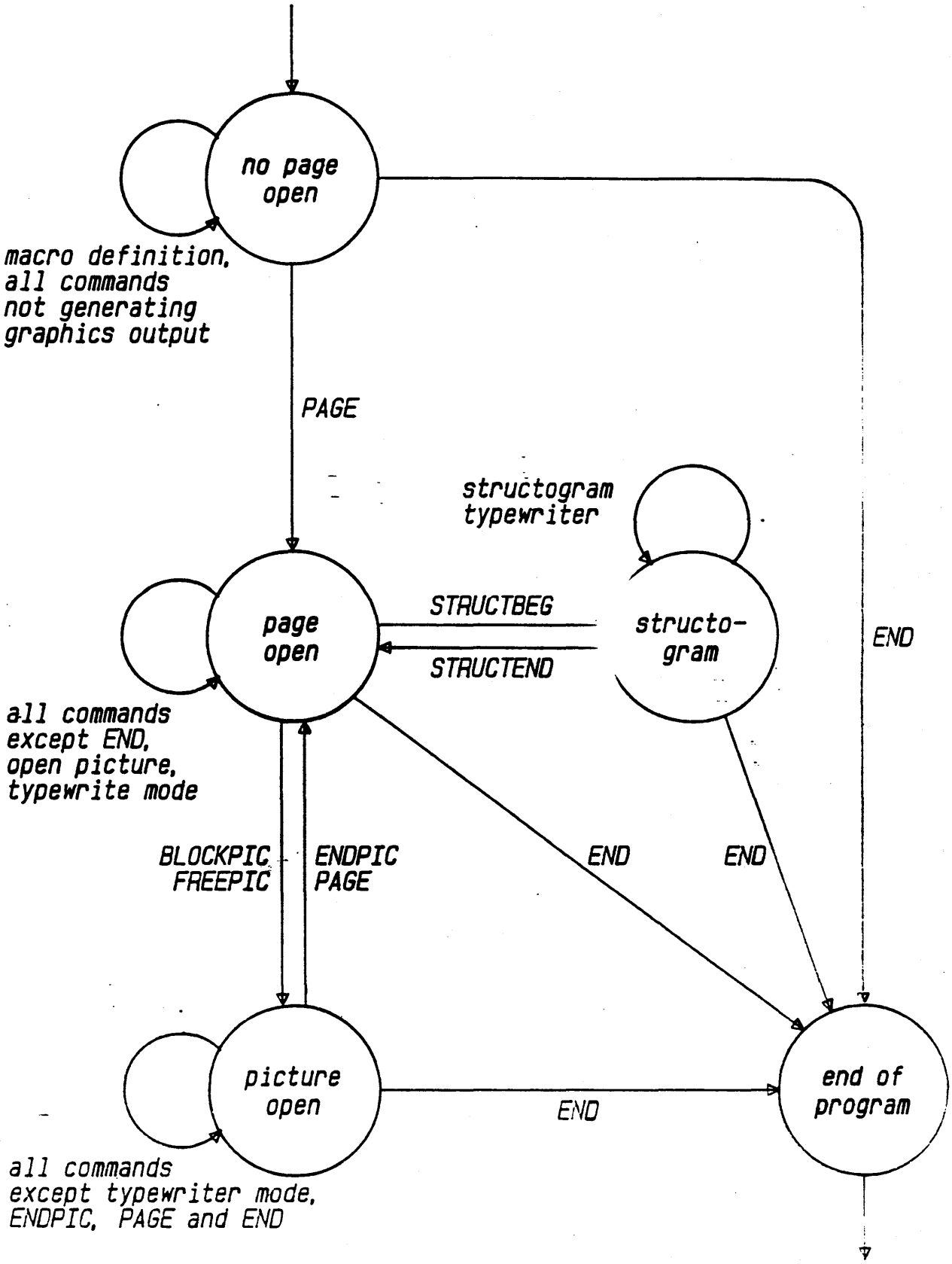
3D Software

Projections
Problem Coordinates
Axes
Area Nets (Hidden Lines)
3D Contour Lines (-"-) — only from Arrays

produced by

SLIDEX erlgraph

SlideX State Diagram



SlideX - Typewriter Mode

PAGESIZE, width, hight	SETAH, n - SETAQ, n
PAGNR, n	NOPAGNR - ONPAGNR
SETCOD, coding	
PAGE (, name)	
COMMENT, text	
LIST	NOLIST
END	
LMARGIN, li	RMARGIN, re
ADJUST (, minfill, maxfill)	NOADJUST
IN, n	OUT,
TOP - SETTOP, dist	BOTTOM - SETBOTTOM, dist
SETLINE, dist	NEGLF - SETLF, fac
SETTHI, hight	SETDST, fac
DOUBLE - HALF	LARGER - SMALLER
TYPE10 (, lines)	TYPE12 (, lines)
ASCII	DIN
CURSIV	NOCURSIV
SETPEN, index	BACKGROUND, index
BLACK RED GREEN BLUE	YELLOW MAGENTA CYAN

produced by SLIDEX ergraph

SlideX - Graphics Output

Picture Window

BLOCKPIC, breite, hoehe

FREEPIC, xlu, ylu, width, hight ENDPIC

NOFRAME ONFRAME

ERLGRAPH Commands

except Drawing Sheet Definition / FORTRAN Functions

Figures (solid filled, hatched, hollow)

BOXf,	xlu, ylu, xsize, ysize, rad,	bound, fill
NGONf,	n, xmid, ymid, rad, strt,	bound, fill
CIRCLEf,	xmid, ymid, rad,	bound, fill
ARCf,	xmid, ymid, rad, strt, ar	bound, fill
ELLIPSEf,	xmid, ymid, a, b, axa,	bound, fill
ELLARCf,	xmid, ymid, a, b, strt, ar, axa,	bound, fill

Connections ans Labels

ARROW, aimx, aimy

ARROWD, dx, dy

BRACES, x1, y1, x2, y2, aimx, aimy

BOOMERANG, x1, y1, aimx, aimy (, arrow)

BOOMERANG2, x1, y1, x2, y2, aimx, aimy (, arrow)

BOOMRADIUS, rad

TEXTCENTER, string

TXTEXT, string

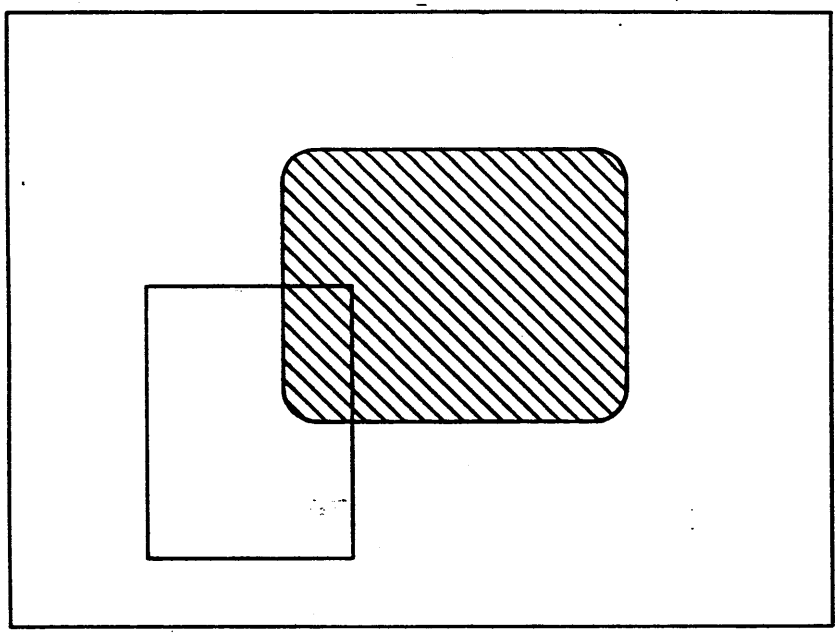
produced by

SLIDEX erlgraph

Example for SlideX - Graphics

Within a Blockpic a Rectangle and a filled Rectangle with "rounded Corners" is drawn.

```
#box, 2, 1, 3, 4
#sethch, 0.2, 135
#boxf, 4, 3, 5, 4, 0.5, 1, 1
```



June 86, Schoe

produced by

SLIDEX erlgraph

SlideX - Variables

DEFINE, name
DEFINE, name, dim1
DEFINE, name, dim1, dim2
PURGE, name
SET, name, expr
SET, name (i1 (, i2)), expr
APPEND, name, liste
EVICT, name, retain
COPY, from, to, start, end, step
CATENATE, name, string1 (, string2)
DUMP, expr
DISPLAY, string

SlideX - Functions

INT (X)	DIV (X1, X2)	MOD (X1, X2)	RND (X)	SGN (X)
ABS (X)	SQRT (X)	MAX (X1, X2)	MIN (X1, X2)	
EXP (X)	LN (X)	LOG (X)		
ATAN (X)	COS (X)	SIN (X)		
NOT (A)	AND (A1, A2)	OR (A1, A2)		
CHAR (I)	ICHAR (C)			

HIGH (0)	WIDE (0)	LINE (0)	LMAR (0)	RMAR (0)
X0 (0)	XSIZE (0)	Y0 (0)	YSIZE (0)	
XACT (0)	YACT (0)	XTACT (0)	YTACT (0)	
PAGE (0)	PEN (0)	THI (0)	CWI (0)	LF (0)
XEXT (0)	YEXT (0)			

produced by

SLIDEX
ergraph

SlideX - Business Graphics

Bar Charts

BarChart, title, xlu, ylu, xsize, ysize BarEnd

Bar, value Bars, array, len

LegendItem, x, y, string1..8

LabeId (xy) Ax, n, ticks (xy) Label, string1...10

MonthXAx, minmonth, maxmonth

Integer (xy) Ax, min, max, ticks

Linear (xy) Ax, min, max

Log (xy) Ax, min, max

BarVerti BarHo iz

BarRow BarCo_umn

BarEvict

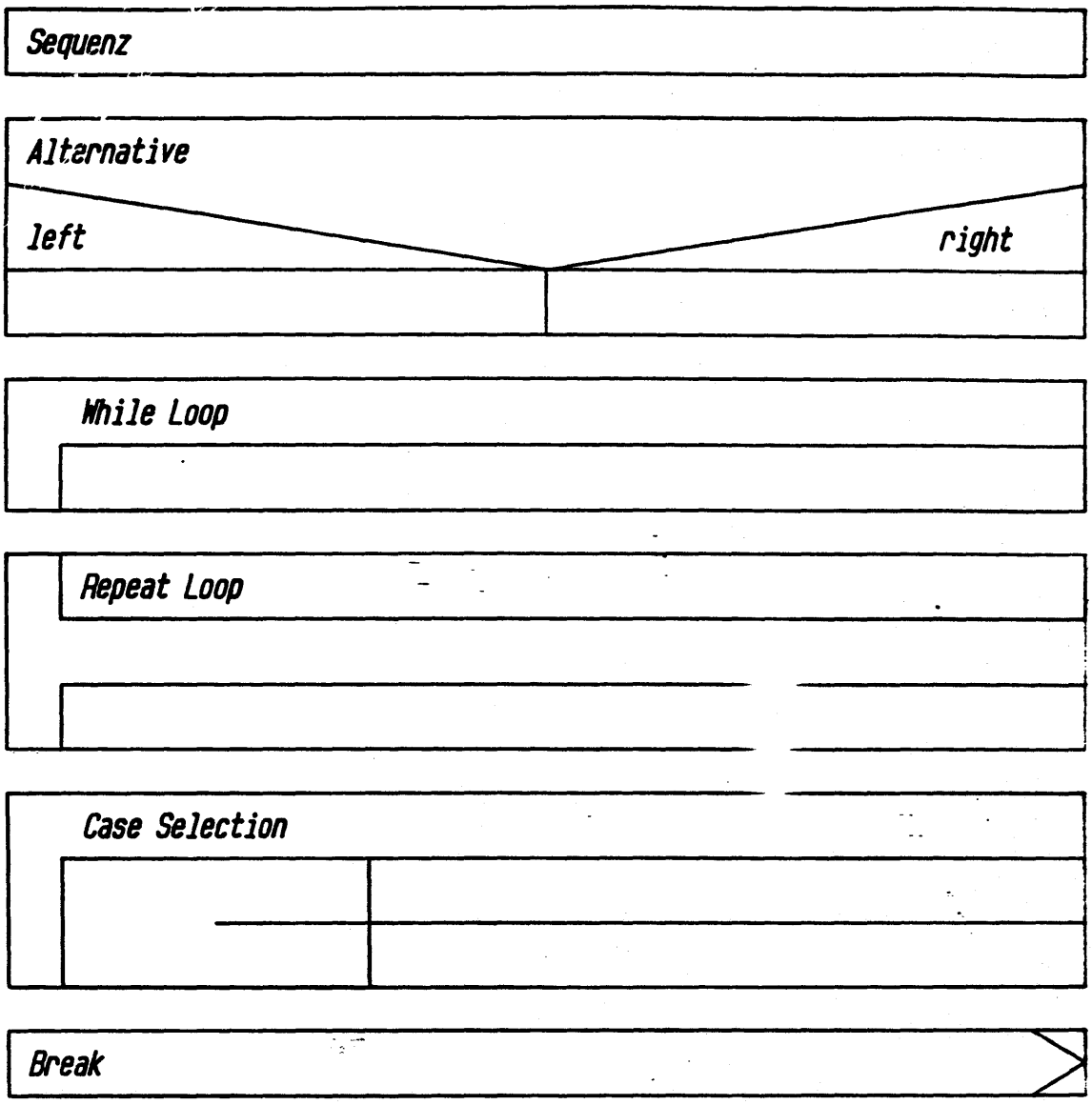
BarCoord CmCoord

BarIndex (, index)

produced b

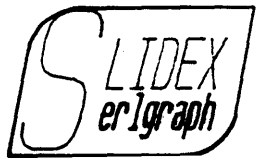
SLIDEX
erlgraph

SlideX -- Struktograms



June 86, Schoe

produced by



SlideX - Business Graphics

General Commands

BusiReset

BusiTitle, topflag

FillSolid	FillHatch
ColorIndex, index	ColorMax, number

HatchIndex, index

Pie Charts

PieChart, total, title	PieEnd
Piece, value, string	
Explode, offset	NoExplode

MinExplode, offset

PieCenter, x, y	PieRadius, r
PieAngle, degrees	
PieceLabel, index	PieTextEll, a, b

SlideX - Macros

Macro Definition, Invocation

MAC, macname	ENDM
ENTRY, macname	RETURN
DELMAC, macname	
macname, (parlist)	CALL, macname (, parlist)

Control Structures

IF,	condition,	command
IFNOT,	condition,	command
Block-IF,	condition	THEN cmds (ELSE cmds) ENDIF
Block-IFNOT,	condition	THEN cmds (ELSE cmds) ENDIF

FOR, varname, start, end, step mds ENDFOR

REPEAT cmds UNTIL, condition

WHILE, condition cmds ENDWHILE

CASE, caseindex, labellist

GOTO, name

LABEL, name