NBSIR 88-3727

A Collection of Technical Studies Completed for the Computer-aided Acquisition and Logistic Support (CALS) Program Fiscal Year 1987 Volume 2 of 4

Sharon J. Kemmerer, Editor

U.S. DEPARTMENT OF COMMERCE
National Bureau of Standards
Institute for Computer Sciences and Technology
Information Systems Engineering Division
Gaithersburg, MD 20899

March 1988

U.S. DEPARTMENT OF COMMERCE
N[illegible] OF STANDARDS

NBSIR 88-3727

A COLLECTION OF TECHNICAL STUDIES COMPLETED FOR THE COMPUTER-AIDED ACQUISITION AND LOGISTIC SUPPORT (CALS) PROGRAM FISCAL YEAR 1987 VOLUME 2 OF 4

Sharon J. Kemmerer, Editor

U.S. DEPARTMENT OF COMMERCE
National Bureau of Standards
Institute for Computer Sciences and Technology
Information Systems Engineering Division
Gaithersburg, MD 20899

March 1988

U.S. DEPARTMENT OF COMMERCE, C. William Verity, *Secretary*
NATIONAL BUREAU OF STANDARDS, Ernest Ambler, *Director*

EXECUTIVE SUMMARY

The overall objective of the Department of Defense Computer-aided Acquisition & Logistic Support (CALS) Program is to integrate the design, manufacturing, and logistic functions through the efficient application of computer technology. CALS is a program to apply existing and emerging communications and computer-aided technologies in DoD and industry to:

- Integrate and improve design, manufacturing, and logistic functions; thereby bridging existing "islands of automation."
- Actively influence the design process to produce weapon systems that are more reliable and easier to support and maintain.
- Shift from current paper-intensive weapon support processes to a highly automated mode of operation, based on a unified DoD interface with industry for exchange of logistic technical information in digital form.

The CALS program was established by the Deputy Secretary of Defense in September 1985 to implement the recommendations of a Joint Industry/DoD Task Force. Management is provided by a DoD Steering Group, an OSD CALS Policy Office, and their counterparts in each Military Department and the Defense Logistics Agency. The CALS Policy Office has obtained the support of the National Bureau of Standards in the selection and implementation of CALS standards. An Industry Steering Group has also been established to focus the work of key industrial associations and the defense contractor community in CALS implementation.

The Bureau has been funded since Spring 1986 to recommend a suite of industry standards for system integration and digital data transfer, and to accelerate their implementation. NBS activities during 1986 were primarily aimed at:

- familiarizing NBS technical staff with key DoD logistic functions and CALS demonstration projects,
- briefing DoD personnel, contractors, and other interested parties on Federal, national, and international standardization efforts that are expected to support CALS objectives,
- identifying a preliminary set of standards required for data interchange in support of CALS, and
- developing reports on the four broad categories of standards required to support the interchange of CALS digitized technical information: (1) product definition data, (2) graphics, (3) text, and (4) data management.

As a result of these efforts, NBS made a preliminary identification of several high-priority standards implementations

needed for CALS data interchange and access.[1] Building on knowledge and experience gained during FY86, NBS focused on the following activities in FY87: developing a CALS Framework, Development Plan and Core Requirements Package; providing technical support for standards development and implementation; and conducting workshops and meetings to promote dialogue with the Services, the Defense Logistics Agency, and industry.

A major FY87 thrust was the completion of initial documentation of the high-priority standards required in the CALS environment. Some of these standards (e.g., SGML, IGES) required tailoring or enhancement. Other standards required a "push" (e.g., CGEM) for their development in a timely fashion. These four volumes are a collection of the final reports presented to the CALS Policy Office.[2] The collection is divided as follows:

VOLUME 1:

Text

Evaluation of Text Interchange Methods

Plan for Conformance Testing for DoD Implementation of SGML

Guidelines for the Development of Tags for SGML

The NBS FIPS - SGML Validation Suite

The NBS FIPS - SGML Reference Parser

Using SGML - Application Guidelines

ODA/ODIF Implementation Agreement a Document Application Profile

Data Management

CALS Report on Data Management Standards

Supporting Logistic Support Analysis (LSA) Using the Information Resource Dictionary System (IRDS)

Media

ICST Recommendations on Optical Disks and Interface Requirements for Planned EDMICS Procurement, Final Report

---

1 Kemmerer, S., Editor, "Final NBS Report for CALS, FY86," U.S. Department of Commerce, National Bureau of Standards, NBSIR 87-3566, May 1987.

2 The publishing of this collection of reports does not imply the CALS Policy Office has endorsed the conclusions and recommendations presented.

Raster Compression
Report on Raster Graphics

Tiled Raster Interchange Format, TRIF Version 1.0, Rev. 1.2

Conformance Testing
NBS Plan for Validation (Conformance Testing) of Computer Products in Support of the CALS Program

VOLUME 2:
Graphics
Raster-to-Vector Conversion: A State-of-the-Art Assessment

Development of CGM Validation Routines

CALS Application Profile for CGM

CALS Requirements Reflected in the Extended CGM (CGEM) Standards Effort

A Reference Implementation for CGM, Functional Requirements and Conceptual Design

IGES to CGM Translator Design Specification

VOLUME 3:
Graphics
CGM Registration For CALS Requirements

VOLUME 4:
Product Data
Guidelines for Testing IGES Translators

Guidelines for IGES Application Subsets

The following are additional deliverables completed by NBS during FY87 but under separate cover. They are available through the CALS Policy Office.

CALS Core Requirements, Phase I.0

CALS Framework

CALS Program Integration of Logistic Support Analysis and Reliability and Maintainability Data Deliverables

CALS Current State of Digital Technology (Phase I.0)

CALS Workshop Proceedings:
Graphics Data Interface for Engineering Design and Technical Publication Systems (January 13/14)
Introduction to the Core Requirements Package (April 23)

MILSTD-1840A, Automated Interchange of Technical Information

MILSPEC-D-28000, Digital Representation for Communication of Product Data: Application Subsets

MILSPEC-M-28001, Manuals, Technical: Markup Requirements and Generic Style Specification for Electronic Printed Output and Exchange

CONTRIBUTIONS

NBS would like to acknowledge the major technical contributors to this volume. In alphabetical order they are:

Daniel Benigni

David Jefferson

Sharon Kemmerer

Mark Skall

GRAPHICS

RASTER-TO-VECTOR CONVERSION:
A STATE-OF-THE-ART ASSESSMENT

DEVELOPMENT OF CGM VALIDATION ROUTINES

CALS APPLICATION PROFILE FOR CGM

CALS REQUIREMENTS REFLECTED IN THE EXTENDED
CGM (CGEM) STANDARDS EFFORT

A REFERENCE IMPLEMENTATION FOR CGM,
FUNCTIONAL REQUIREMENTS AND CONCEPTUAL DESIGN

IGES TO CGM TRANSLATOR DESIGN SPECIFICATION

FINAL REPORT

CALS SOW TASK 2.2.1.2.2

RASTER-TO-VECTOR CONVERSION: A STATE-OF-THE-ART ASSESSMENT

TABLE OF CONTENTS

LIST OF FIGURES

ASSESSMENT OF RASTER-TO-VECTOR CONVERSION TECHNOLOGY

I. PURPOSE

Assess state-of-the-art raster-to-vector conversion technology and develop recommendations for CALS. (Task 2.2.1.2.2)

II. BACKGROUND

Currently, raster data is required to maintain many DOD logistic systems. There was unanimous agreement at the DOD/NBS/Industry Workshop on Automated Technical Manual Systems and Automated Data Repositories, held at NBS on June 24-25, 1986, that the need to accommodate use of raster data was unavoidable, but future directions should be toward the increased use of vector format. This would reduce storage costs, make the pictures more easily modifiable, and allow for the interface to both IGES and CGM, since all of the graphics standards, including IGES, require vector format to be utilized effectively. The workshop participants concluded that CALS principals needed to become more knowledgeable concerning state-of-the-art raster-to-vector conversion and possibly accelerate advancements in this technology. In addition, the general availability of this technology has to be assessed so that alternative strategies can be developed if raster-to-vector conversion is not available in a timely manner.

1.0 Review of CALS-Related Requirements for Standards

In FY86, the NBS contracted with System Development Corporation to review and analyze the requirements of CALS-related projects for graphics standards (This is part of the Graphics Interchange portion of the FY86 Final NBS Report for CALS). The report surveyed a selection of Army, Navy, and Air Force projects that fell into the following three broad application areas: printing and publishing systems, paperless presentation and maintenance aids, and automated engineering data repositories and product definition data. The salient results of that study are summarized in the following. Those conclusions bearing on raster-to-vector conversion are shown in bold type.

> In the technical manuals area, there is an urgent need to define a common DOD approach to SGML, including the use of CGM files to import graphics pictures. IGES is needed for the transport of product data between CAD systems, CGM for transport of pictures between publishing systems, and a raster standard--perhaps a

subset of CGM to store the print-on-demand images, which have already been laid out on the page and formatted. Strong, reliable validation procedures are required to build user confidence in the standard products available from the commercial marketplace.

There is a general need for a raster-to-vector conversion capability, but product definition data need not be carried along. The users want a family of standards, with increasing capabilities available at an increase in complexity and cost. They want text and graphics capabilities in a single standard.

In the engineering data repositories area, there is an expressed need for textual data standards and for data base standards. There are some instances where IGES is used to maintain pictures; CGM could be used instead, with a savings in storage, processing time, and complexity.

In all areas, there is a tremendous backlog of data that must be accessed, manipulated, and outputted, currently in raster format. Much of this data is archived on aperture cards that would have to be digitally scanned. Data in this format will be part of the CALS database for a very long time.

In general, there is a broad, short range need for saving images in raster format. However, in the long term, all users indicate that they'd like to convert to an all-vector format. This would permit them to modify the picture, transform it, and exchange it with otherwise incompatible systems. Whatever format is chosen--facsimile-based, CGM-based, or something entirely new, a strong validation program for file generators and interpreters is required.

Also in the Graphics Interchange portion of the FY86 Final NBS Report for CALS, architectures for four CALS applications areas--engineering design, publishing, procurement support, and interactive delivery systems--were proposed. Use of raster, CGM vector, and CAD databases are shown. These architectures will be used in the Recommendations section of this report to explain where the raster-to-vector conversion process fits into CALS programs.

2.0 Relevant Standards

2.1 CCITT Facsimile Standards

Two CCITT facsimile standards, known informally as Group 3 and Group 4 facsimile, cover the encoding and transmission of raster

data. Both formats deal with black-and-white images only; gray-scale and full color images are not presently covered by these standards.

Group 3 facsimile provides for two options: (a) a one-dimensional Huffman coding compression method and (b) a two-dimensional compression method based on differences between scan lines. Group 4 facsimile specifies the exact same coding method as Group 3 option b, but the protocol is designed for packet switched networks, so the Group 4 protocol assumes error-free transmission.

The encodings support only a restricted number of resolutions. For Group 3 facsimile, horizontal resolution is stated as 1728 dots over 215 mm; vertical resolution is either 3.85 lines per mm or 7.70 lines per mm. For standard A4 paper, this is equivalent to about 200 dots per inch in both X and Y for the fine resolution mode. Group 4 facsimile supports higher resolutions of 240, 300, and 400 dpi square, in addition to the 200 dpi square mode of Group 3 facsimile.

2.2 The Computer Graphics Metafile

The CGM provides a file format suitable for the storage and retrieval of picture description information. The file format consists of an ordered set of elements that can be used to describe pictures in a completely device-independent way. One or more pictures can be stored in a single metafile, and the metafile is defined in such a way that, in addition to sequential access to the whole metafile, random access to individual pictures is well defined. That is, the pictures are completely independent, one from another: their appearance does not depend upon the order in which they are accessed or displayed.

In addition to a functional specification, the CGM standard documents three standard encodings of the metafile semantics. The Character encoding requires minimum metafile size and is suitable for transmission across networks of heterogenous systems but is expensive to encode and decode. The Binary encoding requires minimum effort to generate and interpret but is not well-suited for exchange between computers of different arithmetic data types. It is nearly as efficiently coded as the Character encoding. The Clear-text encoding provides maximum readability and editability for ease of use by humans (e.g., for debugging purposes) but, generally, pays a heavy penalty in size and performance. The size is much larger because English and other natural languages contain a lot of redundancy. The performance is worse because parsing and recognizing text strings and converting text strings to internal numbers for use by a graphics subsystem is expensive in its use of CPU cycles.

In Appendix 1 of the Graphics Interchange portion of the FY86

Final NBS Report for CALS, the standardized CGM elements are listed by type. The ESCAPE and APPLICATION DATA elements have been provided to support uses of the CGM in ways that go beyond the exchange of pictures. Nongraphical data and graphical elements not yet standardized can be incorporated into metafiles in a regular way. When these extended metafiles are exchanged by cooperating processes, standard commercial products can be used to handle the standard metafile elements, and new code needs be written only for the special, non-standardized elements. Large groups of users of extended metafiles can get together and agree upon a set of extensions--just like MAP and TOP users have agreed upon guidelines to the implementation of the OSI standards. For example, the elements of a business chart--like legend entries, tick marks, and axis labels--or the elements of a project schedule--like PERT chart symbols, milestone markers, or title--could be marked in the metafile. An editing program could be written to read such metafiles and allow modifications to them before rendering the chart on a hardcopy device or including it in a report or manual.

In the absence of any facsimile standard capable of handling multicolor images (i.e., those with more than one bit per pixel), a CGM employing only the CELL ARRAY primitive could be used. Images expressed with either indexed or direct color specifications can be represented. In the Character-Coded and Binary encodings, run-length encoding may be used to reduce the size of the resulting CGM files.

The CGM was approved as ANSI/X3.122 in 1986. It became FIPS 128 in March of 1987.

2.3 The Initial Graphics Exchange Specification

The Initial Graphics Exchange Specification (IGES) is a mature standard, first published in 1981, for the digital exchange of database information among present-day CAD systems. Now in its third version, engineering drawings, 3D wireframe and surfaced part models, printed wiring product descriptions, finite element mesh descriptions, and process instrumentation diagrams are application usages addressed by IGES.

IGES information, including drawings and 3D wireframe product models, is intended for human interpretation at the receiving site. However, IGES is often used to attempt interchange between CAD databases and to feed external geometric data into a CAD system, where the data are expected to be processed automatically by computer as well as being worked on by human operators. Consequently, when used for this kind of interchange--a purpose it was not originally designed for, IGES files are often restricted in the kinds of entities used.

III. DISCUSSION

1.0 Methodology

This study was performed in phases, which are described in detail in the following paragraphs.

Phase 1: Examine the Literature. Recent computer graphics literature was examined to find articles describing raster-to-vector conversion methods, techniques, and uses. The articles located are listed in part 1.0 of the Reference section of this report.

Phase 2: Locate and Contact Vendors. From a variety of sources, sixteen companies were identified and contacted. These companies are listed in part 2.0 of the Reference section of this report. Seven companies (ANA Tech, Audre, Autodesk, Optigraphics, Scan-Graphics, Skantek, and SysScan) were sent a special letter requesting their cooperation and assistance. In particular, they were asked to send technical information documenting their proprietary vector output files and to describe what elements of IGES are being used when IGES output is selected. Four of these companies (ANA Tech, Autodesk, Optigraphics, and Scan-Graphics) supplied the requested technical documentation.

Phase 3: Learn the Process. All the vendor literature, technical documentation, and articles were read. A complete picture of the overall raster-to-vector conversion process was worked out. These results are documented in section 2 below.

Phase 4: Analyze the Process. Each stage in the raster-to-vector conversion process was analyzed with respect to some or all of the following characteristics:

- Speed
- Accuracy and Consistency
- System Cost
- Conversion Cost and Time
- Data Interchange Requirements and Opportunities.

Section 3 below contains this analysis.

Phase 5: Analyze the Vector and CAD Entity File. For the four vendors who supplied technical documentation, the global structure and the specific entities used to represent the scanned-in picture or drawing were determined. Then these entities were compared with the representational power of CGM. Finally, those IGES entities which are actually used to transfer

the picture to a CAD database using IGES were examined. Section 4 below reports these results.

Phase 6: Draw Conclusions and Make Recommendations. Based on the analyses of sections 3 and 4, conclusions about the current usability of off-the-shelf, commercial raster-to-vector systems to meet some or all of the CALS requirements are drawn. These can be found in the Recommendations and Impacts and Conclusion sections of this report.

2.0 Review of the Current State-of-the-Art

Figure 1 represents a synthesis of all the processes conventionally included in the phrase "raster-to-vector conversion." Not all vendors provide products that accomplish all these processes; in fact, **most** vendors do not! However, all the stages must be progressed through when converting a representation of a drawing or image of a geometric object to a form whereby it can be incorporated into a design and subjected to engineering analysis and modification. Less ambitious objectives permit some of these stages to be skipped.

The rest of this chapter defines and discusses the current state-of-the-art of each step in the process. We identify the inputs and outputs of each step and explore the possible uses of each output in its native form (as directly produced by the process) and in some modified form (as transformed by some auxiliary conversion process).

[**Note:** These descriptions borrow liberally from the various vendor descriptions; however, the total description represents a synthesis of all the products and does not necessarily represent the full capabilities available from any one vendor.]

Step 1: Scanning and Storing the Document

The first step is the physical scanning process in which the paper sketch or drawing, aperture card, or photograph is raster-scanned. Millions of discrete samples are taken at closely spaced intervals across the entire surface of the document. Each sample is converted to a digital value that indicates the tone (black-and-white, gray-scale, or color value) of the drawing at a given point. This process results in producing a so-called **raster database.**

Such a database simply represents an image by storing, for each horizontal and vertical picture element (pixel), the code for the color representing that pixel. Even when compressed, these raster files vary greatly in size, but they typically are very large, easily 10 to 20 times the size of geometrically defined CAD files for the same picture. When uncompressed, they can be another order of magnitude larger.

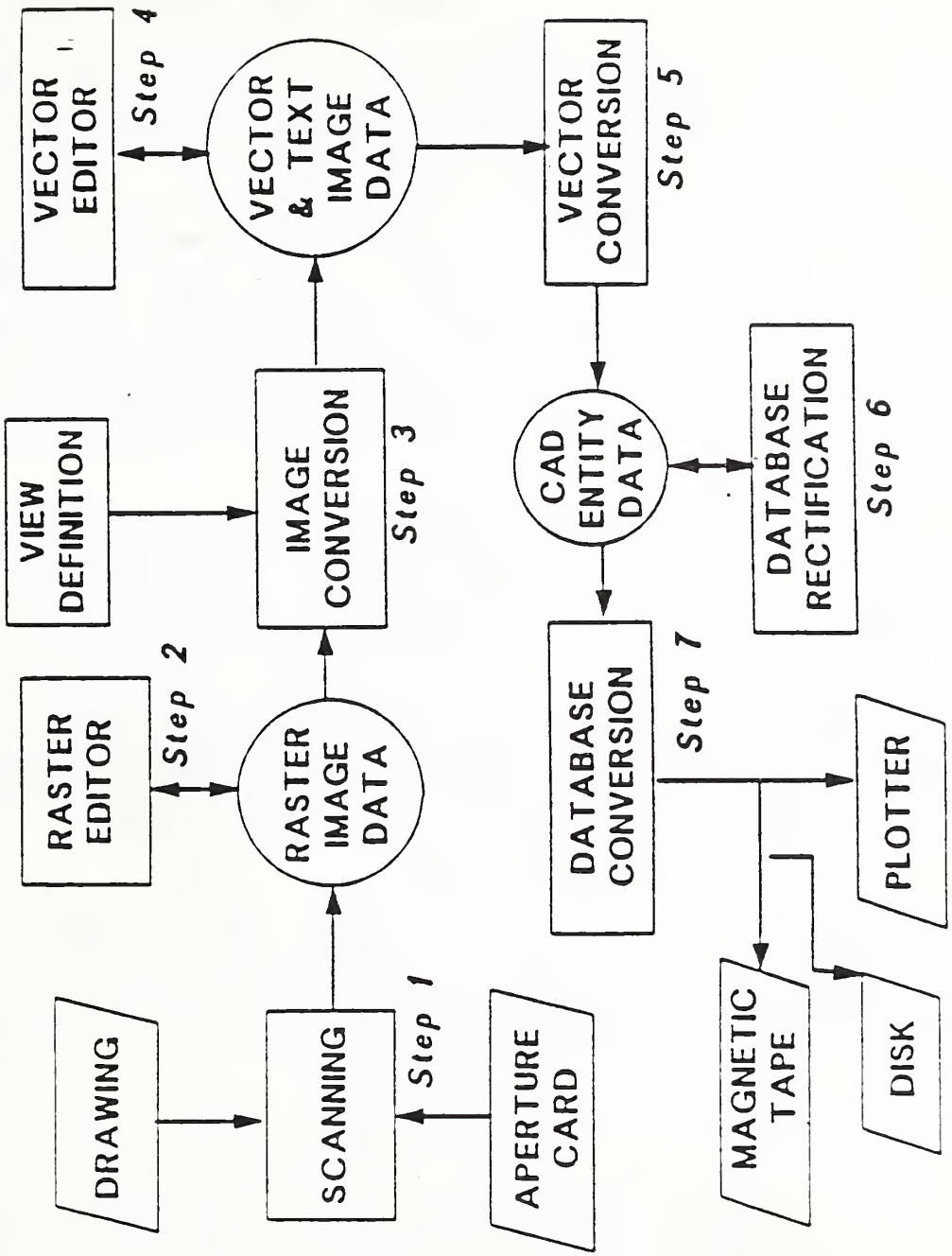


Figure 1. The Raster-to-Vector Conversion Process.

For example, a simple uncompressed black-and-white (1 bit per pixel), PC resolution (640 by 480) picture occupies at least 640x480x1 bits or 38,400 bytes not counting any control and structural information required by the file format. A more typical black-and-white E-size drawing scanned at 200 dots per inch will occupy (36x200)x(48x200)x1 bits or 8,640,000 bytes, 225 times larger than the simple PC image.

When color or gray-scale images are involved, the numbers grow even more impressively. A 16-level (4 bits per pixel) gray scale or color picture of that E-size drawing would occupy over 34 million bytes of storage, while a more typical A-size photograph at the same resolution and color depth would still occupy 7.48 million bytes. For publishing purposes, one must often store 24 bits of color or more. This blows up our A-size color photographic image to more than 22 million bytes.

Raster file compression techniques, such as those specified by the CCITT Group 3 and Group 4 Facsimile standards, can be applied to reduce the amount of space required, but they are costly in computing time and, depending upon the nature of the image, may not achieve more than a 5-15% reduction in size.

These compression techniques were designed originally for and are most effective on documents that evidence a lot of visual uniformity, such as is found in office memos and reports with standard typewriter fonts and in simple engineering drawings. Long runs of all white or all black areas are replaced by counts of the number of all white or all black areas. Compression ratios of 10:1 to 20:1 are common.

However, many maps, satellite images, and photographs do not contain such visual coherence. Furthermore, these standards do not address the coding and compression of gray-scale and full color images. Consequently, compression of less than 20% is often seen for these more complex and life-like images.

There is practically no intelligence in a raster database. It neither represents nor differentiates between line, arcs, and text. This raster type of data can be stored, edited, retrieved, and distributed, but it cannot be sent directly to a CAD/CAM system. A raster storage system is seen either as a replacement for a microfilming system (with appropriate data management and distribution capabilities), a picture reproduction system, or as the first step in the drawing conversion process.

If the image to be captured and vectorized is already in raster format--such as would be delivered by video capture boards or satellite imaging systems, this first step can be by-passed.

Step 2: Editing the Raster Image

Before attempting to apply pattern recognition techniques to find the geometric entities, it is often productive to use a raster editor to modify the raster data by adding, deleting, or altering pixel information. Non-zero pixel values caused by specks of dirt, smudges, and creases in the paper may be removed. Lines of uneven width may be "airbrushed" to a more uniform consistency. Broken lines caused by a faulty pen or pencil in the drawing may be joined.

A raster image that is edited is typically held in working image memory, thus permitting rapid access to the data. With mouse control, the system operator can continuously pan and zoom over the entire image to select an area of interest. For large drawings and images, this requires megabytes of memory.

Step 3: Recognizing the Geometric Entities

This principal step in the raster-to-vector conversion process, sometime called **image conversion** converts raster data to CAD compatible vector data, that is a **geometry definition file** or vector file. In the more sophisticated systems, text information is extracted from the raster data and converted to ASCII character strings by automatic and user interactive processes. Graphic information is extracted from the raster data and automatically converted to geometric entities like lines, polylines, circles, arcs, arrowheads, and solid areas.

Among competing raster-to-vector systems there is great diversity in the nature of the contents of the vector file. The less capable systems produce a file of short line segments combined to represent text, polylines, arcs, circles, etc. Depending on the quality of the vectorization software, real CAD lines may be made up of 10 or 15 short vectors and arcs and circles may be made up of hundreds of line segments. Like raster data, short vector data can be stored, retrieved, edited, and distributed. Additionally, this type of data may be used as input into a CAD/CAM system.

The problem with this type of data is its limited usefulness for most CAD applications. For engineering drawings, the file size may be as large or larger than the corresponding raster image file. Consider the following example.

A black-and-white "D" size drawing scanned at 200 dots per inch requires 4.32 Mbytes in pixel storage. Compressed at a conservatively estimated 10:1 ratio, this drawing would require about 432 Kbytes in its raster form. A sample view in this "D" size drawing could require 50,000 short vectors, especially if there is a lot of annotation or a lot of curved objects in the drawing. Assuming that each short vector occupies 9 bytes (1

byte for the line opcode plus 4 x 2 bytes for each 16-bit coordinate), this same drawing in vector form could occupy 450 Kbytes. Furthermore, many CAD systems--especially the PC-based ones--cannot accept even this much data in a single drawing file.

Because the data are not differentiated as CAD entities (e.g., lines, arcs, text, splines, and dimensions), editing it on a CAD system is nearly impossible. For example, to change the radius of what appears to be a circle on a CAD system, the operator must redefine the group of short vectors as a circle. That is, all the line segments comprising the circle would have to be erased and a new circular element, specified by the CAD system, would have to be defined, before the element could be manipulated directly as a circle. An even more painful example is text. Correcting a simple typographic error or updating a date is nearly impossible if the text characters are known to the system only as short vectors.

Despite their obvious problems, short vector databases have a place in CAD. They can be used to produce backgrounds that won't change and therefore don't have to be edited. A prime example of this application is providing street, sewer, and railroad background for utility mapping. In these cases, vectors do not have to be geometric entities at all. The advantage of short vectors over raster images in this application is that the image is represented in a device-independent form: the picture can be scaled and rotated at will to fit the needs of the application. This has especial value in publishing and procurement applications.

In most of those systems that provide more that just short vectors as a result of the image conversion process, the extracted geometric, textual, and symbolic elements are placed into separate layers. These layers help the operator during the next step--editing the vector file. The layers help sort the mass of information generated during this stage into manageable and more uniform collections of objects.

In some systems, in addition to the geometry definition file, the conversion process will also build a condition or constraint file, as the expert system within the software automatically assigns changeable constraints to the geometry (such as tangents, symmetry, and collinearity).

Step 4: Editing the Vector File

The vector editor allows the system user to examine the quality of data being produced by the image conversion process and make modifications. The editor typically includes provisions for adding, modifying, or deleting geometric primitives in the vector image. The operator may also reorganize the layered data and standardize any of the following information:

- the width of various sets of lines
- the line type--e.g., dashed or dotted--of various lines
- the line join and end characteristics
- the heights of characters
- the fonts of character strings
- the grouping of similar items into symbols of a uniform size and orientation
- the solid or hatched patterns used to fill areas.

Step 5: Converting to a CAD Entity File

In the more expensive and sophisticated systems, steps 3 and 4 may be combined with step 5. That is, the output of the image conversion process is a set of CAD elements that can be accepted directly into a CAD database. However, more frequently, there is a human-assisted, semi-automatic process required to convert the standardized vector file into higher level CAD entities.

Often referred to as "geometric elements," the highest level of database intelligence is referred to as CAD entities. To create this intelligence level, the drawing conversion system must pass to a CAD system neither pixels nor short vectors, but rather the same entities that are used and stored in the target CAD system database--lines, arcs, splines, polygons, ASCII text, notes, dimensions, labels, and the like.

Even the few drawing conversion systems that can produce this type of data automatically vary vastly in the level of usefulness and intelligence within the database. To evaluate the level of database intelligence, it is helpful to ask a series of questions:

- Is there connectivity between the different elements of geometry? Is a given line, for example, connected to the given arc it appears to be connected to?
- Can a geometric entity like a template be moved simply by touching a point and executing the move command? Will the simple entity (the given line) that is touched move or will the entire entity consisting of its sub-entities move?
- When data is passed to the CAD system, are physical conditions such as crucial tangencies, true perpendicularity, true parallelism and collinearity maintained? All the above elements of intellligence are assumed by the appearance of

the drawing until the user tries to use the data he obtains from the drawing conversion system and finds it imprecisely or incompletely defined.

Step 6: Checking the File for Consistency and Accuracy

To be truly useful, a CAD database must not only be intelligent in its maintenance of geometry, connectivity, ancestoral relationships, and physical relationships, but also the database must be accurate and checked for consistency.

When a database is said to be **accurate**, it means that the geometry sent to the CAD system is absolutely accurate to the corresponding dimensions present in the drawing at the specified scale of the drawing. This means that, if the dimension of a line is 9.876 mm, that value is the mathematical norm (i.e., length) of the vector whose coordinates are passed to the CAD system as the geometric representation of that line.

A few raster-to-vector conversion systems can pass accurate geometry that is driven by dimensions and is stored with an accuracy of up to 14 significant figures. Drawing conversion systems based primarily on hardware actually may introduce inaccuracy into the database. This is true if one considers that generally the starting point for these systems (the drawing that was prepared with a pencil) is generally not accurate even to the width of a line (about one-hundredth of an inch). In addition, the accuracy provided by the scanner alone is, at best, equal to its resolution (e.g., about 8 pixels per mm). Furthermore, unless the original mylar (if it exists) is used for the scan, the drawing will not normally be even as accurate as it was drawn.

The only way to guarantee true database accuracy is to attack the problem from a software standpoint. This means taking the scanned data with its dimensions and reproducing a new set of geometry based on the exact dimensions, definitions and physical properties of the geometry. Consequently, to get an accurate database, the system--perhaps with varying amounts of operator assistance--must build a numeric dimension and tolerance file prior to image conversion. To do this, the system must recognize, convert, and store the pixel dimension and associated tolerances.

To build up a 3-D CAD entity file from a drawing consisting of 2-D orthographic or perspective projections, the system needs to be more intelligent than is required for processing 2-D drawings. Furthermore, the operator must participate and specify view information by indicating on the drawing view windows, names of views, the view scale, and a point that is coincident in all views. During the conversion to CAD entities, a "cross-view"

linking process builds a 2 1/2-D model based on this information.

An accurate file is not necessarily checked for consistency; that is, the file is not necessarily:

- free of conflicts in dimensions (e.g., an overall dimensionadds up to several subdimensions) even when considering multi-view drawings;

- free of ambiguities, such as geometry with no dimensions or geometry that has the potential to be confusing to a designer or to a CAD system with regard to tangency points, parallelism, perpendicularity, collinearity, etc.; and

- free of errors in basic design logic.

Step 7: Interchanging the File with Other Systems

The output of step 5 (and step 6, when performed) is an entity file in each vendor's proprietary format. An entity file is not a CAD database. In order to be accepted by any CAD system, it is necessary to translate the vendor-specific CAD entity file into a format that is recognized by the target CAD system. Two approaches are used: one based on standards and one not.

The standards-based approach converts the CAD entity geometry file to an IGES file. The data then can be acquired by any CAD system that can read IGES files. The non-standards-based approach simply provides translators to the more widespread CAD database formats: Autotrol, AutoCAD, CADAM, Computervision, and Intergraph are among the most popular.

The advantages of the first approach are obvious: only one translator is needed. The principal disadvantage is that the various CAD systems don't all accept all the IGES elements specified in the IGES standard. Consequently, a "lowest common denominator" for IGES is often followed. In section 4 below, exactly which IGES entities are actually produced by some of the raster-to-vector conversion systems is reported.

3.0 Analysis of the Raster-to-Vector Conversion Process

3.1 Overview

It is impossible to compare systems on the basis of vendor literature and documentation alone. Not only do each of the systems have different architectures, prices, and mixes of hardware and software, but also they are positioned to serve different niche markets. Consequently, one system may be optimized for engineering drawings, another for well logs, and still another for mapping applications. When asking the question, "Which system is best?", the only answer is It all depends !:

- It all depends on what one wants to do with the picture after it is scanned. Is the drawing only to be included in a technical manual or is the conversion process simply the first step in a redesign of the part shown in the drawing?

- It all depends on how much throughput one needs. How many drawings must be converted over what period of time? Is conversion needed for one project only, for a limited time? Or is this going to be a continuing process?

- It all depends on how much one is willing to pay!

To further complicate the analysis, the performance of each system is directly related to the nature of the drawing being scanned. The density of the lines, the number of symbols, line weights, and text fonts, the amount of text, the extent of and variation in filled areas, the amount of crossing and touching lines--all contribute to the measure of complexity.

In order to make speed comparisons across competitive systems, benchmarks on the identical drawings would have to be performed. But speed is directly related to the price one pays and is not the only consideration. Accuracy, consistency, and richness of the resulting CAD entity file may also be important for many applications.

3.2 Speed

The ANA Tech VANA system, with its hardware vectorizer, can automatically convert an E-size drawing to vectors in 5-7 minutes. Up to an additional hour is typically required to transform the dumb vector file into CAD entities. The other high-end systems also take in the 30 minutes to 2 hour range. At the low end, Autodesk's CAD/Camera will take from 5 to 7 hours on a PC/AT for a drawing of similar complexity.

The scanning process, producing only a raster database, takes from 90 seconds to about 4 minutes at 200 lines per inch. Not

unexpectedly, the more expensive systems have greater throughput or scan at greater resolution.

3.3 Accuracy and Consistency

Minimum detectable line width ranges from .002 to .005 inches. This is a function of the resolution of the scanner. However, as explained in Step 6 of section 2, the accuracy of the resulting vector file depends heavily upon the quality of the original document, unless human-assisted methods are used to correlate the vector geometry as scanned in with the intended absolute dimensions as represented by dimension lines and other annotations on the drawing.

Only by performing the functions from Step 6, can a CAD-accurate database be produced. As yet, no system can perform this step without human intervention.

3.4 System Cost

Prices for automatic scanner systems range from $100,000 to almost $300,000, depending upon speed, sophistication, editing capabilities, and hardware capacity. Where editing is supported, an Apollo, Digital microVAX, or Sun class workstation with stand-alone computing power is provided.

Scanners without workstations are priced in the $40K to $125K range, depending on speed, accuracy, intelligence, and resolution. The highest priced offering--the ANA Tech Eagle 800 for $125K--includes on-the-fly vectorizing of the drawing. The other systems create only a raster database, which is later processed by software to create the vector and CAD databases.

Software-only solutions that run on the IBM PC/AT like Autodesk's CAD/Camera and Professional CAD/CAM's ProCAD+V/R are much less expensive ($3K to $10K), but are less capable and much slower.

Service bureaus are available if the lease or purchase of dedicated in-house systems cannot be justified. They charge from $150 to $300 for each E-size drawing converted.

3.5 Conversion Cost and Time

No formal studies have been published concerning the time and cost of converting drawings. Consequently, one must depend upon anecdotal information and marketing claims; however, there is generally good agreement on the numbers.

Optigraphics claims that the payback for a scanning and conversion system can be achieved by processing about 1000 drawings (about one year's work for most engineering departments).

Audre estimates that conversion of a complex E-size drawing costs about $110 and takes four hours, compared with $1,350 and 30 hours using a standard CAD digitizing approach.

Anderson Reports estimates $225-$550 for the automated path (taking 5-9 hours) and $450-$1000 for manual digitizing (taking 16-40 hours). A reported side benefit of the automatic path is better quality: one experiment reported 30% fewer errors. Anderson also estimates that the tradeoff point between using a service bureau and buying one's own system is about 800 drawings.

For consistent and accurate drawings (the highest level of CAD database), Metagraphics claims that their system is 4 times faster than digitizing via a conventional CAD/CAM system. They also claim a 3:1 cost advantage.

3.6 Data Interchange

Most of the commercial systems studied can accept already digitized raster databases at the front-end (that is, after Step 1). CCITT Group 3 and 4 facsimile formats are generally the only standards supported at this interface. As mentioned in Section 2.1 of the Background section, these formats apply only to black-and-white (one bit per pixel) images. At present, this is not a limiting factor because none of the commercial raster-to-vector systems can handle multiple bit-per-pixel (color or gray scale) images. However, this could become a concern in the future, especially for CALS Interactive Delivery System applications.

No formal standardized format for the CAD-compatible or CAD-accurate vector databases has been adopted across the commercial systems. As described in section 6, each offeror has a proprietary vector file format and structure, but none are compatible with any of their competitors. Autodesk's CAD/Camera product does produce an AutoCAD file--a format that has become a de facto industry standard on PC-based CAD systems.

To export these databases, the suppliers offer translators. As explained in Step 6 of section 2, translators to IGES as well as to the proprietary formats of the major CAD/CAM systems are available. In section 4 below, it is pointed out that, in general, only a few IGES entities are used. Consequently, some of the semantic content of the CAD-compatible database may be lost when using IGES to transfer the data into a CAD/CAM system.

Also in section 4, it is concluded that the Computer Graphics Metafile standard could serve as a common representation of CAD entity databases, especially if enhanced with a few features as concluded in the Recommendations section of this report.

4.0 Analysis of Vector and CAD Entity File Contents

Four vendors--ANA Tech, AutoDesk, Optigraphics, and Scan-Graphics--provided sufficient technical information to permit a detailed examination of the vector and CAD entity file elements generated by their systems. All of these systems perform some automatic recognition of CAD entities. In the following sections, the results of the analysis is presented.

4.1 Entities Used

Line Entities. All four systems support solid lines; only Scan-Graphics appears not to store line width information in the entity file. None appear to have a line type attribute (e.g., dashed, chained); rather, broken lines seem to be represented by a series of short vectors.

Curved Entities. All systems can recognize circles and circular arcs. ANA Tech and Optigraphics also can detect ellipses and elliptical arcs. Optigraphics can also represent curved entities as B-splines.

Area Entities. All systems can recognize outlined or "hollow" filled areas without holes. Only Scan-Graphics cannot detect solid areas. Both ANA Tech and Optigraphics can also detect and represent filled areas with holes. Optigraphics can also detect the special case of a solid filled or hollow rectangle. Autodesk (CAD/Camera) and Optigraphics provide special support for an arrow entity.

Fill Styles. Only outlined ("hollow") and solid fills are supported by some systems. None support patterned or hatch filled regions.

Text. Only Autodesk has no automatic recognition of text entities. From the remaining three systems, support for the various text attributes is uneven. All support character height, orientation, and spacing. Only ANA Tech and Scan-Graphics support recognition of several text fonts, while only Optigraphics and Scan-Graphics support character expansion factor. Only Scan-Graphics stores text alignment information.

Color. None of the systems support the storage and representation of color or gray-scale information.

Symbols. ANA Tech supports only "points" or "dots." Autodesk supports both "points" and a limited concept of symbols or "blocks." Both Optigraphics and Scan-Graphics support a general symbol recognition and symbol representation facility. Symbol rotation and scale can be detected and represented in the Scan-Graphics system, while this feature is planned for but not yet supported on the Optigraphics system.

4.2 Structure

4.2.1 ANA Tech

The output of the image conversion process is structured into layers. As mentioned in section 2 above, layers are used to sort out the mass of information generated during raster-to-vector conversion into more manageable and more uniform collections of objects. For example, all text may be placed in one layer, all geometric elements in another, and all symbols in a third layer. For the ANA Tech system, it is unclear from the documentation what criteria are used to decide which elements are placed in which layer.

ANA Tech's vector file is called Standard Output Format (SOF). Entities can be grouped and connectivity relationships expressed.

4.2.2 AutoDesk

All vectors produced by the image conversion system are stored in an AutoDesk DXB file that is readable by AutoDesk's widely-used AutoCAD program. The vectors are placed on layer "LINES" in the vector database, and solids are placed on layer "SOLIDS." The raster scan lines comprising any text strings or symbols that were identified are placed on layer "SYMBOLS," with borders drawn around them on layer "OUTLINES." Text and symbols are not automatically processed or recognized by the system. Instead,human intervention using the AutoCAD product is required.

4.2.3 Optigraphics

Optigraphics Database Format (ODF) is a more highly structured file format than ANA Tech SOF or AutoDesk DXB. Three sections comprise the file: a header section containing global attributes, a symbol library section containing the definitions of various symbols discovered in the drawing, and the geometry entity section containing the representation of the drawing itself. Within the geometry entity section, layers and groups of primitives may be identified.

4.2.4 Scan-Graphics

The Scan-Graphics vector data format IDS, produced by its RAVE image conversion software, does not provide for any structuring of the picture data. However, an operator using their interactive vector editing software, IGMS, may organize the data into as many as 64 separate levels. The documentation also refers to an element type ASSORTED which may be used in the future to provide structure to the elementary data. Finally, there is an element USER-DEFINED SYMBOL that allows one to

include symbols that have been previously defined and recognized by the RAVE software.

4.3 Comparison with CGM

4.3.1 ANA Tech

The SOF entity types and their Computer Graphics Metafile equivalent are given below:

END-OF-FILE	CGM END METAFILE
DRAW/MOVE	CGM POLYLINE
DATA WINDOW	CGM CLIP RECTANGLE/CLIP INDICATOR
LINE WEIGHT	CGM LINE WIDTH with LINE WIDTH SPECIFICATION MODE equal to "ABSOLUTE"
TEXT with attributes	CGM TEXT with attributes TEXT FONT INDEX, CHARACTER SPACING, CHARACTER HEIGHT, and CHARACTER ORIENTATION
TEXT CONTINUATION	CGM APPEND TEXT
RESOLUTION	CGM SCALING MODE equal to "metric"
AREA with no holes	CGM POLYGON with fill "solid"
AREA with HOLE commands	CGM POLYGON SET with fill "solid"
FONT FILE NAME list	CGM FONT LIST
LAYER	CGM BEGIN PICTURE
POINT	CGM POLYMARKER
POLYGON	CGM POLYGON with fill "hollow"
CIRCLE	CGM CIRCLE and CIRCULAR ARC CENTER
ELLIPSE	CGM ELLIPSE and ELLIPTICAL ARC CENTER)

Two more elements do not map directly into CGM elements: GRAPHIC ITEM NUMBER and CHAIN/CONNECTION. These elements are used to store connectivity relationships between coordinates making up the picture. This information is included in the SOF only if needed and requested by the generator of the SOF. If the CGM were to serve as a replacement for the SOF, APPLICATION DATA

elements of the CGM would need to be specified to contain this additional information.

Other Notes:

1. Each layer can be mapped directly into a CGM picture; one or more CGM pictures comprise a CGM metafile.

2. The SOF may optionally contain font tables, that is, character height and width information about each ASCII character in each font. Inclusion of this information can be suppressed during the generation process.

3. The SOF may also contain line width tables, that is, information describing how lines of various widths are normalized to a constant width. For example, a line width table of [1,3;6,8;] indicates that lines ranging in width from 1 to 5 units should be drawn at 3 units and those of 6 units or greater width should be drawn uniformly at 8 units wide.

4. The generator of the SOF may indicate whether the character height is measured from the baseline to the cap line (the default) or from the bottom line to the cap line. In the CGM, all character heights are measured from the baseline to the cap line only.

4.3.2 AutoDesk

The vector file contents produced by CAD/camera map directly into CGM primitives as detailed in the following:

LINE	CGM POLYLINE with line type "solid" and line width "nominal"
POINT	CGM POLYMARKER with marker type "dot" and marker size "nominal"
CIRCLE	CGM CIRCLE with interior style "hollow"
ARC	CGM CIRCULAR ARC CENTER with line type "solid" and line width "nominal"
TRACE	CGM POLYGON with interior style "hollow"
SOLID	CGM POLYGON with interior style "solid"
POLYLINE with closure flag	No immediate effect on CGM

VERTEX	CGM POLYLINE
BULGE	CGM CIRCULAR ARC 3 POINT
WIDTH	Either (1) CGM LINE WIDTH if width is constant or (2) CGM POLYGON if width is not constant, as in an arrow
SEQEND	Nothing or CGM POLYLINE if closure flag is 1
SCALE FACTOR	CGM SCALING MODE "metric"
NEW LAYER	CGM BEGIN PICTURE
LINE EXTENSION	CGM POLYLINE
TRACE EXTENSION	CGM POLYGON with interior style "hollow"
BLOCK BASE	CGM POLYMARKER with marker type equal to block number
NUMBER MODE	CGM VDC TYPE.

Other Notes:

1. BLOCK BASE maps into CGM POLYMARKER only if the symbols denoted by block number are known to the receiving system as built-in markers.

2. No recognition of text primitives is performed by CAD/Camera.

4.3.3 Optigraphics

Information in the header section maps to the CGM elements SCALING MODE "metric" and VDC EXTENT. The maximum linewidth element cannot be represented directly in the CGM, but would be used during rendering to clamp the width of wide lines to some reasonable maximum.

Information contained in the Symbol Library Definition Section does not directly map to any concept contained in the current CGM specification. However, current work on extending the CGM to include the concept of segments is underway. When complete, these new CGM elements will be able to represent symbols well.

The geometric entities map directly to CGM entities with two exceptions as noted in the list below:

VECTOR STRING	CGM POLYLINE or POLYGON

RECTANGLE	CGM RECTANGLE
CIRCLE	CGM CIRCLE
CIRCULAR ARC	CGM CIRCULAR ARC CENTER [CLOSE]
ELLIPTICAL ARC	CGM ELLIPSE or ELLIPTICAL ARC [CLOSE]
B-SPLINE	No CGM counterpart; use GDP or reduce to POLYLINE's
ARROW	No CGM counterpart; use GDP or reduce to POLYLINE's or POLYGON
FILLED AREA	CGM POLYGON or POLYGON SET
TEXT STRING	CGM TEXT with attributes CHARACTER HEIGHT, CHARACTER EXPANSION FACTOR, CHARACTER SPACING, CHARACTER ORIENTATION
SYMBOL OCCURRENCE	INSERT SEGMENT under transformation or POLYMARKER only in very special circumstances.

Other Notes:

1. The rotation and scale parameters of SYMBOL OCCURRENCE are not yet supported.

2. Generalized Drawing Primitives (GDPs) for B-SPLINE and ARROW would have to be registered and supported for the CGM to perform well as a replacement for ODF. [Both of these are going through the registration process via another CALS task.]

3. All vector entities have a display type: solid (corresponding to CGM INTERIOR STYLE "solid"), hidden (corresponding to CGM interior style "empty" with no boundary visible), centerline (corresponding to "hollow" filled areas with "solid" borders), cutplane and phantom. These last two display types are not explained in the available documentation.

4. All vector entities have line width.

5. No vector entities have color.

6. Although text is recognized, specification of text font is not supported, unlike the ANA Tech and Scan-Graphics systems.

4.3.4 Scan-Graphics

Only seven element types are used by RAVE to represent the scanned image. These are listed below along with their correspondence with CGM elements.

SIMPLE VECTOR (centerline)	CGM POLYLINE
SIMPLE VECTOR (outline)	CGM POLYGON with interior style "hollow"
CIRCLE	CGM CIRCLE
CIRCULAR ARC	CGM CIRCULAR ARC CENTER
TEXT with attributes	CGM TEXT with the attributes CHARACTER HEIGHT, CHARACTER ORIENTATION, CHARACTER EXPANSION FACTOR, CHARACTER SPACING, TEXT ALIGNMENT, and TEXT FONT INDEX
FIDUCIAL	Not supported directly by CGM; would have to be represented by a registered GDP (a fiducial appears to be a text string representing a latitude and longitude)
CENTERED SYMBOL	CGM POLYMARKER with attribute MARKER SIZE and MARKER SIZE SPECIFICATION MODE equal to "absolute"
USER-DEFINED SYMBOL	Would correspond to INSERT SEGMENT under rotation and scale, when the extended CGM has been defined to include a segmentation facility.

Other Notes:

1. Color cannot be represented.

2. Several additional elements--ellipse, hyperbola, parabola, and dimension line--are planned to be used but are not yet generated by the RAVE software.

3. Line width information appears to be specified outside the IDS--at least the documentation provided did not seem to include a place for that information.

4. There was no indication in the documentation of how software reading the IDS would know whether a SIMPLE VECTOR represented a centerline or an outline.

4.4 Interface to IGES

4.4.1 ANA Tech

Only four of the SOF elements are mapped to IGES entities:

MOVE/DRAW	Maps to the IGES COPIOUS DATA entity (106) form 11.
TEXT	Maps to the IGES GENERAL NOTES entity (212).
CIRCLE	Maps to the IGES CIRCULAR ARC entity (100).
ELLIPSE	Maps to the IGES CONIC ARC entity (104) form 1.

What happens to the other graphic primitive information from POLYGON and AREA elements is unstated. Presumably the outline information is converted to lines (IGES COPIOUS DATA form 11), but what happens to the filled portion is not documented.

4.4.2 AutoDesk

CAD/camera cannot directly output IGES files. However, because the CAD/camera vector files can be read by AutoCAD, IGES files can be produced if there is an IGES file generator available from AutoDesk. The IGES entities so created are not described in the available documentation.

4.4.3 Optigraphics

Optigraphics sales literature states that vector data can be expressed in an IGES format optimized for the target CAD system. However, no documentation describing the actual IGES entities used was made available.

4.4.4 Scan-Graphics

RAVE IDS files can be converted to IGES formatted vector files according to the following mapping of IDS data types to IGES entities.

SIMPLE VECTOR (centerline)	IGES Line Entity (110) or Copious Data Entity (106 form 12)
SIMPLE VECTOR (planar outline)	IGES Plane Entity (108)
CIRCLE and ARC	IGES Circular Arc Entity (100)

TEXT IGES General Note Entity (212)

DIMENSION LINE (when supported)
IGES Leader Arrow Entity (214 form 2).

IV. RECOMMENDATIONS AND IMPACTS

1.0 Regarding the Process

As described in Section 2 of the Discussion section above, three types of databases are typically created during the process of converting a physical document (paper, aperture card, etc.) to a digital form. These types are: (1) a **compressed bitmap** file with editing capabilities limited to cosmetic cleaning up of the digitized image; (2) a **CAD-compatible** vector database, which represents the drawing in an efficient manner and which permits easy modification and redisplay at different sizes, resolutions, and orientations on a wide variety of hardcopy and softcopy devices; and (3) a **CAD-accurate** geometric database, which is dimensionally accurate and consistent, permitting engineering analysis and redesign using a CAD/CAM system.

If the goal is to clean up and make a few minor changes to an old drawing, a bitmap database will suffice. In CALS, bitmap databases could be used:

- To provide images and photographs for technical publishing applications. Maintenance manuals and user's manuals, informative newsletters, and technical specifications are just a few of the end-products of the on-line computer-based publishing industry that has emerged in the past few years.

- To support procurement; that is, drawings could be disseminated as part of bid packages.

- To support computer interactive training and maintenance delivery systems. Photographs, drawings, and even animated sequences could be displayed at a raster graphics terminal to explain how to use or fix a piece of military equipment.

Resizing and rotating raster images is expensive and introduces distortions caused by statistical resampling of the image (known as aliasing effects). Similarly, aliasing effects appear if one needs to display an image with a different resolution than the resolution at which it was scanned in and stored. Consequently, limited use can be made of these raster databases. Even if one were to standardize on, say, 200 dot per inch resolution for storing and displaying documents today, in the future, 300, 400, and even 500 dot per inch display technology will be affordable and commonplace. Indeed, already today in the technical publishing industry, 300 dpi is the laser printer standard; 400 dpi is available on some high-end large format electrostatic printers used for CAD/CAM, facilities management, and mapping applications; and 500 dpi and higher is used in the color pre-press and typesetting worlds.

Recommendation 1: The CALS program should continue to participate in a standards development activity to specify a formal raster image file format. The standard should:

- Incorporate CCITT and other compression algorithms.
- Be based on the CGM file structure and encoding techniques.
- Support multiple bit-per-pixel images.
- Not be tied to any particular set of resolutions.

If the drawing will be used in other designs, will be changed often, is needed at various sizes and orientations, and will be displayed on a variety of hardcopy and softcopy devices with different resolutions, a CAD-compatible (vector and fill area) database is most useful. In CALS, CAD-compatible databases could be used:

- In engineering design systems, to input older designs that are intended to provide a conceptual starting point for new designs or to input new conceptual designs developed on stand-alone CAD workstations (perhaps even PC-based).

- To support the technical publications process in all its phases. Pictures represented as CAD-compatible databases can be cropped, scaled, rotated, and composed to fit the needs of the documentation. They can be modified and enhanced by a graphics artist or technical editor to emphasize or illustrate certain features vital for proper maintenance or use.

- To support the procurement process in all its aspects. Drawings can be manipulated just as in the technical publications process for similar purposes. In addition, diagrams, drawings, and displays represented as standardized CAD-compatible databases can be disseminated by the Government to bidders to be used as benchmarks in the evaluation of the graphical speed, capacity, and capabilities of military components and subsystems. Conversely, a bid package may request that bidders submit standardized, CAD-compatible databases representing the display capabilities of their systems. Government employees could then use these data to evaluate the technical merits of the proposals.

- To support computer-based interactive delivery of maintenance and training assistance. Pictures stored as CAD-compatible data are much more suitable for manipulation, enhancement, and animation than are images stored in raster form. Where photographs or high-density maps are needed, a hybrid

approach with the raster data in the background and vectors, filled polygons, and text overlayed in the foreground is a very viable approach.

Within CALS, converting drawings to CAD-accurate databases is needed in three situations:

1. When a major engineering design effort is undertaken and that effort is based on a previous design. For example, an improved design of an airframe would probably want to start from the complete geometrically accurate database of the current airframe.

2. When spare parts need to be procured or reprocured. For example, in the automated manufacturing of reprocurement parts, the database could be included in the bid package for a supplier to pass directly to a numerical control machine for manufacturing.

3. When a part or subsystem needs to undergo extensive engineering analysis. For example, analysis for points of high stress or strain in a helicopter propellor.

Recommendation 2: Suppliers of Raster-to-Vector Conversion systems should be urged to supply CGM (FIPS 128) files as a standard way of exporting both the unprocessed vector information (that is, the output of Steps 3 and 4) and the processed vector information (that is, the output of Steps 5 and 6). This would permit vector representations of drawings to be immediately usable in a wide range of publishing and drawing systems that plan on using ANSI/X3.122 (FIPS 128), CGM, as their standard for importing pictures and diagrams. Publishing systems built around ODA/ODIF already specify the use of CGM.

2.0 Regarding IGES

Exporting CAD-compatible information using IGES is an important feature of present-day systems. However, the weak conformance rules associated with the IGES standard (X3.110-1981) reduce its effectiveness, and therefore, only a few entities from IGES seem to be used by any of the raster-to-vector conversion systems.

Recommendation 3: Using the set of conformance guidelines and application subsets for IGES (DOD-D-IGES) ,work with industry suppliers to raise the level of intelligence that can be understood by all IGES interpreters. Then, develop a set of specifications for raster-to-vector conversion systems conformance. The specifications should indicate how different components of a drawing should map to IGES entities.

3.0 Regarding CGM

Recommendation 2 suggests that CGM be used as the primary vehicle for exporting digital representations of drawings obtained from raster-to-vector conversion systems. Although CGM can be used in its present form as documented in section 4 of the Discussion section above, enhancements to CGM could be made to make the use of CGM more efficient and effective. Several ESCAPE's and GDP's should be specified and registered with the ISO Registration Authority.

These are described in the following:

An ARROW line attribute ESCAPE

This modal attribute would apply to all subsequent polylines and circular and elliptical arcs and would allow four enumerated values: (0) no arrows; (1) arrow at end; (2) arrow at beginning; and (3) arrows at both ends.

An ARROW SHAPE attribute ESCAPE

This modal attribute would control the appearance of any arrows drawn as a result of the ARROW attribute. Control over the length of arrowhead--from tip to base, its width at the base, and whether the arrowhead is filled or outlined should be provided as aspects of this attribute.

A SPLINE GDP element

The parameterization should probably be based on non-rational B-splines. However, experts from the CAD/CAM field should be consulted prior to this GDP's definition.

An ESCAPE control element

Specified as a two-state flag, this element when encountered in a metafile would disable or enable clearing of the viewing surface at the beginning of the next picture and all subsequent pictures. This functionality has been proposed in the TOP 3.0 CGM application profile and is needed to implement layers.

Two of the above items, namely the ARROW line attribute escape and the SPLINE GDP element, have already been identified in Task 2.2.2.2.2 of the CALS SOW, and have been prepared for the registration process. In addition, NBS will be investigating whether there is a future need for representing fiducials, hyperbolas, parabolas, and dimension lines directly in the CGM.

Recommendation 4: As outlined above, CALS should prepare several proposals for Graphical Item registration to support raster-to-vector conversion technology.

The more sophisticated raster-to-vector systems support the creation of a Symbol Library for a drawing by recognizing instances of common geometric forms like diamonds, resistors, and arrowheads. In the current CGM, each instance of such a symbol would have to be described with its full geometry. No references to global definitions of symbols are possible. However, the extended CGM development work taking place within ISO and ASC X3H3 is considering adding a segmentation facility to CGM. The rules regarding where segments can be defined and the scope rules relating to referring to segments have not yet been decided.

Recommendation 5: Continue work through the Standards committees to advocate that the extended CGM permit segments to be defined outside a picture; i.e., in the metafile descriptor. Such segment definitions should be able to be referenced from within any of the pictures comprising the metafile. A facility comparable to GKS's INSERT SEGMENT under transformation (to permit scaling, rotation, and translation of the symbol) should be supported in the CGM.

This recommendation is already being worked on in support of Task 2.2.1.2.1 of the CALS SOW.

V. SUMMARY AND CONCLUSIONS

Three standards are relevant to the raster-to-vector conversion technology. Photographs and drawings may be scanned in and stored in one of the CCITT Group 3 or 4 facsimile formats. Similarly, previously generated raster images may be accepted by a raster-to-vector conversion system in one of the CCITT formats.

Unstructured vector data may be represented using the formats provided by ANSI/X3.122, the Computer Graphics Metafile (CGM). The CGM provides a standard syntax and semantics for storing and transmitting a broad range of color, gray-scale, and black-and-white pictures, represented as both vector drawings and raster images. Three encodings of CGM have been standardized. Each encoding meets different design objectives--compactness, speed of processing, and ease of understanding.

Structured and edited geometry data can be formatted for entry into CAD/CAM systems for subsequent modification and analysis using ANS/Y14.26M, the Initial Graphics Exchange Specification (IGES).

The specific recommendations made above concern CALS sponsorship of changes and improvements in these standards to support CALS requirements associated with scanning, storing, editing, modifying, and exporting drawings in digital form by automated, computer-based raster-to-vector conversion systems.

Generally speaking, the technology has advanced to the point that most raster-to-vector conversion problems can be solved by the application of some combination of automatic and manual methods, at an acceptable speed, with adequate accuracy. Each application requirement has its own price point; whether the price is cost-effective depends completely upon the application and the volume of work required. Where justified, the conversion process should begin now using current raster-to-vector conversion technology.

VI. REFERENCES

1.0 BIBLIOGRAPHY OF STANDARDS DOCUMENTS

CCITT Terminal Equipment and Protocols for Telematic Services (the CCITT "Red Book"), Vol. VII -- Fascicle VII.3

-- CCITT Recommendation T.4, Standardization of Group 3 Facsimile Apparatus for Document Transmission, 1986.

-- CCITT Recommendation T.5, General Aspects of Group 4 Facsimile Apparatus, 1986.

-- CCITT Recommendation T.6, Facsimile Coding Scheme and Coding Control Functions for Group 4 Facsimile Apparatus, 1986.

Computer Graphics Metafile for the Storage and Transfer of Picture Description Information (CGM), ANSI/X3.122-1986, August 27, 1986.

Initial Graphics Exchange Specification (IGES), Version 3.0, National Bureau of Standards Report NBSIR 86-3359, April 1986.

These documents are available from a variety of sources:

> The published ANSI standard CGM can be obtained from ANSI, 1430 Broadway, New York, NY 10018. The CCITT documents may also be obtained through ANSI.
>
> The IGES document can be obtained from the IGES Committee Chairman, Mr. Bradford Smith, National Bureau of Standards, Gaithersburg, MD.

2.0 RELEVANT ARTICLES

"Optigraphics--Drawing Digitizing Price/Performance Leader," F-M Automation Newsletter, September 1985.

"Scanning Imager Offers Advantages Over Other CAD Input Methods, Design Graphics World, January 1986."

"Scanning/Conversion Systems Interface to CAD/CAM for Archiving," Baker, Computer Technology Review, Summer 1986.

"Large-Format Document Conversion Enters Mainstream," Kogan and Stiefel, The S. Klein Computer Graphics Review, Fall 1986.

"Scanning an E-Size Drawing, Graphics Arts Monthly, October 1986.

"Scanned Drawing Modification and Database Creation Made Practical," A-E-C Automation Newsletter, October 1986.

"Automatic Digitizers--A Special Report," The Anderson Report, December 1986.

GLOSSARY

character set The set of displayable symbols mapped to individual character codes in a text string. A character set is independent of the font or typeface.

color In the context of this report, in addition to its ordinary meaning, the word color includes bi-level black-and-white (so called, monochrome) systems and multilevel gray-scale systems.

color table A table for use in mapping from a color index to the corresponding color.

control elements Metafile elements that specify metafile delimiters, address space, clipping boundaries, picture delimiters, and format descriptions of the metafile elements.

descriptor elements Metafile elements that describe the functional content, format, default conditions, identification, and characteristics of a metafile.

device-dependent A system or portion of a system that contains logic, algorithms, or data that are consistent with the behavior of a specific graphical device.

device-independent A system or portion of a system that contains logic, algorithms, or data that do not require nor represent knowledge about the behavior of any particular graphical device.

device coordinates The coordinates native to a device; device-dependent coordinates; physical device coordinates.

direct color A color selection scheme in which the color values are specified directly, without requiring an intermediate mapping via a color table.

escape functions Graphical functions that describe device-dependent or system-dependent elements used to construct a picture, but that are otherwise not standardized.

external functions
: Functions present in some graphics standards that communicate information not directly related to the generation of a graphical image.

metafile
: A mechanism for retaining and transporting graphical data and control information. This information contains a device-independent description of one or more pictures.

metafile generator
: The process or equipment that produces a metafile.

metafile interpreter
: The process or equipment that reads a metafile and interprets the contents to produce again the picture represented in the metafile.

normalized device coordinates (NDC)
: Coordinates specified in a device-independent coordinate system, normalized to some range (typically 0 to 1).

pixel
: The smallest element of a display surface that can be independently assigned color.

prior agreement
: A process whereby the generator of a metafile and the recipient of the metafile come to some understanding regarding the content or format of the metafile, that understanding not being recorded in the metafile itself. In a blind interchange environment, prior agreement can be used to overcome limitations of exchange standards.

segment
: A collection of graphical functions that can be manipulated as a unit. Once functions are grouped into segments, they are referred to as segment elements.

world coordinates
: Coordinates specified in a device-independent coordinate system, whose units are selected by and are meaningful to the client.

APPENDICES

APPENDIX A: VENDOR LIST

The following table lists the companies contacted and the information received.

VENDOR NAME	TECH.DOC.	SALES LIT.
ANA Tech Corp	X	X
Audre, Inc.		X
Autodesk, Inc.	X	X
Automated Scanning, Inc.		X
Eikonix		X
Formative Technologies		X
Impell Corp.		X
Metagraphics		X
Optigraphics	X	X
Professional CAD/CAM		X
QC Data		X
Scan-Graphics	X	X
Scan Group International		X
Scitex		X
Skantek		
SysScan, Inc.		X
Versatec		X
Vidar		X

Complete mailing addresses are given below. Where known, technical or marketing points of contact are provided.

ANA Tech Corporation
10499 Bradford Road
Littleton, CO 80127
303-973-6722
Contact: Mr. Jerry Kasten

Audre, Inc.
10915 Technology Place
San Diego, CA 92127
619-451-2260

Autodesk, Inc.
2320 Marinship Way
Sausalito, CA 94965
415-331-0356
Contact: Mr. Gary Wells

Automated Scanning, Inc.
8000 E. Girard, Suite 402
Denver, CO 80231
303-696-6242
Contact: Mr. Don Van Dyken

Eikonix Corporation
23 Crosby Drive
Bedford, MA 01730
617-275-5070

Formative Technologies, Inc.
Foster Plaza VII
Anderson Drive
Pittsburg, PA 15220
412-682-8000

Impell Corporation
2201 Dwight Way
Berkeley, CA 94704
415-549-9119
Contact: Mr. Jerry Goedicke

Metagraphics, Inc.
30 Commerce Way
Woburn, MA 01801
617-935-6380
Contact: Mr. James Nemecek

Optigraphics, Inc.
9339 Carroll Park Drive
San Diego, CA 92121
619-292-6060
Contact: Mr. Hiram French

Professional CAD/CAM Systems, Inc.
6709 Chokeberry Road
Baltimore, MD 21209
301-486-0644
Contact: Mr. Karl Yatovitz

QC Data Collectors, Inc.
777 Grant Street, #111
Denver, CO 80203
303-837-1444
Contact: Mr. Kenneth Turnbull

Scan-Graphics, Inc.
700 Abbott Drive
Broomall, PA 19008-4373
215-328-1040
Contact: Mr. Larry Krueger

Scan Group International
5200 S. Quebec Street, Suite 300
Englewood, CO 80111
303-771-0017
Contact: Mr. Richard Amico

Scitex America Corp.
Eight Oak Park Drive
Bedford, MA 01730
617-275-5150

Skantek Corp.
150 Bethel Road
Warren, NJ 07060
201-647-7747
Contact: Mr. Jeffrey Arnold

SysScan, Inc.
100 Jerico Quad
Jericho, NY 11753
516-937-0002
Contact: Mr. Darrell Johnson

Versatec
2710 Walsh Avenue
Santa Clara, CA 95051-0982
408-338-0243

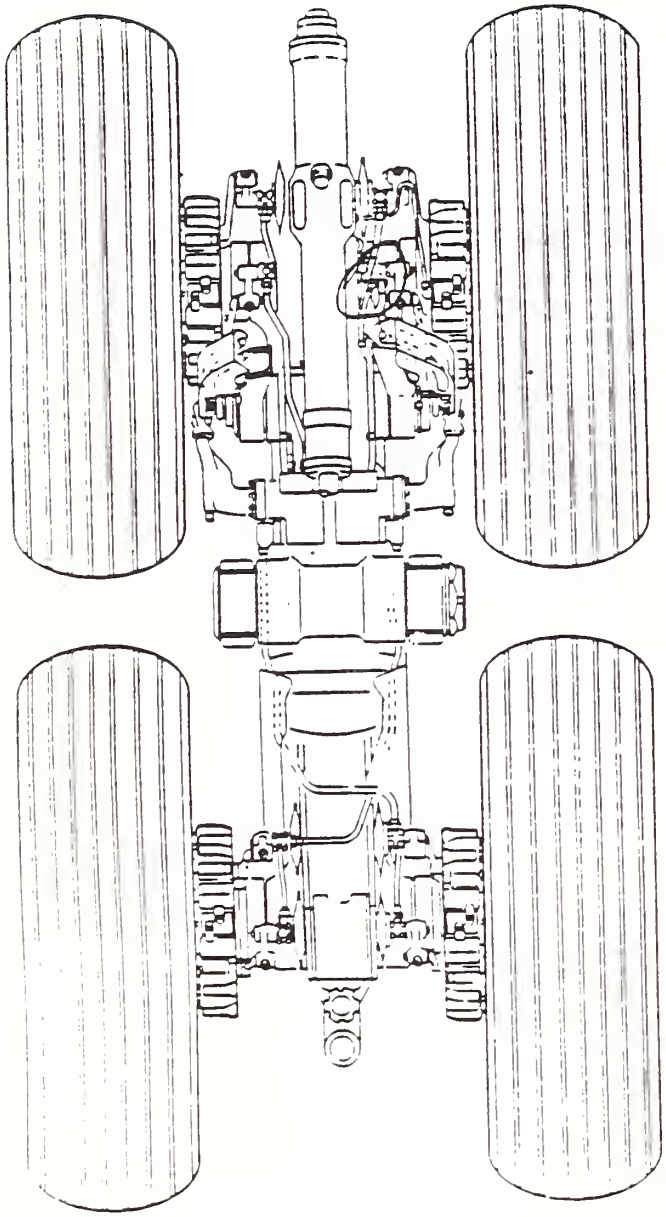
Vidar Systems
520 Herndon Parkway
Herndon, VA 22070
703-471-7070

APPENDIX B: EXAMPLES

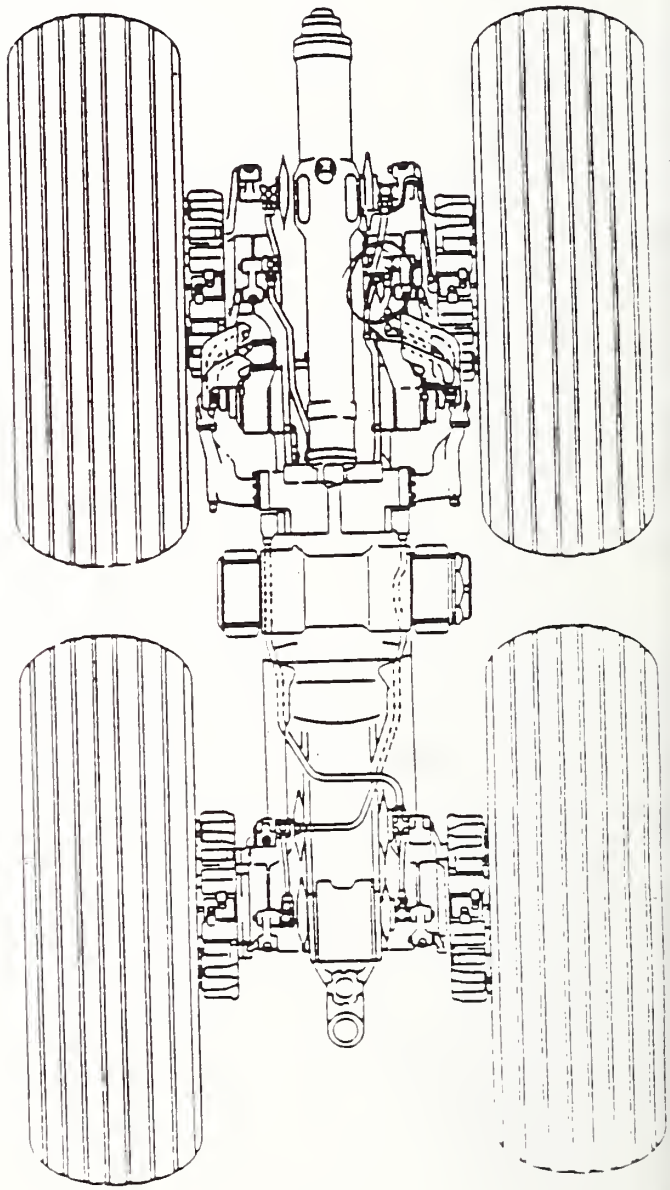
Examples of the Raster-to-Vector conversion process from ANA Tech, Automated Scanning, Scan-Graphics, and QC Data (using an Optigraphics system) are contained in this appendix.

Note especially the final five diagrams from QC Data. This series illustrates how the same drawing is represented with increasing fidelity, accuracy, and completeness as it is manipulated by several stages of processing.

ORIGINAL



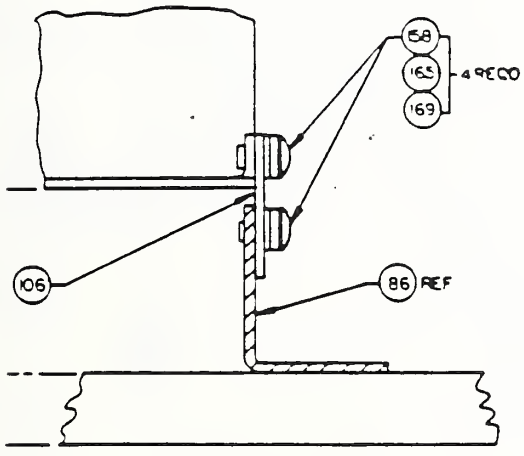
VANA OUTPUT
(Unedited)



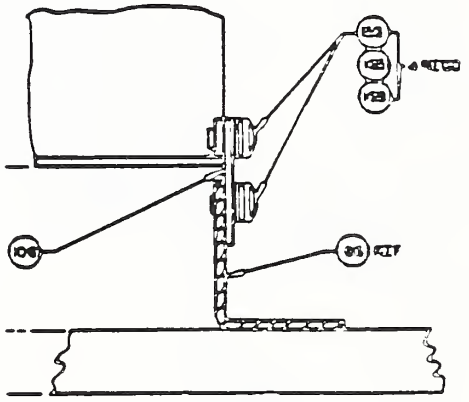
To Hand Digitize on CAD System:
4 Hours

VANA Scan/Vectorize:
7 Minutes

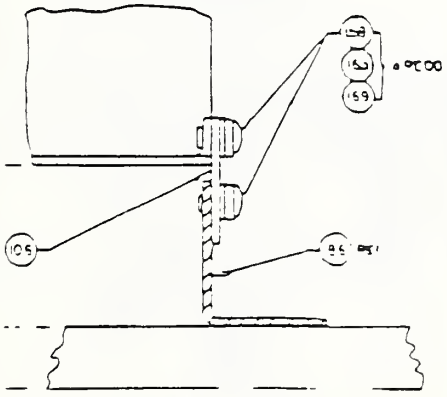
Automated Scanning Offers Several Conversion Options



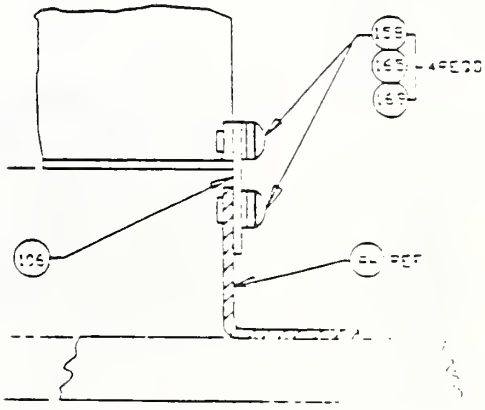
Original Drawing
This is a copy of an ink on mylar second generation black line copy submitted for scanning. Automated Scanning, Inc. can work from original vellums, CB mylars, blue lines, and even sepia prints.



Line Edge or Area Scan
Line edge or AREA reproduces solid areas very well. Using "area fill" this vector representation is an excellent Technical Publications output. In addition, it makes a superb tracing or redrawing "ghost" layer.



Center Line Scan
This is a non-edited center line scan. Notice that center line output does not reproduce solid areas. However, this is excellent output for hatching and general CAD usage. Text and solid features can be restored with simple CAD editing.



Redrawn File
Automated Scanning, Inc. can provide a full range of conversion options including complete redraws. Because we use the latest technologies our prices for redraws are some of the lowest available. We will conform to your drawing conventions or standards, develop symbol libraries and provide text as ASCII strings.

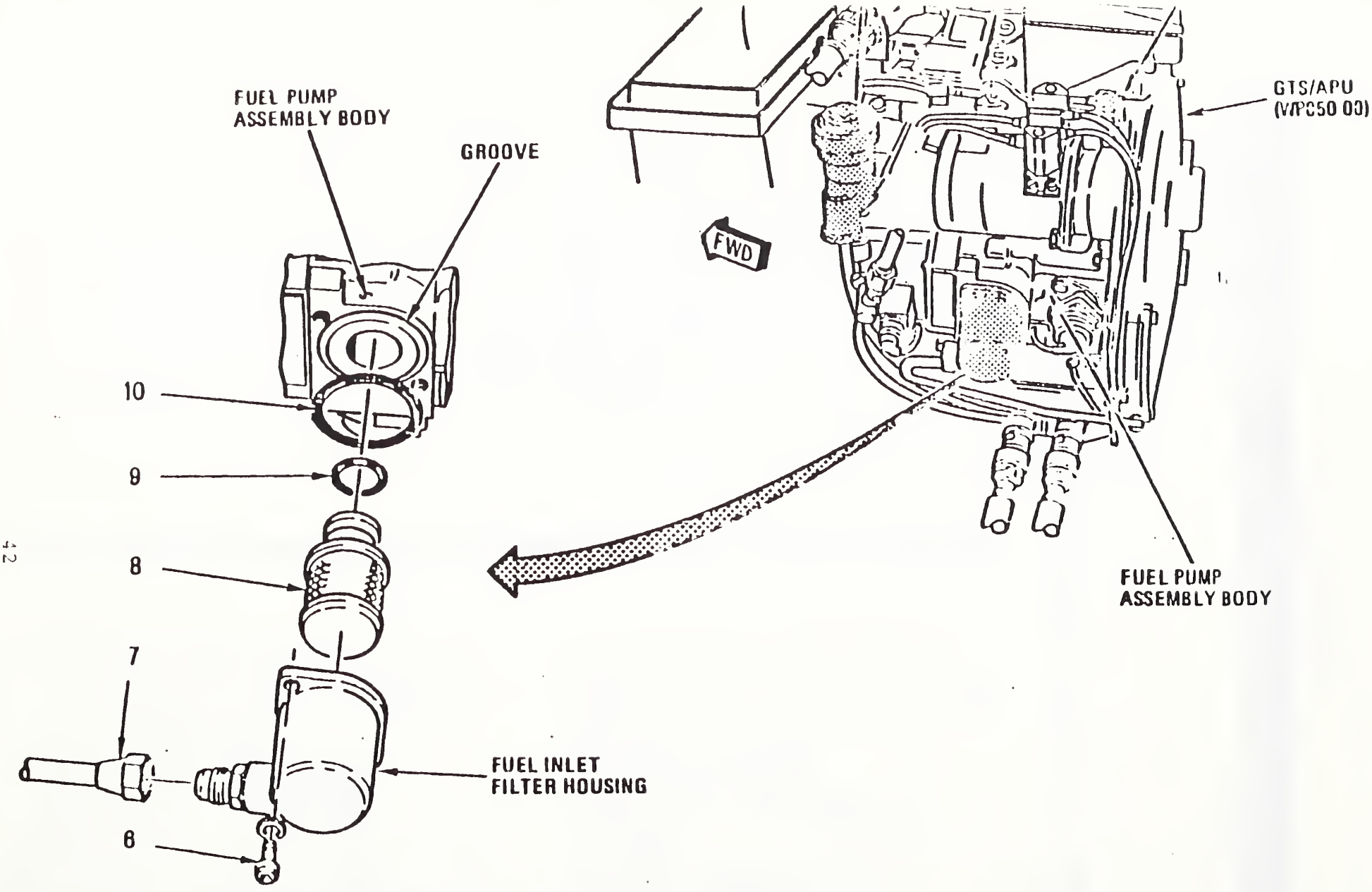


FIGURE 2. GTS/APU FUEL INLET FILTER, OIL PRESSURE FILTER, AND OIL SCAVENGE FILTER CLEANING (RASTER DATA)

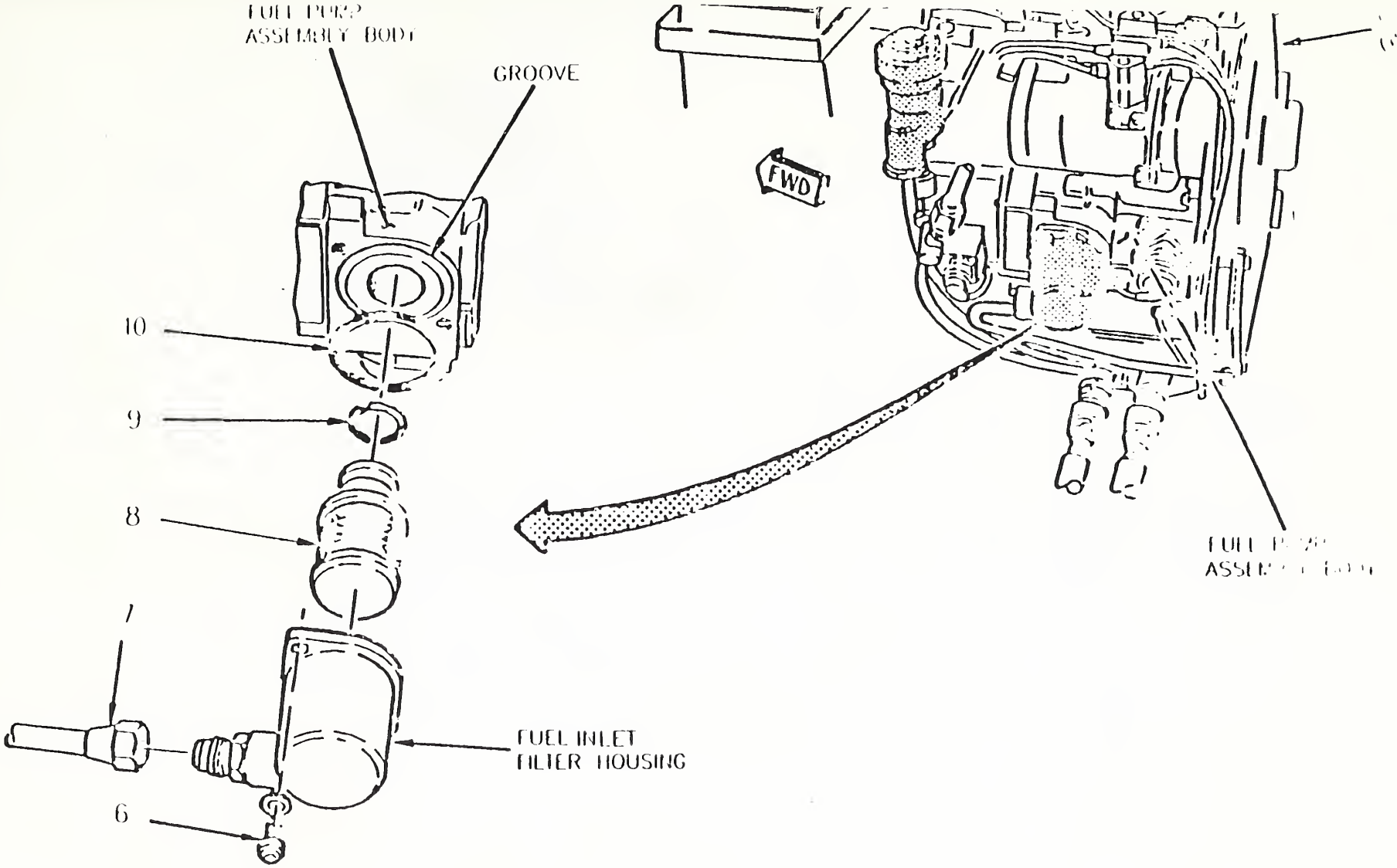
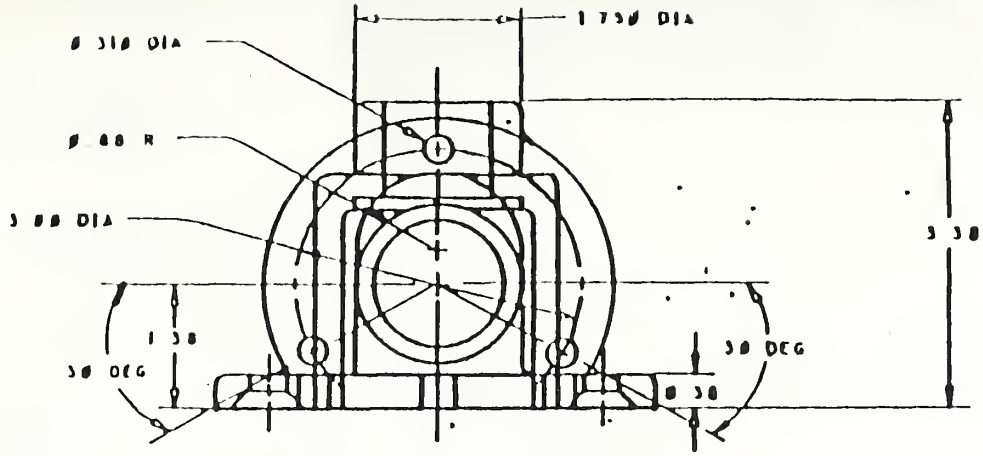


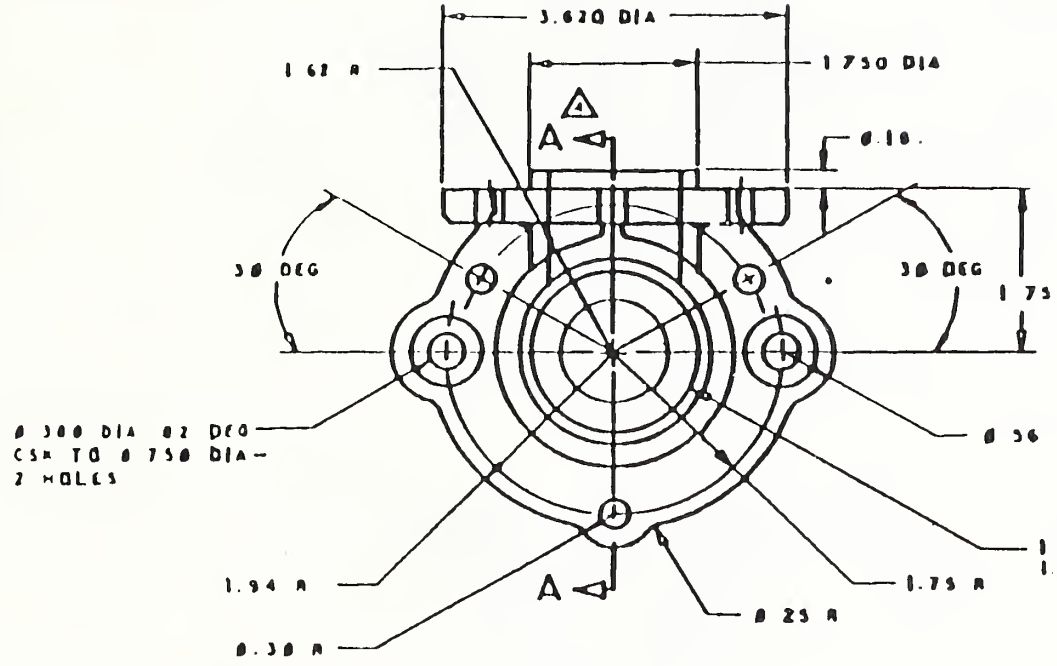
FIGURE 2. GTS/APU FUEL INLET FILTER, OIL PRESSURE FILTER, AND OIL SCAVENGE FILTER CLEANING (VECTOR DATA)

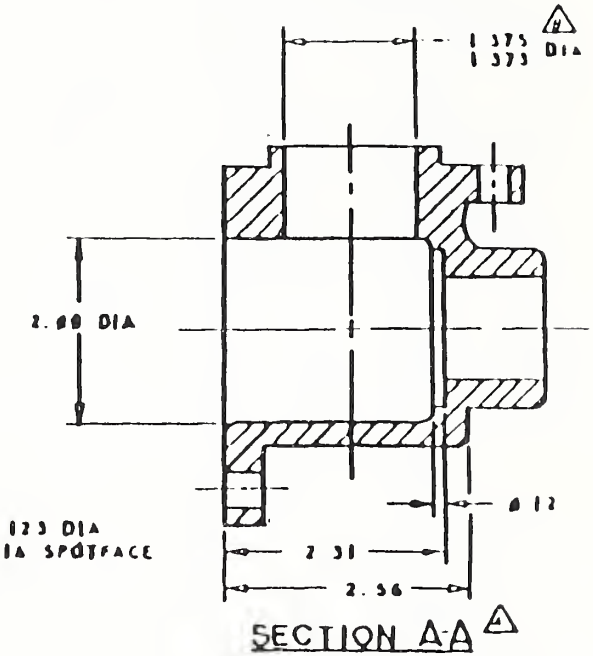
AUTOMATED DATA CAPTURE AND CONVERSION

PROCESSING LEVELS



1. BEND RADII 2.063 MAX
2. FINISH IN BRIGHT ZINC
3. PAINT OPTI GRAY
4. BREAK SHARP EDGES





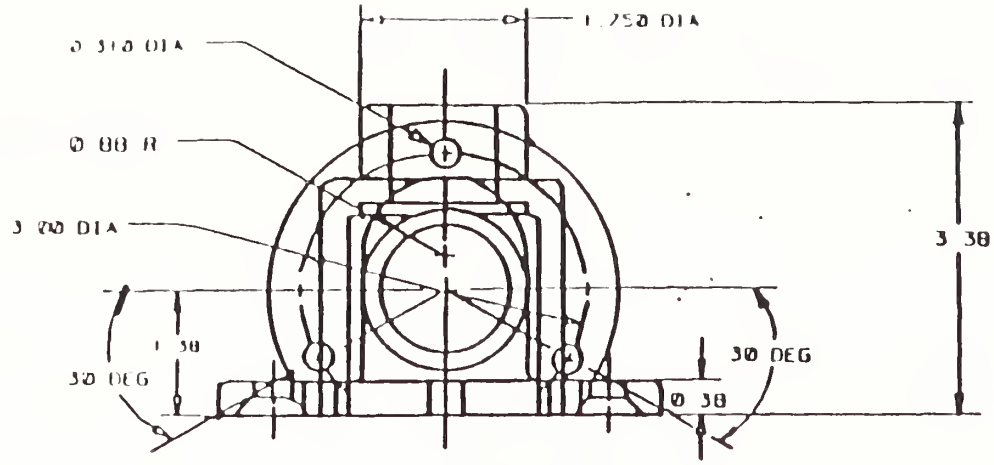
SECTION A-A

NOTE SEE DWGS 8-9 FOR ASSEMBLY DETAILS

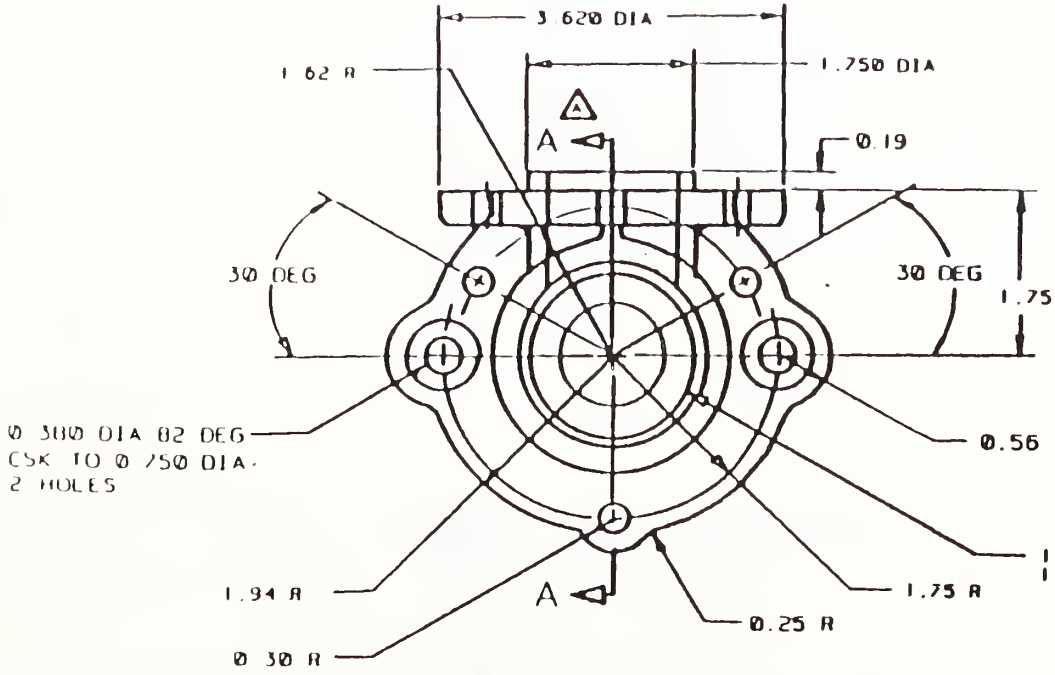
QC DATA

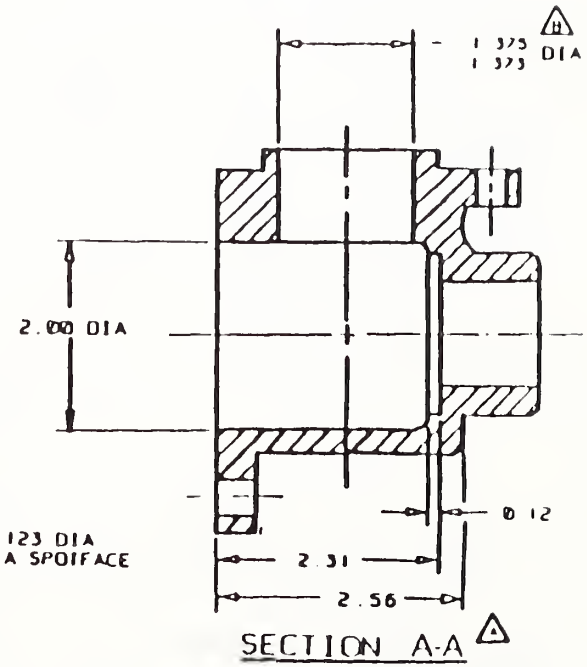
PROCESSED THROUGH:
1. SCAN AND VECTORIZE ONLY
TOTAL TIME = .36 HOURS

QUALITY LEVEL I



1. BEND RADII 2.063 MAX
2. FINISH IN BRIGHT ZINC
3. PAINT OPTI GRAY
4. BREAK SHARP EDGES





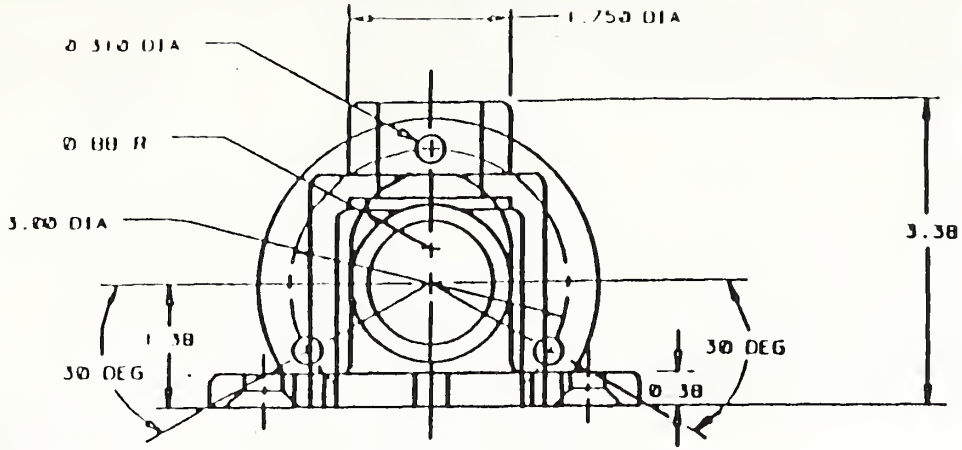
SECTION A-A

NOTE: SEE D*G 2-9 FOR ASSEMBLY DETAILS

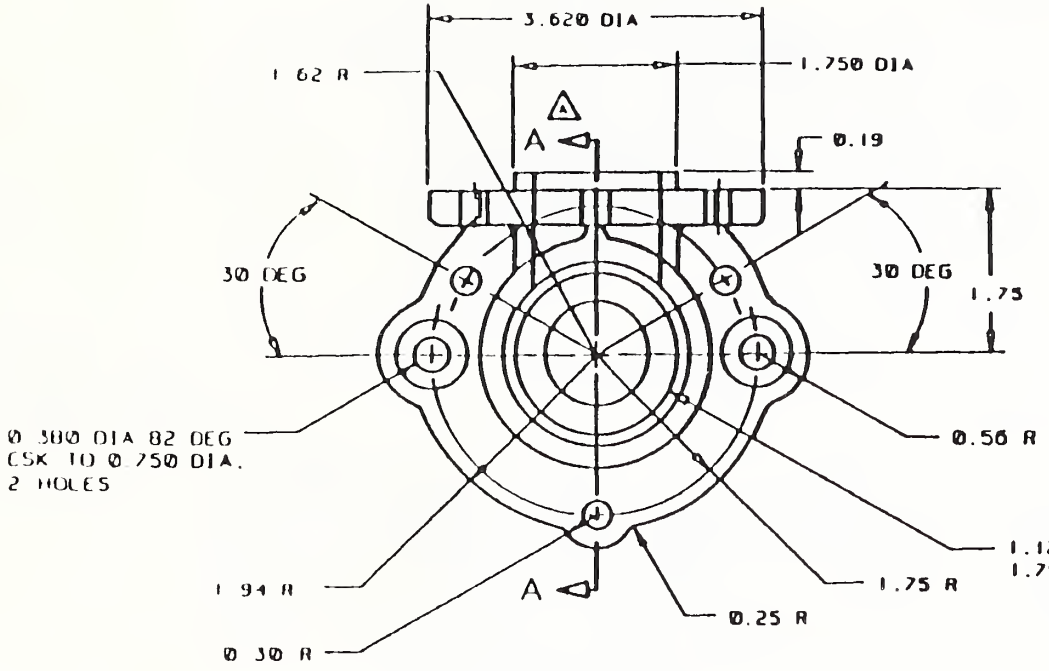
PROCESSED THROUGH:
1. TEXT ONLY
TOTAL TIME = .82 HOURS

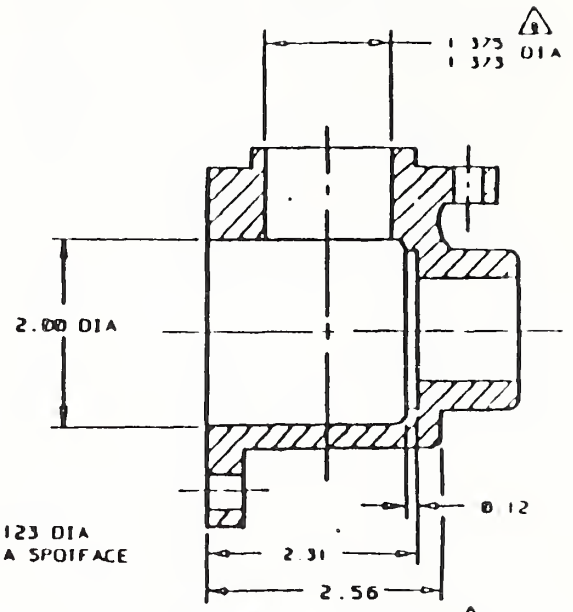
QUALITY LEVEL II

NOTE: TEXT IS NOW ACCESSABLE ASCII CHARACTERS



1. BEND RADII 2.063 MAX
2. FINISH IN BRIGHT ZINC
3. PAINT OPTI GRAY
4. BREAK SHARP EDGES





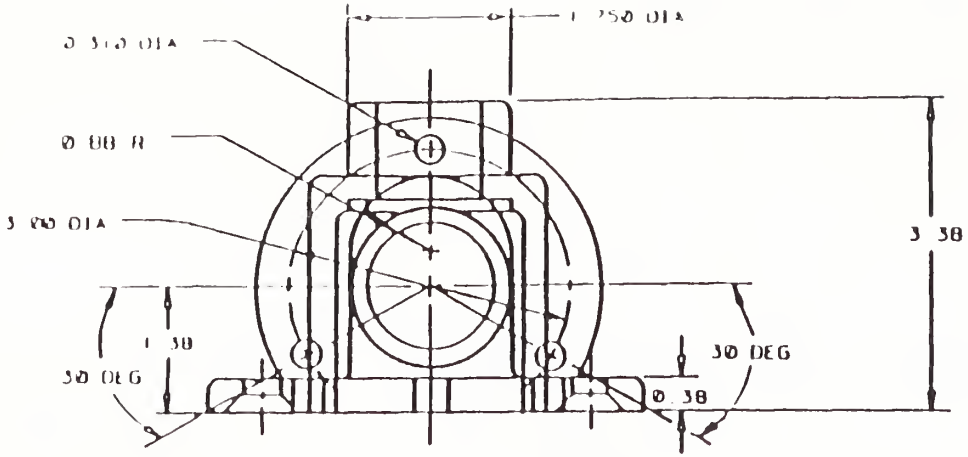
SECTION A-A

NOTE - SEE DWG 2 9 FOR ASSEMBLY DETAILS

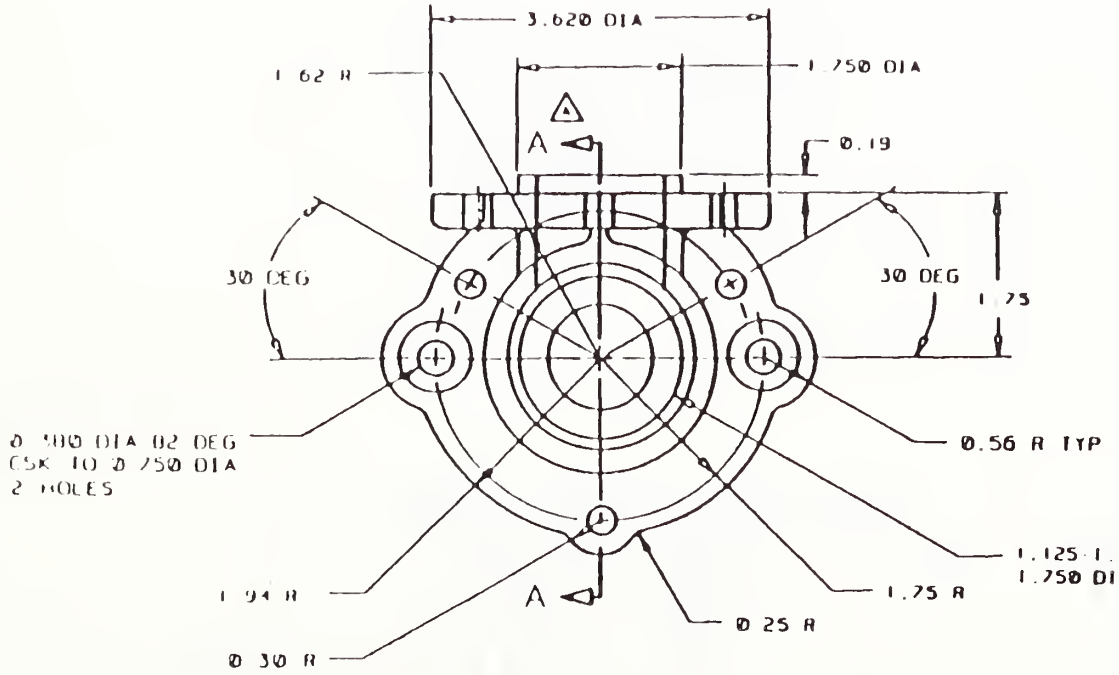
PROCESSED THROUGH:
1. RASTER EDIT
2. TEXT ENTRY
TOTAL TIME - 1.22 HOURS

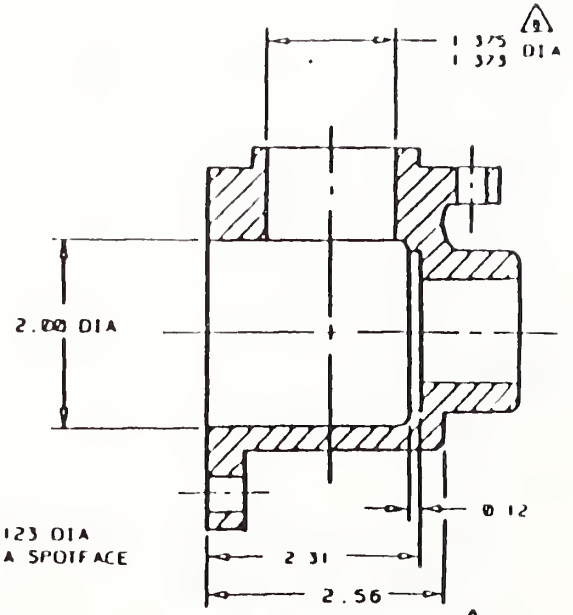
QUALITY LEVEL III

NOTE: 1. ALL LINES ARE SOLID
2. ALL STRAY MARKS ARE ERASED



1. BEND RADII 2.063 MAX
2. FINISH IN BRIGHT ZINC
3. PAINT OPTI GRAY
4. BREAK SHARP EDGES





SECTION A-A

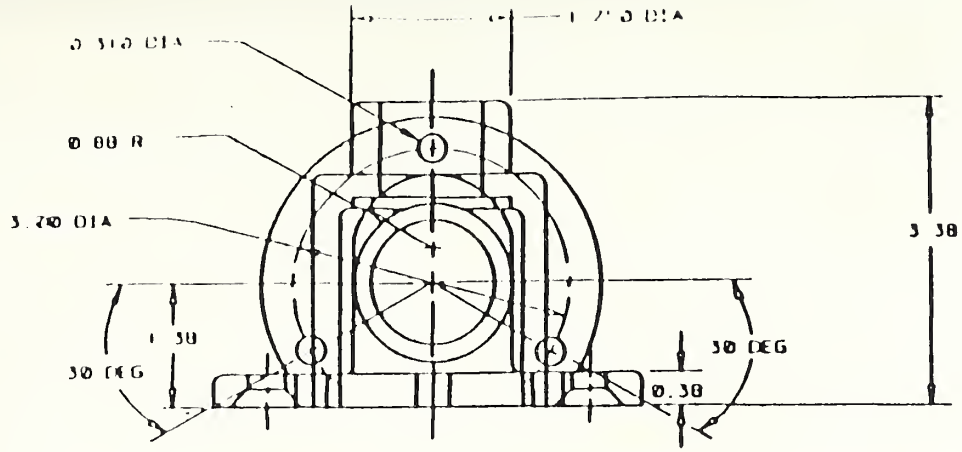
NOTE - SEE DWG 2.9 FOR ASSEMBLY DETAILS

QC DATA

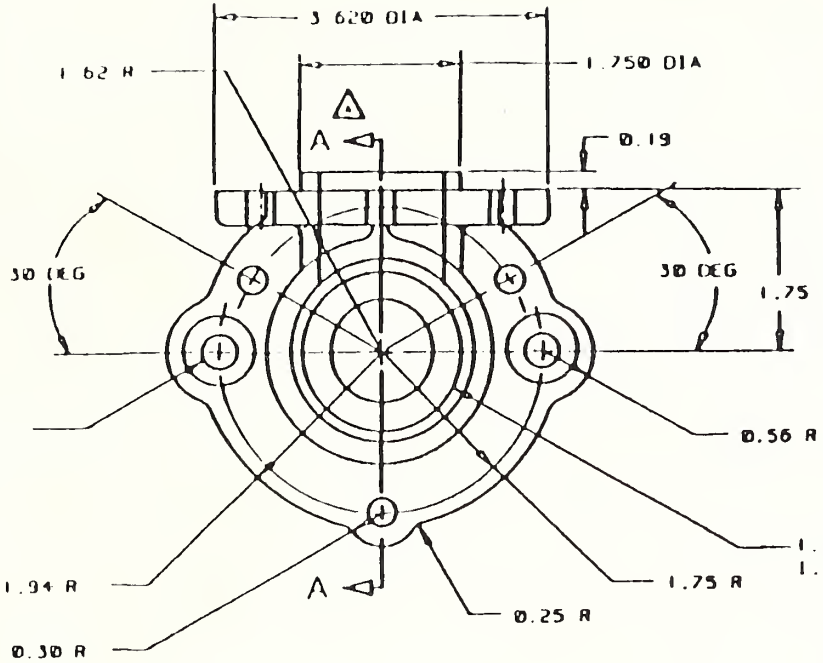
PROCESSED THROUGH:
1. RASTER EDIT
2. TEXT ENTRY
3. VECTOR DATA REVIEW
TOTAL TIME = 1.87 HOURS

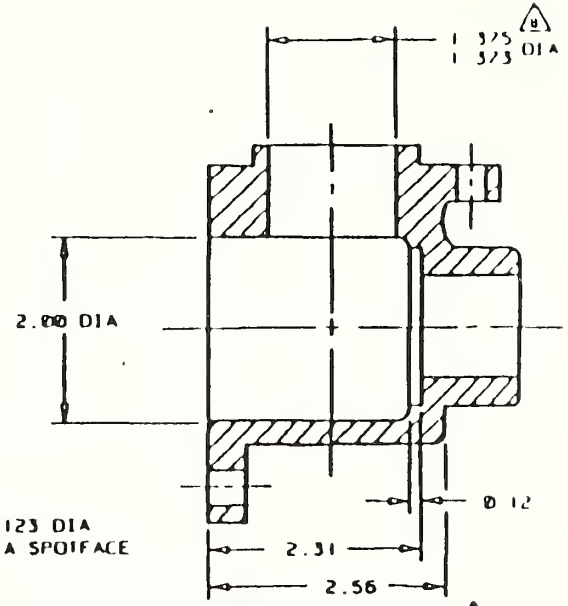
QUALITY LEVEL IV

NOTE - ALL ARCS, CIRCLES, ARROWS, ETC. ARE SINGLE ENTITIES



1. BEND RADII 2.063 MAX
2. FINISH IN BRIGHT ZINC
3. PAINT OPTI GRAY
4. BREAK SHARP EDGES





SECTION A-A

NOTE - SEE DWG 2 9 FOR ASSEMBLY DETAILS

QC DATA

PROCESSED THROUGH:
1. RASTER EDIT
2. TEXT ENTRY
3. VECTOR DATA REVIEW
4. VECTOR DATA EDIT
5. VECTOR POST PROCESS
TOTAL TIME - 2.93 HOURS

QUALITY LEVEL V

NOTE - 1 STANDARDIZED TEXT HEIGHTS AND LINE WIDTHS
2 ADDITION OF SYMBOLS
3 HIGHEST QUALITY DATA CAPTURE

REPORT

CALS SOW TASK 2.2.3.3.2

DEVELOPMENT OF CGM VALIDATION ROUTINES

August 27, 1987

DEVELOPMENT OF CGM VALIDATION ROUTINES

1. PURPOSE

Accelerate development of CGM validation routines. (Task 2.2.3.3.2)

2. BACKGROUND

As a result of NBS/ICST efforts on DoD CALS in FY 86, CGM was recommended as the computer graphics standard of most immediate benefit to DoD CALS. CGM, or the Computer Graphics Metafile, is a standard which provides for the description, storage, and communication of graphical information in a device-independent manner for automated information interchange. CGM will provide CALS with more efficient transfer and more compact storage of illustration files.

But the development of standards is, alone, insufficient for DoD CALS needs. The increasing complexity of computer technology standards such as CGM demands the creation and implementation of conformance tests to ensure that CALS systems will in fact be able to interchange data. Thus, requiring the use of CGM in major DoD weapons and automated systems procurement will ensure the availability of products that due in fact support the CGM standard.

DoD CALS has a very ambitious schedule for having such conformance tests in place to serve CALS needs. However, the NBS Strategic Plan for Validation of Computer Products in Support of the CALS Program predicts that CGM conformance tests will not be ready for at least three years. Validation tests for CGM are in their infancy, but NBS hopes to build on the extensive body of knowledge gained from the development of the GKS validation tests to try and accelerate this process for CGM. This particular task is a continuing effort, which will be accomplished by participating in the development by West Germany of the CGM conformance tests. If direct participation is not possible, then NBS/ICST will input CALS comments concerning the timeliness and quality of these tests on a continuing basis. This report serves as an update to DoD CALS concerning efforts made so far under this task.

3. DISCUSSION

As part of this task, Daniel R. Benigni and Mark Skall of ICST attended a GKS-3D and CGM Certification Workshop in Manchester, U.K. on March 2-4, 1987. The workshop was intended to examine in-depth the strategies for validating conformance to CGM and GKS-3D. Of course, the CGM session was the primary concern because of its possible repurcussions for CALS' needs for CGM

testing. Other participants included representatives of the various test centres in Europe, namely the U.K., France, and West Germany. They are the groups involved in testing implementations of the GKS computer graphics standard, and will be adding conformance testing of CGM implementations once they have been written. Thus they have a great stake in coming up with a CGM conformance strategy that is feasible.

3.1 CGM SESSION RESULTS

The CGM session concluded that the CGM standard has weak conformance requirements and tests for conformance solely to the standard were not sufficient. Test centres agreed with ICST's position that determining conformance of CGM generators and interpreters is essential for a test centre since the essence of CGM is to ensure that it can transfer graphical data between two systems. ICST introduced the concept of Application Profiles (AP's) as a means to define specific user requirements which should be tested against. This strategy envisions that different constituencies such as MAP/TOP and CALS would develop AP's for the CGM which generic test routines would use as input to test for minimum requirements for CGM generators and interpreters. Test centres in Europe indicated that they would be willing to test for different AP's. This is very important for CALS, since it could provide a mechanism for meeting CALS' specific testing requirements for CGM.

A proposed CGM testing architecture including testing for CGM generators and interpreters was defined based on the model for OSI transport layer testing. The conclusion about this architecture was that checking results would still be very manually intensive (i.e., visually comparing two pictures). Another problem concerns how to tell the implementation under test (IUT) how to create a metafile. Possible options include: let the site choose; give the site example graphics programs in GKS to produce metafiles; or give the site a picture from which to derive a metafile. These problems were not resolved at this meeting.

3.2 CGM TESTS STATUS

During the workshop NBS/ICST learned that GTS Gral, under contract to GMD, has begun work on CGM conformance tests to the standard in the form of a syntax checker which only tests the static metafile. Although this is of limited use for CALS, it does provide the basis for building further tests on top of it. Peter Scloendorf of GTS Gral, who is developing this software, gave the status of this development. In brief, he has almost completed this syntax checker for binary encoded metafiles and will be completing the software for all three CGM standard encodings in the next few months. NBS will be monitoring this work as closely as possible.

3.3 NCC AND TESTING OF THE COMPUTER GRAPHICS METAFILE

NBS, in conjunction with Eurographics, is cosponsoring an international workshop entitiled "The CGM in the Real World," which will take place at NBS in September. NBS, as cosponsor, has been able to assign a number of topics to top experts in the field to write position papers on various aspects of the CGM. In part to satisfy this particular task, Jane Pink, GKS Test Service Manager at the National Computing Centre Ltd. (NCC) in the UK, was asked for her opinions on how CGM implementations might be tested, what would be involved, how a test service might be set up for CGM, and what efforts the UK is planning for CGM testing. Her paper, entitled "Testing of the Computer Graphics Metafile" has been received, and here is a brief excerpt from it:

> It has been determined that very little testing for conformance can be carried out on a CGM implementation. Such testing is limited to checking the file format of a given metafile. However, it is likely that useful information could be provided by an evaluation service, providing for testing of generators and interpreters.
>
> ...such a service would be economically viable, ...assuming that funding can be found for development of the test system required, (since) it is likely that the demand from users of CGM implementations will force the implementors to submit their products for testing. The CGM has been included in the TOP (Technical and Office Protocol) Application Profile and it is likely that tests for conformance to this profile will be required.
>
> Future work in the UK on CGM testing will be directed toward ensuring that the tests and test procedures (that are developed for CGM) are accepted on an international basis.

Thus, it appears that the UK is serious about expanding their test services to include the CGM. NBS will be closely monitoring future efforts of the NCC in the area of CGM conformance testing as part of this continuing task.

4. RECOMMENDATION

Continue monitoring this work and provide comments and input concerning the timeliness and quality of the work.

5. IMPACT

NBS/ICST was able to set direction for an architecture for the development of CGM conformance tests which coincides exactly with the elements of the Statement of Work for CALS this year. First is that testing to the standard itself is not enough; the tests must include the CGM generators and interpreters. Part of the work this fiscal year is to develop a plan for the development of

additional conformance tests needed to validate software that generates and reads metafiles, which will be in the form of a reference implementation for CGM. Second, the test centres have initially agreed to test to CGM Application Profiles, a concept that NBS introduced at the CGM session. This is also reflected in the Statement of Work, which has a task to develop conformance definitions for CGM generators and interpreters in the form of an Application Profile. This task is being accomplished by defining an CGM Application Profile for CALS.

6. CONCLUSION

The result of this meeting for CALS is that some thought is being given to developing tests for CGM conformance, now that the standard is in place, and implementations are starting to hit the market which purport to adhere to the standard. NBS plans to continue this work, and through our efforts on the Application Profile for CGM in CALS and the development of the reference implementation, we hope to accelerate this effort in the coming years to meet CALS schedule needs.

7. REFERENCES

A. Proposed Statement of Work, NBS support for DoD Computer Aided Logistic Support Program, December 18, 1987.

B. NBS Strategic Plan, Plan for Validation of Computer Products in Support of the CALS Program, Version 2, January 29,1987.

C. Summary Foreign Trip Report, GKS-3D and CGM Certification Workshop, March 13, 1987.

D. Testing the Computer Graphics Metafile, Report on the Workshop Held at the Moorside Hotel outside Manchester, U.K., March 2-4, 1987.

E. Pink, Jane, "Testing of the Computer Graphics Metafile", to be delivered at The CGM in the Real World Workshop, September, 1987.

8. APPENDICES

Reference D provides an in-depth report of the entire CGM session of the Manchester Workshop, and follows as Appendix A.

APPENDIX A

TESTING THE COMPUTER GRAPHICS METAFILE

REPORT ON THE WORKSHOP HELD AT

THE MOORSIDE HOTEL, DISLEY, ENGLAND

2nd-4th March 1987

Testing the Computer Graphics Metafile

Report on the workshop held at The Moorside Hotel, Dudley, United Kingdom.

The group considered the conformance requirements of the CGM. The CGM recognises two levels of conformance: firstly, Full Conformance where a metafile conforms to both the abstract specification of the CGM (Part 1) and to one of the three encodings (Parts 2-4); secondly, Functional Conformance where the metafile conforms to the abstract specification but uses an encoding other than those described in the standard. It was considered that clearer, and possibly stronger, conformance requirements could have been specified. It was recognised that, in practice, such conformance restrictions are being recognised for application areas, such as MAP/TOP and the Office Document Architecture Standardisation.

Validation testing of a metafile and evaluation of the Generator and Interpreter were considered to be needed by a testing centre. Although only the metafile itself can be tested for conformance there will be value in further evaluation. This may offer testing to application profiles, such as the MAP/TOP profile.

Testing for full conformance of a metafile was discussed in some detail. There is a need to test the semantics and syntax of a metafile. This can be similar for all three encodings. The elements included can be compared with the element list. Consistency of parameters can also be examined.

Functional conformance is more difficult to test. We need to extract information about the test metafiles and thus test their conformance to the abstract specification of the CGM. The group felt that it was reasonable to require a clear text encoding description of the metafile from the system under test. Binary or character encoding could also be supplied but may require more work for the implementor. This clear text metafile could then be tested by the test centre.

An overall testing strategy was envisaged with the test centre having a CGM encoder and decoder capable of producing and interpreting any legal metafile. Illegal metafiles may also be produced for testing. The test centre would take a metafile from the test implementation and check for conformance to the standard as described above.

It was considered useful to be able to specify the content of the test metafiles. The only other alternative is for the test site to choose the metafiles to be tested; this is inconsistent between tests. The test centre could specify the metafiles using one of two approaches: describe pictures to be created by the CGM generator, either in some formal way or simply as pictures; or to give the test suite some GKS programs from which metafiles can be created. This latter choice does not define the precise

nature of the metafile produced but has the advantage of specifying the picture to be produced in a standard way and is more equitable between tests.

Testing the interpreter is not part of the standard but may be a useful evaluation, or necessary for particular application areas. The test centre, having extracted information about an implementation, would generate test metafiles to the alleged support level. The pictures produced by the interpreter could be checked to some standard pictures and script. These may be the same pictures as used for the GKS operator tests where appropriate. In cases where both a generator and interpreter are available an output metafile can also be tested.

Report information is also necessary. A file of statistical information on the content of a metafile would be produced. Information containing details of errors is also necessary. This latter file will contain information on the nature and location of the error; a reference to the standard where appropriate; a list of any parts of the metafile which have been skipped; and a summary of the elements.

The CGM Group

Anne Mumford and Jane Pink made a presentation to the group indicating possible testing strategies.

1. Introduction to the CGM

The CGM consists of 4 parts:

Part 1 Abstract Specification

2 Character Encoding

3 Binary Encoding

4 Clear Text Encoding

The CGM defines two levels of conformance:

- full conformance
  - to Parts 1 and an encoding from Parts 2-4

- functional conformance
  - to Part 1

Tests for conformance are only tests of the generator. It may be useful to have evaluation tests as well.

What approach can be made to CGM testing? We can learn from GKS tests and also from OSI tests. The tests for the CGM may be appropriate for the CGI too.

2. Good test tools

Jane Pink briefly discussed criteria for good test tools. The important points were:

- test tools should be easy to implement. Not too labour intensive, and not too costly to implement.

- result reporting should be as automated as possible. Less manual checking. Less reliance on human judgement.

These criteria make life easier for the test centre.

The presentation then looked at a possible testing strategy for the CGM.

An important part of testing the CGM is to ensure that it can transfer graphical data between two systems.

Looking at testing CGM interpreters and generators. The only test service commercially available, which bears some similarity to the testing of CGM interpreters and generators is OSI testing. In OSI testing, the aim is to check for the correct transfer of

data at some layer of the model between 2 systems. A limited OSI test service has been available at NCC since 1983 and part of this test service is a test system for testing for conformance to layer 4, the Transport layer, of the OSI 7-layer model.

It seemed sensible to investigate the test strategy used for transport layer testing and see if it was applicable to the CGM. It we briefly look at the test model used (figure 1).

3. OSI Transport Layer Testing

3.1 Introduction

The aim is to demonstrate that the Transport Service under test is capable of providing for the transfer of data between users.

The testing involves communication over a live network between two distinct systems, one of which is under the control of NCC.

3.2 Test Method

As with all 3rd party test systems 'black box' testing is used i.e. the internals of the system under test are not examined. OSI testing is carried out remotely, controlled from the NCC test machine.

3.3 Elements of the OSI Test System

Test Driver.

Overall control of the tests to be performed is done by the test driver.

The test driver reads and executes tests held on backing store.

"Reference Implementation"

An implementation of the Transport Service with an additional capability of sending/receiving invalid data.

Data is transmitted to the Implementation Under Test (IUT) by means of the network service, provided by layer 3, the network layer.

The Reference Implementation also receives data from the IUT and decodes it.

Exception Generator

- logs incoming and outgoing data (PDU's)

- generates invalid data.

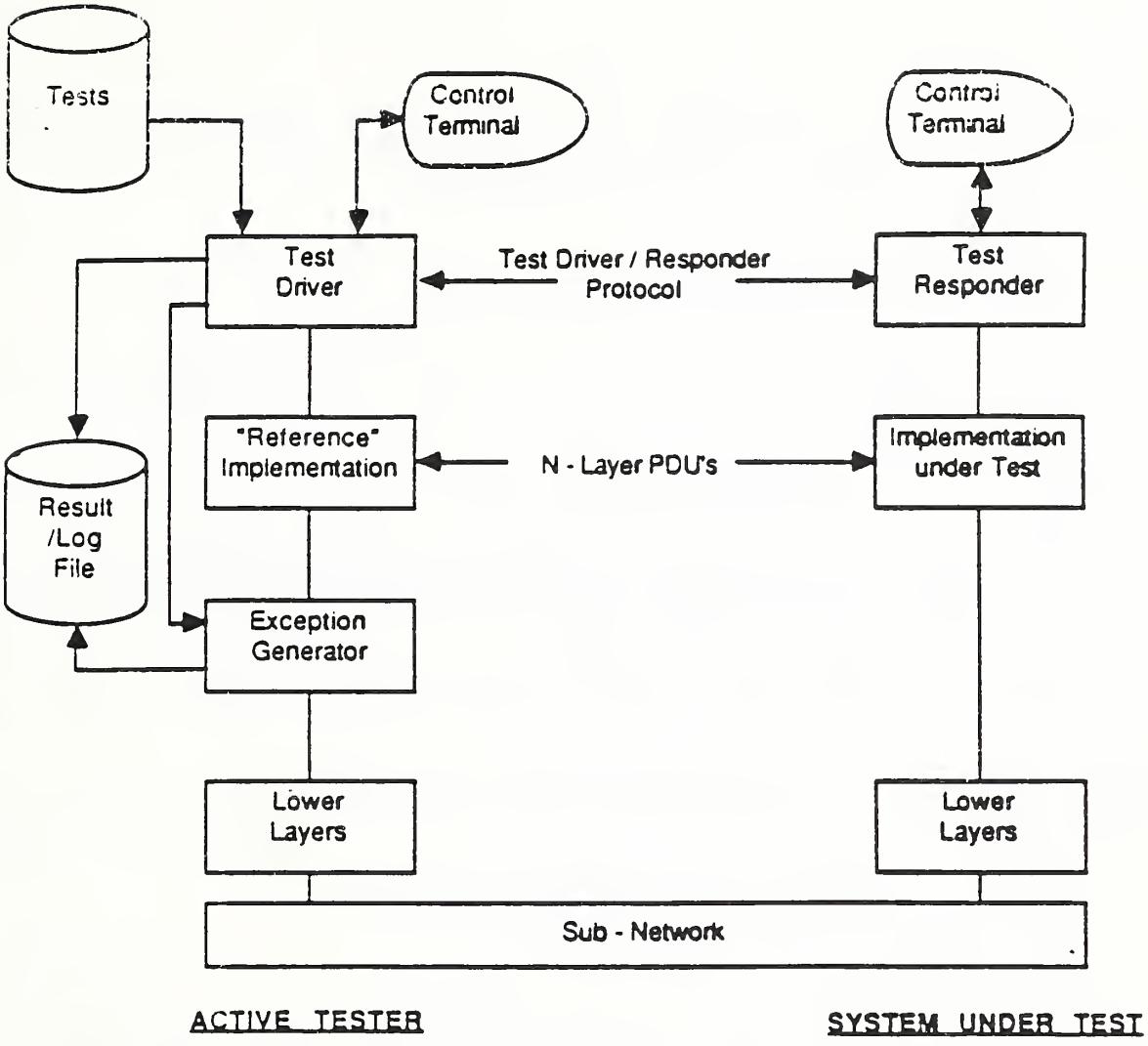


Figure 1
NCC OSI Testing Model

System Under Test

IUT - Implementation Under Test.

The Test Responder is supplied by NCC, in a number of programming languages. It represents a predictable user to the IUT and controls and observes the implementation.

3.4 Summary

In summary the similarities to CGM testing are:

- trying to demonstrate that data can be transferred correctly between systems.

However, in OSI testing, activity can be monitored both at the upper and lower interfaces of the IUT.

Therefore for transport layer testing, we can observe via the network layer (the transport layer uses services of the network layer) and the presentation layer (the transport layer provides a service to the presentation layer).

The CGM is different. We can only examine activity at one interface.

However, Anne and I have attempted to fit CGM testing into an OSI like model.

3.5 A proposed CGM Test Architecture (Figure 2)

The test library contains test data.

The utility program controls the encoder/decoder. The utility program is capable of generating test cases according to information supplied about the capabilities of the IUT.

The encoder/decoder is a CGM implementation with the extra capability of being able to generate incorrect CGM data streams.

3.6 Test process

The encoder/decoder sends a CGM to the IUT. The IUT interprets the metafile and outputs the picture to a display. The display can then be manually checked for correctness. The IUT can then be asked to generate the metafile. The encoder/decoder reads the metafile and outputs the picture to a display for checking.

Figure 3 looks at testing only the CGM generator. This is more difficult. We are asking the implementor to generate a particular test picture from some picture description data. The picture is stored in a metafile. The encoder/decoder reads the metafile and outputs the picture to a display for checking.

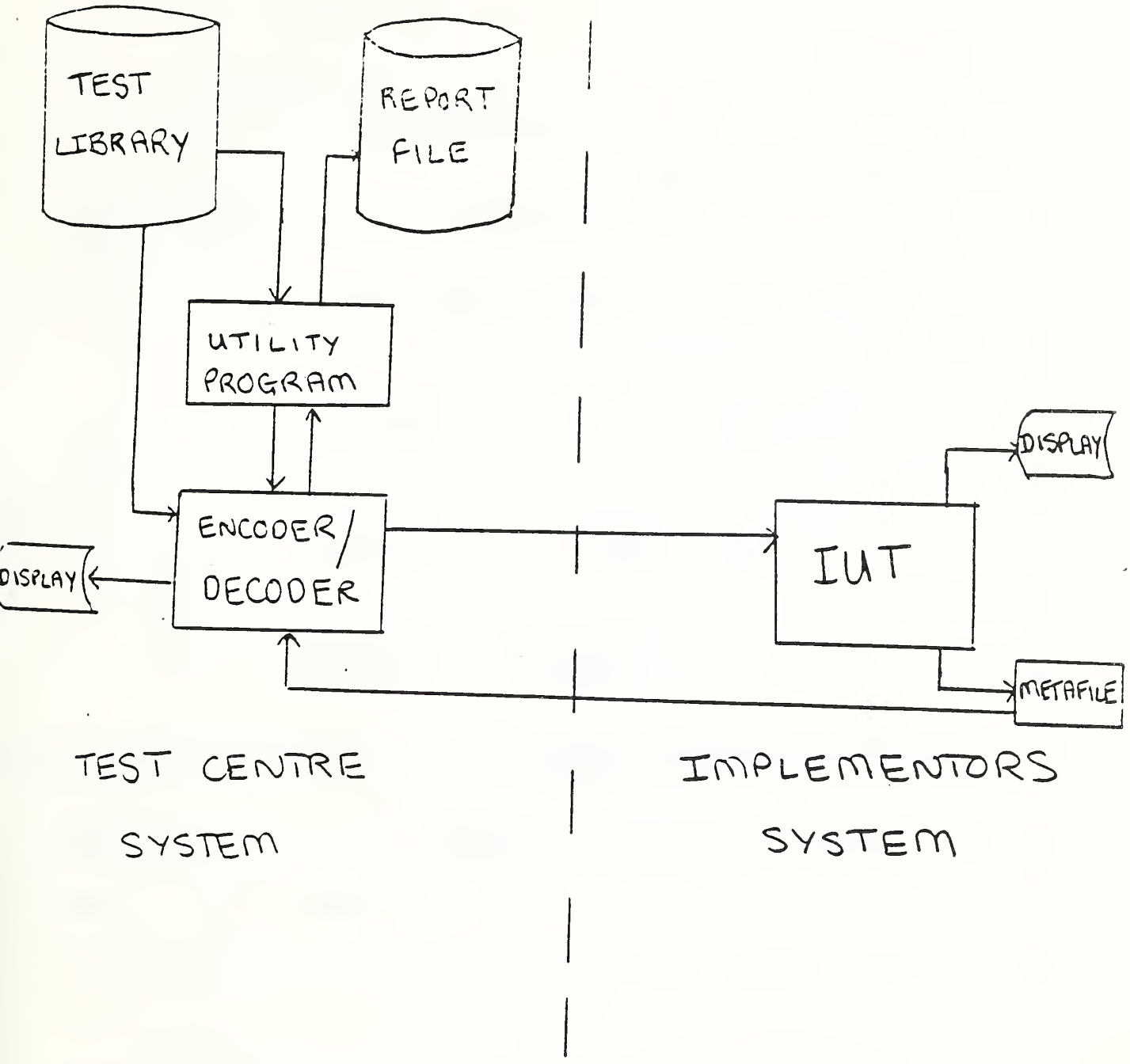


Figure 2
A Proposed CGM Test Architecture

3.7 Conclusions

Process described for CGM testing is very manually intensive. We must look at ways of automating the test model.

4. Testing The CGM Generator

The state diagram from the CGM (figure 4) gives a useful basis for testing.

We can recognise different levels in the metafile.

Metafile Level

BEGIN METAFILE	Metafile Descriptor	Pictures	END METAFILE

Picture Level

BEGIN PICTURE	Picture Descriptor	Picture Body	END PICTURE

Picture Body Level

BEGIN PICTURE BODY	Control Elements	Primitive Elements	Attribute Elements

We can look at the structure independent of the actual elements. Details can be output in the report.

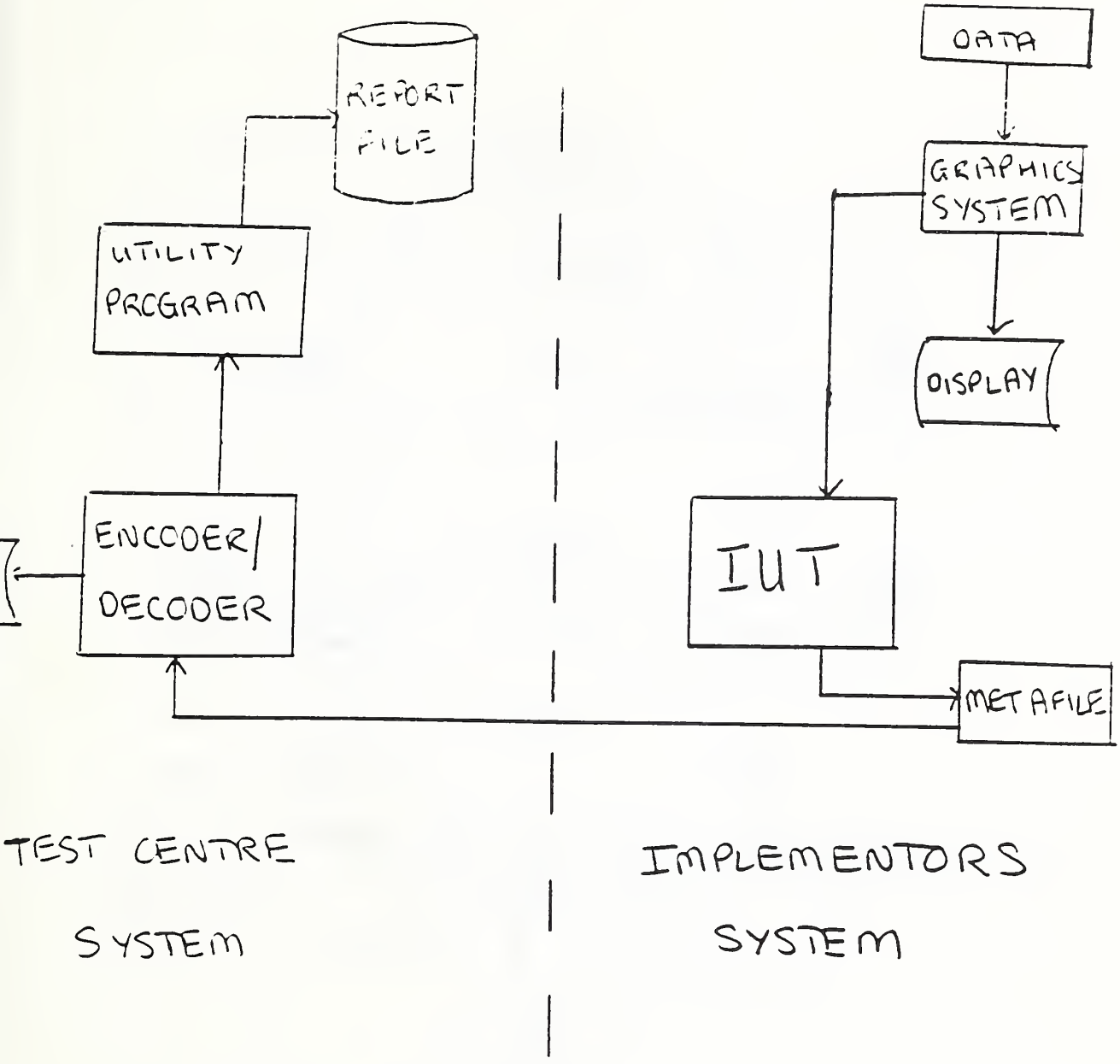
Internal consistencies can also be checked, e.g. if the VDC TYPE is to be integer then are integers used?

Testing can be automatic and recovery attempted to the next element.

Errors can be reported along with the script giving the metafile contents. This could be clear text encoding.

A major problem is how to tell the implementation under test how to create a metafile. Possible options are:

- let site choose
- give site example graphics programs in e.g. GKS to produce metafiles.
- if the system is an interpreter and a generator then give the interpreter a metafile and get the IUT to produce a display and a metafile. This is then checked. This is not ideal as we cannot guarantee output will be the same as input.



TESTING OVER A NETWORK

CGM GENERATOR

Figure 3
Testing the CGM Generator

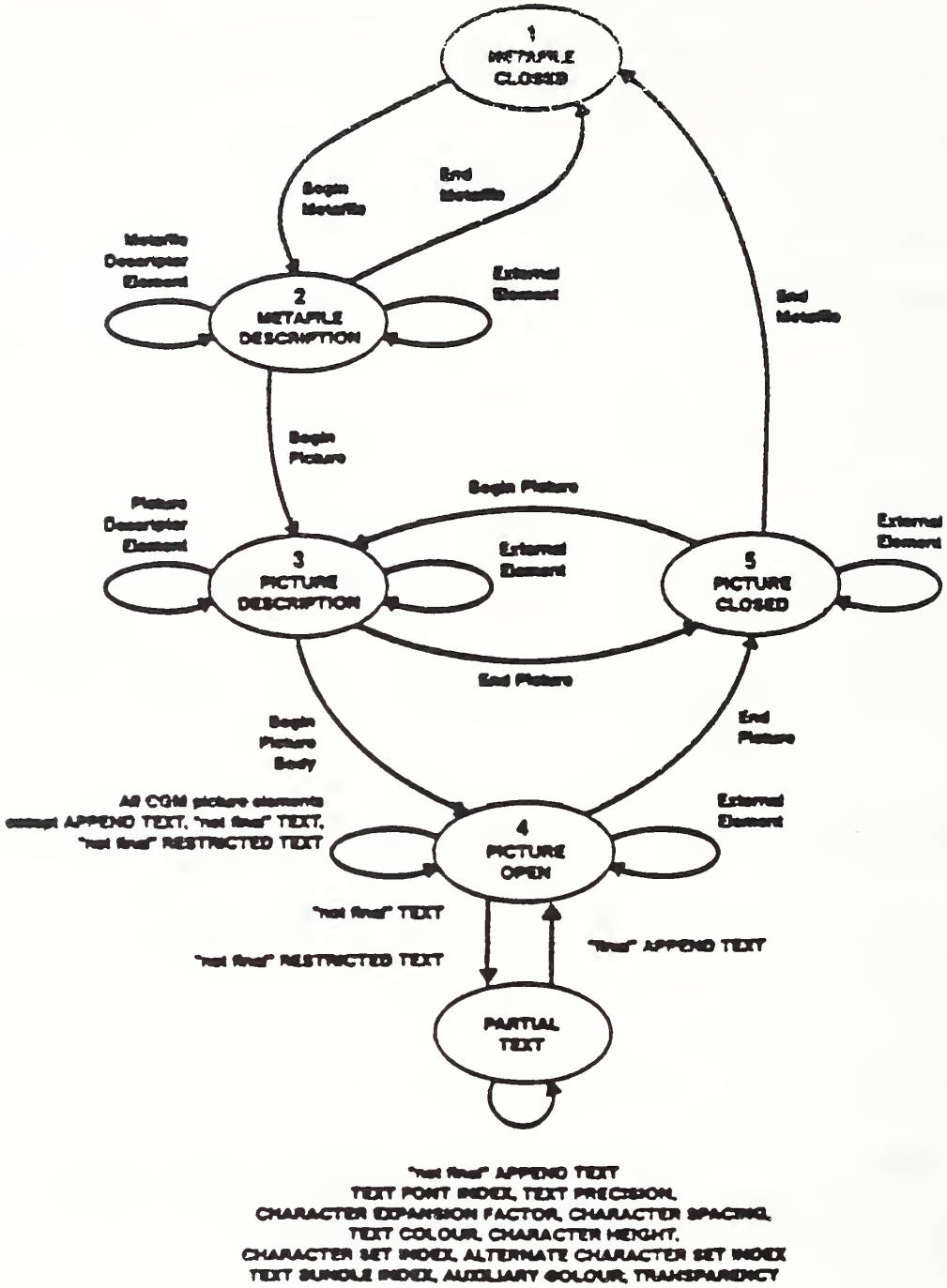


Figure 4
The State Diagram from the CGM Standard

5. Testing the CGM Interpreter

The test centre can send valid metafiles which the interpreter claims to cope with. The graphical output can be checked. Also the interpreter could possibly produce a metafile which is sent back for testing on the test service machine. This could possibly go through another loop back to the IUT.

6. Evaluation

This is also useful and things which might be included are:

- can the interpreter cope with
  - invalid metafiles
  - metafiles with elements beyond those it claims to support
- what elements are supported?
  - DRAWING SET
  - DRAWING Plus CONTROL SET
  - MAP/TOP Application Profile
- does the interpreter have the same capability as the generator.

These presentations were used by the CGM group as a basis for discussion.

Minutes of the CGM Group: Monday 2nd March 1987

1. Members of the Subgroup for the workshop:
   Chair - Anne Mumford, Loughborough University
   Members - Peter Schloendorf, GTS-GRAL
   Clemens Pflueger, GMD
   Jane Pink, NCC
   Martin Goebel, PhG-AGD
   Dan Benigni, NBS
   Mark Skall, NBS

2. The first order of business was to try and decide on an Agenda to work by. Four points were mentioned, namely:
   a) What are the goals and objectives of CGM in the area of conformance?
   b) Discuss Full vs Functional Conformance.
   c) Discuss structure and syntax.
   d) Discuss CGM relationships to other standards - GKS, CGI, Extended CGM.

3. In order to better understand the goals and objectives of the CGM standard with respect to conformance, the following portions of the CGM specification are repeated here so there will be no confusion.

   7.1 Forms of Conformance
   This standard specifies functionality and encodings of Computer Graphic Metafiles, it does not specify operations or required capabilities of metafile generators or metafile readers. Guidelines are provided in Annex D, however, for those striving for uniformity of results.

   A Metafile may conform to this Standard in one of two ways. Full Conformance occurs when the Metafile conforms to one of the encodings specified in the Standard. Functional Conformance occurs when the content of the Metafile corresponds exactly to the Functional Specification given in part 1 of this Standard, but a private encoding is used. These rules are expanded in the following sub-clauses.

   7.2 Functional Conformance of Metafiles.
   A Metafile is said to be functionally conforming to this Standard if the following conditions are met.
   a) All graphical elements contained therein match the functionality of the corresponding elements of this Standard.
   b) The sequence of elements in the Metafile conforms to the relationships specified in this Standard, producing the structure specified in this Standard. For example, the Metafile must begin with BEGIN METAFILE and end with END METAFILE, include exactly one metafile description at the beginning which contains at least all the required elements, and so forth, as specified in this Standard.

c) No elements appear in the metafile other than those specified in the Standard, unless required for the encoding technique. All non-standardized elements are encoded using the ESCAPE elements on the external elements APPLICATION DATA and MESSAGE.

7.3 Full Conformance of Metafiles.
A Metafile is said to be fully conforming to this Standard if the following conditions are met.

a) The Metafile is funtionally conforming, as specified above.

b) The Metafile is encoded in conformance with one of the standardized encodings specified in this Standard.

7.4 Conformance of Other Encodings.
A functionally conforming Metafile may use a private encoding. While it is beyond the scope of this Standard to standardize rules for private encodings, Annex B suggests minimum criteria that private encodings should meet.

4. Also under the topic of the goals and objectives of CGM with respect to CGM Conformance, the first question that arose was:

- Who will apply for CGM testing?

- Will they generally have both generators and interpreters?

At this point Mark Skall introduced the concept of the Application Profile (AP), which is being used in the MAP/TOP arena in support of CGM as the interchange format for computer graphic picture description information in an OSI environment.

An AP defines conformance characteristics or permissible combinations for all possible data streams that conform to that profile. In Mark's words, it defines a set of requirements needed by a generator and interpreter to transfer data for a class of applications. An AP insures inter-operability of implementation of ISO 8632.

The AP for MAP/TOP specifies conformance to the CGM in terms of PERMISSIBLE, BASIC, NONBASIC, and DEFAULT values. Permissible values are the range of values for CGM elements as specified in ISO 8632. Basic values are the range of permissible values that are mandatory for conformance to this AP. Nonbasic values are the remainder of permissible values for CGM elements. The default values for CGM elements are the implicit initial values that are assumed for each parameter. Default values are explicitly overridden by the Metafile Description Elements, Picture Description Elements, Control Elements, and Attribute Elements.

5. After a short discussion, the question was posed to the sub-group: Do we need to test beyond the functional specification? The answer was yes. Then there was some discussion of what the breadth of testing should be. We would like as a start to be able to have a simple, automatic pass/fail test, but that is good for only the syntax testing and for consistency of metafile definition. But then the discussion turned on the need of testing the entire system, of which CGM is only a part. Then we tried to identify what the whole system might be in a CGM testing environment.

   The two figures from Jane's and Anne's talks on their proposed CGM testing architecture were then debated. It was argued that in both of these figures, too much emphasis was placed on the manual side of testing via displays, which could be placed in a number of places.

   It was discussed what form the data should be in if either of these figures were used. The 'Encoder/Decoder', which specifies a full reference implementation capable of producing any legal CGM, could pass data to an Implementation Under Test (IUT) in the form of data, or a metafile, or pictures. The pros and cons of each were discussed. The point was also raised about the IUT - should it create a metafile which the testing service could use to test against? Should an IUT be able to choose from a number of pictures from a particular AP for a class of applications? How does an implementor get these pictures?

6. In terms of interpreters, the test service could have a metafile based on an AP and restricted by what the IUT says it supports. Then the IUT mught have 3 choices of output, either: pictures, another metafile, or to a CGI. If pictures were the choice, then we come around to the same argument about manual comparison with no guarantees.

7. The meeting broke off with Mark expressing the need for developing a list of issues.

Minutes of the CGM Group: Tuesday March 3rd 1987

9.00 am - 11.00 am

1. Introduction - Summary of previous discussions

It is identified that the testing of CGMs requires the testing of the syntax and semantics of the CGM which provides a picture capture mechanism.

Testing according to "application profiles" seems to be useful to reduce the complexity of testing tools and to provide a useful service. Further evaluation of generators and/or interpreters seems to be desirable, although this is very badly (not) described in the standard. For the second purpose the need was seen to separate between generator testing and interpreter testing.

The overall aim of the group is to set up a test strategy for CGM. The first steps will be to consider aspects for testing "Full Conformance". Then it must be evaluated if the tools/ strategy is also applicable for testing "Functional Conformance".

2. CGM - Syntax Checker

Peter introduced us to the work he has done so far for testing CGM syntax. The checker (see appended paper) is made for binary encoded metafiles and assumes a certain file format.

The checker is prepared to test the structure of a CGM, i.e. performing state transitions according to the delimiter elements (e.g. BEGIN METAFILE, BEGIN PICTURE, BEGIN PICTURE BODY, END PICTURE, END METAFILE). Furthermore, within each state the checker is prepared to decide which Metafile Elements are allowed and which are illegal.

As the tool is divided into decoder and checker (state transition machine) it could be used for other encodings assuming the decoder will be exchanged.

The checker recognises fatal errors, i.e. errors in the structure (multiple existence of the BEGIN METAFILE, ...) and other (non fatal) errors, e.g. illegal use of metafile element. An error report is generated. In the case of fatal errors the checker stops whereas in the case of other errors

the checker skips to the next delimiter element. It should be mentioned that the error behaviour is not yet determined and may be changed.

It is recommended to extend the checker to be complete state transition model of CGM, i.e. including the "PARTIAL TEXT" state.

3. Areas precisely defined in CGM (for syntax testing)

It is intended to specify as many aspects as possible that are to be used for precise testing of full conformance. Precise, in this case means that a "PASS/FAIL" decision is possible! Four aspects found so far are:

a) the valid CGM Structure - which is expressed by the state diagram (Figure 4). Delimiter elements perform state transitions.

b) the appearance of illegal elements - within each state a certain class of metafile Elements is allowed while others are illegal. The use of illegal elements results in an invalid metafile.

c) consistency with the CGM element list - the Metafile Descriptor Elements describe the functional capabilities required to interpret a metafile (section 4.3 of the CGM Standard). This Metafile Descriptor potentially defines "application profiles". A metafile must not use other elements than those specified in the descriptor element.

d) consistency of parameters - depending on settings/ selections (e.g. colour selection mode) the parameters of specific metafile elements could be restricted to specific types (e.g. index or RGB values) or limited to specific ranges for parameter values.

The extension of the CGM within ISO defines metafile categories. For different metafile categories, like a GKSM-category with session capture facilities, the testing tools may be different, because different categories (picture capture - session capture) may contain different state transition models or allow/permit the appearance of metafile elements in different states.

Inter-relationships between elements in the metafile

There are complicated interrelationships between elements in the metafile. The aim is to look for possible checks for consistency.

Metafile Descriptor Elements

Element	Elements effected
VDC Type	graphical primitive elements, vdc extent
Integer Precision) Real Precision)	graphical primitive elements and attribute elements
Index Precision	attribute elements
Colour Precision	colour attribute elements, background colour where direct colour is used
Colour Index Precision Maximum Colour Index	colour attribute elements where colour selection mode is indexed
Colour Value Extent	colour attribute elements, background colour where direct colour is used
Metafile Element List	discussed elsewhere
Metafile Defaults Replacement	
Font list	text font index (check that fonts not listed are not then requested in the metafile)
Character Set List	character set index alternate character set index (check that not used one that is not listed in the metafile descriptor)
Character Coding Announcer	character set index alternate character set index (check that only the extension mechanism stated is used)

Picture Descriptor Elements

N.B. Looking at encodings, to see if it can be checked, for example, whether a real number has been given if that is required. This was considered, but left for possible future discussion.

Element	Elements effected
Scaling Mode	graphical primitive elements
Colour Selection Mode	colour attribute elements, auxiliary colour
Line Width Specification Mode	line width elements
Marker Size Specification Mode	marker size element
Edge Width Specification Mode	edge width element
VDC Extent	-

Control Elements

Element	Elements effected
VDC Integer Precision) VDC Real Precision)	graphical primitive elements
Auxiliary Colour	-
Transparency	-
Clip Rectangle	-
Clip Indicator	-

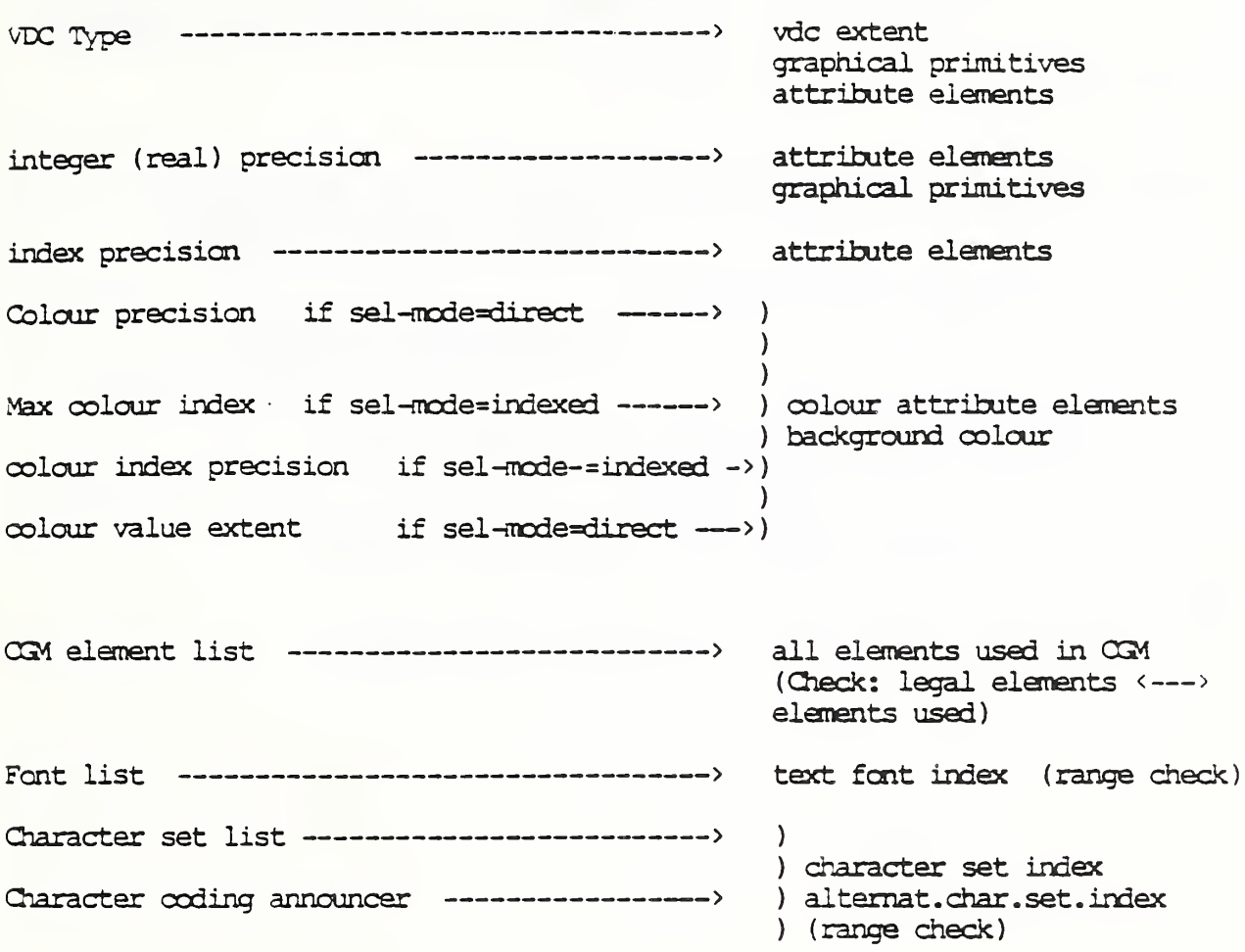
The need to check whether a given value is within a given range appears in several situations, for example

item	valid values
integer precision	8, 16, 24, 32
character set list	0, ...4

So it seems to be sound to provide a routine covering these checks in a unified manner.

These inter-relationships are summarised in the tables over.

Consistency graph for CGM (Source of influence: Descriptor elements)



Consistency graph for CGM (Source of Influence: Picture desciptor elements)

(check: mode announced <-----> mode used)

Scaling mode ----------> graphical primitive elements
graphical attribute elements

Colour Selection Mode ----------> colour element attribute
auxiliary colour element attribute

line width specification mode --------> line width element

marker size specification mode -------> marker size element

edge width specification mode --------> edge width element

Consistency graph for CGM (Source of influence: control elements)

vdc integer precision ---------->	graphical primitive elements attribute elements
vdc real precision ---------->	graphical primitive elements attribute elements
integer precision (range check) --->	provide check that a given value is within the required range
character set list (range check) -->	provide check that a given value is within the required range

Parameter Checks Derived From Descriptions

CGM el Descr.el	graph el[1]	graph attr	VDC extent	colour attr	backgr. col	text font.ix	char set.ix	alt.chr set.ix
*1								
VDC type	X	X	X					
Integer prec	X	X	X					
Real prec	X	X	X					
Index prec		X		X		X	X	X
Colour prec		X		X	X			
Max.col.ix		X		X				
Col.ix.prec		X		X				
Col.val.ext		X		X	X			
Font list		X				X		
Char.set.lst		X					X	X
Char.cod.ann		X					X	X
*2								
Scaling Mode	X	X						
Col.Sel.Mode		X		X				
LW Spec.Mode		X						
MS Spec.Mode		X						
EW Spec.Mode		X						
*3								
VDC Precision	X	X						

*1 MF descr.el
*2 Pic-Descr-el
*3 Cont.el

CGM Testing Subgroup - Tuesday 3rd March 12noon - 1pm

Summary

So far, it has been assumed that we have full conformance. We need to consider how to go about testing functional conformance. We also need to think about the sort of output to be output from a test service. Finally we must consider the overall testing strategy presented on Monday.

Testing for functional conformance

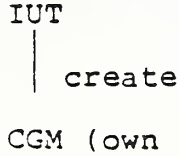
The idea was suggested of splitting off the decoder from the checker. The checker would be supplied by the test laboratory. The implementor would supply the decoder for a private encoding, which would provide mappings to the function calls. The test laboratory would have to define the interface to the checker.

Alternatively, a subroutine library could be supplied to the test site, which would enable them to create a metafile in one of the encodings.

It was agreed that some type of functional interface (language bindings) is what is required.

It is probably reasonable to ask the implementor to provide an interface to the test software.

It was agreed that we need something which is a functional interface for a decoder to call



Implementor has decoder

requirement of test lab

pictures
1. CGM (in any one of 3 encodings)
or 2. File (e.g. BEGIN MF
BEGIN PB etc)
or 3. Clear text encoded metafile

A method is required to translate private encoding to a standard encoding.

It is probably not reasonable to ask them to produce 1. The test lab must provide some software to enable them to produce a file as in 2. Software would be a series of subroutines. For example, when they produce a BEGIN metafile, they call a particular subroutine.

Alternatively the implementor would be asked to output a file in a defined format giving details of the functions called.

It was suggested that the implementor should produce a clear text encoded metafile. The implementor would be presented with part 4 of the standard and they would produce a file of clear text encoding.

In order to test private encodings, some interface is required and that interface might as well be a script in clear text encoding.

People who provide metafiles, should at least be expected to produce a clear text encoding. In effect, the implementor is being asked to produce one of the encodings. The conclusion is that this is as easy as asking the implementor to provide the information in some other form. However, this is back to full conformance.

For the purpose of operating a test service, the suggestions of Annex B of the CGM Standard for designers of private encodings,

i.e. - to ensure the ability to translate a metafile encoded in one of the standardized encodings into a private encoding
- to ensure the corresponding ability to translate a private encoded metafile to one of the standardized encodings

has been reinforced as the only practicable solution to enable testing.

Additional constraints have to be placed on the implementors to enable functional conformance to be tested.

CGM Minutes: 3 March 1987 2.15pm-4.15pm

Functional conformance is contradictory - requires private encodings and allows for no private encoding in the same section.

To verify functional conformance, additional requirements beyond the standard must be met by the implementors - translation of a private encoding to a clear text encoding.

In theory, functional conformance cannot be tested if we stick to the conformance sections within the standard.

Should conformance sections be modified? If so, how?

Should private encodings be allowed? Since "private" language bindings are not allowed in the functional standards.

The Validation Testing & Registration (VTR) Group in ISO TC97/SC21/WG2 should be involved in conformance sections of all standards.

Overall Strategy for Testing

A) The NCC Model for testing the OSI transport layer (layer 4) was presented

   - check for correct transfer of data (same as CGM testing) (see figure 1)

   - note relationship between OSI testing strategy (and ASN.1) and CGM testing

B) A proposed CGM Test Architecture (see figure 2) (CGM Interpreter)

   Note manual checking still required.

   A Proposed CGM Test Architecture (CGM Generator) (see figure 3)

   Test Centre will test metafiles at their site

   How do we define the metafile which is produced from the IUT?

   Study conformance for CGM Generator testing is a syntactically correct metafile.

   Test Centre will generate pictures for IUT to produce a metafile for a generator. Test Centre can then compare pictures with the pictures interpreted from the metafile generated by the IUT.

How do we create a metafile?

1. Test site chooses - not fair test

2. Give example program e.g.GKS

3. Give picture

4. Give picture descriptors (in English Language)

How many tests are necessary to test an Application Profile (AP) for a CGM (i.e. how many pictures need to be described for a generator?) (we may get guidance from GKS test suite developers at Plenary).

Test Centres will have to contact each implementor (or user) to determine what elements he / she is supporting. Pictures must then the developed to test all those capabilities.

3 + 4 can be used to describe a metafile.

These problems are expanded later in this report.

Issue: How to Select Metafiles for Testing CGM Generators

Description: The problem is we should specify a strategy for giving the implementor guidance in producing metafile output for testing purposes.

Alternatives:

1. Let the Implementor choose an appropriate set of metafiles
2. Describe metafile output in forms of application programs, such as GKS
3. Give actual pictures to the implementor
4. Give precise picture description in some formal form like clear text encoding

pro 1: Less work for implementor and tester

pro 1: Only metafiles within the scope of the application profile are generated.

pro 2-3-4:con 1: Not a fair test for different generators

pro 1: The implementor could provide degenerate metafiles that would pass the test

pro 1: Picture comparisons not possible because picture is not known.

pro 2: Easier for implementor to generate metafiles (if application interface supported)

pro 2: Allows a description of picture by tester

con 2: You have to test both the application + the CGM driver

con 2: The mapping between the application + CGM may not be defined (may be defined for GKS programs in Annex but not for others)

con 2: Picture contents comparison very hard, but possible

pro 2: Possible to compare picture content, but very hard automatically and at operator interface.

pro 3: Not limited to specific application interface (less work for tester)

pro 3: Comparison at operator interface very easy

con 3-4: Lot of work to code picture into metafiles

con 3: Requires an implementor to have good interpreter to compare pictures

con 4: Very restrictive - picture level must be at level of metafile elements (e.g. specified with Z EXTENT)

pro 4: Picture contents of metafile are comparable automatically.

con 4: Isolated test of CGM output driver, not of integrated metafile generator

con 4: Requires detailed descriptions by tester

Pro 1-2-3-4: Allow testing of CGM syntax and consistency

Recommendation: Recommend Alt.2

This can be summarised in the table below:

Criteria

	Alt 1	Alt 2	Alt 3	Alt 4
1. Fairness of test	3	0	0	0
2. Amount of work for Implementor	0	1	3	3
3. Amount of work for Tester	0	2	2	2
4. Syntax checking	0	0	0	0
5. Visual testing at Operator I/F	3	1	0	1
6. Automatic check of picture contents	3	1	3	0

where 0 = Best
3 = Worst

CGM Minutes 3rd March 4.45pm-6.30pm

1. We have to accept the fact that we can't ask an implementor to create a metafile. How to create a metafile is a real problem not easily solved, and we recognise that fact.

   Tomorrow, we will develop an issue format for this problem of creating a metafile, with pros and cons. We could not solve this problem.

2. Using Fig.3, the question became of dividing it into distinct pieces to try and attack the problem. Would the IUT have a generator as well as an interpreter? In other words the IUT would be able to accept the tester's CGM and then generate a metafile back to the tester. Cannot guarantee that an IUT will generate a metafile - it is not mandatory in the standard.

   A. Produce a test metafile and Pass Metafile to IUT.
   With questionnaire response information, the tester should be able to produce a metafile that can be sent to the implementor, who tests it on his system. Where possible the metafiles in the test library will follow the type of scripts in the GKS tests as far as general pictures are concerned. We would then expect the IUT to produce "reasonable" pictures. Good operator test mechanism.

   B. Take Metafile back from IUT
   Is it useful to try and automate this? It will come down to only looking at the picture again. It seemed that the metafile should be tested at the IUT site only. In this case, with (A), have both a generator and interpreter, and then it can't be determined where errors occurred.

Individual tasks for the evening:

(1) Diagram looking at the relationship between elements - Peter and Clemens

(2) The issues on to create a metafile - Martin and Dan

(3) What output might be produced by a test centre? - Jane and Mark

(4) Summarize what done and check minutes - Anne

Minutes CGM Group: Wednesday 4th March

The work was reviewed and a report for plenary was prepared.

Relationship with other Standards

We will have 'application profiles' in the metafile categories of the extended metafile work in ISO.

Do we need to check the mapping of metafile generators to CGM? This would make testing easier.

Should the mapping be in the CGM or GKS Standard?

Generation of CGM by GKS should be in GKS. The interpretation is the mapping of CGM to GKS. Therefore it should be in CGM. Some debate on this.

Need for relationship between Standards needs to be specified. Again another WG2 issue.

Felt that work done at this meeting may be extensible to CGI.

How can we test picture contents automatically in CGM/CGI?

Picture must correspond to Application Profile for both CGM/CGI.

Can set up picture description of primitives and their bound primitives. May be useful for functional standards. Possibly need to use bit-map for 'lower' standards e.g. CGM/CGI. Then you need to rely on operator tests or interface testing 'near' to the screen.

FINAL REPORT

CALS SOW TASK 2.2.3.3.4

CALS APPLICATION PROFILE FOR CGM

TABLE OF CONTENTS

CALS APPLICATION PROFILE FOR CGM

I. PURPOSE

Extend Computer Graphics Metafile (CGM) standard conformance definitions to include **generators** and **interpreters** of metafiles. Currently, conformance requirements in the CGM just refer to the syntax of the metafile, **not** to programs which generate metafiles and read metafiles. Conformance criteria are needed for these programs to ensure that the complete graphical image is ported between devices. This is being accomplished by the creation here of an Application Profile (AP) detailing CALS' use of the CGM.

This document comprises the final specification of the CGM Application Profile for CALS. High compatibility with the MAP/TOP V3.0 CGM Application Profile (Appendix 1) has been achieved. There are some differences however, and these are detailed in this report, along with what is being done about them. This final report concludes with a description of additional work which will be needed in order to advance the Profile to a functionally richer "version 2."

II. BACKGROUND

1.0 Overview of CGM

The Computer Graphics Metafile (CGM) standard, ANSI X3.122-1986 and ISO 8632/1-4, specifies the syntax and semantics of a standard file format for storing and communicating computer graphics pictures. By intentional choice of scope, it limits the specification to the syntax and semantics of a set of CGM "elements" for the device-independent description of computer graphics pictures.

In the year that it has been an ANSI standard CGM's use and its incorporation into other standard interface and exchange specifications has been increasing. There are over two dozen implementations existing or known to be in progress in the US alone (there are more internationally). It has been designated as a Federal Information Processing Standard (FIPS) 128, incorporated as the graphical metafile of the MAP/TOP V3.0 specification, and designated as the Geometric Graphic Content Architecture of the ISO compound document standard (ISO 8613, currently in DIS stage), aka "ODA/ODIF."

2.0 The Need for CGM Application Profiles

The syntactic specification in the CGM standard is complete and

unambiguous. It is, as well, redundant in the sense that there are three distinct encodings of the same functionality: binary, character, and clear text. The redundancy serves a useful purpose, as each encoding is tailored to certain computing environments and applications, and so the CGM client has the opportunity to choose a syntax that is optimized to the intended application.

The semantic specification is less complete. The expected overall results of using the geometric primitive elements are well enough specified. However some of the finer details, such as the precise appearance of joints and endpoints in lines, are unspecified. This underspecification of semantics was intentional on the part of the committees formulating the CGM, since it allows a wider range of existing systems to be accommodated and makes the standard more adaptable to the various needs and philosophies of a diverse clientele.

On the other hand, the semantic ambiguity does mean that there will be no single correct interpretation of a given CGM, and hence it will be difficult to unambiguously describe an intended picture using CGM. This is a distinct drawback in certain application environments. The CGM application areas of Technical Illustration and Technical Publishing, which are central to the CALS effort, clearly comprise such environments where unambiguous semantics are critical.

There are further sources of uncertainty in using CGM in an application environment. A CGM is produced by a component of a graphics environment known as a "metafile generator." The content of a CGM is rendered into pictures by a component known as a "metafile interpreter." The CGM standard specifically excludes standardization of the behavior of metafile generators and metafile interpreters. (Most such behavior is described as "implementation dependent.") In doing so, a certain unpredicability of results is introduced into the graphics system viewed as a whole; for example, CGM generators serving GKS (Graphical Kernel System, ANSI X3.124-1985) clients in the product lines of two different vendors might map out-of-range attributes differently.

These two sources of ambiguity in using CGM--incomplete semantics and non-specification of the behavior of generators and interpreters--do not diminish the utility of the CGM for technical illustration and technical publishing. CGM is a sound and suitable basic protocol for these areas. But they do mean that some further specification (beyond that in the published standard) is required in order for the use of CGM to be effective and unambiguous.

Such a specification is precisely what an Application Profile (AP) consists of. In the case of CGM, an AP can specify:

1. complete semantics;

2. the behavior of CGM generators and CGM interpreters;

3. additionally, an AP can extend the functionality by defining additional parameter values, ESCAPE elements, and Generalized Drawing Primitive (GDP) elements.

Some caution must be taken on the points 1 and 3, to avoid specifications incompatible with anticipated extensions to the standard (via Graphical Registration, the Addendum process, or the normal 5-year review and revision process).

An AP specifies minimal and maximal requirements for generators and interpreters, and ties down all implementation dependencies of the CGM. As the name suggests, an AP is a set of specifications appropriate to a given application environment.

One such AP has already been targeted and substantially completed. It is the CGM Application Profile of TOP (Technical Office Protocol), endorsed by a number of major industrial constituents and incorporated into the MAP/TOP V3.0 specification.

For CGM to be used effectively in the CALS Technical Publishing, Administrative Publishing, and Technical Drawing applications, an AP must be designated for CALS as well.

3.0 Contents of the CGM Standard

A survey of the content of the CGM standard is given in Appendix 2 of this report. It is a copy of an article which appeared in the August 1986 issue of the magazine IEEE Computer Graphics and Applications.

4.0 Objectives

4.1 Scope of the CALS Application Profile

There are two categories of specification that should be considered in an Application Profile of CGM:

1. resolution of ambiguities in the metafile and in the behavior of generators and interpreters;

2. extension of the CGM functionality to handle perceived functional deficiencies in the standard.

Any Application Profile must accomplish the first task. The

second task is important for CALS constituents as well, and could be handled by definition of ESCAPE and GDP elements, as well as additional parameter values of existing elements. Presumably such extended CGM functionality would be submitted for Graphical Registration. The CGM is functionally lean when measured against the requirements of automated publishing and technical illustration. Almost any picture from these application areas can be represented. For example lines and areas can be used to represent in the CGM many of the "higher-level" entities of IGES. But the consequence of simulating the entities with very primitive geometric elements is a loss of efficiency and data compaction in the CGM.

In order to be an efficient picture mechanism in the CALS environment, extension of the functional capabilities of CGM is necessary. Such extension is taking place formally now within ISO (the Extended Metafile, Addendum 1 to CGM, and CALS requirements are being injected into this development process). Unfortunately, even the "fast-track" ISO addendum process is a slow process.

Another NBS CALS SOW Task for FY 87, 2.2.2.2.2, entitled CGM Registration for CALS Requirements, has been completed. Its purpose was to:

1. identify needed extensions to CGM, and;

2. prepare and submit Graphical Registration proposals for same.

However, NBS believes that it is (with a few exceptions) inadvisable to include many of the proposed extensions in the current CALS Application Profile. Doing so would encourage their immediate use. While they are useful and needed functionality, they will be examined and likely modified by graphics standards bodies before they have official standing in the Standards arena. Implementations which use the proposals too early in the registration process would likely be non-standard when the proposals eventually complete processing.

This Application Profile for CALS will therefore not include extended functionality, with the following exceptions:

1. the published TOP profile contains two specified ESCAPES, one of which is an encoding of a function that is stable in ISO CGI and in the CGEM (ISO Extended Metafile);

2. the additional linetypes and hatch styles detailed in the CGM Registration report referred to above.

In summary, the CALS CGM Application profile:

1. will specify semantics and syntax that are ambiguous or unspecified in CGM;

2. will not, except as noted, specify extended functionality for CGM.

With regard to point number 2, it must be emphasized again that these extended functionalities are important to the CALS community. The problem at this time is the immaturity of the proposals and the immediate need for an initial AP for the CALS community. Extended functionalities should be included in a "Version 2" upgrade of the CALS AP in the near future; that is, as soon as the work of the ISO and Graphical Registration committees has progressed far enough.

4.2 Relationship to the TOP Profile

Proliferation of "dialects" of CGM is clearly undesirable and contrary to the interests of US industry. In fact, occurrence of such private dialects is directly contrary to the purpose of standard interface specifications such as the CGM and would destroy much of the benefit to be gained by using such a specification.

Accordingly, the highest priority objective of this task was to realize a CALS application profile that is either identical to, or is downwardly compatible with, the TOP AP. In other words, where the APs overlap they should be identical, but CALS may be somewhat richer or may go further in specifying constraints.

Fortunately there is significant common interest and shared requirements between the sectors of industry represented on the MAP/TOP committees and the clientele of the CALS initiative, particularly in the areas of technical illustration and compound document exchange. This means that accomodation and convergence of the profiles (implying changes to both drafts) should be achievable.

4.3 Specific Goals of the Application Profile

Other specific objectives to be achieved in the specification of the CALS CGM AP include:

1. A CALS metafile must be a legal CGM; that is, CALS syntax must be a subset of CGM syntax and CALS semantics must be legal CGM semantics. This means, for example, that the CALS environment cannot assume or specify implicit element defaults that differ from the CGM standard.

2. The picture specified by a CALS metafile should be unambiguous. This means, for example, that private values of attributes (such as private linetypes) cannot be allowed, and private elements (private escapes and GDPs, for example) must be prohibited.

3. The behavior of generators in producing a CALS metafile should be specified so that identical sequences of activity at the application level result in identical metafile contents (intermediate layers in a graphics environment may complicate this).

4. The behavior of interpreters in parsing and rendering CALS metafiles should be as unambiguous as possible. This means that such things as fallback actions when the interpreter lacks capability, or fallback actions in the face of geometric degeneracies, should be specified.

5. The format ambiguities of the CGM, such as the "record size" of the binary encoding (unspecified in CGM) should be specified.

6. The CALS CGM AP should be rich enough to accomplish useful things economically.

7. The AP should be formulated with awareness of the evolution of graphics standards. In particular, the content of the Extended Metafile (ISO 8632 Addendum 1, the first set of extensions to CGM, currently near DP stage) should be carefully followed. No specifications should be made in the CALS AP which compromise compatibility with these standards activities.

8. Similarly the activities of the Graphical Registration process (including CALS requirements submitted via the work done under CALS SOW task 2.2.2.2.2) must be tracked. Future compatibility must again be protected.

9. A CALS metafile should be self-identifying as such.

In some cases, these criteria will be mutually contradictory, which means that it will not necessarily be possible to satisfy all of them at once.

5.0 The TOP Profile and CALS

As stated above, the highest priority in the specification of the CALS profile is to avoid proliferation of dialects of CGM. This was recognized in the work breakdown for this project, which specified these particulars:

- o Analyze the TOP Application Profile with respect to its suitability for CALS;
- o Analyze CALS requirements for CGM interpreters and generators;
- o Either recommend adoption of the TOP Application Profile as is, or negotiate with TOP on a compromise Profile acceptable to both TOP and CALS;
- o Deliver to DOD the recommended Application Profile for CALS.

Accordingly, emphasis in this task was placed on:

- critically reviewing the proposed TOP AP;
- establishing a working relationship with TOP graphics representatives;
- preparing comments and suggestions for changes to TOP with the technical review by and endorsement of ASC X3H3 graphics experts;
- presenting these to TOP during their comment and review period;
- and following through with the cooperative effort to refine the proposals and the resulting TOP profile.

5.1 Historical Overview

When this project commenced, the TOP review process (on the CGM Application Profile for MAP/TOP V3.0) was less than two weeks from completion (January 1987 was the scheduled closing date). Although the draft profile had been available for some time, few graphics experts had taken the considerable time required to carefully read and critically review it, since at that time it was 30+ pages of fairly dense and terse technical material.

The first result NBS obtained was an agreement to extend the review period until 10 February 1987, a week after the close of the January 1987 ASC X3H3 working meeting in Ft. Collins, CO. Prior to that meeting, the TOP AP was studied intensively. The Profile was found suitable for CALS application requirements in stated purpose, direction, and intent. But there were numerous problems in the draft proposal.

Fortunately, many of what at first appeared to be serious technical problems were due to editorial and organizational

problems, as the Profile drafted attempted to replicate much of the technical material of Part 3, the Binary Encoding, of CGM. In this transcription process many errors (some substantial) were introduced. A plan was devised to reorganize the Profile, and this was pursued at the Ft. Collins X3H3 meeting.

In addition, there were a number of technical problems:

1. The concept of conformance, particularly "Basic Conformance," was not clear.

2. Ambiguity was introduced by not prohibiting private values, e.g., private linetypes, in a Basic CGM (although it was not clear if this was intentional or an editorial problem).

3. There were a number of minor problems with allowable precisions, datatypes, etc.

4. There were a number of proposed ESCAPES and GDPs which NBS thought should be withdrawn. Some were not of broad enough interest; some needed to be reformulated with somewhat more deliberation and input from other graphics experts; and the encoding of Data Records for all required improvement.

5. There was some confusion as to whether the Profile specified implicit element defaults different from CGM. If this were true then a TOP metafile would not be a standard CGM.

6. The definitions of the predefined bundle tables (for interpreters to use) needed adjustments.

Proposals were formulated to deal with the editorial and technical problems. These were considered and improved upon by an ad hoc working group, which included CALS, X3H3, and TOP proponents, at the Ft. Collins meeting of X3H3 in the last week of January 1987.

The output of this effort was a list of corrections to the initial parts of the profile, and a redrafting of a key 23-page technical section into 6 concise pages. These changes were presented to X3H3 plenary for endorsement, and sent forward to the TOP committee under the cover letter of the X3H3 committee chair. This packet is included in this report as Appendix 3.

In addition to this liaison statement from X3H3, 11 members of X3H3 (suppliers and consumers in the computer graphics industry) sent individual TOP Comment forms endorsing the results of the Ft. Collins meeting.

The comments were well received by the TOP experts, since they represented significant editorial and technical improvement

without changing the basic emphasis or intent of the profile. Most of the recommended changes were incorporated.

In March NBS had an opportunity to review the complete revised profile. A number of inconsistencies and oversights were caught. These were dealt with by TOP experts and X3H3 experts informally, and carried into the next version of the TOP profile as editorial changes.

Study of the final TOP AP still reveals a number of minor problems. Some of these are editorial. Some are oversights--areas of specification that would be useful but were not considered in time for the publication of this version of the TOP profile.

At the "CGM in the Real World" workshop sponsored by NBS and Eurographics, held in Washington DC in September 1987, a number of these items were discussed between TOP and CALS graphics experts. There were agreed changes to both profiles as a result. It appears that the two profiles will converge and be nearly identical in areas of substance.

III. DEFINITION OF THE CALS APPLICATION PROFILE FOR CGM

1.0 Conformance

The CALS CGM AP specifies conformance in terms of "permissible" and "basic" values. Permissible values are the range of values of CGM elements as specified in ISO 8632. Basic values are a subset of the permissible values and they constitute the "Basic Set." For example, permissible values of MARKER TYPE include all non-zero integers, while basic values include the standardized enumerated values 1 to 5.

CALS defines a conforming basic metafile to be one that contains no elements or parameters outside of the Basic Set. CALS defines a conforming basic generator to be one that produces only conforming basic metafiles (or can be reliably commanded to function in that mode), and additionally conforms to any additional generator requirements in this profile.

CALS defines a conforming basic interpreter to be one that at least correctly interprets any conforming basic metafile, and conforms to any additional interpreter requirements specified in this profile. In addition, any conforming CALS interpreter should be able to parse and skip any elements that it doesn't understand or support, and any parameter values that it does not support.

For interpreters, there are two levels of conformance for the judging what comprises "correct" interpretation of a metafile: minimal level and publication level.

Publication Level: all of the elements specifications of the CGM and this application profile shall be accurately implemented. This includes the specifications of CGM annex D.2 and D.5, and the treatment of indeterminate specifications of circular and elliptical primitives in D.4.5. The results should be completely predictable across implementations conforming at this level; that is, suitable for publication as the name implies.

Minimal Level: the specifications of CGM annex D.2 (degeneracies) and D.3 (mapping color to black-and-white) shall be implemented. The treatment of indeterminate specifications of circular and elliptical primitives in D.4.5 shall be followed. The capabilities of annex D.5 of CGM and of the Basic set as defined in this profile shall be present. However the following interpreter fallback actions of D.4 may be taken:

AUXILIARY COLOR
APPEND TEXT
RESTRICTED TEXT
CELL ARRAY
LINE TYPE
LINE WIDTH
MARKER TYPE
MARKER SIZE
TEXT PRECISION
CHARACTER EXPANSION FACTOR
CHARACTER SPACING
CHARACTER HEIGHT
CHARACTER ORIENTATION
CHARACTER SET INDEX
FONT DESIGN
HATCH INDEX
EDGE TYPE
EDGE WIDTH
PATTERN SIZE

CGM (ISO 8632) specifies three encodings of the abstractly specified metafile functionality: binary, character, and clear text. This application profile is an application profile for the CGM Binary Encoding, ISO 8632/3. Future application profiles may be developed (or this profile extended) for the other encodings of CGM.

2.0 Metafile Constraints

The Basic Set is defined by the limitations on Basic Values noted below. Where an element is not mentioned, it is implied that the Basic Set includes all values permitted in the CGM.

2.1 Delimiter Elements

TABLE 1. Delimiter Element Constraints

Element	Basic Values
no-op	An arbitrary sequence of n octets, n=0..32767.

2.2 Metafile Descriptor Elements

TABLE 2. Metafile Descriptor Element Constraints

Element	Basic Values
Metafile Description	(Note 1)
Integer Precision	16
Real Precision	(1,16,16) (fixed) (0,9,23) (floating point)
Index Precision	16
Colour Precision	8, 16
Colour Index Precision	8, 16
Font List	(Note 2)
Character Set List	(0,4/2) (Note 3) (1,4/1) (Note 4)
Character Coding Announcer	0 (Basic 7-bit) 1 (Basic 8-bit)

Note 1: The Metafile Description element's string: should include a substring briefly identifying company or product, so that interpreters can account for known idiosyncrasies of generators; shall contain the substring "CALS/BASIC-1".

Note 2: The character set is ANS X3.4, 7-bit American National Standard Code for Information Interchange (7-bit ASCII).

Note 3: The character set is ANS X3.134/2, 8-bit American National Standards Code for Information Interchange (8-bit ASCII). This is equivalent to ISO 8859/1, Right-Hand Part of Latin Alphabet Number 1.

Note 4: Four simultaneous fonts are supported. The font names are selected from the basic font names in Section 6 below.

2.3 Picture Descriptor Elements

Note that the scale-factor parameter of SCALING MODE is always a floating point number, even when REAL PRECISION has selected fixed-point for other real numbers. This is not an error--a floating-point parameter is needed for some situations where low precision fixed-point reals would not encompass the range at all (for example, scaling a plot done with 32-bit integer coordinates onto a 1-meter piece of paper) and for other situations where using a fixed-point scale factor would produce unacceptable loss of resolution. It is not apparent in the CGM standard what the precision of this floating point parameter is when fixed point reals have been selected: its precision is (0,9,23).

2.4 Control Elements

TABLE 3. Control Element Constraints

Element	Basic Values
VDC Integer Precision	16, 32
VDC Real Precision	(1,16,16) (fixed) (0,9,23) (floating point)

2.5 Graphical Primitives

To ensure portability and predictable results, CALS metafiles may contain only those GDP elements that are defined in CALS profiles. This revision of the profile does not contain any such GDPs.

2.6 Attribute Elements

TABLE 4. Attribute Element Constraints

Element	Basic Values
Line Bundle Index	1-5
Line Type	1-5 (Note 1)
Marker Bundle Index	1-5
Marker Type	1-5
Text Bundle Index	1-2
Text Font Index	1-4
Character Set Index	1-2
Alternate Character Set Index	1-2
Fill Bundle Index	1-5
Hatch Index	1-6 (Note 2)
Edge Bundle Index	1-5
Edge Type	1-5
Pattern Table	Starting Index, 1-8 nx, 1-16 ny, 1-16
Colour Table	start index 0-255

Note 1: Additionally, the linetypes (and edge types) defined in section 3.1, which have been submitted for Graphical Registration, are in the Basic Set of this profile.

Note 2: Additionally, the hatch styles (indexes) defined in section 3.2, which have been submitted for Graphical Registration, are in the Basic Set of this profile.

2.7 Escape Element

To ensure portability and predictable results, CALS metafiles may contain only those ESCAPE elements that are defined in section 7.0 below.

2.8 External Elements

The 'action required' flag of the MESSAGE element is restricted to the value 'no action required'.

3.0 Additional Attribute Values

3.1 Linetypes

The following additional linetypes were specified for CALS in the

Final Report for CALS SOW Task 2.2.2.2.2, and have been submitted for graphical registration. Refer to that report for a complete definition. The name of the linetype is given, followed by the numeric value (the linetype parameter) by which it shall be referenced prior to registration.

TABLE 5. Additional CALS Linetypes

linetype	CGM parameter value
chain line	-11301
center line	-11302
hidden line	-11303
phantom line	-11304
double arrow	-11305
single dot	-11306
single arrow	-11307
stitch line	-11308

3.2 Hatch Styles

The following additional hatch styles were specified for CALS in the GSC report, and have been submitted for graphical registration. Refer to that report for a complete definition. The name of the hatch style is given, followed by the numeric value (the hatch index parameter) by which it shall be referenced pending registration.

TABLE 6. Additional CALS Hatch Styles

hatch style	CGM parameter value
across grain wood	-11401
with grain wood	-11402
bronze, brass, copper, and compositions	-11403
cast iron or malleable iron and general use for all materials	-11404
steel	-11405
concrete	-11406
cork, felt, fabric, leather, and fiber	-11407
earth	-11408
magnesium, aluminum, and aluminum alloys	-11409
marble, slate, glass, porcelain, etc.	-11410
rock	-11411
rubber, plastic, and electrical insulation	-11412
sand	-11413
sound insulation	-11414
thermal insulation	-11415
titanium and refractory material	-11416
water and other liquids	-11417
white metal, zinc, lead, babbitt, and alloys	-11418

4.0 CALS CGM Defaults

The CGM specifies a complete set of defaults. In some cases, these defaults do not match CALS application requirements. However, any CALS metafile must be a legal CGM, including implicit defaults, thus each deviation requires that the affected element either:

1. Appear in the METAFILE DEFAULTS REPLACEMENT element, or

2. be explicitly specified for its value to be applicable.

Therefore, each CALS metafile shall contain in the Metafile Descriptor a METAFILE DEFAULTS REPLACEMENT element that includes (at a minimum):

TEXT PRECISION element; precision 2 (stroke).

Each CALS metafile shall also contain in the Metafile Descriptor the CHARACTER SET LIST element, with the first two indices set to (0,4/2), (1,4/1).

The CGM leaves the definition of the default color table implementation dependent. Sections 8.2.1 and 8.2.2 specify how conforming CALS generators and interpreters shall initialize their color tables. This removes any implementation dependencies in color selection while in a closed CALS environment. While it is not required by this profile, generators may include this color table in the defaults replacement. This will give predictable results even in those cases where the metafile may leave the CALS environment (for color at least -- there are other environment dependencies which cannot be resolved in this way).

5.0 Specification of Semantic Ambiguities

The CGM leaves the semantics of a number of graphical details unspecified or "implementation dependent." This is unacceptable where predictable interchange is required. The following specifications are made for CALS generators and interpreters:

5.1 View Surface Clearing

The view surface shall be cleared at the beginning of each picture (see however the section on ESCAPE elements).

5.2 Clipping

Clipping shall be done to the intersection of the viewport and the device view surface limits when the clipping indicator is

'off'. Clipping shall be done to the intersection of the clip rectangle, the viewport and the device view surface limits when the clipping indicator is 'on'.

5.3 Linetype Continuation

Linetype shall be maintained (continued) across the interior vertices of a polyline.

5.4 Edge Type Continuation

Edge type shall be maintained (continued) across the vertices of a filled area boundary.

6.0 CGM Font Specifications

The fonts in Table 7 are public domain fonts, available from NTIS # PB251845 (NBS Special Publication 424, April 1976). All of these fonts are considered to be basic capabilities of a CALS conforming basic metafile. Any of these fonts may appear in the Font List element in a CGM that conforms to this AP. The font names are specified in a manner compatible with ISO 9541, Font and Character Information Interchange. The font name (Font Identifier for Base Font) is a concatenated string of the Universal Font Name and a User Readable Font Name. The User Readable Font Name is the concatenated string "HERSEY:" to designate one of the Hersey fonts, and "name string" to designate the particular typeface.

It is recognized by CALS that the Hersey fonts may not be of adequate quality for modern publication requirements. The CALS profile considers any rendering of a requested font conforming if the rendering is "metrically identical" to the font metrics of the requested font. This means that the placement and alignment of the string and the placement, size, and shape of individual characters (i.e., the drawn portions of the character cells) are measurably identical. This would allow a good quality filled font to be substituted for a stroked Hersey font, for example.

Finally, the Hersey "fonts" are really a mixture of fonts and character sets (e.g., Greek is a character set). The requirements of the CALS profile are served by providing that the necessary character sets be supported in part, and the necessary typefaces be supported in part, so that the combinations required to render the listed 16 Hersey "fonts" are supported in full.

TABLE 7. Basic Font Names

1.	HERSEY:CARTOGRAPHIC_ROMAN
2.	HERSEY:CARTOGRAPHIC_GREEK
3.	HERSEY:SIMPLEX_ROMAN
4.	HERSEY:SIMPLEX_GREEK
5.	HERSEY:SIMPLEX_SCRIPT
6.	HERSEY:COMPLEX_ROMAN
7.	HERSEY:COMPLEX_GREEK
8.	HERSEY:COMPLEX_SCRIPT
9.	HERSEY:COMPLEX_ITALIC
10.	HERSEY:COMPLEX_CYRILLIC
11.	HERSEY:DUPLEX_ROMAN
12.	HERSEY:TRIPLEX_ROMAN
13.	HERSEY:TRIPLEX_ITALIC
14.	HERSEY:GOTHIC_GERMAN
15.	HERSEY:GOTHIC_ENGLISH
16.	HERSEY:GOTHIC_ITALIAN

7.0 Escape Elements

Support of the following ESCAPE elements is required in CALS conforming implementations.

7.1 Disable Clearing of View Surface

The normal interpretation of a CGM is such that the view surface of a device is cleared on each Begin Picture Body element. This Escape element will disable the clearing of the view surface for all of the pictures in the metafile. The effect of this Escape element is to permit multiple metafile pictures to be imaged on the same view surface with a mapping as described in the CGM standard. The pictures may have different VDC Extents. Each picture will be mapped into the current device viewport (whether default or specified by the Device Viewport Escape element). If used, this Escape element must appear in the Metafile Descriptor. This Escape element is a basic capability of this profile.

Escape Identifier: -301

Escape Data Record: null

7.2 Device Viewport

The default device viewport for interpreting a picture in CGM is the largest rectangle which maintains the aspect ratio of the VDC Extent. This Escape element redefines the device viewport for the picture to some portion of the available view surface. If used, it must appear in the Picture Descriptor. The

specification units for the viewport are real fractional [0.0,1.0] -- the fraction is applied to the default device viewport. If the scaling mode has been set to metric, then the device viewport has precedence -- the scaling mode is ignored.

Escape Identifier: -302

Escape Data Record: A single string of text containing the specification of the viewport. Parameters in the viewport are separated by at least one blank character and/or a single comma character. The decimal point of the real fraction is required. Leading zeroes of the real fraction are optional. There are four parameters:

P1: First corner x-coordinate. Real fraction of the default device viewport, in the range [0.0,1.0].

P2: First corner y-coordinate. Real fraction of the default device viewport, in the range [0.0,1.0].

P3: Second corner x-coordinate. Real fraction of the default device viewport, in the range [0.0,1.0].

P4: Second corner y-coordinate. Real fraction of the default device viewport, in the range [0.0,1.0].

Example: a viewport equal to the upper right quarter of the default viewport could be coded as:

Escape Identifier -301

Escape Data Record ".5 .5 1. 1."

This Escape element is a basic capability of this profile.

8.0 CGM Implementation Dependencies

This section specifies implementation dependencies and environmental constraints for this AP.

8.1 General Guidelines for CGM Elements

Unless otherwise noted in this application profile, the guidelines of CGM Annex D shall be treated by CALS generators and

interpreters as specified in section 1.1.

Name: Metafile Defaults Replacement

Description: The Metafile Defaults Replacement element shall not be partitioned. Note that the CGM standard permits multiple occurrences of this element, so that partitioning should not be required.

Name: Restricted Text
Description: Minimal capability of a basic conforming CALS interpreter shall be to render the complete restricted text string (including appended text), scaled isotropically (i.e., specified aspect ratio for the text is not distorted) such that the string fits into the text extent parallelogram.

Name: Color Table

Description: The Color Table element has an unspecified effect when it appears in a picture subsequent to any graphical primitives. The Color Table element should appear prior to any graphical primitive elements to insure that interpreting systems without dynamic color update capabilities can render the intended effect.

Name: Pattern Table

Description: The Pattern Table element has an unspecified effect when it appears in a picture subsequent to any graphical primitives. The Pattern Table element should appear prior to any graphical primitive elements to insure that interpreting systems without dynamic pattern update capabilities can render the intended effect.

8.2 Implementation Requirements for Generators and Interpreters

The specifications in this section augment those of ISO 8632/1 annex D.5 and ISO 8632/3 clause 8.

8.2.1 Additional Generator Specifications

This CALS AP specifies that the values of attributes (e.g., linetype) are restricted to a certain set. When a CALS generator receives (from the application or graphics system client) a value outside of the Basic set, it should be handled as follows:

--If the index is selecting an attribute (e.g., linetype), then the CGM generator should map it by MODULO onto the Basic range;

--If the index is defining an attribute (e.g., color table), then it should be ignored if outside the Basic range.

These choices for the generator are consistent with annex D of CGM.

In the absence of specific application requirements to the contrary, CALS metafile generators shall initialize their color tables as described in the next section.

8.2.2 Additional Interpreter Specifications

In the absence of any color table elements (in either the defaults replacement or the body of the metafile) in a metafile, CALS interpreters shall initialize their color tables as follows: the starting color index is set to 2 and the remaining 254 entries are a repetition of the following 8 entries:

TABLE 8. Default Color Table

Index	Values	Meaning
2	(255,0,0)	Red
3	(0,255,0)	Green
4	(0,0,255)	Blue
5	(255,255,0)	Yellow
6	(255,0,255)	Magenta
7	(0,255,255)	Cyan
8	(0,0,0)	Black
9	(255,255,255)	White

Color table defaults for colour indices 0 and 1 are defined in the CGM standard as corresponding to the nominal background and nominal foreground colours, respectively.

8.2.3 Minimum Data Structure Support

Name: Maximum Color Array Dimension

Description: The basic value for the number of color values that can appear in a color array or color list parameter. CELL ARRAY and PATTERN TABLE have color array parameters and COLOR TABLE has a color list parameter.

Basic Value: 1048576 for CELL ARRAY (one 1024x1024 image);
2048 for PATTERN TABLE (eight 16x16 patterns);

256 for COLOR TABLE (entries 0-255).

Name: Maximum Point Array Length

Description: The basic value for the number of points and VDC that can appear in parameters for metafile elements.

Basic Value: 1024

Name: Maximum String Length

Description: The basic value for the length of an individual string of characters.

Basic Value: 256 for all string parameters except data records; 32767 for data records.

Name: Bundle Table

Description: Bundle representations are not settable in the current version of the CGM. To insure predicatable results, CGM interpreters and generators conforming to this profile shall use the following default bundle tables.

Basic Value: See Table 9.

TABLE 9. CALS Default Bundle Tables

Bundle Type	Bundle Index 1	2	3	4	5
Line Bundle					
Line Type	solid	dash	dot	dash-dot	dash-dot-dot
Line Width	1	1	1	1	1
Line Color	1	1	1	1	1
Marker Bundle					
Marker Type	dot	plus	asterisk	circle	cross
Marker Size	1	1	1	1	1
Marker Color	1	1	1	1	1
Text Bundle					
Font Index	1	1			
Text Precision	stroke	stroke			
Character Expansion Factor	1	0.5			
Character Spacing	0	0			
Text Color	1	1			
Fill Bundle					
Interior Style	hatch	hatch	hatch	hatch	hatch
Fill Color	1	1	1	1	1
Hatch Index	1	2	3	4	5
Pattern Index	1	1	1	1	1
Edge Bundle					
Edge Type	solid	dash	dot	dash-dot	dash-dot-dot
Edge Width	1	1	1	1	1
Edge Color	1	1	1	1	1

8.2.4 Metafile Transfer Format

Operating system dependencies for file formats can often be more of a burden for interoperability than differences in interchange formats. To ensure CGM interoperability some conventions are required for exchanging metafiles.

For transfer purposes the CGM shall consist of fixed length 80 octet records. If the transfer medium is magnetic tape, the 80-octet logical records should be blocked into 800-octet physical records.

In addition, the bit/octet/word order of section 4.3 of Part 3 of

CGM (ISO 8632/3) is the correct order for interchange between conforming CALS implementations.

8.2.5 Error Processing

A CALS conforming interpreter should gracefully recover from any exception condition. If there is something which is not understood by the interpreter, then if possible that element should be skipped, appropriate error warnings generated or logged, and interpretation continue with the next element following the problem element.

8.2.6 The Use of OSI Data Transfer Services

To transfer a CGM file between two CALS systems the services provided by either FTAM (File Transfer, Access and Management) or MHS (Message Handling System) can be used. Remote access to part of a CGM file is not addressed at this time.

8.2.6.1 Using FTAM to Transfer Metafiles

One should specify the CGM file Document Type entry number as NBS-1 or FTAM-3, Document Type Name as '{ISO standard 8571 document type (6) unstructured binary (4)}' for Contents-Type-Attribute or Contents-Type-List parameters. The contents of the CGM file should be mapped onto a sequence of octet strings. The boundary of octet strings has no significant meaning.

Note: FTAM does not provide a standard document type for a CGM file. Therefore the Presentation Layer can not be fully used and it is left up to the user or application programs that remotely access using FTAM to know that a given file contains CGM formatted information.

8.2.6.2 Using MHS to Transfer CGM Files

Specify the Body as USABodyPartsBodyPartNumber'7'. The contents of the CGM file shall be mapped on to the body of an IPM (Inter-Personal Message) as a sequence of octet strings. The boundary of the octet strings has no particular meaning.

IV. RECOMMENDATIONS AND IMPACTS

1.0 Differences between the TOP and CALS APs

At this point there are a number of differences between the CALS and MAP/TOP V3.0 profiles. These are expected to be eliminated in the revision and review cycle of the TOP profile. Collaborative development between TOP and CALS graphics experts has led to apparent consensus on a number of changes to the TOP profile. The substantive differences are:

1. The Metafile Description element of a CALS metafile contains the substring CALS/BASIC-1.

2. The CALS profile allows both floating and fixed point reals and real VDC. TOP currently allows only floating.

3. In the CALS profile, the implicit default viewport (for ESCAPE -302) is the largest area of the available view surface that has the same aspect ratio as the implicit default VDC EXTENT. The latter is square, so the implicit default viewport is the largest square area of the available view surface. The specification of isotropy is missing from TOP (but believed to be intended).

4. The data structure support for Color Table in CALS is 256. TOP has 254 (believed to be an error).

5. The data structure support for Pattern Table in CALS is 2048, eight 16x16 patterns. TOP has 1024, four 16x16 patterns (believed to be an error).

6. As specified in the TOP profile, the only fonts available for predictable interchange are the Hersey fonts. The CALS profile considers any rendering of a requested font conforming if the rendering is "metrically identical" to the font metrics of the requested font.

7. CALS has added a number of additional linetypes and hatch styles. It is not known if these are appropriate for the TOP constituents and whether TOP will adopt them. They have been submitted for graphical registration.

8. CALS adds a number of additional requirements for generators and interpreters. In particular, the default color table of TOP becomes a generator and interpreter conformance requirement in CALS.

9. CALS specifies two conformance levels for interpreters, minimal and publication.

2.0 Recommended Future Work on the CALS Profile

2.1 Extended Functionality

Significant functional deficiencies have been pointed out in CGM, when it is considered for **efficient** use in technical publishing and technical drawing applications. Other CALS studies have listed the deficiencies and proposed solutions through graphical registration. Some of these are fairly non-controversial. Some, such as the text proposals, may be controversial and may be at variance with some solution proposals in the graphics standards community.

The controversial proposals need further detailing and analysis. The non-controversial ones need to be expedited through the registration process (and into the standards themselves) and then incorporated into the CALS Application Profile when they are stable. They should **not** be incorporated until they appear to be fairly stable. Sometime in the next year the second revision of this profile should begin, and it should continue as a collaborative effort with the MAP/TOP community.

The user defined linetypes and hatch styles, presented at the end of this report, should be submitted for registration.

2.2 Fonts

A good and useful set of mandated fonts is the highest priority functional extension for the next year. There should be serious consideration given to developing a good high-quality public domain font set to replace the Hersey fonts. The Hersey fonts are a precedent for this proposal.

At the least, there should be investigation of the possibility of metric specification of fonts, following the work of ISO 9541, to allow widespread and uniform implementation of a set of high-quality fonts. A priority is to accomplish this free of any proprietary claims of current patent or copyright holders.

2.3 Conformance

At least two levels of conformance for interpreters are needed by CALS. The number may be higher but this has not been ascertained. More levels lead to confusion in the market place. It should be determined whether one or more intermediate conformance levels are needed.

2.4 Encodings

A single encoding has the advantage of low implementation cost and thus reduces the barrier to adoption of CGM technology into CALS environments. There is some controversy over whether the

binary encoding is adequate to serve the CALS needs. There are two issues: data compactness and network communications. Little quantitative information exists on these questions. This information should be generated and the question of encodings reconsidered during the next year.

2.5 File Format

Early implementation experience, as revealed at the "CGM in the Real World" workshop, indicates some confusion over the meaning of the fixed length 80-octet file format for binary files. There are alleged to be some system-dependent problems with achieving this. Some pre-TOP CGM implementors chose a continuous byte stream as the file format. This in turn cannot be handled in some language/operating system environments. This question should be studied and the profile adjusted and enhanced (if necessary) during the next year.

3.0 User Defined Linetype and Hatch Style Proposals

During study of the TOP profile and the CGM registration task, the need for user defined linetype and hatch style elements became apparent. As part of this project general purpose and flexible formulations of these were developed. These are presented here. They should be submitted for graphical registration as linetypes with corresponding ESCAPEs.

3.1 User Defined Linetype

This element defines a linetype and associates it with an index for future reference:

Parameters:

linetype (IX) - index of linetype being defined;
number of dash elements (I) - number of elements in the defined line pattern;
list of dash elements (nI) - I>=0, n>=1

n=1 means a solid line;
I=0 interpreted as a dot;
First element is a dash, second a space, etc;

Additional parameters (or ESCAPE attributes):

duty cycle unit selector = (VDC, mm, native device units,abstract)
the value of 'abstract' indicates that the implementation may normalize and map the sum of the dash pattern elements at its discretion.
duty cycle (repeat length in units of '..selector')
These two controls define the length of the dash pattern.
adaptive flag = (no, yes) - an "adaptive" linetype is one

where every vertex falls on an inked portion of the line. This is accomplished in plotters by temporarily modifying the duty cycle for each line segment (ceiling function) such that there is always an integral number of repeats (and all predefined linetypes have their gaps_array defined such that they begin and end with inked or "pen down" portions). Default is "no" or non-adaptive, so that the duty cycle is always the same regardless of line segment length, unless the user requests otherwise.

3.2 User Defined Hatch Style

This element defines a hatch style and associates it with an index for future reference:

Parameters:

hatch index (IX) - index of hatch style being defined;
style indicator (E) - {parallel, crosshatch};
number of hatch elements (I) - number of elements in the defined hatch style;
list of hatch elements (nI) - I>=0, n>=2
the array gives alternating (line width, gap width) - a direct analogy to the linetype array. Center of the first hatch line is matched up with PATTERN REFERENCE POINT, if implemented. 0 interpreted as thinnest line width available. Error if sum of hatch elements is 0.

Additional parameters (or ESCAPE attributes):

units indicator = {VDC, mm, device units, abstract}
specifies the units in which 'angle' and 'duty cycle length' are specified. Also controls the manner of transformation of the hatching: If VDC, then the hatching transforms with segment transform and anisotropic transforms (as if you had done POLYLINES); otherwise, the hatching is like "wallpaper" that shows through the polygon-shaped hole - you've mapped all that's necessary into device units and are doing hatching in device space. The value of 'abstract' indicates that the implementation may normalize and map the sum of the dash pattern elements at its discretion.
angle (dx, dy)* - default is horizontal;
duty cycle (repeat length)
Specifies the distance measured perpendicular to the hatch line. The sum of hatch elements in the hatch element list is normalized to this distance before presentation of the hatch on the view surface.

V. SUMMARY AND CONCLUSIONS

The TOP Profile was not suited to CALS requirements at the start of this effort, so most of the efforts for this NBS CALS Task were expended studying and recommending changes in the TOP Application Profile of CGM.

Collaborative efforts with TOP graphics experts have been ongoing throughout most of 1987, right up through the "CGM in the Real World" workshop (Sept 1987, a joint NBS/Eurographics workshop). Adjustments made to the TOP profile (and further adjustments pending) created an adequate basis for an initial specification of a CALS profile. As part of another NBS CALS Task, functional extensions to CGM have been recommended, by the process of Graphical Registration of ESCAPE and GDP elements.

A number of these are reasonably stable: specifically the proposals for additional hatch styles and linetypes. These are adopted into this CALS AP as "private" linetypes and hatch styles, pending completion of graphical registration. Many other of the extension proposals need close examination by graphics experts.

When this process has completed or at least stabilized, this initial CALS profile should be amended or extended to include such functionality. Similarly, CALS should begin adopting Extended Metafile (ISO Addendum 1 to CGM) functionality referenced in the NBS study on needed extensions (e.g., symbol library facilities) as such stabilizes.

The most serious functional deficiency is the lack of text fonts of decent quality. The work of ISO draft standard 9541 should be followed closely here. Future work for CALS should focus on a method of specifying an adequate and useful set of fonts in a way which relieves conformers to the profile of obligations to holders of copyrights on some of the more useful fonts.

Finally, it may prove desirable that all CGM encodings be available as conforming CALS interchange. This does create a greater implementation cost for conformers to the profile. Hence before this is mandated, the results of using only the binary encoding should be evaluated; and the properties of the encodings should be carefully studied to ascertain whether the claims of the CGM standard are valid.

APPENDIX 1

MAP/TOP V3.0 CGM APPLICATION PROFILE

Chapter 6 Graphics Specifications

The computer graphics community is faced with a number of interface and interconnection choices and a number of different ways of implementing those choices. Device-independence guarantees that a diverse set of computer graphics hardware can be used in a manner transparent to the operator. It is in the interest of the end-user to have standards specified for all the interfaces. Failure to include any of the interfaces will only result in further proliferation of non-standard practices and will lessen the benefit of specifying any standard.

The selection of such standards is not simply a case of determining the "best" from a number of candidates. Some graphics standards are not sufficiently well developed in the standards definition process to be considered for this TOP Specification. It is important that emerging standards reach a level of maturity, such as that represented by the International Organization for Standardization (ISO) Draft International Standard (DIS) level, in order to avoid confusion among the vendor and user communities. Under these guidelines, ISO 7942, Graphical Kernel System (GKS), and ISO 8632, Computer Graphics Metafile (CGM), have been selected for this version of the TOP Specification. As the emerging graphics standards mature, their inclusion will be considered for future versions of TOP. Refer to Appendix D for more information on the status of current and emerging graphics standards.

6.1 TOP Computer Graphics Reference Model

Computer graphics can be defined as the generation and manipulation of pictures using computers. An application program which produces or manipulates computer graphic pictures requires computer graphics services which provide the following:

- Graphics output, the generation and display of pictures on graphics hardware,
- Graphics input, the acquisition of pictorial information from an operator,
- Graphics interaction, the control of interaction between graphics input and graphics output in a single interactive application,
- Graphical data bases, storage and retrieval of (possibly structured) graphical data, and
- Graphics metafiles, the generation, interchange and interpretation of picture descriptions.

Computer graphics standards provide these services in an application-independent and device-independent manner. Application independence guarantees that the graphics requirements of a wide variety of applications can be met with a small number of Application Program Interface (API) standards. Device-independence guarantees that a diversity of computer graphics hardware can be used in a manner transparent to the operator.

The computer graphics standards developed by ISO TC97/SC21/WG2 attempt to provide a well-defined, internationally-accepted functionality to meet each of these general requirements. No single standard can perform all of the above computer graphics services adequately for all classes of applications. The requirements of applications with respect to computer graphics are very diverse; therefore, a compatible family of standards exists to serve particular constituencies with specialized requirements.

TOP specifies the use of the CGM standard ISO 8632 [CGMRef7-10] to provide 2-dimensional picture interchange between End Systems and the GKS standard ISO 7942 [CGMRef13] to provide an API and Device-Independent Graphics Services (DIGS) for 2-dimensional graphics. Standards addressing 3-dimensional graphics have not reached a level of maturity for specification in TOP, at this time. The TOP Graphics Reference Model illustrates the recommended CGM and GKS implementation, providing inter-operability between graphic applications and hardware. GKS is necessary to provide the portability of application programs and programmers, which is identified as an important user requirement. Future revisions of TOP may refine this computer graphics reference model. In particular, there are multiple standards concerned with the API currently under development within ISO TC97/SC21/WG2. For their communication requirement, graphics applications will make use of the services provided by the Application Layer of the OSI Reference Model. Standard interfaces between computer graphics and communication aspects of applications do not exist today. Such topics will be the subject of future attention (refer to Appendix G).

Figure 6.1-1 shows an overall view of the computer graphics environment. There is a graphics application process and, optionally, an operator (if the application is interactive or input-driven). The computer graphics functionality is provided by computer graphics services, which may use both graphics hardware and graphics software. This is decomposed into a device-independent layer and a device-dependent layer.

The graphics application may be either a self-contained commercial product or an application program or set of programs designed and written by the user. Common applications found in the technical office include desk-top publishing systems, drafting programs and integrated spreadsheets which create business graphs. Common technical applications include Computer-Aided Design packages, simulation and scientific data analysis. It is not within the domain of any standards body or specification at this time to standardize particular application programs.

Applications use the computer graphics services through invocation of an API. Different APIs will be appropriate for different kinds of applications. This document specifies the use of one API: GKS (ISO 7942). GKS supports the needs of many graphics applications. There are programming Language Application Bindings (LABs) for the API. TOP specifies which of the proposed LABs are mandatory for conformance.

The APIs are implemented to provide the DIGS of the Device-Independent Graphics Layer (DIGL). DIGS are standardized in the functional specification of an API standard. The GKS functional specification is Part 1 of ISO-7942.

The DIGL provides services common to all graphics architectures and hardware. Its use makes device-dependencies transparent to the user.

A significant feature of current computer graphics systems is the ability to maintain and modify internally resident graphics data structures. This capability off-loads the overhead of maintaining the graphical data base from the application program onto the DIGS. GKS (at output levels higher than 0) supports a graphical data structure called the segment. At output level 2, GKS supports a specialized GKS workstation into which segments can be stored and later retrieved and copied to some other GKS workstation.

The device-independent nature of the API is achieved by segregating all of the device-dependent graphics functionality into a separate layer of the TOP Graphics Reference Model, below the DIGL (refer to Figure 6.1-1). This Device-Dependent Graphics Layer (DDGL) supports the interfacing of the DIGS to specific graphics devices or workstations. The interface between the DIGL and the DDGL is the Device Interface. The Device-Dependent Graphics Service (DDGS) is often implemented as a device driver.

While ISO TC97/SC21/WG2 is currently standardizing the graphics functionality of the DDGL, the Computer Graphics Interface (CGI) standard [CGMRef16] is not specified at this time due to the incomplete nature of this project. The CGI defines a Virtual Device Interface (VDI) between the DIGL and the DDGL.

CGM (ISO 8632) is also at the level of the Device Interface. While CGI is an interactive interface, the CGM is a static analog of much of this functionality. The CGM is a picture description metafile, i.e., it contains, in device-, system- and installation-independent form, the picture description information represented by the graphics functions invoked through the API. Rather than record an audit trail of the functions invoked through the API, the CGM stores the output primitives and attributes which compose the picture. The CGM can be used for archiving and transferring such picture description information.

The CGM, which is graphics system and device-independent, is created by a CGM generator. The CGM generator resides at the level of device driver and is invoked by the application-callable layer. In turn, the CGM may be interpreted by an application-callable device-independent graphics system. GKS is an application callable, device-independent graphics system recommended in the reference model. A mapping between GKS and CGM identifies a subset of the GKS functions which are used to generate and interpret a CGM. However, the CGM generator/interpreter can be independent of any graphics system.

Device drivers access actual graphics hardware or metafiles through an Operating System Interface. Physical devices may be distributed on a network.

The TOP Computer Graphics Reference Model of Figure 6.1-1 also depicts the existence of file stores outside the domain of the computer graphics services. The graphics applications programs may maintain application-dependent data bases in system-dependent file stores.

Other relevant emerging standards such as GKS-3D (ISO 8805, [CGMRef14]) and PHIGS (ISO 9592, [CGMRef15]), which are shown in Figure 6.1-1, are briefly discussed in Appendix D.

6.1.1 Services Required/Affected by Graphics

Pictures stored in a CGM encoding may be transferred across a network using the services of File Transfer, Access and Management (FTAM) or Message Handling Systems (MHS).

The following application is intended to demonstrate how selected standards specified in this version of TOP can be used together and to demonstrate their importance to the end-user. It is not intended to be an all encompassing view of graphic requirements. It is only one example of a TOP requirement.

The purpose of this application is to create a technical report to be delivered to a customer. The report is to be created in a totally electronic form, with the eventual goal of delivering the report in electronic form. The report is to contain objects from several source systems. These systems include: CAD/CAM applications, engineering analysis applications, graphic arts development systems, existing hardcopy artwork (image) and text from PC and Host-based systems.

To capture CAD/CAM and engineering drawings for inclusion in the report requires the generation of a metafile. The CGM (ISO 8632) is specified in this version of TOP to satisfy this requirement. Similarly, artwork created by graphics arts personnel via a presentation graphics application also requires the generation of a CGM. Existing hardcopy artwork is captured in electronic form via a scanner and software. This input will also be stored in a CGM.

For each of these objects to be included in the technical report, they must be transferred to the target system. The target system is where all of the entities will be merged into an editable form. The CGM data will be transferred to the target system from the wide variety of source systems using services such as FTAM or MHS.

A graphics editor is required on the target system to finalize the graphics and image for inclusion in the report. This editor must be capable of importing and exporting a TOP conforming CGM. An editor of this type requires an API to a graphics subsystem. GKS (ISO 7942) has been selected as the TOP API. The functionality required of this editor includes scaling, rotating, text annotation, etc.

These objects will then be merged with the text into an electronic compound document format. This format should be one that can be transferred to users on other systems. TOP is specifying Office Document Architecture/Office Document Interchange Format (ODA/ODIF, ISO 8613) for this purpose.

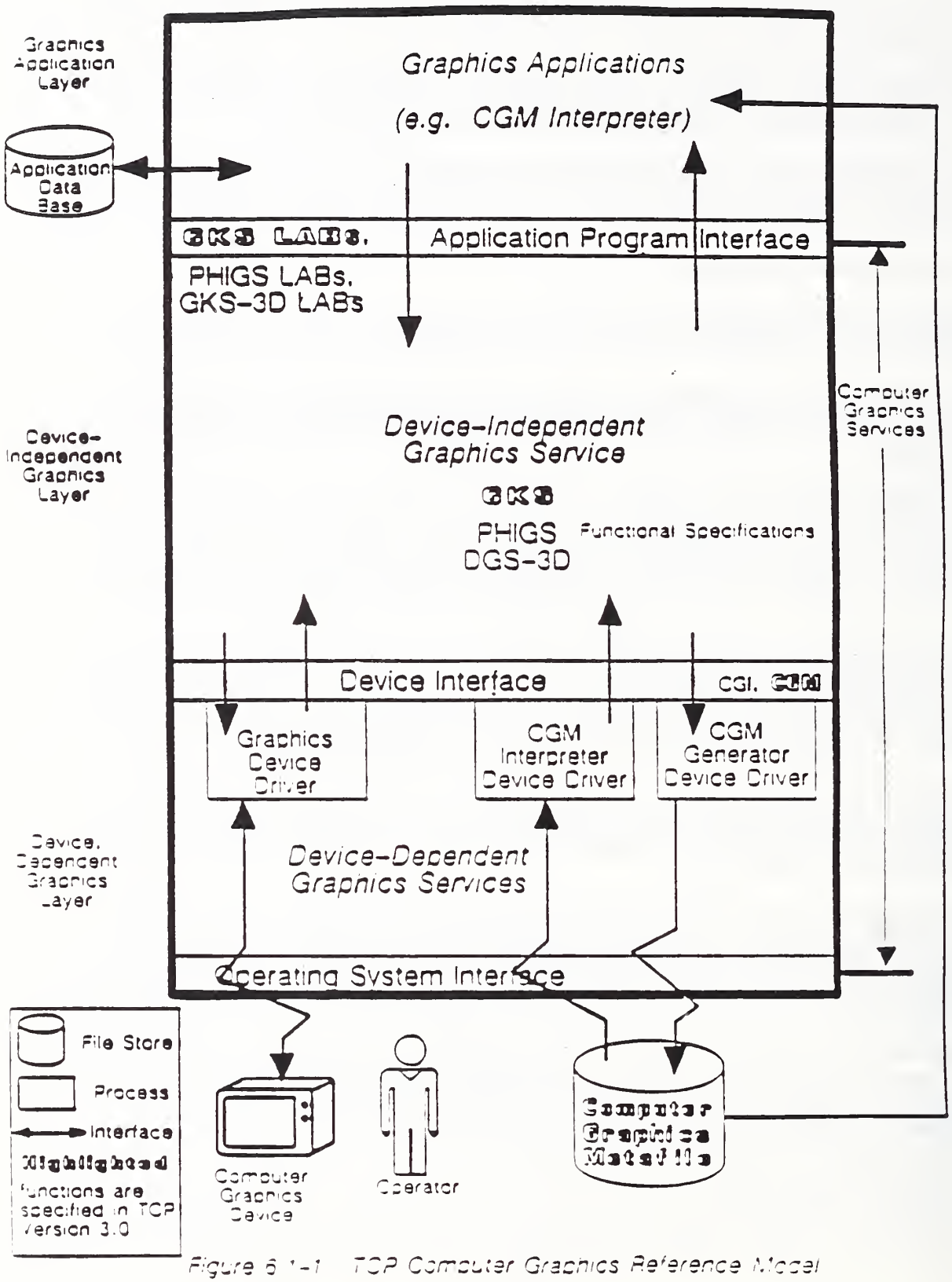


Figure 6-1 TCP Computer Graphics Reference Model

6.2 Computer Graphics Metafile

This section defines the TOP CGM Application Profile (AP) for the Computer Graphics Metafile Interchange Format Building Block. The functional description and binding rules for this building block are found in Chapter 2, Section 2.4.10.

6.2.1 CGM Introduction

The TOP CGM AP defines the conformance characteristics or permissible combinations for all possible data streams that are specified in the profile. In addition, the TOP CGM AP defines additional requirements for transmitting, receiving, interpreting and handling valid CGM data streams. The definition of such implementation constraints is usually outside the scope of an ISO standard. However, such APs are required and necessary to insure uniform implementation of such standards, especially where interchange in an open systems environment is concerned.

6.2.2 CGM Scope

The TOP CGM AP defines the CGM implementation that is required for computer graphics picture information interchange. CGM implementations that conform to this AP will be able to be integrated into other application processes such as compound document interchange. This AP can, in the future, be supplemented by additional CGM APs.

6.2.3 Definitions

APPLICATION PROFILE (AP) - A specification that defines the use of an International Standard, with a definition of all possible data streams that conform to that profile. An AP insures interoperability of implementations of an International Standard.

BASIC VALUE - The subset of permissible values for parameters of a CGM element that are mandatory for conformance to this AP.

CGM MI - A CGM metafile input workstation.

CGM MO - A CGM metafile output workstation.

COMPOUND DOCUMENT - A digital analog of a document containing more than one component objects (such as character, computer graphics, image or facsimile data).

COMPOUND DOCUMENT INTERCHANGE FORMAT - The specification for a mechanism for storing and transferring a compound document. Refer to ISO 8613.

COMPUTER GRAPHICS INTERFACE (CGI) - The specification for interface techniques with graphical devices.

COMPUTER GRAPHICS METAFILE (CGM) - The specification for a mechanism for storing and transferring picture description information. Refer to ISO 8632.

DATA INTERFACE - The communication boundary (i.e., interface) between software modules or devices comprising one or more operation codes and data (as contrasted with a subroutine call interface).

DEFAULT VALUE - The implicit value for a parameter of a CGM element. For example, default Metafile Name in Begin Metafile element is a null string.

DEVICE DRIVER - The device-dependent portion of a graphics system which supports a physical device. The device driver generates device class specific output.

GRAPHICAL KERNEL SYSTEM - A standardized application programmer's interface to graphics systems. Refer to ISO 7942.

METAFILE - Synonymous with CGM. A representation for the storage and transfer of graphical data and control information. This information contains a device-independent description of one or more pictures.

METAFILE GENERATOR - Synonymous with CGM Generator. The software or hardware that creates a picture or conveys information in the CGM representation.

METAFILE INTERPRETER - Synonymous with CGM Interpreter. The software or hardware that reads the CGM and interprets the contents.

PERMISSIBLE VALUES - The range of valid values for a parameter of a CGM element as specified in ISO 8632.

VIRTUAL DEVICE - An idealized computer graphics device that presents a set of graphics capabilities to graphics software of systems via the CGI.

Note: Refer to ISO 8632, clause 3 and ISO 7942, clause 3, for further definitions of computer graphics terms.

6.2.4 CGM Architectural Concepts

The CGM is designed to be usable and useful to a wide range of applications, graphics systems and devices or workstations. The CGM is graphics system independent, as well as device-independent. The CGM is created by a CGM Generator. The CGM Generator resides at the level of the device driver and is invoked by the application callable layer. The CGM Generator can be used to record device-independent picture descriptions, conceptually in parallel with the presentation of images on actual devices. Figure 6.2-1 illustrates the TOP Graphics Reference Model for creation of the CGM.

The CGM is designed to be interpreted in one of two ways. First, the CGM can be interpreted by a special application program, that in turn invokes a device-independent graphics system to render the CGM. Second, a device-independent graphics system may have functions that can be invoked by an application to get, read and interpret metafile elements using the facilities of a CGM metafile input workstation. Figure 6.2-2 illustrates a CGM Generator (primary) Reference Model. Figure 6.2-3 illustrates the CGM Interpreter (alternate) Reference Model.

The GKS may be the device-independent graphics system that is used in Figures 6.2-2 and 6.2-3. GKS (ISO 7942), however, does not specifically refer to the CGM any more than it does to another specific class of graphics device.

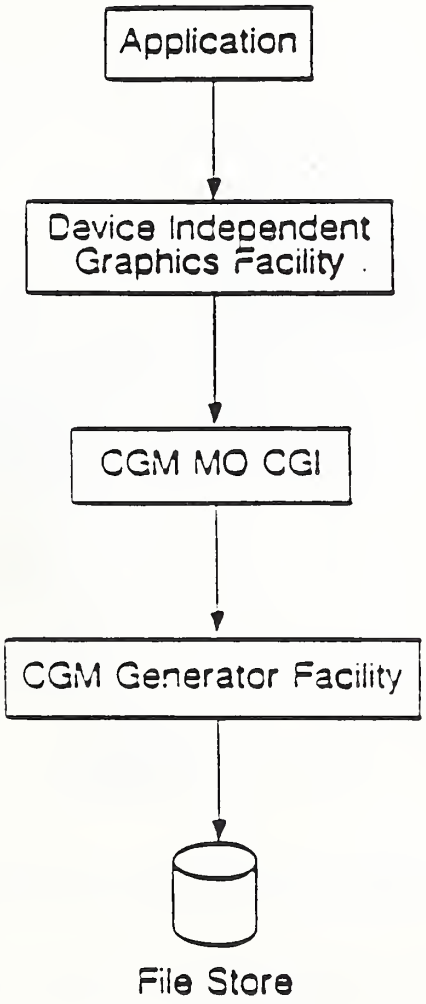


Figure 6.2-1: CGM Generator Reference Model

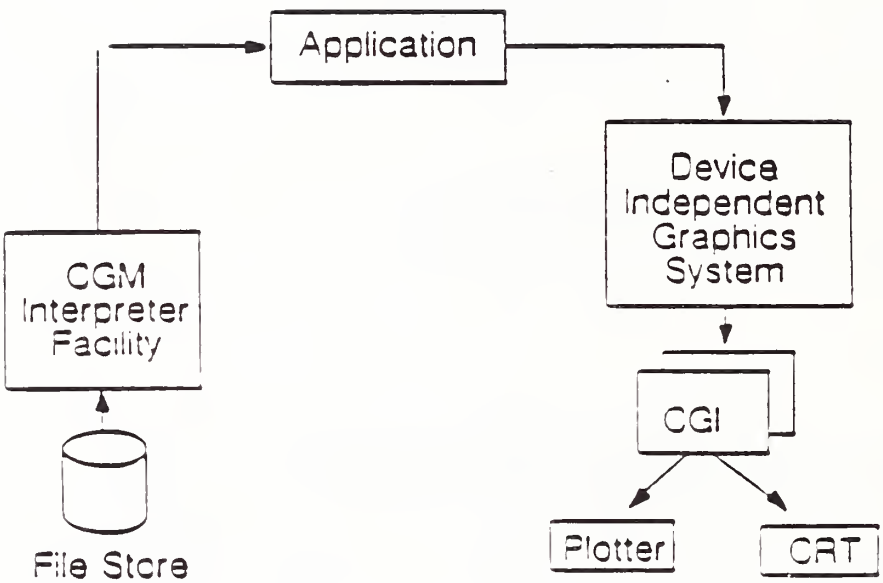


Figure 6.2-2: CGM Interpreter Reference Model

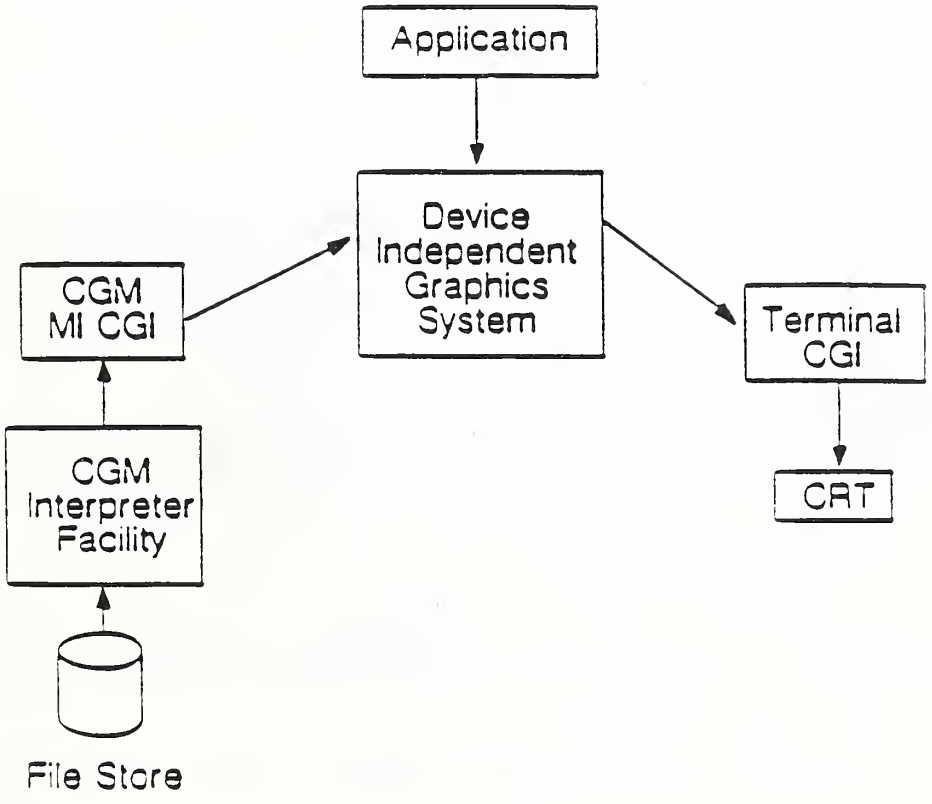


Figure 6.2-3: Alternate CGM Interpreter Reference Model

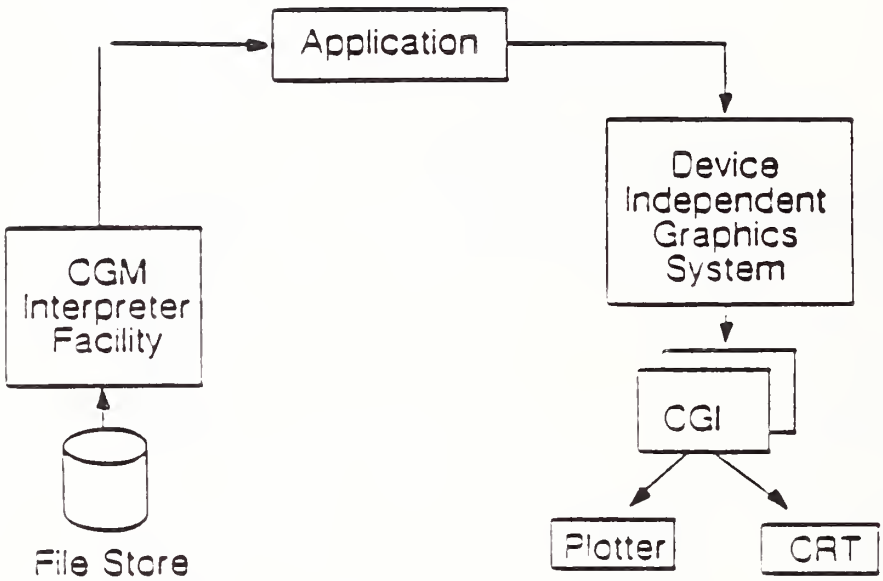


Figure 6.2-2: CGM Interpreter Reference Model

6.2.5 CGM Conformance

The TOP CGM AP specifies conformance in terms of "permissible" and "basic" values. Permissible values are the range of values of CGM elements as specified in ISO 8632. Basic values are a subset of the permissible values that constitute the "basic set". For example, permissible values of LINE TYPE include all non-zero integers, while basic values include the standardized enumerated values 1 to 5.

TOP defines a conforming "basic metafile" to be one that contains no elements or parameter values outside of the basic set. TOP defines a conforming "basic interpreter" to be one that correctly interprets any conforming basic metafile and may have more capability as well. TOP defines a conforming "basic generator" as one that produces only conforming basic metafiles, or can reliably be directed to function in a mode of producing basic metafiles.

In addition, any TOP conforming basic interpreter should correctly parse and ignore any elements and parameter values that it does not support.

CGM (ISO 8632) defines the form (syntax) and the functional behavior (semantics) of the ordered set of metafile elements. There are three different encodings of the CGM that have been standardized. These include Clear Text Encoding, Character Encoding and Binary Encoding. This AP specifies the CGM Binary Encoding, ISO 8632/3. Future APs may be developed for the other encodings.

For interchange of CGM files on a TOP network the binary encoding is required.

The basic form for the command header and string parameter header is the long form. The long form of the command header is detailed in ISO 8632/3, subclause 4.4. The long form of the string parameter header is detailed in ISO 8632/3, subclause 6, note 6.

6.2.6 Metafile Constraints

The basic set is defined by the limitations on permissible values below. Where an element is not mentioned, it is implied that the basic set includes all values permitted in the CGM.

6.2.6.1 Delimiter Elements

Element	Basic Value
NO-OP	An arbitrary sequence of n octets, n=0,1,2,....,32767

Table 6.2-1: Delimiter Element Constraints

6.2.6.2 Metafile Descriptor Elements

Element	Basic Value
Metafile Description	(Note 1)
Integer Precision	16
Real Precision	0,9,23 (floating point)
Index Precision	16
Color Precision	8,16
Color Index Precision	8,16
Maximum Color Index	255
Font Index	(Note 4)
Character Set List	0,4/2 (Note 2) 1,4/1 (Note 3)
Character Coding Announcer	0,1

Note 1: Implementors should use the Metafile Description element's string to include a brief identification of their company or product, so that interpreters can account for known idiosyncrasies of generators. The string "TOP/BASIC-1" should be used to label the metafile as conforming to this profile.

Note 2: The character set is ANS X3.4, 7-bit American National Standard Code for Information Interchange (7-bit ASCII).

Note 3: The character set is ANS X3.134/2, 8-bit American National Standards Standard Code for Information Interchange (8-bit ASCII). This is equivalent to ISO 8859/1, Right-Hand Part of Latin Alphabet Number 1.

Note 4: Four simultaneous fonts are supported. The font names are selected from the basic font names in Table 6.2-8.

Table 6.2-2: Metafile Descriptor Element Constraints

6.2.6.3 Picture Descriptor Elements

Element	Basic Value
Scaling Mode	(Note 1)

Note 1: Implementors should use care in specifying the value of the metric scaling factor to ensure that it has sufficient significant resolution to specify the intended accuracy.

Table 6.2-3 Picture Descriptor Element Constraints

6.2.6.4 Control Elements

Element	Basic Value
VDC Integer Precision	16,32
VDC Real Precision	0,9,23 (floating point)

Table 6.2-4: Control Element Constraints

6.2.6.5 Graphics Primitive Elements

To ensure portability and predictable results, TOP conforming basic metafiles may not contain any Generalized Drawing Primitive (GDP) elements.

6.2.6.6 Attribute Elements

Element	Basic Value
Line Bundle Index	1-5
Line Type	1-5
Marker Bundle Index	1-5
Marker Type	1-5
Text Bundle Index	1-2
Text Font Index	1-4
Character Set Index	1-2
Alternate Character Set Index	1-2
Fill Bundle Index	1-5
Hatch Index	1-6
Pattern Index	1-8
Edge Bundle Index	1-5
Edge Type	1-5
Pattern Table	Pattern Table Index, 1-8 nx, 1-16 ny, 1-16
Color Table	Starting Color Index, 0-255

Table 6.2-5: Attribute Element Constraints

6.2.6.7 Escape Elements

To ensure portability and predictable results, TOP conforming basic metafiles may contain only those ESCAPE elements that are defined in Section 6.2.8.2 of this AP.

6.2.6.8 External Elements

Element	Basic Value
Message	Action Required Flag, 0

Table 6.2-6: External Element Constraints

6.2.7 CGM Defaults

The CGM specifies a complete set of defaults. In a few cases, these defaults are not appropriate for TOP requirements. However, any TOP metafile must be a legal CGM. This includes implicit defaults specified in ISO 8632/1, clause 6 and ISO 8632/3, clause 8. Therefore, each deviation from the implicit defaults requires that the affected element either:

1. Appear in the Metafile Defaults Replacement element, or
2. Be explicitly specified for its value to be applicable.

Each TOP conforming basic metafile shall contain in the Metafile Descriptor a Metafile Defaults Replacement element that includes at a minimum:

1. VDC Real Precision element, where precision is set to (0,9,23) (floating point).
2. Text Precision element, where precision is set to 2 (stroke).
3. Color Table element, where the starting color table index is set to 2 and the list of direct color values for the remaining 254 entries are a repetition of the following eight entries:

Index	Values	Meaning
2	(255, 0, 0)	Red
3	(0, 255, 0)	Green
4	(0, 0, 255)	Blue
5	(255, 255, 0)	Yellow
6	(255, 0, 255)	Magenta
7	(0, 255, 255)	Cyan
8	(0, 0, 0)	Black
9	(255, 255, 255)	White

Table 6.2-7: Default Color Table

Color table defaults for color indices 0 and 1 are explicitly defined in the CGM standard as corresponding to the nominal background and nominal foreground colors, respectively.

Each TOP conforming basic metafile shall also contain in the Metafile Descriptor, the following elements:

1. Real Precision element, where precision is set to (0,9,23) (floating point).
2. Maximum Color Index element,where the maximum color index is set to 255.
3. Character Set List element, where the first two character set indices are set to (0,4/2) and (1,4/1).

It is not apparent in the CGM standard what the default value for the precision of the floating point real parameter of Scaling Mode should be. TOP conforming generators and interpreters shall assume that the real precision for this parameter is (0,9,23).

6.2.8 CGM Related Private Use of Elements

6.2.8.1 Fonts

The fonts in Table 6.2-8 are public domain fonts, available from the U.S. National Bureau of Standards (NBS) [CGMRef5]. All of these fonts are considered to be basic capabilities of a TOP conforming basic metafile. Any of these fonts may appear in the Font List element in a CGM that conforms to this AP. The font names are specified in a manner compatible with ISO 9541, Font and Character Information Interchange [CGMRef11]. The font name (Font Identifier for Base Font) is a concatenated string of the Universal Font Name and a User Readable Font Name. The Universal Font Name for these fonts is assumed to be "NBS", pending the registration of NBS with an Organization Name. The User Readable Font Name is the concatenated string "HERSHEY:", to designate one of the Hershey fonts and "name string", to designate the particular typeface.

1.	NBS HERSHEY:CARTOGRAPHIC_ROMAN
2.	NBS HERSHEY:CARTOGRAPHIC_GREEK
3.	NBS HERSHEY:SIMPLEX_ROMAN
4.	NBS HERSHEY:SIMPLEX_GREEK
5.	NBS HERSHEY:SIMPLEX_SCRIPT
6.	NBS HERSHEY:COMPLEX_ROMAN
7.	NBS HERSHEY:COMPLEX_GREEK
8.	NBS HERSHEY:COMPLEX_SCRIPT
9.	NBS HERSHEY:COMPLEX_ITALIC
10.	NBS HERSHEY:COMPLEX_CYRILLIC
11.	NBS HERSHEY:DUPLEX_ROMAN
12.	NBS HERSHEY:TRIPLEX_ROMAN
13.	NBS HERSHEY:TRIPLEX_ITALIC
14.	NBS HERSHEY:GOTHIC_GERMAN
15.	NBS HERSHEY:GOTHIC_ENGLISH
16.	NBS HERSHEY:GOTHIC_ITALIAN

Table 6.2-8: Basic Font Names

6.2.8.2 CGM Escape Elements

The following Escape elements are required in all TOP conforming basic metafiles.

6.2.8.2.1 Disable Clearing of View Surface

The normal interpretation of a CGM is such that the view surface of a device will be cleared on each Begin Picture Body element. This Escape element will disable the clearing of the view surface for all of the pictures in a metafile. The effect of this Escape element is to permit multiple metafile pictures to be imaged on the same view surface in a temporal manner (i.e., last picture overlays previous picture) with a mapping as described in the CGM standard. Pictures in the metafile may have different VDC Extents. Each picture will be mapped into the current Device Viewport for the picture. This Escape element must appear in the metafile description (that is, between the Begin Metafile element and the first Begin Picture element). The Device Viewport is defined by the Device Viewport Escape element. The default Device Viewport is the available view surface.

This Escape element will have no effect on the resetting of the metafile defaults on each Begin Picture element. This Escape element is a basic capability of this AP.

Escape Identifier: -301
Escape Data Record: N/A

6.2.8.2.2 Device Viewport

The default Device Viewport for interpreting a picture in the CGM is the available view surface of the interpreting device. This Escape element will redefine the Device Viewport for the picture to some portion of the available view surface. This Escape element must appear in the Picture Description (that is, between the Begin Picture element and the Begin Picture Body element). The specification units for the Device Viewport definition is a real fraction, in the range [0.0, 1.0], of the default Device Viewport. The default Device Viewport is the available view surface. This Escape element is a basic capability of this AP. If this Escape element is specified when the Scaling Mode has been set to metric units, then the Device Viewport Escape will cause portions of the resulting picture that do not fit into the specified Device Viewport to be clipped. The VDC Extent is mapped into the specified Device Viewport such that the origin of the VDC Extent coincides with the origin of the specified Device Viewport.

Escape Identifier: -302

Escape Data Record: A single string of text. This string is the specification of the Device Viewport. Parameters in the string are separated by at least one blank character and/or a single comma character. The decimal point of the real fraction is required. Leading zeroes of the real fraction are optional. There are four parameters. Invalid parameters will result in this Escape element being ignored.

P1: First corner x-coordinate. Real fraction of the default Device Viewport, in the range [0.0, 1.0].

P2: First corner y-coordinate. Real fraction of the default Device Viewport, in the range [0.0, 1.0].

P3: Second corner x-coordinate. Real fraction of the default Device Viewport, in the range [0.0, 1.0].

P4: Second corner y-coordinate. Real fraction of the default Device Viewport, in the range [0.0, 1.0].

For example, a Device Viewport equal to the upper right quarter of the default Device Viewport would be coded with the following Escape element:

Escape Identifier: -302

Escape Data Record: ".5,.5,1.,1."

OR

Escape Identifier: -302

Escape Data Record: "0.50 0.50 1.0 1.0"

A Device Viewport equal in width to the left one tenth of the default Device Viewport and equal in height to the default Device Viewport would be coded with the following Escape element:

Escape Identifier: -302

Escape Data Record: "0., 0., .1, 1."

6.2.9 CGM Implementation Dependencies

This section describes the implementation dependencies and environmental constraints for this AP. Specifying the nominal values for implementation practices, defaults and options will facilitate uniform generation and interpretation of the CGM.

6.2.9.1 General Guidelines for CGM Elements

Unless otherwise noted in this AP, all of the guidelines of ISO 8632, Annex D, shall be adhered to by TOP CGM generators and interpreters. In particular, the minimum interpreter capabilities of ISO 8632, Annex D.5, plus the interpreter functions defined in Section 6.2.6, should be the minimum supported capabilities.

Name: Metafile Defaults Replacement

Description: The Metafile Defaults Replacement element shall not be partitioned. In addition, no part of the element will be partitioned.

Name:	Restricted Text
Description:	Minimal capability of a basic conforming TOP interpreter shall render the complete restricted text string (i.e., Append Text elements permitted), scaled isotropically (i.e., specified aspect ratio for the text is not distorted), such that the text string fits into the Text Extent parallelogram.
Name:	Color Table
Description:	The Color Table element has an unspecified effect when it appears in a picture, subsequent to any graphical primitive elements. The Color Table element should appear prior to any graphical primitive elements to insure that interpreting systems without dynamic color update capabilities can render the intended effect.

6.2.9.2 Implementation Guidelines for Generators and Interpreters

This section is meant to augment ISO 8632/1, Annex D.5 and ISO 8632/3, clause 8.

6.2.9.2.1 Minimum Data Structure Support

Name:	Maximum Color Array Dimension
Description:	The basic value for the number of color values that can appear in a color array or color list parameter. CELL ARRAY has a color list parameter. PATTERN TABLE has a color array parameter. COLOR TABLE has a color list parameter.
Basic Value:	1048576 for CELL ARRAY (i.e., one 1024 x 1024 image) 1024 for PATTERN TABLE (i.e., four 16 x 16 patterns) 254 for COLOR TABLE (i.e., entries 2-255)
Name:	Maximum Point Array Length
Description:	The basic value for the number of points and VDC that can appear in parameters for metafile elements.
Basic Value:	1024
Name:	Maximum String Length
Description:	The basic value for the length of an individual string of characters.
Basic Value:	256 for all string parameters except data records 32767 for data records.

Name:	Bundle Table
Description:	The bundle representations are not settable in the current version of the CGM. This implementation dependency detracts from the open interchange of the CGM. The following default bundle table values will permit a picture to be uniformly rendered by all conforming basic TOP interpreters.
Basic Value:	Refer to Table 6.2-9

6.2.9.2.2 CGM Transfer Format

Operating system dependencies for file formats can often be more of a burden on interoperability than differences in interchange formats. To ensure CGM interoperability, some conventions for file formats are required.

The file containing the CGM should be formatted into fixed length 80 octet records. If the record length is less than 80 octets, even octet records are required.

Note: When the files are transferred on magnetic tape, the 80 octet records should be formatted into blocks of 800 octets.

6.2.10 CGM Error Processing

A TOP conforming interpreter should gracefully recover from any exception condition. If there is something which is not understood by the interpreter, processing of the metafile should continue with the metafile element preceding that which caused the exception. Exact details for exception handling are outside the scope of this specification.

6.2.11 CGM Conformance Testing

Conformance testing recommendations for the conforming TOP basic metafile will be addressed by subsequent releases of this specification.

Bundle Type / Bundle Representation	Bundle Index 1	2	3	4	5
Line Bundle					
Line Type	Solid	Dash	Dot	Dash-dot	Dash-dot-dot
Line Width	1	1	1	1	1
Line Color	1	1	1	1	1
Marker Bundle					
Marker Type	Dot	Plus	Asterisk	Circle	Cross
Marker Size	1	1	1	1	1
Marker Color	1	1	1	1	1
Text Bundle					
Font Index	1	1			
Text Precision	Stroke	Stroke			
Character Expansion Factor	1	0.5			
Character Spacing	0	0			
Text Color	1	1			
Fill Area Bundle					
Interior Style	Hatch	Hatch	Hatch	Hatch	Hatch
Fill Color	1	1	1	1	1
Hatch Index	1	2	3	4	5
Pattern Index	1	1	1	1	1
Edge Bundle					
Edge Type	Solid	Dash	Dot	Dash-dot	Dash-dot-dot
Edge Width	1	1	1	1	1
Edge Color	1	1	1	1	1

Table 6.2-9: Basic Bundle Table

6.2.12 CGM References

These references relate to documents applicable to this specification and are in addition to those referenced in ISO 8632.

CGMRef1 ANS X3.4 - 1986, 7-bit American National Standard Code for Information Interchange.

CGMRef2 ANS X3.41 - 1974, American National Standard Code Extension Techniques for Use With the 7-bit Coded Character Set of American National Standard Code for Information Interchange.

CGMRef3 ANS X3.134/1-1987, American National Standard Code for 8-bit ASCII Structure

CGMRef4 ANS X3.134/2-1987, American National Standard Code for 7-bit and 8-bit ASCII Supplemental Multilingual Graphic Character Set.

CGMRef5 NBS Special Publication 424 - April 1976, Hershey Fonts.

CGMRef6 ISO DIS 8613, Information Processing - Text and Office Systems - Office Document Architecture and Interchange Format (ODA/ODIF).

CGMRef7 ISO 8632/1, Information Processing Systems - Computer Graphics - Metafile (CGM) for the Storage and Transfer of Picture Description Information. Part 1; Functional Specification.

CGMRef8 ISO 8632/2, Information Processing Systems - Computer Graphics - Metafile (CGM) for the Storage and Transfer of Picture Description Information. Part 2; Character Encoding.

CGMRef9 ISO 8632/3, Information Processing Systems - Computer Graphics - Metafile (CGM) for the Storage and Transfer of Picture Description Information. Part 3; Binary Encoding.

CGMRef10 ISO 8632/4, Information Processing Systems - Computer Graphics - Metafile (CGM) for the Storage and Transfer of Picture Description Information. Part 4; Clear Text Encoding.

CGMRef11 ISO 9541, Information Processing Systems - Font and Character Information Interchange.

CGMRef12 ISO DIS 8859/1, Information Processing - 8-Bit Single Byte Coded Graphic Character Sets. Part 1; Latin Alphabet Part 1.

CGMRef13 ISO 7942, Information Processing Systems - Computer Graphics - Graphical Kernel System (GKS) Functional Description.

CGMRef14 ISO DIS 8805, Information Processing Systems - Computer Graphics - Graphical Kernel System for Three Dimensions (GKS-3D) Functional Description.

CGMRef15 ISO DP 9592, Information Processing Systems - Computer Graphics - Programmers Hierarchical Interactive Graphics System (PHIGS).

CGMRef16 ISO TC97/SC21 N1179, Information Processing System - Computer Graphics - Interfacing Techniques for Dialogues with Graphical Devices (CGI).

6.2.13 The Use of OSI Data Transfer Services

To transfer a CGM file between two TOP End Systems, the services provided either by FTAM or by MHS can be used. Remote access to part of a CGM file is not addressed at this time.

Using FTAM to transfer CGM files:

One should specify the CGM file Document Type entry number as NBS-1 or FTAM-3, Document Type Name as '{ISO standard 8571 document type (6) unstructured binary (4)}' for Contents-Type-Attribute or Contents-Type-List parameters. The contents of the CGM file should be mapped onto a sequence of octet strings. The boundary of octet strings has no significant meaning.

Note: FTAM does not provide a standard document type for a CGM file. Therefore, the Presentation Layer can not be fully used and it is left up to the user or application programs that remotely access files using FTAM to know that a given file contains CGM formatted information.

Using MHS to transfer CGM files:

Specify the Body as USABodyParts BodyPartNumber '7'. The contents of the CGM file shall be mapped on to the body of an IPM as a sequence of octet strings. The boundary of the octet strings has no particular semantics.

APPENDIX 2

SURVEY ARTICLE ON CGM TECHNICAL DETAILS

The Computer Graphics Metafile

Lofton Henderson, Henderson Software

Margaret Journey, Precision Visuals

Chris Osland, Rutherford Appleton Laboratory

The Computer Graphics Metafile is the only standard for graphical database specification designed to serve a wide range of applications. As such, this standard for capturing multiple, device-independent picture definitions is necessarily complex. CGM is a precise, concise, and sometimes terse specification, and it is hard to grasp all the ramifications even after several readings. The purpose of this article is to provide some idea of the basic goals of CGM, its structure, and what it does and does not standardize, as well as to illustrate how it works when applied.

The Computer Graphics Metafile, or CGM, will soon become the first standard for a general-purpose graphical metafile. The computer graphics industry will have for the first time, a versatile and standard definition of a file for the capture, transfer, and archiving of pictorial information. Understanding CGM is no easy task, however: its definition has the conciseness and precision of a legal document, containing numerous subtleties and intricate detail. Even with careful study the proper interpretation and usage of the metafile is easily lost in the details.

This article presents an overview of CGM's basic philosophy, describes its contents, and outlines its uses.

Topics include the general concepts of graphical metafiles, including what type of metafile CGM is, and the structure and contents of CGM and the functionality of the metafile itself. Several detailed CGM application scenarios are developed to illustrate the adaptability of CGM and how CGM elements may be used to achieve this versatility.

What is a graphical metafile?

Consider two graphical sessions on a local area network. On one computer an application program is generating graphical representations of technical data and saving them for later off-line plotting. The file format is binary, resembling a low-level graphics device protocol but in fact corresponding to no existing graphics device. Elsewhere in the network a technical writer is interactively generating illustrations with a drawing package based on GKS. During the session a complete ASCII cleartext script of the dialogue across the GKS workstation interface is being recorded for later editing and session restart.

Though seemingly dissimilar, both graphical files produced by these hypothetical applications are *graphical metafiles*. Because the concept of a graphical metafile encompasses a wide range of objects, the definition is very broad: a graphical metafile is a mechanism for the capture, storage, and transport of graphical information.

Timetable for CGM

1976	Seillac I workshop explores standardization for computer graphics; identifies need for several levels of graphics standards
1979	GSPC 79 issued containing first proposal for a graphics metafile standard
1980	Work begins on the Virtual Device Metafile (VDM)
1982	VDM proposed as ISO work item
1983	VDM accepted as ISO work item
1984	First ANSI public review; first ISO DP ballot; name changed from VDM to Computer Graphics Metafile (CGM); cleartext encoding added
1985	Second ANSI public review; second ISO DP ballot; ISO begins investigation of work item for compatible revision of CGM incorporating at least GKS dynamics and session capture
1986	CGM becomes ANSI standard
1987	CGM becomes ISO standard

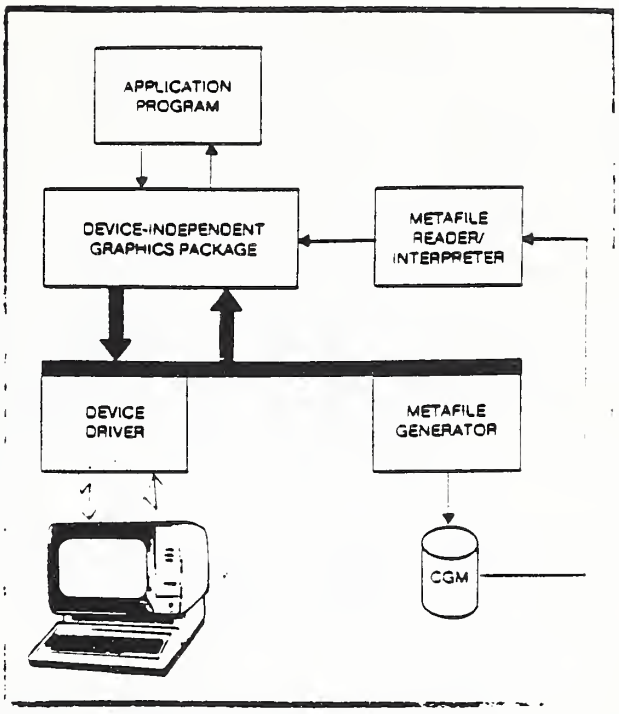


Figure 1. CGM as it relates to a graphics application environment.

Later we will see some detailed application examples. For now a brief list of some metafile uses will suffice to illustrate the concept. Graphical metafiles provide

- a data format for picture archiving
- a graphical protocol for off-line and off-site plotting
- a single format for spooling to multiple dissimilar plotting devices
- the possibility and impetus for a single standard interface to picture-generating devices
- a way to reuse the same picture without recomputing it
- a means to preview pictures before committing output to slow or expensive media
- a basis for debugging and quality assurance tools
- a basis for session save/restart mechanisms
- the glue for unifying and integrating distinct graphics applications and hardware/software systems in a distributed computing environment

Relationship to a graphics system

A metafile is a graphical database. Consequently, there must be a component of a graphical system for generating the database concurrently with the execution of an application (the metafile generator). There must also be a component for reading, interpreting, and rendering the graphical information in a metafile (the metafile interpreter). Figure 1 illustrates the relationship of a CGM-like metafile and these processing components to the rest of a graphical system. In this figure the interpreter as an

entity is not as well defined as the generator. In different environments the interpreter may be a stand-alone process, it may be a component of the graphics system, or it may be a combination of the two.

Figure 1 implies that the generator and interpreter are software components—which is the current state of technology. However, there is no reason that their functions (particularly the functions of the interpreter) should not migrate into hardware. This migration is, in fact, one of the purposes and anticipated benefits of metafile standardization.

Implicit in the definitions, examples, and context diagram just described is the separation of the processes generating metafiles and the processes using them. There is only a single, one-way connection between the two types of processes—the metafile itself. This constraint has implications for the design of metafiles and the selection of metafile elements, and is one of the primary distinctions between a metafile and an interface such as Computer Graphics Interface (CGI). In particular, the functions of a metafile must be independent of the final output device.

Types of metafiles

To understand what CGM is and is not, we should first distinguish two types of metafiles: picture capture and session capture. This classification must not be taken too strictly. The types are not exclusive and a metafile definition could share characteristics of the two types.

A picture-capture metafile is one whose primary function is the capture of multiple, device-independent picture definitions. It provides a mechanism well suited for storing or transmitting randomly accessible and concisely defined collections of independent images. CGM is a picture-capture metafile.

A session-capture metafile is designed to capture the complete output dialogue across some interface in a graphical system. It thus provides a mechanism suitable for recording the exact state of a graphical system during a graphics session. The GKS metafile (GKSM) annex of the Graphical Kernel System contains such a definition, but it is not part of the standard.

Definition of CGM

CGM is a *static* picture-capture metafile. That is, it contains no elements (functions) with dynamic effects on partially defined pictures. A change of transformation to achieve zoom is an example of a dynamic effect.

The definition of CGM scope, particularly its limitation, was the most difficult and persistent issue during CGM specification. A large constituency wanted a picture-capture metafile that could be standardized relatively quickly. Another important constituency—including users of the nonstandard GKSM—desired dynamics, session capture, segmentation, and other advanced features with useful functionality. The limited scope was finally adopted, and more advanced and complex functionality was put off for a second phase of standardization. Work is already underway on this advanced (but backwardly compatible) metafile definition. The first priority of this effort is to extend CGM to serve as a fully capable GKSM.

CGM is not an application programmer standard, as are GKS and PHIGS. Rather, it is a specification for system designers and system implementers. Figure 1 shows the (conceptual) placement of the metafile generator at approximately the level of device drivers in a graphics hierarchy.

Generality is a key attribute of CGM. It is designed for use with a wide variety of devices, applications, and systems. The same metafile can be interpreted on a low-resolution monochrome terminal, a high-resolution multipen plotter, or a raster device with high functionality.

CGM can be used in simple and minimal applications, in which the priority is blind interchange of substantively correct pictures to a variety of devices, and precise tuning of output is unimportant. But it is also highly tailorable: The generator can target a particular device or class of devices and tailor the metafile contents (such as the metafile coordinate system) accordingly. The mechanisms of tailoring are such that device independence of the resulting metafile is preserved.

What does CGM standardize?

CGM standardizes the semantics and syntax of a set of elements for the device-independent definition of pictures. The standard is organized in four parts. Part 1 is a functional specification. All standardized elements are identified, their parameterizations are described (in an abstract fashion), and their meanings are defined. An appendix (annex A) gives a highly concise definition in a formal grammar.

The remaining three parts present data encodings of the functionality of Part 1. Different applications have conflicting needs: compactness of the metafile vs. speed of generation and interpretation vs. readability, editability, and ease of transfer, etc. These conflicting needs are met by providing three distinct encodings: character, binary, and cleartext.

Part 2 is a character encoding. Opcode and parameter data are encoded by characters from the ASCII character set (CGM actually specifies "standard national character set based on ISO 646." ASCII is the US version and is identical to ISO 646 but for a small number of code positions, which ISO 646 says are to be defined by the national standards bodies of the ISO members.) The resulting encoded data consists of printable characters. Elements and parameter lists in this encoding are self-delimiting; that is, there is no leading length indication. The encoding is compact and can be transmitted directly through standard character-oriented communications services. There are no unusual escape or control sequences to confuse the communications service.

Part 3 is a binary encoding. It is intended for applications in which speed of generation and speed of translation are most important. It is reasonably compact as well. The formats for encoded data are either chosen for their similarity to data formats in computers, or designed for fast decoding and processing. For example, integers are encoded as 2's complement binary integers; reals can be encoded with either a floating-point format based on ANSI/IEEE standard P754 or a fixed-point format. Encoded elements are aligned so that parameter and opcode

entities will, on most computers, align conveniently with respect to computer word boundaries (for speed of parsing).

Part 4 is a cleartext encoding. It is human readable. For example, a circle centered at (0,0) with radius .25 would be encoded as CIRCLE 0. 0. .25;. It is transmittable with standard character-oriented services, like character encoding, but is not very compact and is relatively slow to generate and interpret. An important feature is its ease of comprehension and manipulation using standard text editors.

A vital feature and design principle of encodings is their translatability—any CGM in one encoding must be translatable to an equivalent metafile in the other two encodings. "Equivalent" means that no information is lost, and interpretation of the metafiles will yield the same picture. Hence, translation from one encoding to another and back again will yield the same picture, although the exact bit streams of the two copies might differ.

Understanding what is *not* standardized with CGM is as important as understanding what is standardized. In the functional specification, a set of picture-defining elements is standardized, along with parameterizations. Metafile generators and metafile interpreters are not standardized. This point has led to considerable confusion. The question often arises "Would an interpreter be a conforming interpreter if it . . .?" Within CGM such a question cannot be answered. The semantics and syntax of the metafile elements are standardized, but not the behavior of processes that manipulate metafiles.

Part 1 does contain an appendix (annex D) that gives some useful guidelines to implementers of interpreters, but it is not part of the actual standard. The annex describes what minimal functionality should be provided and the reasonable responses to exception conditions in a metafile.

Although Parts 2, 3, and 4 standardize the encodings of elements in the appropriate style, they do not specify anything about physical record formats of the encoded data. These formats are acknowledged to be important for successful metafile interchange, but specification is beyond the scope of graphics standards committees—it is in the domain of groups standardizing file structure, transfer, and management.

CGM structural overview

A computer graphics metafile is an ordered sequence of elements with a simple two-level structure (see Figure 2). Every metafile consists of a *metafile descriptor* and a collection of logically independent pictures. Each picture consists of a *picture descriptor* and a picture body containing the actual picture definition. The MD contains descriptive information that applies to all pictures in the metafile. The information enables the interpreter to correctly parse the metafile and identifies the resources that may be required to render the pictures correctly. The PD also contains descriptive information, but PD elements pertain only to the picture in which the PD resides.

Each picture definition is self-contained and logically independent of all other picture definitions in the metafile. After the MD is interpreted, pictures may be randomly accessed and correctly interpreted without the need to interpret any of their predecessors. Such picture independence is possible because CGM defines elements that can be thought of as having a state as being in their default state at the start of each picture. Hence, changes of state in previous pictures have no effect on later ones. Picture independence was one of the most significant design criteria of CGM.

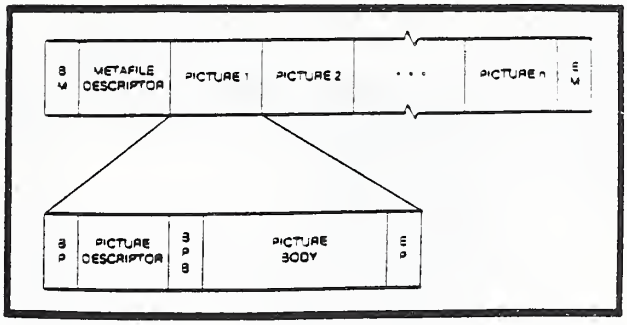


Figure 2. CGM structure: BM = begin metafile; EM = end metafile; BP = begin picture; BPB = begin picture body; EP = end picture.

Coordinate systems

The coordinates of CGM elements are called *virtual device coordinates*. VDC space is a two-dimensional Cartesian coordinate space. As we will discuss in more detail later, the VDCs of particular CGMs are highly configurable. The representation of VDC space can be integer or real, and the precision (which determines the range and granularity) can be varied. The coordinate space can even be inverted and mirrored by reversing the normal senses of the $+x$ and $+y$ directions.

Color specification

Both indexed and direct selection of color are supported in CGM. In indexed mode the color specifier is an index into a color table (CGM contains a function for defining the contents of the color table). In direct mode the color specifier is an RGB triple. RGB is the only color system supported by CGM. Other systems such as HLS (hue, lightness, saturation) are more user-friendly, but CGM is not a user-level standard, and the other systems are easily converted to equivalent RGB specifications.

The metafile elements

Tables 1 and 2 list the CGM elements. Many of the elements will be familiar to those who have worked with standard device-independent graphics systems (such as GKS). Only elements that are unusual or that are associated with major CGM features are of interest here.

As Table 1 shows, there are eight classes of elements. The first three are unusual (relative to such specifications as GKS) in that they do not describe any graphical functionality. Rather, they structure the metafile and describe its format and contents. Their function is to communicate nongraphical but vital syntactic information from generator to interpreter. It is by these elements

Table 1. CGM elements by element class.

Delimiter	BEGIN METAFILE END METAFILE BEGIN PICTURE BEGIN PICTURE BODY END PICTURE
Metafile Descriptor	METAFILE VERSION METAFILE DESCRIPTION VDC TYPE INTEGER PRECISION REAL PRECISION INDEX PRECISION COLOUR PRECISION COLOUR INDEX PRECISION MAXIMUM COLOUR INDEX COLOUR VALUE EXTENT METAFILE ELEMENT LIST METAFILE DEFAULTS REPLACEMENT FONT LIST CHARACTER SET LIST CHARACTER CODING ANNOUNCER
Picture Descriptor	SCALING MODE COLOUR SELECTION MODE LINE WIDTH SPECIFICATION MODE MARKER SIZE SPECIFICATION MODE EDGE WIDTH SPECIFICATION MODE VDC EXTENT BACKGROUND COLOUR
Control	VDC INTEGER PRECISION VDC REAL PRECISION AUXILIARY COLOUR TRANSPARENCY CLIP RECTANGLE CLIP INDICATOR
Graphical Primitive	See Table 2
Attribute	See Table 2
Escape	ESCAPE
External	MESSAGE APPLICATION DATA

Table 2. Primitive types, primitives, and applicable attributes.

Line	POLYLINE DISJOINT POLYLINE CIRCULAR ARC 3 POINT CIRCULAR ARC CENTRE ELLIPTICAL ARC	LINE BUNDLE INDEX LINE TYPE LINE WIDTH LINE COLOUR
Marker	POLYMARKER	MARKER BUNDLE INDEX MARKER TYPE MARKER SIZE MARKER COLOUR
Text	TEXT RESTRICTED TEXT APPEND TEXT	TEXT BUNDLE INDEX TEXT FONT INDEX TEXT PRECISION CHARACTER EXPANSION FACTOR CHARACTER SPACING TEXT COLOUR CHARACTER HEIGHT CHARACTER ORIENTATION TEXT PATH TEXT ALIGNMENT CHARACTER SET INDEX ALTERNATE CHARACTER SET INDEX
Filled Areas	POLYGON POLYGON SET RECTANGLE CIRCLE CIRCULAR ARC 3 POINT CLOSE CIRCULAR ARC CENTRE CLOSE ELLIPSE ELLIPTICAL ARC CLOSE	FILL BUNDLE INDEX INTERIOR STYLE FILL COLOUR HATCH INDEX PATTERN INDEX EDGE BUNDLE INDEX EDGE TYPE EDGE WIDTH EDGE COLOUR EDGE VISIBILITY FILL REFERENCE POINT PATTERN TABLE PATTERN SIZE
	GENERALIZED DRAWING PRIMITIVE	Appropriate standard attributes
	All above	ASPECT SOURCE FLAGS
	CELL ARRAY	No attributes
	All colour elements	COLOUR TABLE

that the metafile is tailored for different constituencies, and its contents announced to its recipients.

Most of the elements in the remaining classes are the familiar graphical functions that specify picture components and their appearances. GKS functions form the kernel for these elements. The basic set of GKS functions was considerably augmented to serve wider constituencies and provide functions appropriate for a low-level standard such as CGM.

Delimiters

Five elements delimit the structure of the CGM, as illustrated in Figure 2. Two of the elements have meaning beyond simply delimiting structure: all elements assume their default values upon BEGIN PICTURE. BEGIN PICTURE BODY tells the interpreter that the view surface is to be cleared if the interpreter intends to present the picture on a clean view surface.

Metafile descriptors

The elements of this class make several important declarations that apply to the entire metafile. First, the type and format of the metafile parameter data types are described: VDC TYPE is used to declare whether the representation of VDC space will be integer or real; SET <xxx> PRECISION functions are used to declare the precision of each metafile data type. The exact nature depends on the encoding, but precisions are typically a field width in some units meaningful to the encoding—bits, characters, etc. For example, INTEGER PRECISION of binary encoding gives the number of bits used to represent parameters of type integer in the metafile.

METAFILE ELEMENT LIST gives a list of all elements that might be found in the metafile (it is an upper bound, but need not be the least upper bound). Hence, inter-

preters can be told up front what resources they will need to interpret the metafile.

Part 1 of the standard gives a default value for each element for which a default makes sense. These defaults, which are the values that elements assume at the start of each picture, may be replaced for the entire metafile with METAFILE DEFAULTS REPLACEMENT. Thus, a common set of initial attribute values need not be included at each picture start.

As mentioned earlier, the color system of CGM is RGB. Abstractly the range of each component (red, green, blue) is the real range [0.0,1.0]; whereas a given encoding may represent the components with a different data type or a different numerical range (e.g., integers in binary encoding). COLOUR VALUE EXTENT defines the mapping between the abstract range and the numerical range of the encoding.

Other MD elements allow specification of the character sets and fonts to be referenced in the metafile, as well as the mechanism by which character sets will be selected.

Picture descriptors

This class contains descriptive elements that apply on a picture-by-picture basis. VDC EXTENT defines a window in VDC space in which the image will be defined (as well as allowing the mirroring and inverting of VDC space). This element plus the MD element VDC TYPE and the control elements VDC <xxx> PRECISION allow complete tailoring and customizing of the metafile coordinate space. It can be configured as an abstract normalized address range for maximum device independence. But it can also be configured to mimic the addressability of a particular target device to take advantage of particular device characteristics. In the latter case, the mechanisms of CGM ensure device independence.

CGM contains no coordinate or mapping transformations. However SCALING MODE allows the specification of abstract or scaled VDC interpretation: abstract implies that VDC may be correctly rendered at any size to display the picture; scaled specifies that VDC must have a given metric size for the picture to be properly rendered. Thus scaled mode allows for the specification of precisely sized drawings. This is the only presentation directive for interpreters that CGM contains.

COLOUR SELECTION MODE declares indexed or direct mode on a picture-by-picture basis. Mixing color modes within a picture is not allowed. <xxx> SPECIFICATION MODE allows a choice of modes for specifying such sizing elements as line width and marker size. BACKGROUND COLOUR specifies the initial view surface color for the picture and implicitly defines color index 0 if the color mode is indexed.

Controls

Elements of the control class and all remaining classes may appear anywhere within a picture body. The VDC <xxx> PRECISION elements complete the family of metafile coordinate tailoring functions. The clipping functions are formulated in a manner that differs slightly from but is designed to support higher level graphics specifications such as GKS.

A final pair of elements, TRANSPARENCY and AUXILIARY COLOUR, give access to capabilities available in some raster display devices—the ability to specify the local background color of character cells, dashed lines, etc.

Graphical primitives

The graphical primitive elements define the geometric objects that make up the pictures—lines, text, circles, etc. Table 2 presents and categorizes all graphical primitive elements. CGM contains a rich set of primitives for lines, markers, filled areas, text, and a generalized raster function.

The graphical primitive elements of GKS form the basis for this class, but GKS contains only a single element in each category. The reader familiar with both standards will also notice differences in parameterization. The parameterizations of GKS functions are designed for a user interface; those of the low-level CGM are designed on the assumption that CGM is generated at the bottom of the transformation pipeline of a system such as GKS.

CGM contains a POLYGON SET element as well as the basic POLYGON. With this element multiple disconnected polygonal regions may be defined, making it easy to define, for example, an annulus (which is awkward with the basic POLYGON).

To give users access to nonstandardized drawing functions, CGM contains (as does GKS) a GENERALIZED DRAWING PRIMITIVE (GDP). CGM goes further than GKS, however, standardizing a set of the basic geometric objects (CIRCLE, RECTANGLE, ELLIPSE, etc.), which would be GDPs in GKS. These occur often enough in available graphics devices and their value in data compression is high enough to warrant standardizing them in a low-level standard such as CGM.

CGM contains two text functions not usually found in higher level systems: RESTRICTED TEXT and APPEND TEXT. Both are supplied because the INQUIRE TEXT EXTENT function of higher level systems, which allows accurate sizing and concatenation of text strings, is not possible in a metafile environment (the generator and interpreter may be separated in time and space). RESTRICTED TEXT ensures that the displayed text will not exceed a specified area (parallelogram) within the picture. APPEND TEXT allows a text string to be built, aligned, and displayed as a single unit from several pieces. It also allows text attributes (color, font, etc) to be altered between the pieces.

CELL ARRAY is a generalized raster function, like its GKS equivalent.

Circular arc elements in CGM exist in two different parameterizations, centered and three-point. Dual parameterization is a good illustration of how CGM functionality has been driven in part by its level in the graphics hierarchy. Direct users of the standard are system implementers and tool builders, and implementation details can be critical. The two forms of arc elements differ as to where computational error is incurred during rendering. Hence, both are provided so that the implementer can choose according to the demands of the application.

Attributes

Attribute elements describe how the graphical primitive elements are to appear, for example, the color of the lines or the style used to fill areas. As with the graphical primitives, GKS attributes provided a kernel, which was extended to form the CGM attribute set. Table 2 presents the categories of primitives, the primitives in those categories, and the attributes controlling the appearance of the primitives.

CGM allows either bundled or individual selection of attributes. In individual selection of attributes, the more familiar style, each attribute is adjusted individually and all subsequent primitives are displayed according to the new value. In bundled selection of attributes, for each primitive type <xxx>, a single attribute called <xxx> BUNDLE INDEX is manipulated. <xxx> BUNDLE INDEX is conceptually an index into a table, each of whose entries contains a complete and distinct combination of the individual attributes of the primitive. Hence, the index selects all attributes at once—as a bundle.

In CGM, as in level 0 of GKS, all bundles are predefined; that is, the system implementer initially defines the contents of the bundle tables, which the user cannot later adjust. Predefined bundles are a good way to guarantee the distinct appearance of primitives in environments where applications cannot ascertain target device capabilities (such as metafile environments).

For some primitives, such as line elements, all attributes can be bundled. For others, such as text elements, some can be bundled and some cannot. Although the correspondence breaks down if pursued too far, the attributes that cannot be bundled tend to be geometric (CHARACTER HEIGHT, for example) and those that can be bundled tend to control appearance or rendering (color attributes, for example).

A set of attributes called ASPECT SOURCE FLAGS, with one ASF for each attribute that can be bundled, determines whether attribute selection is individual or bundled.

As indicated earlier, sizing attributes such as LINE WIDTH and MARKER SIZE may be specified in one of two modes. In the scaled mode, the size is a scale factor to be applied to the nominal size of the target device at display time. In the absolute mode, the size is specified in VDCs, and thus should have a fixed relationship to the rest of the defined picture (or to the window defined by VDC EXTENT) as the picture is scaled and displayed.

Most CGM attributes are patterned after similar GKS attributes, with two notable additions. CGM has a set of attributes for controlling the edges of filled areas. It also separates the concept of character set from the concept of font (typeface) and provides attributes for independently selecting character sets.

CGM has parameter values that are not standardized in GKS. For example, HATCH INDEX standardizes six basic hatch styles; INTERIOR STYLE has an empty style, and TEXT ALIGNMENT has values for continuous alignment as well as the discrete alignment of GKS.

Escapes and externals

There will always be cases in which an application would benefit from using a standard but also needs some nonstandardized functionality. It is for these cases that every graphics standard has included an ESCAPE element. ESCAPE gives the user a catch-all for specifying nonstandard functions within the rules of the standard. GDPs are intended for access to nonstandardized graphical primitives (graphical objects), and ESCAPE is intended for all other nonstandardized graphical functions, such as control functions and transformations.

CGM adds an element APPLICATION DATA, which is provided for any nongraphical purpose the user desires. For example, the element might be used to embed documentation of the picture or to embed the raw data used to generate the picture. APPLICATION DATA corresponds to the user item of GKS.

Applications

CGM has been designed to serve a wide range of applications, even though the requirements may differ greatly. For example, in some applications performance may be the most important factor, while for others the ability to communicate with other systems may be paramount.

One way CGM matches the needs of different applications is to specify different encoding schemes. The details of precisely how values are encoded can also be specified within CGM. Finally, some value types, such as color, can be specified in a number of ways, each corresponding to a large area of application.

CGM can be used in an unlimited number of ways both in centralized and distributed computer graphics systems. Three ways are described here as examples of how CGM may be applied to (and tailored for) particular types of use:

- access to graphics devices via a spooling system
- archiving of computer-generated pictures
- description of pages containing mixed text and graphics

For each of these applications, we will describe the advantages of using CGM, the relationship between the graphics system and CGM, and CGM functions that are especially useful to the application.

Each of these examples suggests a CGM generated by and output from a computer program. While this style of using the metafile is likely to be the most common, it may not always be the one used. CGM has the facilities, for example, to serve as the file format for an image input system—a scanner or digitizer that produces raster representations of scanned images.

Accessing spooled graphics devices

One of the most common uses of any graphics metafile is to transfer information from a computer program to a graphics device that is either too slow, too remote, or too busy to handle the drawing requests as fast as the program generates them. Here, as with line printer output from the earliest days, a spooling system is used. Adopting CGM as the accepted format for the spooling system is advantageous for a number of reasons:

- If character encoding is used, the format is highly compact, comparing favorably with most common manufacturers' encodings.
- If binary encoding is used, the computational effort required for generation and interpretation can be minimized, and the encoding is still quite compact.
- The large range of graphical functions available in CGM (the many primitives and all their attributes) allow much greater data compression than is typical of earlier formats.
- Data is transportable through networks that accept only seven-bit entities (assuming the appropriate encoding is used).
- Since the file need not contain any information specific to the intended device, the output can be rerouted to other graphics devices if the original device has been incorrectly specified or is out of service.

The generating program can either use or ignore any known information about the target device. For example, in a GKS system CGM could be either an OUTPUT workstation or a METAFILE OUTPUT workstation. If CGM were generated by an MO workstation, the application would be blind to the characteristics of the target device.

If the generating workstation were OUTPUT, then its workstation description table would describe the characteristics of the target device, and the information would be available to the application. The CGM generator itself could tailor the metafile contents (such as address space) to exploit known target device capabilities.

The full range of CGM functions can be used to obtain precise and compact descriptions of the pictures. The metafile descriptor elements may be set to reflect the range of facilities required in the final output device if there is any chance that the file may be rerouted.

Picture archiving

In many applications, graphics pictures are stored for periods of minutes up to several years. The shortest times are typically encountered when previewing graphical output from a batch program. Intermediate times are encountered when holding information on line that might otherwise be printed out. The longest times are for the archiving of pictures. For all these applications, the viewer needs access to the picture from any suitable device, using the capabilities of that device to the best advantage. For long-term storage the picture must still be viewable when it is restored.

The design of CGM ensures that these requirements are met.

- Pictures stored in CGM format may be completely device-independent, avoiding problems of file format and device incompatibility
- Any device-specific functions are stored in a way that allows the viewing system to skip them when their use would be inappropriate.
- Every CGM file is self-identifying, both as to originator and the version of CGM being used. This self-identification, together with the status of CGM, provides the best insurance that viewing mechanisms will still exist when CGM pictures are restored from ancient archives.

The archival CGM is normally generated in the same way as for the spooling system application described earlier. Since we are not likely to know when the picture is generated what device will be used to view it after it is restored from the archive, the metafile descriptor elements are used to record the facilities required for successful interpretation of the metafile.

Mixed text and graphics

Documents that contain text and graphics may be stored in revisable form, suitable for input to a layout system, or in nonrevisable form, output from such a system. CGM can assist at both stages.

A revisable document may contain pictures in CGM format or instructions to merge in pictures from other files in CGM format. Pictures generated by computer or input by a scanner are likely to be stored using character encoding or binary encoding; pictures could also be prepared by hand using cleartext encoding. This approach to preparing documents has been adopted by the International Organization for Standardization groups developing a standard for composite document architecture (ISO DIS 8613/1-6). The architecture currently specifies that pictorial information is to be encoded using CGM. The wide range of elements available in CGM allows efficient encoding of most types of pictures.

CGM could also be used for a nonrevisable document form. A number of approaches are possible:

- CGM could be used just for graphical information; the text would be in a different format.
- Each page of the document could be fully converted to a bitmap; CGM would be used to store the sequence of pages as pictures, each containing CELL ARRAY elements.
- CGM could be used to store both text and graphics, using the wide range of text attribute elements to obtain the different typefaces, character sets, and character sizes.

The last method allows top-quality text reproduction only if the layout system can assume a particular output device and knows all the font attributes of that device. The advantage is that the document can be previewed on other graphics devices.

Extensibility is built into CGM via such elements as APPLICATION DATA, which allows nongraphical, application-specific information to be embedded in the metafile. Any of the three methods described could use APPLICATION DATA elements to pass nongraphical directives from the layout system to a layout postprocessor or output device.

For a more comprehensive discussion of mixed text

and graphics, see the article by W. Horak referenced at the end of this article.

Other standards

There are a number of other graphical database specifications besides CGM in the computer graphics industry. We have already mentioned GKSM, a metafile specification tailored to certain needs of GKS. North American Presentation Level Protocol Syntax and Initial Graphics Exchange Specification are two other specifications tailored to and used within specialized applications sectors.

The primary distinction between CGM and these specifications is that CGM is a versatile and general-purpose specification designed to serve a wide range of applications in diverse environments. NAPLPS and IGES are tailored to specific applications. Still, compatibility between formal and de facto standards is highly desirable, regardless of the intended constituency of the specification—files always seem to find their way across the boundaries of specific application areas into other environments.

Accordingly, some of the diverse groups working on generalized and special-purpose standards are beginning to devote considerable effort to maintaining close liaison. Such liaison pays off in greater compatibility and reduction of effort, since existing work is incorporated instead of equivalent specifications reinvented. The adoption of CGM as the picture-defining protocol of the current ISO specifications for ODA, ODIF (office document architecture and office document interchange format) is a good example of the benefits of such close cooperation.

Conclusion

The Computer Graphics Metafile is a standard designed for diverse graphics environments requiring a mechanism for the capture, transfer, and archiving of pictures. All of the technical issues regarding CGM have been resolved, and the final editorial review is in progress. Follow-on work is already underway to extend CGM to better serve some needs such as GKSM, which were outside the scope of the initial effort. Thus, CGM will be the first member of an anticipated family of compatible graphical metafile standards. ■

Additional reading

Further information on the following standards can be obtained from ANSI, 1430 Broadway, New York, NY 10018, 212/642-4900.

ANS X3.110-1983 North American Presentation Level Protocol Syntax (NAPLPS)

ANS X3.122-1986 Computer Graphics Metafile (CGM)

ANS X3.124-1985 Graphical Kernel System (GKS)

ANS Y14.26M-1981 Initial Graphics Exchange Specification (IGES)

More discussion on issues related to the standards can be obtained from the following sources.

Bono, Peter. "A Survey of Graphics Standards and Their Role in Information Interchange," *Computer*, Vol. 18, No. 10, Oct. 1985, pp.63-75.

Horak, W., "Office Document Architecture and Office Document Interchange Formats: Current Status of International Standardization," *Computer*, Vol. 18, No. 10, Oct. 1985, pp.50-62.

Smith, Bradford, and Joan Wellington, "IGES: A Key Interface Specification for CAD/CAM System Integration," *Computer Graphics 84: Proc. Fifth Ann. Conf. and Expo., Vol. II*, pp.548-555.

Reed, Theodore N., "A Metafile for Efficient Sequential and Random Display of Graphics," *Computer Graphics*, Vol. 16, No. 3, July 1982, pp.39-44.

Reed, Theodore N., "Standardization of the Virtual Device Interface and the Virtual Device Metafile," *Computers and Graphics*, Vol. 9, No. 1, 1985, pp.33-38.

Lofton Henderson is an independent computer graphics consultant. From 1978 to 1986 he was the graphics project leader at the National Center for Atmospheric Research in Boulder, Colorado. Prior to 1979 he was an applications programmer there. His interests include graphics for scientific data representation, metafiles and distributed graphics, user interfaces, and graphics standards.

Since 1980 Henderson has been a member of technical committee X3H3 working on ANSI graphics standards and since 1982 has been a member of the U.S. delegation to the ISO Working Group on graphics (ISO TC97, SC21, WG2). He is CGM technical editor for both ANSI and ISO. He received an AB in mathematics from Cornell University and an MS in applied mathematics from the University of Colorado. He is a member of ACM SIGGRAPH.

Margaret Journey is manager of Engineering Service for Precision Visuals. Her interests include computer graphics software systems and programmer productivity tools. From 1979 to 1985 she was a member of ANSI X3H3 and from 1983 to 1985 chaired the X3H3.3 task group for CGM and Computer Graphics Interface (CGI). She received a BA in mathematics from the University of Colorado.

Chris Osland is head of the computer graphics section at Rutherford Appleton Laboratory in the U.K. He was involved in pioneering work on distributed computer graphics at the London Institute of Computer Science prior to moving to RAL in 1971 to work on operating systems and computer graphics. From its formation in 1982 until July 1985 he chaired the subgroup of ISO TC97 SC21 WG2, which developed CGM within ISO. He received a BA in physics from Oxford University.

Questions about this article can be directed to L. Henderson, PO Box 4036, Boulder, CO 80306.

APPENDIX 3

X3H3 LIAISON DOCUMENT ON DRAFT TOP AP

Accredited Standards Committee
X3, INFORMATION PROCESSING SYSTEMS*

Doc. No.: X3H3/87-39 (cover)
Date: 4 February 1987
Project:
Ref. Doc.: TOP TC010
Reply to: Dr. Peter Bono
Bono Associates
PO Box 648
Gales Ferry, CT 06335

Tom Haug
TOP Technical Review Committee
Boeing Computer Services
M/S 7C-16
PO Box 24346
Seattle, WA 98124-5720

Dear Mr. Haug:

Thank you for extending the closing date for comments on the TOP Specification Change: "TC010 Graphics." We appreciate this opportunity to comment. Because almost all major graphics suppliers and many graphics users are represented on X3H3 (the current membership of X3H3 is over 80 members), we are confident that our reviewing TC010 will result in greater consensus and compliance to TOP version 3.0.

During the week of January 26, a group of X3H3 experts spent over 80 man-hours reviewing in detail this TOP Change. Overall, we have found very few technical details that we recommend be changed. Our aim in doing this review was to follow closely the apparent intent of the document, correcting obvious mistakes and "tightening up" the specification so that adherance to TOP 3.0 will achieve the goal of completely predictable results when interchanging graphics pictures.

Most of the recommended changes in the TOP-CGM profile are editorial and organizational. In particular, we recommend replacing many pages which duplicate the CGM with a short description of how the TOP-CGM profile *differs* from the full CGM standard. This change, along with explicitly pulling out the Conformance and Defaults to their own sections, will make the pertinent information easier to find, and eliminate many of the transcription errors we found in the current document.

There are a number of recommended technical changes as well. These include:

1. The "basic set" of parameter values should be limited to those values with standardized meanings (e.g., linetype 1 to 5) — private values should not be allowed in the basic set.
2. Floating point reals should be added to the basic set, since they are already required to decode the SCALING MODE parameter.
3. The defaults for COLOR PRECISION and COLOR INDEX PRECISION should be changed to 8.
4. The right-hand portion of the "Latin 1 Character Set" (ISO 8859/1) should be added to the basic CHARACTER SET LIST.

5. Elements that only had one legal basic value, that being the default, should be eliminated from the basic set — this includes PATTERN INDEX and PATTERN TABLE.
6. Most of the proposed GDPs and ESCAPES should be withdrawn from the proposal so that wider consensus on the proper formulation and functionality may be attained. The 'disable clear' and 'viewport' were left in, in the latter case with minor changes to track the current CGI and extended CGM documents.
7. Encoded parameters in data records should be separated by blanks.
8. The predefined fill bundles should be all hatch and vary by hatch indexes 1 to 5.
9. The default foreground and background colors should be specified as being "nominal" (as in CGM) rather than black (0,0,0) and white (1,1,1). Black and white should be explicitly added to the default color table. The default color table is included in the Metafile Defaults Replacement, for indexes 0 to 9, with repetition of 2 through 9 to fill the table implicitly.

On Friday, January 30, 1987, Accredited Standards Committee X3H3 unanimously approved the detailed revisions suggested for TC010 attached to this letter.

Sincerely yours,

Peter R Bono

Dr. Peter R. Bono
Chair, X3H3
Technical Committee on Computer Graphics

cc: Sylvan Chasen, Chair, TOP Graphics Subcommittee
Lockheed-Georgia Co.
Dept. 72-92, Zone 419
Marietta, Georgia 30063

Alan Francis, CGM Rapporteur
Cyclops Group
Open University - Walton Hall
Milton Keynes MK7 6AA
ENGLAND

X3H3:
Debby Cahn
Frank Dawson
Andrea Frankel
Lofton Henderson
John McConnell (for X3H3 distribution)
Tom Powers
Barry Shephard

Section 3.4

On page 8, reword the paragraph:

> "The CGM is designed to be interpreted in one of two ways: firstly, by a special application program that, in turn, invokes a device independent graphics system to render the CGM; alternatively, the device independent graphics system may have application callable functions to get, read, and interpret metafile elements, using the facilities of a CGM metafile input workstation."

Conformance

The conformance statement in the second paragraph of 3.1 needs clarification and strengthening, and should be relocated. Delete the paragraph at its current location. Retitle 3.5 as "Conformance" and insert the following text at the beginning of the section:

> "The application profile specifies conformance in terms of of PERMISSIBLE and BASIC values. Permissible values are the range of values of CGM elements as specified in ISO 8632. Basic values are a subset of the permissible values. For example, permissible values of linetype include all non-zero integers, while basic values include the standardized linetypes 1 to 5.
>
> Any legal CGM is a legal TOP metafile. TOP defines a conforming "basic metafile" to be one that contains no elements or parameter values outside of the basic set, and that obeys any noted restrictions on the presence of ESCAPE elements. A conforming "basic interpreter" is one that at least correctly interprets any conforming basic metafile (and may have more capability as well). A conforming "basic generator" is one that produces only conforming basic metafiles, or can reliably be directed to function in the mode of producing basic metafiles.
>
> In addition, any TOP metafile interpreter should correctly parse and pass over any elements that it does not support and any parameter values that it does not support."

In the current first paragraph of 3.5, replace the last sentence with:

> "This application profile is an application profile for the CGM Binary Encoding, ISO 8632/3. Future application profiles may be developed for the other encodings of CGM."

Delete the page references in the last sentence of section 3.5; they may not be correct in the final ISO text.

Section 3.6

Note: The following text is intended to replace the current section 3.6.

The Basic Set is defined by the limitations on Basic Values noted below. Where an element is not mentioned, it is implied that the Basic Set includes all values permitted in the CGM.

3.6.1 Delimiter Values

3.6.2 Metafile Descriptor Elements

Element	Basic Values
INTEGER PRECISION	16
REAL PRECISION	(1,16,16) (*fixed*)
	(0,9,23) (*floating point*)
INDEX PRECISION	16
COLOUR PRECISION	8, 16
COLOUR INDEX PRECISION	8, 16
FONT LIST	*Refer to clause 4*
CHARACTER SET LIST	(0, 4/2) (*ASCII*)
	(1, 4/1) *Note 1*
CHARACTER CODING ANNOUNCER	0 (*Basic 7-bit*)
	1 (*Basic 8-bit*)

Note 1: This set is named "Right-Hand Part of Latin Alphabet Number 1", contained in ISO 8859/1, ECMA 94, and ANSI X3.?.

Note 2: We also suggest that implementors use the METAFILE DESCRIPTION element's string to include a brief identification of their company or product, so that interpreters can account for known idiosyncrasies of generators. TOP may also wish to establish a string (e.g., "TOP/BASIC") which labels the metafile as conforming to this profile. The metafile may always be interpreted without using the contents of the METAFILE DESCRIPTION; its use is optional.

3.6.3 Picture Descriptor Elements

Note that the scale-factor parameter of SCALING MODE is always a floating point number, even when REAL PRECISION has selected fixed-point for other real numbers. This is not an error — a floating-point parameter is needed for some situations where low precision fixed-point reals would not encompass the range at all (for example, scaling a plot done with 32-bit integer coordinates onto a 1-meter piece of paper) and for other situations where using a fixed-point scale factor would produce unacceptable loss of resolution.

3.6.4 Control Elements

Element	Basic Values
VDC INTEGER PRECISION	16, 32
VDC REAL PRECISION	(1,16,16) (*fixed*) (0,9,23) (*floating point*)

3.6.5 Graphical Primitives

To ensure portability and predictable results, TOP-conforming metafiles may contain only those GDP elements that are defined in TOP profiles.

3.6.6 Attribute Elements

The PATTERN TABLE and PATTERN INDEX elements are not included in the Basic Set, as patterned fill is not able to be supported easily in a device-independent manner (e.g., on pen plotters). The default pattern table is described in the CGM as having one entry, a solid "foreground colour" in the first position; therefore, selection of interior style 'pattern' results in solid fill.

Element	Basic Values
LINE BUNDLE INDEX	1-5
LINE TYPE	1-5
MARKER BUNDLE INDEX	1-5
MARKER TYPE	1-5
TEXT BUNDLE INDEX	1-2
TEXT FONT INDEX	*Refer to clause 4*
CHARACTER SET INDEX	1-2
ALTERNATE CHARACTER SET INDEX	1-2
FILL BUNDLE INDEX	1-5
HATCH INDEX	1-6
EDGE BUNDLE INDEX	1-5
EDGE TYPE	1-5
COLOUR TABLE	start index 0-1023

3.6.7 Escape Element

To ensure portability and predictable results, TOP-conforming metafiles may contain only those ESCAPE elements that are defined in TOP profiles.

3.6.8 External Elements

The 'action required' flag of the MESSAGE element is restricted to the value 'no action required'.

In line with the recommnendations for simplifying and restructuring 3.6, add this material on defaults after section 3.6:

3.7 TOP-CGM Defaults

The CGM specifies a complete set of defaults. In a few cases, these defaults are not appropriate for TOP requirements. However, any TOP metafile must be a legal CGM, including implicit defaults, thus each deviation requires that the affected element either:

1. Appear in the METAFILE DEFAULTS REPLACEMENT element, or
2. be explicitly specified for its value to be applicable.

Therefore, each TOP metafile shall contain in the Metafile Descriptor a METAFILE DEFAULTS REPLACEMENT element that includes (at a minimum):

1. TEXT PRECISION element; Precision = 2 (stroke).
2. COLOUR TABLE element;

Index	Values	Meaning
2	(255,0,0)	Red
3	(0,255,0)	Green
4	(0,0,255)	Blue
5	(255,255,0)	Yellow
6	(255,0,255)	Magenta
7	(0,255,255)	Cyan
8	(0,0,0)	Black
9	(255,255,255)	White

This sequence of colours is implicitly repeated, thus 256 default colour indices are specified. Color table defaults for colour indices 0 and 1 are explicitly defined in the CGM standard as corresponding to the nominal background and nominal foreground colours, respectively.

3. CHARACTER SET LIST element; (0, 4/2), (1, 4/1)

It is not apparent in the CGM standard what the default value for the precsion of the floating point real parameter of SCALING MODE should be. TOP generators and interpreters should assume that the real precision for this parameter is (0, 9, 23). If a precision other than the default is desired for this floating point parameter and if the rest of the reals in the metafile are to be fixed point, then the mechanism described in the CGM standard, clause 5.3.12, for setting defaults for multiple mode parameters should be used.

Section 4

We strongly recommend that all of section 4.2 GDPs should be withdrawn for now. The need for the particular GDPs chosen and their specification has not been adequately reviewed, and serious technical problems have been identified with the ones proposed.

For the same reason, delete the ESCAPEs specified in 4.3.1 through 4.3.5. More general and useful proposals for algorithmic specification of linetype and hatch style are now

being formulated within X3H3 and should be available for an early addendum to the TOP profile.

In the "viewport" ESCAPE, the parameter should be 'fraction', real between 0.0 and 1.0, not percentage. 'Fraction' is what is now in CGI, and this formulation should be compatible. Add the following statement:

> "A TOP-conforming metafile may include the "viewport" ESCAPE only if it does not include the SCALING MODE element with the value 'metric'.

The encoding of data into ESCAPE and GDP data records should have better readability. Specify that at least one blank shall separate parameters.

On p.31, 4th line from the bottom — metafile description should be Metafile Descriptor. The next sentence should read "That is, between the BEGIN METAFILE and the first BEGIN PICTURE."

Section 5

Section 5.1, insert at the beginning:

> "Unless otherwise noted in this application profile, all of the guidelines of CGM Annex D shall be adhered to by TOP generators and interpreters. In particular, the interpreter minimum capabilities of D.5 should be be the minimal capability of a basic-conforming TOP interpreter, unless richer capabilities are specified in the "basic set" of section 3.6 of this TOP application profile. The interpreter fallback actions, such as those for APPEND TEXT, are to be applied as well."

Section 5.1: METAFILE DEFAULTS REPLACEMENT — use "shall" in place of "will". APPEND TEXT — delete — it's covered in Annex D. COLOUR TABLE — change "indeterminate" to "unspecified".

Section 5.2, change the categories "Default:" to "Basic value:".

Section 5.2, Maximum string array length — a data record is a single string, not an array of strings. The only place where we can see that this applies is FONT LIST, whose datatype is nS.

Section 5.2, Maximum string length — see previous comment. 256 is inadequate for data records, since they are single strings. Suggestion: "1) 256 for all strings but data records; 2) 32767 for data records."

Section 5.2, for the Fill area bundle, using hatch interior style for all entries and varying the hatch index makes much more sense — use hatch indexes 1 through 5.

Section 8

Section 8.0, the font working group in SC18 is WG8. Check and correct the references.

FINAL REPORT

CALS SOW TASK 2.2.1.2.1

INJECT CALS REQUIREMENTS INTO EXTENDED CGM (CGEM) STANDARDS WORK

TABLE OF CONTENTS

TABLE OF CONTENTS (continued)

I. PURPOSE

Inject CALS requirements for extended CGM (CALS SOW Task 2.2.1.2.1). Work is just beginning to develop a more powerful and consolidated metafile for graphics picture transfer. This metafile would allow much more substantial modifications to be made on pictures being transferred and would be more compatible with existing graphics standards (GKS, GKS-3D, PHIGS). It is imperative that CALS requirements with respect to picture modification be input to this effort in its early stages of development. This task is a continuing effort, and is partially accomplished by representing CALS in the standards effort to design the extended CGM.

This work was accomplished through a contractor, Mr. Lofton Henderson, of Henderson Software, who is a member of the following:

- the Accredited National Standards Committee X3H3 on Computer Graphics Standards;
- the sub-committee, X3H3.3, Computer Graphics Metafile, and Computer Graphics Interface (CGI);
- the U.S. delegation to the International Standards Organization Working Group 2 (ISO/WG2), Computer Graphics Standards;
- the ISO/WG2 CGM Rapporteur Group, responsible for processing comments, refining the standard, and issuing interpretations of the standard; and
- the ISO/WG2 sub-group developing the Extended Metafile standard.

As is apparent, Mr. Henderson is in a unique position to ensure that CALS requirements get addressed and injected into the extended metafile (CGEM) work at the national and international levels. He will be referred to in the remainder of this report as the NBS representative, which serves to properly identify the role that he has played in furthering, under NBS direction, the needs of CALS in the development of CGEM.

II. BACKGROUND

After six years of deliberation, circulation, balloting, and refinement the Computer Graphics Metafile (CGM) is now a standard. It became an ANSI standard (X3.122-1986) on 27 August 1986 and was published a few weeks later; it received final approval from ISO WG2 in September 1986 and its publication is just completed (ISO 8632/1-4 1987). It became FIPS 128 in March 1987.

The excessively long time period for completion of the standard is typical of the standards making process. It is due in part to

the time required to resolve the conflicting interests of the diverse constituencies that participate in the standards process --ANSI and ISO standards are by consensus and compromise among many factions.

Another consequence of the consensus process is that the standard tended to be a **least common denominator** graphical metafile for the various constituents. It is, to a large degree, the area of overlap that all participants in its formulation agreed should be in a graphical metafile. As a result, it is functionally lean (although not as much so, in primitives and attributes, as other standards like GKS). There was in fact a faction that believed that this is exactly what the first standard metafile should be, servicing common needs of diverse constituencies, with sufficient overlap and leanness that processing could be completed relatively quickly.

The disadvantage of a lean CGM is that it is difficult to use the CGM **efficiently** in some application environments. Much useful additional functionality, particularly to support clients such as GKSM, technical illustration and publishing, and compound document exchange, was proposed for CGM during its formulation. Most of the proposals were deferred or deflected. The ANSI and ISO committees decided that the lean **first generation** CGM should be standardized as quickly as possible, and an addendum or extension process should immediately commence and begin sorting through the proposals to enrich CGM functionality in the direction of the requirements of more advanced metafile applications. (This agreement to start on CGEM immediately was the deal that finally unblocked much European opposition to CGM, due to perceived deficiencies, and allowed CGM to become and ISO standard.)

This approach was first defined in the ISO meeting at Timberline, Oregon in July 1985. Technical work producing a working draft was carried forth at an ISO meeting in Egham, England in September 1986. At this meeting it was also decided to process the extensions, known as Computer Graphics Extended Metafile, as an addendum to CGM (ISO 8632); the addendum process is the fastest processing path through the ISO standards labyrinth. Two addenda were actually identified: Addendum 1 extending CGM sufficiently to serve as a GKS Metafile; Addendum 2 extending the metafile to 3D.

The Working Draft was circulated for international comment and balloting in early 1987, with the intention that the national standards bodies (e.g., ASC X3H3 in the US) should generate positions and submit comments. The plan was that the comments be processed at an ISO/TC97/SC21/WG2 meeting in France in May 1987, and the CGEM addendum be advanced to Draft Proposal (DP) status at that meeting.

ACRONYMS AND TERMS

ASC X3H3 Accredited Standards Committee X3H3, the ANSI accredited committee responsible for computer graphics standards in the US.

X3H3.3 The subcommittee of X3H3 that is responsible for CGM and CGI.

ISO TC97/SC21/WG2 International Standards Organization, Technical Committee 97, Standing Committee 21, Working Group 2, the international counterpart to X3H3.

Working Draft (WD) The first complete draft of a proposed ISO standard, the starting document for subsequent work and review.

Draft Proposal (DP) The second stage in the ISO processing pipeline. After national bodies have commented on the WD, it is altered and refined and then registered as a DP. Another round of ballot and comment takes place on the DP.

CGM Computer Graphics Metafile, ANSI standard X3.122-1986 and ISO standard ISO 8632/1-4 1987.

CGI Computer Graphics Interface, another ANSI/ISO standards project, currently at the DP stage, which exists about at the level of the CGM in the graphics pipeline (device level).
CGI is an interactive (input) and highly extended and enriched interface specification, whereas CGM has output-only functionality (for picture definition) and is a picture description protocol (a graphical database). CGI embeds CGM output functionality as a subset.

GKS Graphical Kernel System, an application programmer interface to computer graphics, now an ANSI and ISO standard.

GKSM A metafile for use with GKS. One was proposed in non-standard Annex E of GKS. Work on it was deferred in favor of CGM, and now of extended CGM (CGEM).

CGEM Computer Graphics Extended Metafile, a set of addenda and extensions to CGM, being processed by ISO, currently nearing DP stage.

BSI British Standards Institute, the British equivalent of ASC X3H3.

DIN The German equivalent of ASC X3H3.

AFNOR The French equivalent of ASC X3H3.

III. DISCUSSION

1.0 CALS Requirements, CGM, and CGEM

CALS has identified the CGM one protocol for capture, archival, and transmission of computer generated vector graphics in technical illustration and publishing applications. Under another NBS task (2.2.3.3.4), a CGM application profile (AP) for CALS has been developed -- it removes the ambiguities in the standard, limits use of "private" information (e.g., private linetypes), and puts conformance requirements on generators and interpreters. The AP both makes the results of using CGM predictable for picture interchange, and makes conformance testing of generators and interpreters possible.

In the reports of two other CALS tasks, the need for greater functional richness in CGM has been identified. In particular, the Registration report (CALS Task 2.2.2.2.2) recommends specific functional extensions and has injected these into the Graphical Registration process of ISO; the functionality will be attained initially by registered ESCAPE and GDP elements.

While this will serve the short term needs of CALS, it is not the most desirable solution. These escape elements are private, and will not in general be recognized outside of the CALS community. It is therefore highly desirable that such functionality get standing through an official standard. The CGEM addenda, 1 for 2D functional extensions and 2 for extension to 3D, are the obvious place to achieve this. This is the reason for this task.

The original identified scope of Addendum 1 was to be limited to that functionality necessary to directly support GKSM (the metafile of GKS). This meant basically the addition of segmentation and settable bundles, and not much more. The key formative stages of the scope and functionality of CGEM were late 1986 and early-mid 1987. This was the proper time for the needs of CALS metafile users to be made known in the standards process.

The NBS representative was the document editor for both the ANSI and ISO versions of CGM (they are identical in content but differ some in formatting),and has been the leader within ASC X3H3 of the processing of CGEM. He has led the metafile sub-delegations at the last several ISO WG2 meetings, which has put him in an excellent position to bring the needs of the CALS constituency into the standards making community.

The approach of this deliverable is to report the representative standards processes and ensuing results beneficial to a more robust transfer within a CALS environment.

2.0 Specific CALS Objectives for Metafile Extensions

CALS requirements for extended metafiles can be summarized into a few broad categories. These categories are defined in terms of the major issues that were dealt with at the Tulsa and Valbonne meetings.

2.1 Broaden Scope

The original scope of the CGEM was tightly limited to supporting the requirements of GKSM (a GKS metafile). The additional functionality basically consisted of the GKS segmentation model and settable bundles, as well as a few additional control functions.

As long as the scope was limited in such a manner, it would not be possible to include the sorts of additional functionality required by CALS. This includes symbol libraries, spline drawing primitives, additional drawing control attributes (line cap, hatch styles, ...); the list is detailed in the final report of the Registration task.

It was high on the CALS priority list that the CGEM should be more general and serve broader constituencies.

2.2 Symbol Libraries

The CALS need for a symbol library facility has been previously identified in other CALS tasks. This was an important issue at Tulsa, and the outcome was a US position in favor of such a facility. Such a facility is important in technical illustration because it allows definition once and for all of the symbols common to engineering and technical drawing, and then instancing them in pictures from the single definition.

The (at that time current) segmentation proposals did not support such a facility.

2.3 Additional Functionality

There were several possible sources of additional functionality specifications for inclusion in CGEM. One of the most obvious was new drawing functionality in CGI. Other sources included some of the PDL specifications (e.g., PostScript), and the collected recommendations of the CALS/NBS study mentioned above.

Another source of additional functionality was CGI, and included mainly the CLOSE FIGURE primitive and drawing modes.

2.3.1 Additional Drawing Controls

A richer set of drawing functions and controls needs to be included in CGEM to serve technical illustration and publishing needs. Many defacto PDL (page description language) standards such as postscript already contained such. A list of needed functionality would include:

1. user defined linetype;
2. user defined hatch style;
3. a number of additional linetypes;
4. a number of additional hatch styles;
5. several types of spline curves;
6. conics and conic arcs;
7. closed figure primitive;
8. arbitrary clipping boundary;
9. a number and variety of fonts;
10. a completely new text model;
11. raster "input" and associated attributes for image processing;
12. additional line attributes, e.g., line cap, line join ...

2.3.2 Better Text and Fonts

Publication quality graphics cannot be done without a varied repertoire of standard, commonly available fonts, and better methods of text composition and placement.

3.0 The CGEM Working Draft

Appendix 1 to this report contains a copy of the CGEM Addendum 1 Working Draft, as produced at the Egham meeting and circulated for national comment. The draft is a little hard to read, as it is in the form of an additions document to the original CGM text. This is as per the ISO rules for addenda to standards. It also serves the critical need of insuring that nothing in the existing CGM is changed -- addenda can add to but not change their base standards. This has been a critical requirement of the US from the beginning of the CGEM effort.

A comparison of CGEM Addendum 1 with GKS shows strong similarities, in fact much of the text was lifted directly from GKS (without even revising the wording to be appropriate for a metafile standard). This is as per the original identified scope --limit the CGEM to functionality required for GKSM support. There is also a key target of constituents like CALS, that the scope must be broadened.

4.0 Preliminary U.S. Processing of CGEM Issues

4.1 Background -- LB-47 and LB-49, and the Ft Collins Meeting

In December 1986 the Working Draft was circulated within the membership of ASC X3H3 for voting "on the suitability for registration as a DP at the May 1987 ISO WG2 meeting". The real purpose of the ballot was to solicit comments and issues from US graphics experts, to be used to formulate the US position for the WG2 meeting and progress the document to DP status.

These ballot results were in hand for the ASC X3H3 meeting at Ft. Collins, late January 1987. A working group of 5-8 (varying membership), led by the NBS representative, processed the results of this ballot. The output of this working group was a document containing:

1. additional arguments and clarifications to the existing ISO issues log for CGEM;

2. generation of a set of new ANSI issues on CGEM.

The identification and logging of issues was critical in several respects. Basically, the open issues were to be submitted to ISO (without positions or recommendations) as the US national comment. This served the purpose of placing the issues on the agenda for discussion. Officially, it is a WG2 rule that technical topics cannot be discussed and resolved unless they have been put on the agenda and pre-circulated to the membership -- in reality this rule gets ignored but the US metafile experts assumed it would not be and wanted to have their concerns efficiently dealt with.

In addition, the issues were used to develop the US positions for the ISO WG2 meeting. The combined issues log, containing 23 ISO issues and 28 new ANSI issues, was packaged with some explanatory text and circulated to X3H3 membership for voting. This package is contained in Appendix 2 to this report. The ballot results (contained in Appendix 3) were in hand at the start of the ASC X3H3 meeting in Tulsa during the last week of May 1987.

Processing of issue ballot results at the Tulsa meeting was a strategic point for CALS requirements to be represented. This is particularly so because the needs of CALS and similar constituencies had not been strongly represented in CGM/CGEM work to date, in the formulation of the scope of CGEM and generation of the initial (ISO) issues set.

4.2 Issues Important to CALS

The following issues are considered important to CALS (the issues log from which the following discussion is derived can be found in its entirety in Appendix 2 of this report).

ANSI.6 (Also CGMA2). The resolution of this issue favorably would enable a symbol library capability in CGEM; this has been identified as a critical CALS requirement.

ANSI.1 This is a conceptual issue with some important and far-reaching implications. Some have suggested that the CGEM should just be a syntactic framework, others that semantics vary by category. NBS feels the first makes CGM results unpredictable, and the second would complicate conformance checking.

ANSI.4 This opens the way for CGEM to properly support the segmentation and symbol needs of diverse clients, including CALS, rather than just being limited to GKSM.

ANSI.14 The need for better font definition facilities has been identified for CALS. This is a good idea at some time, but the "premature" argument may be compelling because of the newness and instability of the work.

ANSI.15 This is very important for opening up additional useful functionality, such as CLOSED FIGURE, that exist in CGI now and were added after development of CGM was frozen. It could also open the door to additional useful functionality (splines, line cap, hatch styles, ...) that are not in CGI but are needed (as per other NBS studies mentioned above).

ANSI.25 NBS strongly feels that conformance should be handled as in CGM, by application profiles.

ANSI.26 Once again, proper resolution will free CGEM from the functional limitations of GKSM and allow it to grow in the direction of CALS needs.

The rest of the issues are of a more detailed technical nature. Their resolution is important from the standpoint of a good, consistent, and usable standard, but they do not have the importance to CALS as the above.

One issue with some import for CALS, that did not make it into the official issues list, is the nature of the 3D extensions. CALS needs here are not clear as yet.

4.3 Processing LB-49 at the Tulsa Meeting

Copies of all letter ballots and comments were brought to the Tulsa X3H3 meeting, along with a tally and classification of comments. A working group of 4-5 worked on ballot results processing -- two representatives of McDonnell Douglass Corp, one representative from MagiCorp, and sometimes a representative of Peter Bono R. Associates. The NBS representative led the subgroup. The goal of the subgroup was to have all issues closed, i.e., have positions or recommendations on all issues.

First, issues with clear consensus were identified and declared closed. On issues that were more badly split in the vote the subgroup deliberated and came up with recommendations from the majority of the subgroup, to the X3H3 membership at large. Past experience showed that these would most likely be accepted.

On a small handful of "closed" issues and issues with a majority leaning one way, the subgroup disagreed with the majority position and recommended for reversal. It was the subgroup's position that the issue was not being properly understood (some were badly worded or lacking arguments), or that there were significant new arguments that recommended against the majority position. A couple of these were important to CALS, such as ANSI.6 (the important capability for a symbol library).

In summary, the group processed 51 issues. 32 were closed by clear majority. 19 either did not have a clear consensus, or there were new arguments and the majority position was reversed in the recommendations.

These recommendations were presented to X3H3.3 for endorsement. There was some discussion on the important ones, and the recommendations were unanimously endorsed. The NBS representative then presented the recommendations to X3H3 plenary, and after some discussion they were unanimously endorsed again. This then became the US position for the ISO WG2 meeting at Valbonne.

The following subsections review results that were achieved during the week in Tulsa that were favorable and important for CALS constituents.

4.3.1 Scope

A number of issues were resolved that broaden the scope of CGEM beyond the limited scope from the WG2 Egham meeting. In particular, the US position at the May 1987 WG2 meeting was for alignment with CGI when possible, so that more diverse client applications could be supported than would be possible if CGEM were required to have a 1-to-1 relationship with GKSM.

Subsequent subsections detail other important issues of scope,

such as added functionality beyond that needed to support GKSM. The broader scope comes with the proviso, however, that it not adversely impact the schedule of Addendum 1. This is a reality that comes from the impatience of (particularly European) constituents to have GKSM support as soon as possible.

4.3.2 Symbol Libraries

It was a reversal of a majority position that determined there should be a symbol library facility in the CGEM. This is the issue of where segments are defined and from where they are referenceable. GKSM implies it should only be within a picture (there is actually no well-defined concept of randomly accessible pictures in GKSM, as there is in CGM and CGEM). In the existing segment model, segments disappear at END PICTURE. Such a segment model cannot be used to support symbol libraries. For issue resolution the US position is that segments should also be definable in some place like the Metafile Descriptor and then referenceable from any picture.

4.3.3 Additional Stable CGI Functionality

It was resolved as part of the US position that additional stable functionality from CGI, beyond that required to support GKSM, should be adopted into CGEM. This applies to additional output primitives (e.g., CLOSED FIGURE), primitive attributes, control functions, drawing mode, etc.

4.3.4 Additional Drawing Functionality

This is the issue of additional functionality, needed by CALS but not yet in CGI or other standards. Such elements as splines, additional hatch and linetypes, additional attributes (e.g., line join), user definable hatch styles, etc., are included here. This was not raised and presented as a specific issue in the balloting, i.e., there was no specific issue on LB-49. But there was interest among X3H3 and the US delegation. It was agreed informally by the metafile sub-delegation to look for an opportunity to advocate for such in CGEM at Valbonne in May.

4.3.5 Fonts and Text

There was an issue about incorporating the superior (to the CGM/GKS text model) font specification techniques being developed by SC18. It was voted and recommended **not** to try to incorporate such at this time. The proposals themselves are still in development at this time, and it was not believed to be a good idea to attach the work before it is stabilized. There is keen

interest in this work however, and at some point in the near future it may be appropriate to try to incorporate it into CGEM.

4.3.6 3D Metafiles

Once again there was no specific issue on LB-49 dealing with what form of 3D support is desirable, needed, or achievable. There was a position paper circulate by BSI (British Standards Institute) as a proposal for the Working Draft of Addendum 2. This was basically exactly the 3D GKSM, copied straight from GKS-3D. There is considerable uncertainty as to what the role of a 3D metafile is. What need is to be fulfilled? Simple 3D primitives? Full viewing? What coordinate system in the 3D pipeline?

There has been relatively little interest in 3D metafiles in the US, or at least among 3D graphics experts. Most of the interest is from Europe, and most of that because of 3D GKS. After presenting the results to X3H3 for endorsement, the NBS representative called for participation in an ad hoc meeting, by X3H3 members interested in 3D (GKS, PHIGS, ..). In a one hour meeting no consensus could be reached. It was decided the position was not to initiate any 3D work at the WG2 meeting, but to respond critically to other proposals and evaluate and choose the best.

The CALS requirements are not clear here. To provide some support for IGES-to-CGEM translation, probably a rudimentary 3D system of any kind would do. That is, a GKSM-3D or a PHIGS archive or a more general 3D metafile supporting varied 3D clients. The simple minded metafile of Annex E of GKS-3D is too specific to GKS and too limited, and that a more general proposal should be supported if one were forthcoming.

This is an area where more work is needed on CALS behalf, both within ASC X3H3 and within ISO.

5.0 The Valbonne Meeting

The results of the Tulsa meeting became the US positions for the ISO WG2 meeting. The detailed minutes of the metafile subgroup at the ISO WG2 meeting in Valbonne are contained in Appendix 4 of this report. A summary is presented here.

The major activity was the processing of national-body comments on the Working Draft of CGEM, which was circulated along with the initial ISO issues log in late 1986 and early 1987. X3H3 had two ballots on these documents. The first was on the suitability of the working draft for registration as a DP. The comments that the US submitted during the ISO Working Draft comment period

consisted mainly of an additional 28 new open issues which were identified during the processing of the comments on this X3H3 letter ballot.

On the second X3H3 letter ballot, each of the open ISO and ANSI issues was voted. The results were processed at Tulsa to develop the US position for Valbonne.

The CGEM Rapporteur Group (or metafile working group) consisted of 2 US members, 1 UK delegate, 1 from France, 3 from Germany, 1 from Italy, and 1 from Japan. This group worked on a number of major technical areas during the meeting:

1. Scope of Addendum 1:
   a. incorporation of additional stable functionality from CGI;
   b. incorporation of additional functionality to begin addressing the needs of technical illustration and publishing;
2. Relationship of CGEM and GKSM -- 1-to-1, or is a well defined and "reasonable" mapping sufficient;
3. Relationship of CGEM and CGI;
4. What segment model should CGEM use;
5. Resolution of all open issues;
6. Scope of Addendum 2 (3D).

5.1 Scope of Addendum 1

Everyone agreed that GKSM support was the minimal requirement of Addendum 1. It was also generally agreed (UK dissenting, with the position that CGEM should not be expanding its scope and generating new work) that the slowness of the standards-making process made it highly desirable to include additional stable functionality, for support of constituents other than GKSM users, where such functionality could be identified. This endorsement of wider scope for Addendum 1 was with the proviso that consideration of such functionality not slow down the processing of Addendum 1.

Specifically, the CGEM group endorsed the inclusion of "additional stable output functionality of CGI," and the HOD/RAP (Heads of Delegations and Rapporteurs, a sort of "steering committee" overseeing the working groups) concurred.

Early in the meeting the US added to the agenda consideration of functionality for better support of user requirements in technical illustration and publishing (e.g., ODA/ODIF) applications. This includes such functionality as additional hatch and patterns, splines, and attributes like line cap, line join, etc., and such capability as "symbol libraries" or "global segments." The argument was made that CGM and CGEM were in

danger of being ignored by certain important constituencies if their needs were not addressed.

There was general support in the CGEM group for looking at such needs during the meeting, both additional drawing controls and a symbol or segment library capability (UK dissenting again). This was loosely termed "Addendum 3." With the exception of UK, all delegations in CGEM placed Addendum 3 at higher priority than Addendum 2. HOD/RAP advised CGEM, however, to discontinue consideration -- an ad hoc group of interested parties from within WG2 should be convened to draw up user requirements, and perhaps a project could be organized at the next WG2 (SC24) meeting.

The US metafile experts and NBS feel that this is a critical issue for continued acceptance of CGM/CGEM in the US. To that end NBS has directed the NBS representative to generate a proposal for an additional addendum to CGM on user requirements for metafiles in technical illustration and publishing applications (see Appendix 7).

The CGEM group decided that a symbol library capability could be provided with no additional elements beyond those already being added for GKSM support. A proposal was written up and issues generated and resolved by a sub-group led by the NBS representative. This was unanimously endorsed by CGEM for inclusion in Addendum 1. A copy of this proposal is in Appendix 5 of this report.

Basically, segments may be defined in the Metafile Descriptor (these are Global Segments) as well as in picture bodies (Local Segments). Segments defined in the MD may be referenced (via COPY SEGMENT) from any picture in the metafile. All issues pertaining to attribute binding, legal operations on Global Segments, etc., were addressed and resolved. Global Segments should be a valuable tool for the technical illustration constituency, and this result is in accord with the X3H3 position from Tulsa.

5.2 Relationship of CGEM and GKSM

A major conceptual issue for Addendum 1 was whether CGEM must have a direct 1-to-1 relationship to GKSM, or whether support could be more general with a well-defined mapping between CGEM and GKSM. The resolution of this issue impacts such questions as whether combined forms of certain separate CGM elements must be provided (e.g., character height and orientation) -- this was a major debate at Egham (particularly for AFNOR and DIN). The relationship also has major bearing on what segmentation model must be used -- the GKS model or the more generally useful CGI model.

With basically no debate (even DIN and AFNOR positively agreeing, which was a positive shift in position since the Egham meeting of Sept 1986), CGEM unanimously resolved that general and well-defined support is adequate and desirable. Addendum 1 will contain in an annex the mappings between CGEM elements and GKSM items that must be performed by generators and interpreters.

Wherever possible, the functionality of CGI will be used to support the requirements of GKSM. This will in some cases lead to a one-to-many mapping between GKSM items and CGI/CGEM functions.

5.3 Relationship of CGEM and CGI

Another major conceptual issue is the relationship between CGEM and CGI. Should every CGEM be generatable (directly or via mapping) from a CGI? Should only some categories, e.g., GKSM, be supported? Should CGEM be an audit trail of a CGI? At the CGEM/CGI liaison meeting there was strong support that GKSM should be generatable through CGI -- there was a vote (almost unanimous) that CGI should add new functions if such were required to use CGEM as a GKSM.

Further liaison work during the week resulted in presentation of the new CGI pipeline model to CGEM (particularly as it pertains to the segmentation model), presentation of a list of output functions of CGI considered stable enough for inclusion in Addendum 1, and some interchange on CGEM needs for new functions in CGI.

Following these liaison exchanges, CGEM experts identified the priority location of CGEM in the CGI pipeline as just prior to segment storage.

The CGEM group presented to CGI the need for 2 new functions -- UPDATE and MODIFY FONT LIST -- in order to generate a CGEM/GKSM through CGI. By the end of the meeting CGI had resolved **not** to include such functions at this time -- the relationship between CGEM and CGI now stands that some metafiles will be generatable through CGI, but not all. In particular, not all GKSMs will be generatable through CGI. CGI will have adequate facilities to interpret CGEMs, with some mapping required.

CGEM will adopt most of the functions suggested by CGI. There is some concern about the stability of some, and the usefulness of some others in a metafile environment. The stability question is particularly sensitive, as it is a firm principle that expansion of the scope of Addendum 1 shall not slow down its progress.

In summary then, the relationship between CGEM and CGI

functionality is "close," but uncoupled in the sense that CGEM generators cannot necessarily reside below CGI, even in the GKSM case (because of lack of the 2 functions mentioned).

5.4 Segment Model of CGEM

The US presented to CGEM a position paper (by Vanderschel and Gerety of the US CGI sub-delegation) demonstrating that the CGI segmentation model was adequate to support GKSM needs. Together with study and explanation of the new CGI pipeline model, explicit definition of the mapping between CGI functions and GKSM items, and the resolution of the 1-to-1 GKSM support issue, it was unanimously resolved to adopt the CGI segmentation model for CGEM. The UPDATE function of GKS/GKSM must be added, however.

In addition, as mentioned earlier the US position on Global Segments (symbol libraries) was accepted and they will be incorporated into Addendum 1.

5.5 GKSM Item Types

A distinction is made between the "logical items" of a client of CGEM, such as GKS/GKSM, and the "physical items" which are the elements of CGEM. The CGEM group felt that it is the responsibility of the client standard to define logical items and logical item types, and the responsibility of CGEM to define the mapping between physical items and logical items. HOD/RAP felt that this should be done by CGEM Addendum 1 in the case of GKSM, because of expediency.

Addendum 1 will contain a new annex detailing the mapping between CGEM elements and GKSM items, and will specify the item types (as in GKS annex E), and that this new annex will be part of the standard.

5.6 Resolution of all Open Issues

All open issues, both in the initial ISO issues log and those generated by national review, were resolved. The US submitted most of the latter issues, with some also from UK, and a small number from France and Germany.

The US delegation was effective here. All the resolutions agree with the US positions on the issues, as developed at the X3H3 Tulsa meeting, with the following exceptions. (Please note: None of these exceptions are on issues of major importance.)

1. ANSI.5 -- "What elements may be included in segments?" The resolution was alternative 3, "restricted set by category".

2. **ISO.12 and ANSI.11** -- It was resolved that the TEXT FONT INDEX element could be used by GKS, with the proper use of the FONT LIST element and the addition of a new MODIFY FONT LIST element (which is a picture element). The mechanism should be explained in the new GKSM annex of Addendum 1.

3. **ANSI.16** -- "What is result when both SCALING MODE and DEVICE VIEWPORT appear?" This was basically resolved as per the preferred position of the US, alternative 1, with the additional proviso that when neither element appears in the metafile, the default viewport has precedence in those metafile categories in which both elements are allowed.

4. **ISO.4** -- "Should SEGMENT DISPLAY PRIORITY be integer or real?" This was resolved as per the preferred US position, integer. However a SEGMENT PRIORITY EXTENT function will be included (similar to VDC EXTENT, COLOR VALUE EXTENT, etc) to declare the minimum and maximum values (i.e., they are not simply implicit from the range of representable integers).

5. **ANSI.9** -- "Should the encoding of the METAFILE DEFAULTS REPLACEMENT element in the Binary Encoding be improved?" After much discussion, the consensus was that the change: 1) would not accomplish what was desired, because generators and interpreters would still have to honor the old encoding for category 'cgm' metafiles; and 2) is not really needed. The worst difficulty with the current encoding is the awkwardness of generating it. But there is a solution in the current CGM -- a generator can produce multiple occurrences of the MDR element, and can set one default with each occurrence, and this is fairly easy to generate.

6. **ISO.19** -- "How should item types be defined?" As per the above discussion, a standard annex of Addendum 1 will give the logical item types of GKSM logical items, and these will agree with the current annex E of GKS.

7. **BSI.2.1** -- "What are the dynamic effects of the REPRESENTATION elements [and of the current COLOR TABLE and PATTERN TABLE in category GKSM]?" This is a new issue. In Addendum 1 the new elements will have dynamic effects, as in GKS. The existing elements must also have dynamic effects for category 'gksm', but the current CGM states that the dynamic effects are unspecified. The GKSM annex of Addendum 1 will be used to specify the dynamic nature of these two elements.

8. **DIN.2.1, DIN.2.2, DIN.2.3** -- These three issues pertain to how bundled geometric attributes can be made to properly transform, how CIRCLE and similar elements transform, and how such attributes as absolute linewidth transform. The CGI solutions to these issues will be adopted by CGEM, and will be

explained by the new CGI pipeline model (which CGI presented and explained to CGEM at Valbonne).

5.7 Scope of Addendum 2 (3D)

Position papers on 3D were presented by BSI and DIN. The BSI paper was discussed at Tulsa briefly -- it is basically just the annex E of GKS-3D. The DIN paper was more ambitious -- it proposed a 3D metafile for GKS-3D, and for PHIGS, and for general 3D work. This proposal is definitely the more interesting one when diverse metafile applications such as CALS, IGES-to-CGM, GKS, etc. are being considered.

There was no clear resolution of the scope question. The preference seemed to be toward the minimal GKS-3D scope. There was much concern expressed that Addendum 2 not preclude or complicate extension to support PHIGS or other 3D constituents in the future. An interim meeting will prepare a Working Draft for circulation.

5.8 Status and Schedules

For Addendum 1, all open issues were resolved and enough of the drafting work was completed that the document editor will be able to have the DP text ready in August. The DP text will point heavily to CGI text, rather than replicating it. The current CGI text will be used. The next CGI draft would be preferable, but it will not be available in time for CGEM to meet its schedule for the first DP of Addendum 1. Where significant technical changes to CGI are known, these will be noted.

The CGEM schedule calls for a 3-month DP Registration ballot, and a 3-month DP ballot, both to be completed by 1 April 1988. This should allow national bodies to process comments at a domestic meeting prior to the summer 1988 SC24 meeting (SC21/WG2 is becoming SC24).

There will be an interim working meeting for Addendum 2, probably in the UK, probably in November 1987. The purpose is to produce Working Draft text.

6.0 The NBS/Eurographics Metafiles Workshop

6.1 Background and Objectives

In September 1987 NBS and Eurographics (a professional organization for computer graphics, based in Europe) jointly sponsored a workshop, **CGM in the Real World**, held at NBS in Gaithersburg, MD. The results of this workshop will be published as an edited volume by Springer-Verlag. Participation in the workshop was by invitation. Attendance included a spectrum of US and foreign experts, representing industry, academia, the standards community, and government. There was much implementation experience represented, and much awareness of the benefits and problems of using CGM.

CALS participation in the CGEM effort has generally had positive results. However, at the Valbonne meeting, ISO WG2 had declined to deal with extended drawing functionality such as that needed by the application areas of technical illustration and technical publishing. Making CGEM more suitable in these areas is one of the priority CALS objectives.

An effective strategy for continuing to pursue the addition of such functionality within ISO would be to first get a strong US position endorsing such extensions, and submit a detailed draft of such a position to ISO as a "strawman" proposal as an official US input document. The NBS workshop presented a good opportunity to begin formulating such a position. The workshop included many experts in the field, and it also included some key members of the graphics standards community.

6.2 Results of the Workshop

6.2.1 Presentations

Very preliminary minutes of the workshop are included as Appendix 6 to this report. An attendance list is included. Each attendee submitted a paper and presented it to the workshop. These are the papers that will be printed by Springer-Verlag. The presentations commenced on the morning of the first day and continued until the middle of the second day. Group discussion during and after each paper identified key issues and areas where further discussion and action are required.

The presentations were organized into six topical areas:

1. Performance: performance considerations and issues in using CGM;

2. **CALS:** the topics of the CALS Application Profile, CGM extensions for efficient technical drawing and publishing application, and CGM's role in raster to vector conversion;

3. **Testing:** testing of the CGM, conformance testing of ODA, and NBS graphics conformance testing;

4. **Commercial Applications:** CGM encodings issues and presentation graphics requirements;

5. **Implementations:** implementatations within an academic environment and a corporate environment.

6.2.2 Evaluation of Issues

The issues generated by the presentations and discussions were divided into 4 principal categories:

1. Technical Barriers to Interchange Using the Current CGM. There are a number of problems that were identified that are presenting barriers to CGM usage. Among these are:

- o inability to specify high quality text and kerning information;
- o color to black-and-white mapping;
- o lack of settable bundles;
- o handling of cell array;
- o relationship of CGM elements and GKSM item types;
- o lack of specification of physical file format;
- o misuse of the VDC Extent element.

2. User Requirements & Needed Future Capabilities. The CGM does not have sufficient graphical capabilities to efficiently support some application areas. Application areas considered as sources of requirements included:

- o CALS;
- o business graphics;
- o computing centers (e.g., large government funded research labs);
- o office systems;
- o publishing.

3. Educational Guidelines and Application Profiles. Issues and topics identified in this area were:

- o the need for a CGM bibliography;
- o educating raster-to-vector system producers to CGM opportunities;
- o guidance on which encodings to use for what;
- o guidance on how to interface CGM and GKS;

- guidance on how to interface CGM and GKS;
- need to specify the content of APs as a general specification;
- need for public agreement on font metrics and definitions;
- user guidelines for particular APs;
- need for maintaining consistency between APs.

4. Testing and Validation. A number of important topics were identified, including:

- testing CGM generators;
- testing CGM interpreters;
- the role of formal testing;
- the role of extended testing;
- testing mixed content (graphics, raster, text);
- testing for conformance to Application Profiles;
- testing private encodings;
- the role of the Control Board for CGM;
- how to make meaningful conformance statements for CGM.

The workshop participants split into 4 groups and worked during the afternoon of the second day identifying and studying problems and issues. The results were presented and summarized on the third (last) day of the workshop.

There is considerable interaction between the major topic areas, as well as between the specific issues within those areas. The most important area for future CALS requirements was clearly the second, since further work to influence CGEM and inject advanced capabilities needed by CALS and similar constituents had to start with a definition of user requirements. It is in this working group that the NBS representative participated.

The results of this working group are presented in the table that comprises the last two pages of Appendix 7. The table gives the results in terms of a matrix -- needed extension on one axis and application area requiring the extension on the other.

This table, and the output of the workshop in general, represent an important step toward the goal of getting CGM extended to be a more effective mechanism for CALS drawing and publishing applications. It is the first time that a user requirements study has been done for metafiles in such application areas. As mentioned before, it is a prerequisite and first step to getting a proposal for further CGM addendum work into the ISO standards committees.

IV. RECOMMENDATIONS AND IMPACTS

The work detailed above formally completed the program of work for this task for FY87. However, two weeks after the workshop (the first week of October 1987) there was an ASC X3H3 meeting in Lowell, MA. This included a working meeting of X3H3.3, within which metafile work is carried out in the U.S. If CALS requirements for additional drawing functionality are to make any progress in the ISO arena, then the U.S. must take the lead. This requires that the U.S. develop and submit a proposal to ISO.

The NBS CGM workshop output represented the basis of such a proposal. This X3H3 meeting came at a critical time in formulating a new proposal, since there is a meeting of ISO SC24 (formerly SC21/WG2, the ISO graphics standards committee) in December 1987, at which such a proposal could be approved. Missing this meeting would cause at least a 9 month delay in getting a project initiated.

Accordingly, the NBS representative participated with and led a handful of interested parties in producing a proposal for the SC24 meeting. The proposal is for an Addendum 3 to CGM, with a scope that includes the capabilities required for effective CGM use in technical illustration and publishing. This proposal is contained in Appendix 7 to this report.

The writing of this brief proposal is only the first step of a successful addendum process. It proposes a fairly aggressive, but realistic, timetable for achieving completion of the addendum. It is CALS participation in the standards process that has achieved the progress to date on Addendum 3 (as well as the important results through the Valbonne meeting). If the work is to continue to progress smoothly and rapidly, CALS needs to continue working actively in the standards making process at least through the DP draft stage of the project, which will occur at the SC24 meeting in summer of 1988.

Recommendation: CALS should continue, without delay or interruption, to inject its requirements into and provide leadership for the CGM Addendum 3, up to and including the SC24 meeting in summer 1988 in Tucson, Arizona.

V. SUMMARY AND CONCLUSIONS

Before presenting technical positions to the ISO WG2 subgroup writing the extended metafile, those positions had to become part of the US national position for that meeting. Formulating the US position was the agenda for the metafile experts at the Tulsa X3H3 meeting in late April 1987.

Specifically, the metafile experts had to process the results of an X3H3 letter ballot in which X3H3 members voted individually on 51 ANSI and ISO issues for CGEM. Processing meant coming to consensus on what position the US metafile sub-delegation should support at the next ISO WG2 working meeting (May 87).

In a working group led by the NBS representative, all open issues were resolved (positions taken). These recommendations were endorsed by X3H3 and became the US position for the WG2 meeting. Among the issues were several that were important for the CALS community. Without exception these issues were resolved for the alternatives that met the CALS requirements.

On an important general issue of the scope of the CGEM it was resolved that the scope of Addendum 1 (and other addenda) should be broadened beyond the limited scope that was current in the ISO WG2 CGEM project. Related to this was the decision to adopt the CGI segmentation model, to include additional stable functionality of CGI, and to not be constrained to a 1-to-1 relationship between CGEM and GKSM.

It was resolved to support some sort of symbol library facility. This is a facility that is very much needed and useful in CALS environments, and this was an important result. There was keen interest in better font specification work, but the decision was that the proposals should stabilize some before serious consideration for inclusion in CGEM. CALS does need better text facilities. Finally, the decision was taken to adopt a passive posture on 3D. This means basically reacting to and responding to other proposals, of which the first (from the UK) was in hand at the Tulsa meeting.

These were the major US positions taken into the ISO WG2 meeting. The results of a week of work by ISO metafile experts were favorable on most, but not on all. In general, there was considerable consensus on all of the important CALS issues, particularly between the US, Germany, and France. There was some support on many from the UK as well, but the UK did oppose some other important ones (such as richer drawing controls to support technical publishing).

On the important topic of broadening the scope of the CGEM so that it serves a broader clientele, the US position was accepted.

This conceptual resolution was the prerequisite for dealing with specific functionality, such as symbol libraries, drawing controls, CGI functionality, etc.

On the issue of a symbol library facility the US position was accepted. A small task group drafted the proposal and presented it to the metafile subgroup. It introduced no new CGM elements, but defined the semantics and a syntax of existing elements to achieve the functionality; this was probably key in diverting any opposition to the proposal.

The issue of inclusion of additional stable CGI functionality was resolved as per the US position. The DP draft of CGEM which is now in preparation will include a number of primitives, attributes, and controls from CGI.

The issue of additional functionality needed to support technical illustration and publication quality graphics was resolved against the US position. This is unfortunate, because all delegations with the exception of one all agreed with the US and placed such extensions at higher priority than 3D. That delegation (UK) apparently prevailed on procedural grounds at the "steering committee" level.

Finally, there was no definite resolution on 3D. It appears that the UK proposal, which is basically just GKS-3D Annex E, has the momentum. There was a more interesting and general proposal from Germany. Although CALS and other needs in 3D are not yet completely clear, it appears that the German proposal would be the wiser choice, allowing more options and wider clientele in the future.

Overall, the results were good for CALS and similar constituents in the US graphics community. On the topics of expanded drawing capability and control, and 3D, more work is needed.

The groundwork for an additional CGM addendum (Appendix 7) was laid in the form of a user requirements specification that was produced, with NBS direction, at the "CGM in the Real World" workshop sponsored by NBS in September 1987. This was injected into the ASC X3H3 meeting in early October 1987, resulting in a US position for the next ISO meeting that proposes another addendum to CGM. An opportunity now exists for CALS to achieve those objectives that were not met through the Valbonne meeting.

APPENDIX 1

WORKING DRAFT CGEM

ISO/TC97/SC21/N1403
X3H /86-187

Information Processing Systems

Computer Graphics

Metafile for the Storage and Transfer of Picture Description Information

Addendum 1

Part 1

Functional Description

Working Draft November 1986

Page X

Sub-clause 0.1: Add the following at the end of the sub-clause:

This picture description includes static images and session capture requirements.

Page X

Sub-clause 0.3: Add the following at the end of item c):

It should also not preclude further extensions to support future standards.

Page X

Sub-clause 0.3: Add the following at the end of item d):

It should include the capability to support both GKS picture and session requirements.

Page X

Sub-clause 0.8: Add the following at the end of the first paragraph:

The extended CGM also specifies the elements required to support GKS session capture.

Page X

Clause 1: Add the following at the end of the first paragraph:

This picture description includes static image and session capture requirements.

Page X

Clause 1: Add the following at the end of the second paragraph:

The extended CGM also contains elements that delimit and manipulate groups of elements within pictures. Capability is provided for segment structure and dynamic picture regeneration such as is required for session capture.

Page X

Sub-clause 4.1: Add the following at the end of the list of classes of elements:

- Segment Elements, which enable the manipulation and appearance of elements within pictures

Page X

Sub-clause 4.1: Add the following after the third paragraph:

Graphical output primitives and attributes may be grouped in segments. Segment attribute elements control the appearance of segments.

Page X

Sub-clause 4.2: Add the following at the end of the sub-clause:

Groups of elements within pictures, called segments, are delimited by BEGIN SEGMENT and END SEGMENT. Each segment is uniquely identified by a segment identifier.

Page X

Sub-clause 4.3: Add the following immediately above 4.3.1:

Each metafile falls into a particular metafile category. The metafile category may be announced at the start of the metafile. This information may be used by the interpreter to decide if the metafile can be interpreted. The default metafile category is of the type 'cgm' as defined by IS 8632. The category implies that the metafile conforms to the semantics and formal grammar of that category. The metafile categories may overlap. The category does not imply particular default settings; these must be explicitly stated via the METAFILE DEFAULTS REPLACEMENT element.

Page X

Sub-clause 4.3.2.1: Add the following to the end of the list of elements included in the drawing set:

BEGIN SEGMENT
END SEGMENT
METAFILE CATEGORY
VDC NORMALIZATION
DEVICE VIEWPORT
SET DEFERRAL STATE
CLEAR
UPDATE
TEXT FONT AND PRECISION
CHARACTER VECTORS
PICK IDENTIFIER
LINE REPRESENTATION
MARKER REPRESENTATION
TEXT REPRESENTATION
FILL REPRESENTATION
EDGE REPRESENTATION
RENAME SEGMENT
DELETE SEGMENT
REDRAW ALL SEGMENTS
SEGMENT TRANSFORM
SEGMENT VISIBILITY
SEGMENT HIGHLIGHTING
SEGMENT DISPLAY PRIORITY

Page X

Add the following sub-clauses after sub-clause 4.3.2.2:

4.3.2.3 GKSMO Set

The GKSMO set includes all elements conforming to GKS level 0a in IS 7942.

The elements included in the GKSMO set are:

BEGIN METAFILE
END METAFILE
BEGIN PICTURE
BEGIN PICTURE BODY
END PICTURE
BEGIN SEGMENT
END SEGMENT
METAFILE VERSION
METAFILE DESCRIPTION
VDC TYPE
INTEGER PRECISION
REAL PRECISION
INDEX PRECISION
COLOUR PRECISION
COLOUR INDEX PRECISION
MAXIMUM COLOUR INDEX
METAFILE ELEMENT LIST
METAFILE DEFAULTS REPLACEMENT
FONT LIST
CHARACTER SET LIST
CHARACTER CODING ANNOUNCER
VDC EXTENT
BACKGROUND COLOUR
VDC NORMALIZATION
VDC INTEGER PRECISION
VDC REAL PRECISION
CLIP RECTANGLE
CLEAR WORKSTATION
UPDATE WORKSTATION
SET DEFERRAL MODE
DEVICE VIEWPORT
POLYLINE
POLYMARKER
TEXT
POLYGON
CELL ARRAY
GDP
LINE BUNDLE INDEX
LINE TYPE
LINE WIDTH
LINE COLOUR
MARKER BUNDLE INDEX
MARKER TYPE
MARKER SIZE

MARKER COLOUR
TEXT BUNDLE INDEX
TEXT FONT AND PRECISION
CHARACTER EXPANSION FACTOR
CHARACTER SPACING
TEXT COLOUR
CHARACTER VECTORS
TEXT PATH
TEXT ALIGNMENT
FILL BUNDLE INDEX
INTERIOR STYLE
FILL COLOUR
HATCH INDEX
PATTERN INDEX
FILL REFERENCE POINT
PATTERN TABLE
PATTERN SIZE
COLOUR TABLE
ASPECT SOURCE FLAGS
ESCAPE
MESSAGE
APPLICATION DATA

4.3.2.4 GKSM Set

The GKSM set includes all elements conforming to GKS IS 7942.

The elements included in the GKSM set are:

BEGIN METAFILE
END METAFILE
BEGIN PICTURE
BEGIN PICTURE BODY
END PICTURE
BEGIN SEGMENT
END SEGMENT
METAFILE VERSION
METAFILE DESCRIPTION
VDC TYPE
INTEGER PRECISION
REAL PRECISION
INDEX PRECISION
COLOUR PRECISION
COLOUR INDEX PRECISION
MAXIMUM COLOUR INDEX
METAFILE ELEMENT LIST
METAFILE DEFAULTS REPLACEMENT
FONT LIST
CHARACTER SET LIST
CHARACTER CODING ANNOUNCER
VDC EXTENT
BACKGROUND COLOUR
VDC NORMALIZATION
VDC INTEGER PRECISION
VDC REAL PRECISION
CLIP RECTANGLE

CLEAR WORKSTATION
UPDATE WORKSTATION
SET DEFERRAL MODE
DEVICE VIEWPORT
RENAME SEGMENT
DELETE SEGMENT
REDRAW ALL SEGMENTS
POLYLINE
POLYMARKER
TEXT
POLYGON
CELL ARRAY
GDP
LINE BUNDLE INDEX
LINE TYPE
LINE WIDTH
LINE COLOUR
MARKER BUNDLE INDEX
MARKER TYPE
MARKER SIZE
MARKER COLOUR
TEXT BUNDLE INDEX
TEXT FONT AND PRECISION
CHARACTER EXPANSION FACTOR
CHARACTER SPACING
TEXT COLOUR
CHARACTER VECTORS
TEXT PATH
TEXT ALIGNMENT
FILL BUNDLE INDEX
INTERIOR STYLE
FILL COLOUR
HATCH INDEX
PATTERN INDEX
FILL REFERENCE POINT
PATTERN TABLE
PATTERN SIZE
COLOUR TABLE
ASPECT SOURCE FLAGS
PICK IDENTIFIER
LINE REPRESENTATION
MARKER REPRESENTATION
TEXT REPRESENTATION
FILL REPRESENTATION
SEGMENT TRANSFORM
SEGMENT VISIBILITY
SEGMENT HIGHLIGHTING
SEGMENT DISPLAY PRIORITY
SEGMENT DETECTABILITY
ESCAPE
MESSAGE
APPLICATION DATA

Page X

Add the following sub-clause after sub-clause 4.3.3:

4.3.4 VDC Normalization

VDC NORMALIZATION defines a mapping of a subspace of the VDC range with the normalized coordinate space of a target graphics system.

Page X

Add the following sub-clause after sub-clause 4.5.2:

4.5.3 Device Control

The extended CGM may contain information for the interpreter to use in controlling the output of the graphical information stored in the metafile.

DEFERRAL STATE allows the storing on the metafile of information relating to the control of buffering and deferred actions of a graphics system. Deferral state controls the possible delaying of output functions: for example, data sent to a device may be buffered to optimize data transfer. The values of deferral state (in increasing order of delay) are:

a) ASAP: The visual effect of each function will be achieved As Soon As Possible (ASAP).

b) BNIG: The visual effect of each function will be achieved Before the Next Interaction Globally (BNIG), i.e. before the next interaction with a logical input device gets underway.

c) BNIL: The visual effect of each function will be achieved Before the Next Interaction Locally (BNIL).

d) ASTI: The visual effect of each function will be achieved At Some Time (ASTI).

An implicit regeneration is equivalent to an invocation of the function REDRAW ALL SEGMENTS. Its possible delay is controlled by the implicit regeneration mode. The values of implicit regeneration mode are:

a) SUPPRESSED Implicit regeneration of the picture is suppressed until it is explicitly requested.

b) ALLOWED: Implicit regeneration of the picture is allowed.

Deferred actions can be made visible at any time by the use of the UPDATE function or by an appropriate change of the deferral state.

The CLEAR element gives the capability for clearing the display surface. The precise meaning of this element is interpreter dependent. Some indication of the expected meaning for this element may be gauged from the METAFILE CATEGORY element.

Page X

In table 1, column "May Be Bundled", add the following at the end:

TEXT FONT AND PRECISION

Page X

Sub-clause 4.7: Add the following immediately above sub-clause 4.7.1:

The attribute elements LINE REPRESENTATION, MARKER REPRESENTATION, FILL REPRESENTATION, EDGE REPRESENTATION and TEXT REPRESENTATION are used to set all of the attribute values in a bundle table entry at the same time.

Page X

In table 2, column "Aspects" add the following at the end of the list of the "TEXT" bundle.

TEXT FONT AND PRECISION

Page X

Sub-clause 4.7.3: Add the following at the end of the sub-clause:

f. TEXT FONT AND PRECISION determines the style of the graphical display of text characters and the fidelity with which characters need to be displayed and positioned.

Page X

Sub-clause 4.7.6: Add the following in the first paragraph after the first sentence:

The TEXT FONT AND PRECISION and CHARACTER VECTORS elements may also control the representation and placement of text characters.

Page X

Sub-clause 4.7.6: Add the following in the first paragraph after the second sentence:

The placement and orientation of text strings may also be controlled by the CHARACTER VECTORS element.

Page X

Sub-clause 4.7.6: Add the following at the end of the first paragraph:

The rendering may also depend on the value set by the TEXT FONT AND PRECISION element.

Page X

Sub-clause 4.7.6: Add the following paragraph after the first paragraph in this sub-clause:

Some of the text attributes may be specified in two forms. The font and precision aspects may either be specified by the two elements TEXT FONT INDEX and TEXT PRECISION or by the single element TEXT FONT AND PRECISION. Similarly, the character height (text height) and up/base vectors (text vectors) may be specified either by the two elements CHARACTER HEIGHT and CHARACTER ORIENTATION or by the single element CHARACTER VECTORS.

Page X

Sub-clause 4.7.6: Add the following at the end of the fifth paragraph after the sentence ending ".....Metafile Descriptor element FONT LIST."

For the extended metafile this functionality can also be achieved with the single element TEXT FONT AND PRECISION.

Page X

Sub-clause 4.7.6: Add the following after the seventh sentence of the sixth paragraph, after the sentence which ends: ".....of the font (see figure 3).":

For the extended metafile this functionality can also be achieved via the length of the up vector of CHARACTER VECTORS.

Page X

Sub-clause 4.7.6: Add the following at the end of the sixth paragraph, after the sentence ending: ".....as a fraction of the CHARACTER HEIGHT.":

For the extended metafile the height can also be derived from the element CHARACTER VECTORS.

Page X

Sub-clause 4.7.6: Add the following paragraph after the end of the seventh paragraph, after the sentence which ends: "..... the ratio of their lengths are significant.":

The generation and interpretation of CHARACTER VECTORS is similar to the generation and interpretation of CHARACTER HEIGHT and CHARACTER ORIENTATION. However, the properly sized vectors which are given to the metafile generator are used directly to generate the CHARACTER VECTORS element. In addition, the absolute lengths of the vectors of the CHARACTER VECTORS element are significant to the interpreter-the length of the up vector gives the text height.

Page X

Sub-clause 4.7.6: Add the following in the tenth paragraph after the first sentence which ends: "..... of the up vector of CHARACTER ORIENTATION.":

The up vector of the CHARACTER VECTORS may also give this information in the extended metafile.

Page X

Sub-clause 4.7.6: Add the following in the twenty third paragraph after the first sentence which reads: "TEXT PRECISION is..... and the clipping currently applicable.":

This may also be specified by the precision part of the TEXT FONT AND PRECISION element for the extended metafile.

Page X

Add the following sub-clause after sub-clause 4.11:

4.12 Segment Elements

In the CGM, graphical output primitives and attribute setting elements may be grouped in segments as well as being invoked outside segments. Each segment is identified by a unique segment identifier. Segments may be:

a. transformed;

b. made visible or invisible;

c. highlighted;

d. ordered front to back;

e. made detectable or undetectable;

f. deleted;

Only functions stored inside segments are affected by these operations.

Segments are the units for manipulation and change. Manipulation includes creation, deletion and renaming. Change includes transforming a segment, making a segment visible or invisible, highlighting a segment, and changing the order of overlapping segments.

The appearance of segments is controlled by segment attributes, which include segment transformation, visibility, highlighting, and segment display priority. Such segment attributes can be a basis for feedback during manipulations (for example, highlighting). The pick input properties of segments are also controlled by segment attributes, which include detectability and pick priority.

The segment elements are:

RENAME SEGMENT
DELETE SEGMENT
REDRAW ALL SEGMENTS
SEGMENT TRANSFORM
SEGMENT VISIBILITY
SEGMENT HIGHLIGHTING
SEGMENT DISPLAY PRIORITY
SEGMENT DETECTABILITY

Page X

Add the following to Figure 12:

figure 12 modifications to add segments

Page X

Sub-clause 5.1: Add the following after the ninth paragraph which starts with the sentence: "The External Elements.....":

The Segment Elements (described in sub-clause 5.10) provide for the grouping and manipulation of elements.

Page X

Sub-clause 5.1: Add the following at the end of the table of abbreviations of data type names:

N	Name	Identifier of type Integer
DP	Device Point	A point expressed in a coordinate system that is device dependent. DP units are metres or other appropriate device units.

Page X

Add the following sub-clauses after sub-clause 5.2.5:

5.2.6 BEGIN SEGMENT

Parameters:

Segment Identifier (I)

Description:

This is the first element of a segment. All subsequent elements until the next END SEGMENT will be collected into this segment.

5.2.7 END SEGMENT

Parameters:

None

Description:

Subsequent elements will no longer be part of a segment.

Page X

Sub-clause 5.3.11: Add the following shorthand names at the end of the list given in the third paragraph of the "Description":

GKSM
GKSMO
CGMEXT1

Page X

Add the following sub-clauses after sub-clause 5.3.15:

5.3.16 METAFILE CATEGORY

Parameters:

category (one of: cgm, gksmO, gksm, cgmext1) (E)

Description:

This function sets the metafile category to the type indicated by the parameter.

5.3.17 VDC NORMALIZATION

Parameters:

low value (VDC)
high value (VDC)

Description:

The parameters define a mapping of a sub-space of the VDC range defined by (low,low) and (high,high) and the virtual coordinate space of a graphics system, e.g. NDC, such that the (low,low) corner is equivalent to the lower left corner of NDC, and the (high,high) corner with the upper right corner of NDC. The low value is less than the high value.

Add the following sub-clauses after sub-clause 5.5.6:

5.5.7 DEVICE VIEWPORT

Parameters:

first corner (DP)
second corner (DP)

Description:

The two parameters define the opposite corners of a rectangular viewport on the device's display surface.

5.5.8 DEFERRAL STATE

Parameters:

deferral mode (one of: asap, bnig, bnil, asti) (E)
implicit regeneration mode (one of: suppressed, allowed) (E)

Description:

Deferral mode controls the possible delaying of output functions: for example, data sent to a device may be buffered to optimize data transfer. The values of deferral mode (in increasing order of delay) are:

a) ASAP: The visual effect of each function will be achieved As Soon As Possible (ASAP).

b) BNIG: The visual effect of each function will be achieved Before the Next Interaction Globally (BNIG) i.e. before the next interaction with a logical input device gets underway.

c) BNIL: The visual effect of each function will be achieved Before the Next Interaction Locally (BNIL).

d) ASTI: the visual effect of each function will be achieved At Some Time (ASTI).

An implicit regeneration is equivalent to an invocation of the function REDRAW ALL SEGMENTS. Its possible delay is controlled by the implicit regeneration mode. The values of implicit regeneration mode are:

a) SUPPRESSED: Implicit regeneration of the picture is suppressed until it is explicitly requested.

b) ALLOWED: Implicit regeneration of the picture is allowed.

5.5.9 CLEAR

Parameters:

Control Flag (one of: conditionally, always) (E)

Description:

This element gives the capability for clearing the display surface on interpretation. The precise meaning of this element is dependent on the environment within which the metafile is being interpreted.

5.5.10 UPDATE

Parameters:

update regeneration flag (one of: perform, postpone) (E)

Description:

This element indicates that the interpreter should make deferred actions visible. The meaning of the parameter is interpreter dependent.

Page X

Subclause 5.6.4: Add the following at the end of the second paragraph of the "Description":

For the extended metafile the character height and orientation may be derived from the CHARACTER VECTORS element.

Page X

Sub-clause 5.6.4: Add the following at the end of the third paragraph of the "Description":

For the extended metafile the CHARACTER VECTORS and TEXT FONT AND PRECISION may also be changed within a string.

Page X

Sub-clause 5.6.5: Add the following at the end of the third paragraph of the "Description":

For the extended metafile the character height and orientation may be derived from the CHARACTER VECTORS element.

Page X

Sub-clause 5.6.5: Add the following at the end of the fourth paragraph of the "Description":

For the extended metafile the orientation may be obtained from the base vector component of the CHARACTER VECTORS element and the height from the up vector of the CHARACTER VECTORS element.

Page X

Sub-clause 5.6.5: Add the following in the fifth paragraph of the "Description" after the second sentence which ends: "..... to achieve the required restriction.":

For the extended metafile the values of the text attributes CHARACTER VECTORS and TEXT FONT AND PRECISION may also be varied.

Page X

Sub-clause 5.6.5: Add the following at the end of the sixth paragraph of the "Description":

For the extended metafile the CHARACTER VECTORS and TEXT FONT AND PRECISION may also be varied within a string.

Page X

Sub-clause 5.6.6: Add the following at the end of the second paragraph of the "Description":

For the extended metafile the orientation may be obtained from the base vector component of the CHARACTER VECTORS element and the height from the up vector of the CHARACTER VECTORS element.

Page X

Sub-clause 5.6.6: Add the following at the end of the third paragraph of the "Description".

For the extended metafile the values of the text attributes CHARACTER VECTORS and TEXT FONT AND PRECISION may also be varied.

Page X

Sub-clause 5.7.9: Add the following in the first paragraph of the "Description" after the second paragraph which ends: "..... attributes are set to 'bundled'.":

For the extended metafile the values of TEXT FONT AND PRECISION are taken from the corresponding components of the indexed bundle if the ASFs for the attributes are set to 'bundled'.

Page X

Sub-clause 5.7.12: Add the following sentence to the NOTE after the second sentence which ends: "..... by the CHARACTER EXPANSION FACTOR.":

For the extended metafile the character height may be obtained from the up component of the CHARACTER VECTORS element.

Page X

Sub-clause 5.7.13: Add the following in the fourth paragraph of the "Description" after the second sentence which ends: "..... of the current CHARACTER HEIGHT attribute.":

For the extended metafile the character height may be obtained from the up component of the CHARACTER VECTORS element.

Page X

Sub-clause 5.7.35: Add the following to the list of ASF types:

text font and precision ASF

Page X

Add the following sub-clauses after sub-clause 5.7.35:

5.5.36 TEXT FONT AND PRECISION

Parameters:

text font (I)
precision (one of: string, char, stroke) (E)

Description:

The text font and precision is set to the value specified by the parameter. When the TEXT FONT AND PRECISION ASF is 'individual' subsequent text elements are displayed with this text font and precision. When the TEXT FONT AND PRECISION ASF is 'bundled', this element does not affect the display of subsequent text elements until the ASF returns to 'individual'.

5.7.37 CHARACTER VECTORS

Parameters:

x character height component (VDC)
y character height component (VDC)
x character width component (VDC)
y character width component (VDC)

Description:

The two vectors define orientation, height, width and skew of the character body of subsequent text elements. For the purposes of alignment and path, 'up' is the direction of the character height vector and 'right' is in the direction of the character width vector.

Valid values of the vectors include any which have non-zero length, and do not have the same direction, and do not have opposite directions.

5.7.38 PICK IDENTIFIER

Parameters:

pick identifier (N)

Description:

The pick identifier is associated with all of the graphical primitive elements of a segment until the next PICK IDENTIFIER element.

With pick input, a structure is returned consisting of the picked segment identifier and a pick identifier. This pick identifier represents the graphical elements that were associated with it during creation of the segment. This pick structure is returned only if the picked segment is both VISIBLE and DETECTABLE. The default pick identifier is zero.

5.7.39 LINE REPRESENTATION

Parameters:

line bundle index (IX)
line type indicator (IX)
line width specifier, either
    absolute line width (VDC)
        or
    line width scale factor (R)
line colour specifier, either
    line colour index (CI)
        or
    line colour value (CD)

Description:

In the line bundle table. the given line bundle index is associated with the specific parameters.

Line type is specified and behaves as indicated in the LINE TYPE attribute function.

Line width is defined in the current LINE WIDTH SPECIFICATION MODE and is stored in the bundle table along with that mode. Thus, the definition is immune to subsequent changes to the selection mode.

The line bundle table has predefined entries. Each entry renders a distinct appearance from other predefined entries. Any table entry (including the predefined entries) may be redefined with this function. Redefining a table entry or adding a new table entry may eliminate the ability to render a distinct appearance from other table entries.

When line functions are displayed the line bundle index refers to an entry in the line bundle table.

Which aspects in the entry are used depends upon the setting of the corresponding ASFs, see the ASPECT SOURCE FLAGS function.

5.7.40 MARKER REPRESENTATION

Parameters:

marker bundle index (IX)
marker type indicator (IX)
marker size specifier, either
absolute marker size (VDC)
or
marker size scale factor (R)
marker colour specifier, either
marker colour index (CI)
or
marker colour value (CD)

Description:

In the marker bundle table, the given marker bundle index is associated with the specified parameters.

Marker type is specified and behaves as indicated in the MARKER TYPE attribute function.

Marker size is defined in the current MARKER SIZE SPECIFICATION MODE and is stored in the bundle table along with that mode. Thus, the definition is immune to subsequent changes to the specification mode.

Marker colour is defined in the current COLOUR SELECTION MODE, and is stored in the bundle table along with that mode. Thus

the definition is immune to subsequent changes to the selection mode.

The marker bundle table has predefined entries. Each entry renders a distinct appearance from other predefined entries. Any table entry (including the predefined entries) may be redefined with this function. Redefining a table entry or adding a new table entry may eliminate the ability to render a distinct appearance from other table entries.

When polymarkers are displayed the marker bundle index refers to an entry in the marker bundle table.

Which aspects in the entry are used depends upon the setting of the corresponding ASFs, see the ASPECT SOURCE FLAGS function.

5.7.41 TEXT REPRESENTATION

Parameters:

text bundle index (IX)
text font index (IX)
text precision (one of: string, character, stroke) (E)
character expansion factor (R)
character spacing (R)
text colour specifier, either
text colour index (CI)
or
text colour value (CD)

Description:

In the text bundle table, the given text bundle index is associated with the specified parameters.

Text font index is specified and behaves as indicated in the TEXT FONT INDEX attribute function.

Text precision is specified and behaves as indicated in the TEXT PRECISION attribute function.

Character expansion factor is specified and behaves as indicated in the CHARACTER EXPANSION FACTOR attribute function.

Character spacing is specified and behaves as indicated in the CHARACTER SPACING attribute function.

Text colour is defined in the current COLOUR SELECTION MODE, and is stored in the bundle table along with that mode. Thus, the definition is immune to subsequent changes to the selection mode.

The text bundle table has predefined entries. Each entry renders a distinct appearance from other predefined entries. Any table entry (including the predefined entries) may be redefined with this function. Redefining a table entry or

adding a new table entry may eliminate the ability to render a distinct appearance from other table entries.

When text is displayed the text bundle index refers to an entry in the text bundle table.

Which aspects in the entry are used depends upon the setting of the corresponding ASFs, see the ASPECT SOURCE FLAGS function.

5.7.42 FILL REPRESENTATION

Parameters:

fill area bundle index (IX)
interior style (one of: hollow, solid, pattern,hatch, empty)(E)
fill colour specifier, either
fill colour index (CI)
or
fill colour value (CD)
hatch index (IX)
pattern index (IX)

Description:

In the fill bundle table, the given fill bundle index is associated with the specified parameters.

Interior style is specified and behaves as indicated in the INTERIOR STYLE attribute function.

Hatch index indicator is specified and behaves as indicated in the HATCH INDEX attribute function.

Pattern index indicator is specified and behaves as indicated in the PATTERN INDEX attribute function.

Fill colour is defined in the current COLOUR SELECTION MODE, and is stored in the bundle table along with that mode. Thus, the definition is immune to subsequent changes to the selection mode.

The fill bundle table has predefined entries. Each entry renders a distinct appearance from other predefined entries. Any table entry (including predefined entries) may be redefined with this function. Redefining a table entry or adding a new table entry may eliminate the ability to render a distinct appearance from other table entries.

When fill areas are displayed the fill bundle index refers to an entry in the fill bundle table.

Which aspects in the entry are used depends upon the setting of the corresponding ASFs, see the ASPECT SOURCE FLAGS function.

5.7.43 EDGE REPRESENTATION

Parameters:

edge bundle index (IX)
edge type indicator (IX)
edge width specifier, either
absolute edge width (VDC)
or
edge width scale factor (R)
edge colour specifier, either
edge colour index (CI)
or
edge colour value (CD)

Description:

In the edge bundle table, the given edge bundle index is associated with the specified parameters.

Edge type is specified and behaves as indicated in the EDGE TYPE attribute function.

Edge width is defined in the current EDGE WIDTH SPECIFICATION MODE and is stored in the bundle table along with that mode. Thus, the definition is immune to subsequent changes to the specification mode.

Edge colour is defined in the current COLOUR SELECTION MODE and is stored in the bundle table along with that mode. Thus, the definition is immune to subsequent changes to the selection mode.

The edge bundle table has predefined entries. Each entry renders a distinct appearance from other predefined entries. Any table entry (including predefined entries) may be redefined with this function. Redefining a table entry or adding a new table entry may eliminate the ability to render a distinct appearance from other table entries.

When fill areas are displayed the edge bundle index refers to an entry in the edge bundle table.

Which aspects in the entry are used depends upon the setting of the corresponding ASFs, see the ASPECT SOURCE FLAGS function.

Page X

Add the following sub-clause after sub-clause 5.9:

5.10 Segment Elements

5.10.1 RENAME SEGMENT

Parameters:

old segment identifier (I)
new segment identifier (I)

Description:
An existing segment is associated with a new segment identifier.

5.10.2 DELETE SEGMENT

Parameters:

segment identifier (I)

Description:
The identified segment is deleted.

NOTE - The segment identifier may appear in a subsequent BEGIN SEGMENT element.

5.10.3 REDRAW ALL SEGMENTS

Parameters:

None

Description:
This function is intended to result in a redraw of all defined segments. However, if a segment's visibility attribute is INVISIBLE, that segment is not drawn. Segments of higher display priority should always appear to cover overlapping segments of lower display priority.

5.10.4 SEGMENT TRANSFORM

Parameters:

segment identifier (I)
transformation matrix (4R 2VDC)

Description:
The matrix is stored in the identified segment as a segment attribute. The segment transform replaces the old segment transform. There is no accumulation of matrices.

When a segment is displayed, the segment transform is applied to all reference points in VDC space with the following matrix

$$\begin{bmatrix} X' \\ Y' \end{bmatrix} = \begin{bmatrix} M11 & M12 & M13 \\ M21 & M22 & M23 \end{bmatrix} * \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}$$

where X an Y is the original coordinate pair and X' and Y' is the new coordinate pair.

Reference points may refer to both output primitive coordinate pairs as well as to geometric attributes. Note that the reference points for all output primitives are defined to be input parameters of type POINT. In the case of reference points for geometric attributes that are of vectors, such as CHARACTER ORIENTATION, the translation portion of the matrix M13 and M23 is not applied.

The default segment transform is the identity matrix. The segment transform may be set after a segment has been open or created. It is permissible to change the transform of the open segment.

5.10.5 SEGMENT VISIBILITY

Parameters:

segment identifier (I)
visibility (one of: visible, invisible) (E)

Description:

When the visibility attribute is set to 'visible', the segment may be displayed. When this attribute is set to 'invisible' the segment must not be displayed.

NOTE - Invisible segments cannot be picked.

5.10.6 SEGMENT HIGHLIGHTING

Parameters:

segment identifier (I)
highlighting (one of: normal, highlighted) (E)

Description:

When the highlighting attribute is set to 'highlighted', the visual appearance of the segment is implementation dependent. When the highlighting attribute is set to 'normal', the segment is displayed according to the segment and primitive attributes.

5.10.7 SEGMENT DISPLAY PRIORITY

Parameters:

segment identifier (I)
segment display priority (R)

Description:

The segment display priority for the identified segment is set to the specified value.

Segments with higher segment display priority appear to be in front of segments with lower segment display priorities. When the segment display priorities of two overlapping segments are the same, the order in which they appear is implementation dependent.

5.10.8 SEGMENT DETECTABILITY

Parameters:

segment identifier (I)
detectability (one of: detectable, undetectable) (E)

Description:

When the detectability attribute is set to 'detectable' and the visibility attribute to 'visible', the segment can be picked. 'detectable' but 'invisible' or 'undetectable' segments cannot be picked.

Page X

Clause 6: Add the following at the end of clause 6:

METAFILE CATEGORY	basic cgm
VDC NORMALIZATION	0,32767 for VDC type integer 0.0,1.0 for VDC type real
DEFERRAL MODE	asap.suppressed
DEVICE VIEWPORT	i.d.
RENAME SEGMENT	n/a
DELETE SEGMENT	n/a
REDRAW ALL SEGMENTS	n/a
TEXT FONT AND PRECISION	1,string
CHARACTER VECTORS	0,1/100 of the length of the longest side of the rectangle defined by VDC EXTENT,1/100 of the length of the longest side of the rectangle defined by VDC EXTENT,0
PICK IDENTIFIER	n/a
LINE REPRESENTATION	i.d.
MARKER REPRESENTATION	i.d.
TEXT REPRESENTATION	i.d.

FILL REPRESENTATION	i.d.
EDGE REPRESENTATION	i.d.
SEGMENT TRANSFORM	1.0,0 0,1,0
SEGMENT VISIBILITY	visible
SEGMENT HIGHLIGHTING	normal
SEGMENT DETECTABILITY	undetectable

Page X

The following forms clause E7

E.7 GKS Item Types

The item type returned to the application will be based on the binary op-code which appears in Part 3 of this standard. The algorithm for calculating the item type is:

1024*class+subclass

The following annex forms a new annex F.

F Formal Grammar of the Functional Specification of the CGMEXT1 Category

NOTE - This annex is not part of the Standard; it is included for information purposes only.

F.1 Introduction

This grammar is a formal definition of a standard CGM extended syntax. The encoding-independent and the encoding-dependent productions are separated, and there are subsections showing the syntax of each of the standardized encoding schemes. Details on the encoding of terminal symbols can be found in parts of this Standard that deal with the particular encoding schemes.

F.2 Notation Used

<symbol>	- nonterminal
<SYMBOL>	- terminal
<symbol>*	- 0 or more occurrences
<symbol>+	- 1 or more occurrences
<symbol>o	- optional (0 or 1 occurrences)
<symbol>(n)	- exactly n occurrences, n=2,3,...
<symbol-1> ::= <symbol-2>	- symbol-1 has the syntax of symbol-2
<symbol-1> ¦ <symbol-2>	- symbol-1 or alternatively symbol-2
<symbol: meaning>	- symbol with the stated meaning
{comment}	- explanation of a symbol or a production

F.3 Detailed Grammar

F.3.1 Metafile Structure

```
<metafile>                  ::= <BEGIN METAFILE>
                                  <identifier>
                                <metafile descriptor>
                                <metafile contents>
                                <END METAFILE>

<metafile contents>         ::= <extra element>*
                                <picture>
                                <extra element>*

<extra element>             ::= <external element>
                              ¦ <escape element>

<picture>                   ::= <BEGIN PICTURE>
                                  <identifier>
                                <picture descriptor element>*
                                <BEGIN PICTURE BODY>
                                <picture content>*
```

```
                                    <END PICTURE>

<picture content>           ::= <picture element>
                              | <segment>

<identifier>                ::= <string>

<picture element>           ::= <control element>
                              | <graphical element>
                              | <attribute element>
                              | <escape element>
                              | <external element>
                              | <segment element>

<segment>                   ::= <BEGIN SEGMENT>
                                  <name>
                                <picture element>*
                                <END SEGMENT>
```

F.3.2 Metafile Descriptor Elements

```
<metafile descriptor>       ::= <identification>
                                <characteristics>

<identification>            ::= <METAFILE VERSION>
                                  <integer>
                                <metafile description>o
                                  <metafile category>o

<metafile category>         ::= <METAFILE CATEGORY>
                                  <category enumerated>

<metafile description>      ::= <METAFILE DESCRIPTION>
                                  <string>

<category enumerated>       ::= <BASIC CGM>
                              | <CGMEXT1>
                              | <GKSM>

<characteristics>           ::= <element list>
                                <optional descr elmt>*

<element list>              ::= <METAFILE ELEMENT LIST>
                                  <element name>*

<optional descr elmt>       ::= <VDC TYPE>
                                  <vdc type>
                              | <MAXIMUM COLOUR INDEX>
                                  <colour index>
                              | <COLOUR VALUE EXTENT>
                                  <red green blue>(2)
                              | <METAFILE DEFAULTS REPLACEMENT>
                                  <element default>+
                              | <FONT LIST>
                                  <font name>+
```

```
                                  ¦ <CHARACTER SET LIST>
                                      <character set definition>+
                                  ¦ <CHARACTER CODING ANNOUNCER>
                                      <coding technique enumerated>
                                  ¦ <scalar precision>*
                                  ¦ <escape element>
                                  ¦ <external element>
                                  ¦ <VDC NORMALIZATION>
                                      <point> (2)

<vdc type>                        ::= <INTEGER>
                                    ¦ <REAL>

<element default>                 ::= <eligible control element>
                                    ¦ <picture descriptor element>
                                    ¦ <attribute element>
                                    ¦ <escape element>

<font name>                       ::= <string>

<character set definition>        ::= <char set enumerated>
                                      <designation sequence>

<index>                           ::= <standard index value>
                                    ¦ <private index value>

<standard index value>            ::= <non-negative integer>
<non-negative integer>            ::= <integer>     {greater or equal to 0}
<positive integer>                ::= <integer>     {greater than 0}
<private index value>             ::= <negative integer>
<negative integer>                ::= <integer>     {less than 0}
<positive index value>            ::= <positive integer>

<char set enumerated>             ::= <94 CHAR>
                                    ¦ <96 CHAR>
                                    ¦ <MULTI-BYTE 94 CHAR>
                                    ¦ <MULTI-BYTE 96 CHAR>
                                    ¦ <COMPLETE CODE>

<coding technique enumerated>     ::= <BASIC 7-BIT>
                                    ¦ <BASIC 8-BIT>
                                    ¦ <EXTENDED 7-BIT>
                                    ¦ <EXTENDED 8-BIT>

<designation sequence>            ::= <string>

<scalar precision>                ::= <INTEGER PRECISION>
                                      <integer precision value>
                                    ¦ <REAL PRECISION>
                                      <real precision value>
                                    ¦ <INDEX PRECISION>
                                      <index precision value>
                                    ¦ <COLOUR PRECISION>
                                      <colour precision value>
                                    ¦ <COLOUR INDEX PRECISION>
                                      <col index precision value>
```

```
                                {these elements have encoding}
                                {dependent parameters        }

<eligible control element>   ::= <**********>

<point>                      ::= <vdc value> (2)
```

F.3.3 Picture Descriptor Elements

```
<picture descriptor element> ::= <SCALING MODE>
                                   <scaling spec mode>
                                   <metric scale factor>
                               ¦ <COLOUR SELECTION MODE>
                                   <colour select mode>
                               ¦ LINE WIDTH SPECIFICATION MODE>
                                   <spec mode>
                               ¦ <MARKER SIZE SPECIFICATION MODE>
                                   <spec mode>
                               ¦ <EDGE WIDTH SPECIFICATION MODE>
                                   <spec mode>
                               ¦ <VDC EXTENT>
                                   <point>(2)
                               ¦ <BACKGROUND COLOUR>
                                   <red green blue>
                               ¦ <escape element>
                               ¦ <external element>

<colour select mode>         ::= <INDEXED>
                                 ¦ <DIRECT>

<scaling spec mode>          ::= <ABSTRACT>
                                 ¦ <METRIC>

<metric scale factor>        ::= <real>

<spec mode>                  ::= <ABSOLUTE>
                                 ¦ <SCALED>

<point>                      ::= <vdc value> (2)
```

F.3.4 Control Elements

```
<control element>            ::= <vdc precision>
                                 ¦ <AUXILIARY COLOUR>
                                     <colour>
                                 ¦ <TRANSPARENCY>
                                     <on-off indicator enumerated>
                                 ¦ <CLIP RECTANGLE>
                                     <point>(2)
                                 ¦ <CLIP INDICATOR>
                                     <on-off indicator enumerated>
                                 ¦ <VDC EXTENT>
                                     <point>(2)
                                 ¦ <DEVICE VIEWPORT>
                                     <device point>(2)
                                 ¦ <DEFERRAL STATE>
```

```
                                    <deferral mode enumerated>
                                    <implicit regeneration mode enumerated>
                                  ¦ <CLEAR>
                                    <control flag enumerated>
                                  ¦ <UPDATE>
                                    <update regeneration flag enumerated>

<on-off indicator enumerated>::= <ON>
                                 ¦ <OFF>

<colour>                      ::= <colour index>
                                ¦ <red green blue>

<vdc precision>               ::= <VDC INTEGER PRECISION>
                                    <vdc integer precision value>
                                ¦ <VDC REAL PRECISION>
                                    <vdc real precision value>
                                  {these elements have encoding}
                                  {dependent parameters        }

<device point>                ::=<real>(2)

<deferral mode enumerated>    ::= <ASAP>
                                ¦ <BNIG>
                                ¦ <BNIL>
                                ¦ <ASTI>

<implicit regeneration mode>  ::= <SUPPRESSED>
                                ¦ <ALLOWED>

<control flag enumerated>     ::=<CONDITIONALLY>
                                ¦ <ALWAYS>

<update regeneration flag>
      enumerated>             ::= <PERFORM>
                                . ¦ <POSTPONE>
```

F.3.5 Graphical Elements

```
<graphical element>           ::= <polypoint element>
                                ¦ <text element>
                                ¦ <cell element>
                                ¦ <gdp element>
                                ¦ <rectangle element>
                                ¦ <circular element>
                                ¦ <elliptical element>

<polypoint element>           ::= <POLYLINE>
                                    <point pair>
                                    <point list>
                                ¦ <DISJOINT POLYLINE>
                                    <point pair>
                                    <point pair list>
                                ¦ <POLYMARKER>
                                    <point>
                                    <point list>
```

```
                                  | <POLYGON>
                                      <point>(3)
                                      <point list>
                                  | <POLYGON SET>
                                      <point edge pair>(3)
                                      <point edge pair list>

<point list>                  ::= <point>*

<point pair list>             ::= <point pair>*

<point pair>                  ::= <point>(2)

<point edge pair>             ::= <point><edge out flag>

<point edge pair list>        ::= <point edge pair>*

<edge out flag>               ::= <INVISIBLE>
                                  | <VISIBLE>
                                  | <CLOSE_INVISIBLE>
                                  | <CLOSE_VISIBLE>

<text element>                ::= <TEXT>
                                      <point>
                                      <text tail>
                                  | <restricted text element>

<restricted text element>     ::= <RESTRICTED TEXT>
                                      <extent>
                                      <point>
                                      <text tail>

<extent>                      ::= <vdc value>(2)

<text tail>                   ::= <final character list>
                                  | <nonfinal character list>

<final character list>        ::= <FINAL>
                                      <character list>

<nonfinal character list>     ::= <NOT FINAL>
                                      <string>
                                      <character attribute element>*
                                      <spanned text>

<spanned text>                ::= <APPEND TEXT>
                                      <text tail>

<cell element>                ::= <CELL ARRAY>
                                      <point>(3)
                                      <integer>(2)
                                      <local colour precision>
                                      <colour>(integer1 x integer2)

                                    {this element has an encoding}
                                    {dependent parameter          }
```

```
<local colour precision>     ::= <colour precision value>
                               | <col index precision value>
                               | <default col precision indicator>

<gdp element>                ::= <GDP>
                                   <gdp identifier>
                                   <point list>*
                                   <data record>

<gdp identifier>             ::= <integer>

<rectangle element>          ::= <RECTANGLE>
                                   <point pair>

<circular element>           ::= <CIRCLE>
                                   <point>
                                   <radius>
                               | <CIRCULAR ARC 3 POINT>
                                   <point>(3)
                               | <CIRCULAR ARC 3 POINT CLOSE>
                                   <point>(3)
                                   <close type>
                               | <CIRCULAR ARC CENTRE>
                                   <point>
                                   <vdc value>(4)
                                   <radius>
                               | <CIRCULAR ARC CENTRE CLOSE>
                                   <point>
                                   <vdc value>(4)
                                   <radius>
                                   <close type>

<radius>                     ::= <non-negative vdc value>

<non-negative vdc value>     ::= <vdc value>  {greater or equal to 0}

<close type>                 ::= <PIE>
                               | <CHORD>

<elliptical element>         ::= <ELLIPSE>
                                   <point>(3)
                               | <ELLIPTICAL ARC>
                                   <point>(3)
                                   <vdc value>(4)
                               | <ELLIPTICAL ARC CLOSE>
                                   <point>(3)
                                   <vdc value>(4)
                                   <close type>
```

F.3.6 Attribute Elements

```
<primitive attribute elem>   ::= <line attribute element>
                               | <marker attribute element>
                               | <text attribute element>
                               | <filled area attribute element>
```

```
                                  ¦ <colour table element>
                                  ¦ <aspect source flags>
                                  ¦ <representation element>

<line attribute element>      ::= <LINE BUNDLE INDEX>
                                    <positive index>
                                  ¦ <LINE TYPE>
                                    <index>
                                  ¦ <LINE WIDTH>
                                    <size value>
                                  ¦ <LINE COLOUR>
                                    <colour>

<size value>                  ::= <non-negative vdc value>
                                  ¦ <non-negative real>

<non-negative real>           ::= <real>  {greater or equal to 0}

<marker attribute element>    ::= <MARKER BUNDLE INDEX>
                                    <positive index>
                                  ¦ <MARKER TYPE>
                                    <index>
                                  ¦ <MARKER SIZE>
                                    <size value>
                                  ¦ <MARKER COLOUR>
                                    <colour>

<text attribute element>      ::= <char attribute element>
                                  ¦ <string attribute element>

<char attribute element>      ::= <TEXT BUNDLE INDEX>
                                    <positive index>
                                  ¦ <TEXT FONT INDEX>
                                    <positive index>
                                  ¦ <CHARACTER EXPANSION FACTOR>
                                    <real>
                                  ¦ <CHARACTER SPACING>
                                    <real>
                                  ¦ <TEXT COLOUR>
                                    <colour>
                                  ¦ <CHARACTER HEIGHT>
                                    <non-negative vdc value>
                                  ¦ <CHARACTER ORIENTATION>
                                    <vdc value>(4)
                                  ¦ <CHARACTER SET INDEX>
                                    <positive index>
                                  ¦ <ALTERNATE CHARACTER SET INDEX>
                                    <positive index>
                                  ¦ <TEXT FONT AND PRECISION>
                                    <index>
                                    <text precision enumerated>
                                  ¦ <CHARACTER VECTORS>
                                    <vdc value>(4)

<string attribute element>    ::= <TEXT PATH>
                                    <path enumerated>
```

```
                                | <TEXT PRECISION>
                                    <text precision enumerated>
                                | <TEXT ALIGNMENT>
                                    <horizontal align enumerated>
                                    <vertical align enumerated>
                                    <continuous align value> (2)

<path enumerated>             ::= <RIGHT>
                                | <LEFT>
                                | <UP>
                                | <DOWN>

<text precision enumerated>   ::= <STRING>
                                | <CHARACTER>
                                | <STROKE>

<horizontal align enumerated> ::= <NORMAL HORIZONTAL>
                                | <LEFT>
                                | <CENTRE>
                                | <RIGHT>
                                | <CONTINUOUS HORIZONTAL>

<vertical align enumerated>   ::= <NORMAL VERTICAL>
                                | <TOP>
                                | <CAP>
                                | <HALF>
                                | <BASE>
                                | <BOTTOM>
                                | <CONTINUOUS VERTICAL>

<continuous align value>      ::= <real>

<filled area attribute elem>  ::= <FILL BUNDLE INDEX>
                                    <positive index>
                                | <INTERIOR STYLE>
                                    <interior style enumerated>
                                | <FILL COLOUR>
                                    <colour>
                                | <HATCH INDEX>
                                    <index>
                                | <PATTERN INDEX>
                                    <positive index>
                                | <EDGE BUNDLE INDEX>
                                    <positive index>
                                | <EDGE TYPE>
                                    <index>
                                | <EDGE WIDTH>
                                    <size value>
                                | <EDGE COLOUR>
                                    <colour>
                                | <EDGE VISIBILITY>
                                    <on-off indicator enumerated>
                                | <FILL REFERENCE POINT>
                                    <point>
                                | <PATTERN TABLE>
                                    <positive index>
```

```
                                    <integer>(2)
                                    <local colour precision>
                                    <colour>(integer1 x integer2)
                                  {this element has an encoding}
                                  {dependent parameter          }
                                | <PATTERN SIZE>
                                    <vdc value>(4)

<interior style enumerated>  ::= <HOLLOW>
                                | <SOLID>
                                | <PATTERN>
                                | <HATCH>
                                | <EMPTY>

<colour table element>       ::= <COLOUR TABLE>
                                    <starting index>
                                    <red green blue>+

<starting index>             ::= <colour index>

<aspect source flags>        ::= <ASPECT SOURCE FLAGS>
                                    <asf pair>+

<asf pair>                   ::= <asf type>
                                 <asf>

<asf type>                   ::= <LINE TYPE ASF>
                                | <LINE WIDTH ASF>
                                | <LINE COLOUR ASF>
                                | <MARKER TYPE ASF>
                                | <MARKER SIZE ASF>
                                | <MARKER COLOUR ASF>
                                | <TEXT FONT ASF>
                                | <TEXT PRECISION ASF>
                                | <TEXT FONT AND PRECISION ASF>
                                | <CHARACTER EXPANSION FACTOR ASF>
                                | <CHARACTER SPACING ASF>
                                | <TEXT COLOUR ASF>
                                | <INTERIOR STYLE ASF>
                                | <FILL COLOUR ASF>
                                | <HATCH INDEX ASF>
                                | <PATTERN INDEX ASF>
                                | <EDGE TYPE ASF>
                                | <EDGE WIDTH ASF>
                                | <EDGE COLOUR ASF>

<asf>                        ::= <INDIVIDUAL>
                                | <BUNDLED>

<pick identifier>            ::= <PICK IDENTIFIER>
                                    <integer>

<representation element>     ::= <LINE REPRESENTATION>
                                    <positive index>
                                    <index>
                                    <size value>
```

```
                                <colour>
                              ¦ <MARKER REPRESENTATION>
                                <positive index>
                                <index>
                                <size value>
                                <colour>
                              ¦ <TEXT REPRESENTATION>
                                <positive index>
                                <index>  {font}
                                <text precision enumerated>
                                <real>   {character spacing}
                                <real>   {expansion factor}
                                <colour>
                              ¦ <FILL REPRESENTATION>
                                <positive index>
                                <interior style enumerated>
                                <index>            {hatch index}
                                <positive index>   {pattern index}
                                <colour>
                              ¦ <EDGE REPRESENTATION>
                                <positive index>
                                <index>
                                <size value>
                                <colour>
                              ¦ <PATTERN TABLE>
                                <positive index>
                                <integer>(2)
                                <local colour precision>
                                <colour>(integer1*integer2)
                               {this element has an encoding}
                               {dependent parameter}
                              ¦ <COLOUR TABLE>
                                <starting index>
                                <red green blue>+
```

F.3.7 Escape Elements

```
<escape element>          ::= <ESCAPE>
                                <identifier>
                                <data record>

<identifier>              ::= <integer>
```

F.3.8 External Elements

```
<external element>        ::= <MESSAGE>
                                <action flag>
                                <string>
                              ¦ <APPLICATION DATA>
                                <integer>
                                <data record>

<action flag>             ::= <YES>
```

```
                                  | <NO>
```

F.3.9 Segment Elements

```
<segment element>            ::= <RENAME SEGMENT>
                                   <old segment name>
                                   <new segment name>
                                 | <DELETE SEGMENT>
                                   <segment name>
                                 | <REDRAW ALL SEGMENTS>
                                 | <SEGMENT TRANSFORM>
                                   <name>
                                   <transformation matrix>
                                 | <SEGMENT VISIBILITY>
                                   <name>
                                   <visibility enumerated>
                                 | <SEGMENT HIGHLIGHTING>
                                   <name>
                                   <highlighting enumerated>
                                 |<SEGMENT PRIORITY>
                                   <name>
                                   <segment priority>
                                 |<SEGMENT DETECTABILITY>
                                   <name>
                                   <detectability enumerated>

<old segment name>           ::= <name>

<new segment name>           ::= <name>

<segment name>               ::= <name>

<transformation matrix>      ::= <real>(6)

<visibility enumerated>      ::= <VISIBLE>
                                 | <INVISIBLE>

<highlighting enumerated>    ::= <NORMAL>
                                 | <HIGHLIGHTED>

<segment priority>           ::=<real>   {0-1}

detectability enumerated     ::= <DETECTABLE>
                                 | <UNDETECTABLE>
```

F.4 Terminal Symbols

The following are the terminals in this grammar.
Their representation is dependent on the encoding scheme used.
In annex A of the subsequent parts of this Standard, these
encoding-dependent symbols are further described.

```
<element name>
<integer>
<real>
```

<vdc value>
<string>
<colour index>
<red green blue>
<integer prec value>
<real prec value>
<index prec value>
<colour prec value>
<col index prec value>
<default col prec indicator>
<vdc integer prec value>
<vdc real prec value>
<colour list>
<data record>
<device point>
<name>

The CGM extended opcodes are encoding dependent. A complete list of them can be found in the productions for <element name enumerated> below.

The enumerated types:

<BASIC CGM>
<CGMEXT1>
<GKSM>
<GKSMO>
<INTEGER>
<REAL>
<ON>
<OFF>
<INDEXED>
<DIRECT>
<ABSTRACT>
<METRIC>
<ABSOLUTE>
<SCALED>
<94 CHAR>
<96 CHAR>
<MULTI-BYTE 94 CHAR>
<MULTI-BYTE 96 CHAR>
<COMPLETE CODE>
<BASIC 7-BIT>
<BASIC 8-BIT>
<EXTENDED 7-BIT>
<EXTENDED 8-BIT>
<ASAP>
<BNIG>
<BNIL>
<ASTI>
<SUPPRESSED>
<ALLOWED>
<POSTPONE>
<PERFORM>
<CONDITIONALLY>
<ALWAYS>

<PIE>
<CHORD>
<FINAL>
<NOT FINAL>
<INDIVIDUAL>
<BUNDLED>
<HOLLOW>
<SOLID>
<PATTERN>
<HATCH>
<EMPTY>
<STRING>
<CHARACTER>
<STROKE>
<RIGHT>
<LEFT>
<UP>
<DOWN>
<NORMAL HORIZONTAL>
<CENTRE>
<CONTINUOUS HORIZONTAL>
<NORMAL VERTICAL>
<TOP>
<CAP>
<HALF>
<BASE>
<BOTTOM>
<CONTINUOUS VERTICAL>
<YES>
<NO>
<LINE TYPE ASF>
<LINE WIDTH ASF>
<LINE COLOUR ASF>
<MARKER TYPE ASF>
<MARKER SIZE ASF>
<MARKER COLOUR ASF>
<TEXT FONT ASF>
<TEXT PRECISION ASF>
<TEXT FONT AND PRECISION ASF>
<CHARACTER EXPANSION FACTOR ASF>
<CHARACTER SPACING ASF>
<TEXT COLOUR ASF>
<INTERIOR STYLE ASF>
<HATCH INDEX ASF>
<PATTERN INDEX ASF>
<FILL COLOUR ASF>
<EDGE TYPE ASF>
<EDGE WIDTH ASF>
<EDGE COLOUR ASF>
<VISIBLE>
<INVISIBLE>
<NORMAL>
<HIGHLIGHTED>
<DETECTABLE>
<UNDETECTABLE>

```
<element name enumerated>    ::= <BEGIN METAFILE>
                               | <END METAFILE>
                               | <BEGIN PICTURE>
                               | <BEGIN PICTURE BODY>
                               | <END PICTURE>
                               | <BEGIN SEGMENT>
                               | <END SEGMENT>
                               | <METAFILE VERSION>
                               | <METAFILE DESCRIPTION>
                               | <VDC TYPE>
                               | <INTEGER PRECISION>
                               | <REAL PRECISION>
                               | <INDEX PRECISION>
                               | <COLOUR PRECISION>
                               | <COLOUR INDEX PRECISION>
                               | <MAXIMUM COLOUR INDEX>
                               | <METAFILE ELEMENT LIST>
                               | <METAFILE DEFAULTS REPLACEMENT>
                               | <FONT LIST>
                               | <CHARACTER SET LIST>
                               | <CHARACTER CODING ANNOUNCER>
                               | <SCALING MODE>
                               | <COLOUR SELECTION MODE>
                               | <LINE WIDTH SPECIFICATION MODE>
                               | <MARKER SIZE SPECIFICATION MODE>
                               | <EDGE WIDTH SPECIFICATION MODE>
                               | <VDC EXTENT>
                               | <BACKGROUND COLOUR>
                               | <VDC NORMALIZATION>
                               | <VDC INTEGER PRECISION>
                               | <VDC REAL PRECISION>
                               | <AUXILIARY COLOUR>
                               | <TRANSPARENCY>
                               | <CLIP RECTANGLE>
                               | <CLIP INDICATOR>
                               | <CLEAR WORKSTATION>
                               | <UPDATE WORKSTATION>
                               | <SET DEFERRAL MODE>
                               | <DEVICE VIEWPORT>
                               | <RENAME SEGMENT>
                               | <DELETE SEGMENT>
                               | <REDRAW ALL SEGMENTS>
                               | <POLYLINE>
                               | <DISJOINT POLYLINE>
                               | <POLYMARKER>
                               | <TEXT>
                               | <RESTRICTED TEXT>
                               | <APPEND TEXT>
                               | <POLYGON>
                               | <POLYGON SET>
                               | <CELL ARRAY>
                               | <GDP>
                               | <RECTANGLE>
                               | <CIRCLE>
                               | <CIRCULAR ARC 3 POINT>
```

<CIRCULAR ARC 3 POINT CLOSE>
<CIRCULAR ARC CENTRE>
<CIRCULAR ARC CENTRE CLOSE>
<ELLIPSE>
<ELLIPTICAL ARC>
<ELLIPTICAL ARC CLOSE>
<LINE BUNDLE INDEX>
<LINE TYPE>
<LINE WIDTH>
<LINE COLOUR>
<MARKER BUNDLE INDEX>
<MARKER TYPE>
<MARKER SIZE>
<MARKER COLOUR>
<TEXT BUNDLE INDEX>
<TEXT FONT INDEX>
<TEXT PRECISION>
<TEXT FONT AND PRECISION>
<CHARACTER EXPANSION FACTOR>
<CHARACTER SPACING>
<TEXT COLOUR>
<CHARACTER HEIGHT>
<CHARACTER ORIENTATION>
<CHARACTER VECTORS>
<TEXT PATH>
<TEXT ALIGNMENT>
<CHARACTER SET INDEX>
<ALTERNATE CHARACTER SET INDEX>
<FILL BUNDLE INDEX>
<INTERIOR STYLE>
<FILL COLOUR>
<HATCH INDEX>
<PATTERN INDEX>
<EDGE BUNDLE INDEX>
<EDGE TYPE>
<EDGE WIDTH>
<EDGE COLOUR>
<EDGE VISIBILITY>
<FILL REFERENCE POINT>
<PATTERN TABLE>
<PATTERN SIZE>
<COLOUR TABLE>
<ASPECT SOURCE FLAGS>
<PICK IDENTIFIER>
<LINE REPRESENTATION>
<MARKER REPRESENTATION>
<TEXT REPRESENTATION>
<FILL REPRESENTATION>
<SEGMENT TRANSFORM>
<SEGMENT VISIBILITY>
<SEGMENT HIGHLIGHTING>
<SEGMENT DISPLAY PRIORITY>
<SEGMENT DETECTABILITY>
<ESCAPE>
<MESSAGE>
<APPLICATION DATA>

| <DRAWING SET>
| <DRAWING PLUS CONTROL SET>

The following annex forms the new annex G.

G Formal Grammar of the Functional Specification of the GKSM Category

NOTE - This annex is not part of the standard; it is included for information purposes only.

G.1 Introduction

This grammar is a formal definition of GKSM syntax. The encoding-independent and the encoding-dependent productions are separated, aand there are subsections showing the syntax of each of the standardized encoding schemes. Details on the encoding of terminal symbols can be found in parts of the CGM Standard that deal with the particular encoding schemes.

G.2 Notation Used

<symbol>	- nonterminal
<SYMBOL>	- terminal
<symbol>*	- 0 or more occurrences
<symbol>+	- 1 or more occurrences
<symbol>o	- optional (0 or 1 occurrences)
<symbol>(n)	- exactly n occurrences, n=2,3,...
<symbol-1> ::= <symbol-2>	- symbol-1 has the syntax of symbol-2
<symbol-1> ¦ <symbol-2>	- symbol-1 or alternatively symbol-2
<symbol: meaning>	- symbol with the stated meaning
{comment}	- explanation of a symbol or a production

Detailed Grammar

Metafile Structure

```
<metafile>                  ::= <BEGIN METAFILE>
                                  <identifier>
                                <metafile descriptor>
                                <metafile contents>*
                                <END METAFILE>

<metafile contents>         ::= <extra element>*
                                <picture>
                                <extra element>*

<extra element>             ::= <external element>
                              ¦ <escape element>

<picture>                   ::= <BEGIN PICTURE>
                                  <identifier>
                                <picture descriptor element>*
                                <BEGIN PICTURE BODY>
                                <picture content>*
                                <END PICTURE>
```

```
<picture content>           ::= <picture element>
                              ¦ <segment>

<picture identifier>        ::= <string>

<picture element>           ::= <control element>
                              ¦ <graphical element>
                              ¦ <attribute element>
                              ¦ <escape element>
                              ¦ <external element>
                              ¦ <segment element>

<segment>                   ::= <BEGIN SEGMENT>
                                  <name>
                                <picture element>*
                                <END SEGMENT>
```

G.3.2 Metafile Descriptor Elements

```
<metafile descriptor>       ::= <identification>
                                <characteristics>

<identification>            ::= <METAFILE VERSION>
                                  <integer>
                                <metafile description>o
                                  <metafile category>o

<metafile category>         ::= <METAFILE CATEGORY>
                                  <category enumerated>

<metafile description>      ::= <METAFILE DESCRIPTION>
                                  <string>

<category enumerated>       ::= <GKSM>

<characteristics>           ::= <element list>
                                <optional descr elmt>*

<element list>              ::= <METAFILE ELEMENT LIST>
                                  <element name>*

<optional descr elmt>       ::= <VDC TYPE>
                                  <vdc type>
                              ¦ <MAXIMUM COLOUR INDEX>
                                  <colour index>
                              ¦ <COLOUR VALUE EXTENT>
                                  <red green blue>(2)
                              ¦ <METAFILE DEFAULTS REPLACEMENT>
                                  <element default>+
                              ¦ <FONT LIST>
                                  <font name>+
                              ¦ <CHARACTER SET LIST>
                                  <character set definition>+
                              ¦ <CHARACTER CODING ANNOUNCER>
```

```
                                    <coding technique enumerated>
                                  ¦ <scalar precision>*
                                  ¦ <escape element>
                                  ¦ <external element>
                                  ¦ <VDC NORMALIZATION>
                                    <point> (2)

<vdc type>                        ::= <INTEGER>
                                  ¦ <REAL>

<element default>                 ::= <eligible control element>
                                  ¦ <picture descriptor element>
                                  ¦ <attribute element>
                                  ¦ <escape element>

<eligible control element>        ::= <******************>

<font name>                       ::= <string>

<character set definition>        ::= <char set enumerated>
                                      <designation sequence>

<index>                           ::= <standard index value>
                                  ¦ <private index value>

<standard index value>            ::= <non-negative integer>
<non-negative integer>            ::= <integer>     {greater or equal to 0}
<positive integer>                ::= <integer>     {greater than 0}
<private index value>             ::= <negative integer>
<negative integer>                ::= <integer>     {less than 0}
<positive index value>            ::= <positive integer>

<char set enumerated>             ::= <94 CHAR>
                                  ¦ <96 CHAR>
                                  ¦ <MULTI-BYTE 94 CHAR>
                                  ¦ <MULTI-BYTE 96 CHAR>
                                  ¦ <COMPLETE CODE>

<coding technique enumerated>     ::= <BASIC 7-BIT>
                                  ¦ <BASIC 8-BIT>
                                  ¦ <EXTENDED 7-BIT>
                                  ¦ <EXTENDED 8-BIT>

<designation sequence>            ::= <string>

<scalar precision>                ::= <INTEGER PRECISION>
                                      <integer precision value>
                                  ¦ <REAL PRECISION>
                                      <real precision value>
                                  ¦ <INDEX PRECISION>
                                      <index precision value>
                                  ¦ <COLOUR PRECISION>
                                      <colour precision value>
                                  ¦ <COLOUR INDEX PRECISION>
                                      <col index precision value>
                                    {these elements have encoding}
```

```
                                {dependent parameters        }

<point>                         ::= <vdc value> (2)
```

G.3.3 Picture Descriptor Elements

```
<picture descriptor element> ::= <BACKGROUND COLOUR>
                                   <red green blue>
                                 ¦ <escape element>
                                 ¦ <external element>
                                 ¦ <VDC NORMALIZATION>
                                   <point>(2)

<point>                         ::= <vdc value>  (2)
```

G.3.4 Control Elements

```
<control element>               ::= <vdc precision>
                                  ¦ <CLIP RECTANGLE>
                                    <point>(2)
                                  ¦ <workstation window>
                                  ¦ <workstation viewport>
                                  ¦ <DEFERRAL STATE>
                                    <deferral mode enumerated>
                                    <implicit regeneration mode enumerated>
                                  ¦ <clear workstation>
                                  ¦ <update workstation>

<vdc precision>                 ::= <VDC INTEGER PRECISION>
                                    <vdc integer precision value>
                                  ¦ <VDC REAL PRECISION>
                                    <vdc real precision value>
                                   {these elements have encoding}
                                   {dependent parameters        }

<workstation window>            ::=<VDC EXTENT>
                                    <point>(2)

<workstation viewport>          ::=<DEVICE VIEWPORT>
                                    <device point>(2)

<device point>                  ::=<real>(2)

<deferral mode enumerated>      ::= <ASAP>
                                  ¦ <BNIG>
                                  ¦ <BNIL>
                                  ¦ <ASTI>

<implicit regeneration mode>    ::= <SUPPRESSED>
                                  ¦ <ALLOWED>

<clear workstation>             ::= <CLEAR>
                                    <control flag enumerated>

<update workstation>            ::= <UPDATE>
                                    <update regeneration flag enumerated>
```

```
<control flag enumerated>     ::=<CONDITIONALLY>
                                | <ALWAYS>

<update regeneration flag>
      enumerated>             ::= <PERFORM>
                                | <POSTPONE>
```

G.3.5 Graphical Elements

```
<graphical element>           ::= <polypoint element>
                                | <text element>
                                | <cell element>
                                | <gdp element>

<polypoint element>           ::= <POLYLINE>
                                    <point pair>
                                    <point list>
                                | <POLYMARKER>
                                    <point>
                                    <point list>
                                | <POLYGON>
                                    <point>(3)
                                    <point list>

<point list>                  ::= <point>*

<point pair list>             ::= <point pair>*

<point pair>                  ::= <point>(2)

<text element>                ::= <TEXT>
                                    <point>
                                    <text tail>
                                  | <restricted text element>

<text tail>                   ::= <final character list>

<final character list>        ::= <FINAL>
                                    <string>

<cell element>                ::= <CELL ARRAY>
                                    <point>(3)
                                    <integer>(2)
                                    <local colour precision>
                                    <colour>(integer1 x integer2)

                                  {this element has an encoding}
                                  {dependent parameter         }

<local colour precision>      ::= <colour precision value>
                                | <col index precision value>
                                | <default col precision indicator>

<gdp element>                 ::= <GDP>
```

```
                                    <gdp identifier>
                                    <point list>*
                                    <data record>

<gdp identifier>                ::= <integer>
```

G.3.6 Attribute Elements

```
<attribute element>             ::= <line attribute element>
                                  | <marker attribute element>
                                  | <text attribute element>
                                  | <filled area attribute element>
                                  | <aspect source flags>
                                  | <pick identifier>
                                  | <representation element>

<line attribute element>        ::= <LINE BUNDLE INDEX>
                                      <positive index>
                                  | <LINE TYPE>
                                      <index>
                                  | <LINE WIDTH>
                                      <size value>
                                  | <LINE COLOUR>
                                      <positive index>

<size value>                    ::= <non-negative real>

<non-negative real>             ::= <real>  {greater or equal to 0}

<marker attribute element>      ::= <MARKER BUNDLE INDEX>
                                      <positive index>
                                  | <MARKER TYPE>
                                      <index>
                                  | <MARKER SIZE>
                                    <size value>
                                  | <MARKER COLOUR>
                                    <positive index>

<text attribute element>        ::= <char attribute element>
                                  | <string attribute element>

<char attribute element>        ::= <TEXT BUNDLE INDEX>
                                      <positive index>
                                  | <CHARACTER EXPANSION FACTOR>
                                      <real>
                                  | <CHARACTER SPACING>
                                      <real>
                                  | <TEXT COLOUR>
                                      <positive index>
                                  | <TEXT FONT AND PRECISION>
                                      <index>
                                      <text precision enumerated>
                                  | <CHARACTER VECTORS>
                                      <vdc value>(4)
```

```
<string attribute element>   ::= <TEXT PATH>
                                   <path enumerated>
                                 ¦ <TEXT ALIGNMENT>
                                   <horizontal align enumerated>
                                   <vertical align enumerated>

<path enumerated>            ::= <RIGHT>
                                 ¦ <LEFT>
                                 ¦ <UP>
                                 ¦ <DOWN>

<text precision enumerated>  ::= <STRING>
                                 ¦ <CHARACTER>
                                 ¦ <STROKE>

<horizontal align enumerated>  ::= <NORMAL HORIZONTAL>
                                 ¦ <LEFT>
                                 ¦ <CENTRE>
                                 ¦ <RIGHT>

<vertical align enumerated>  ::= <NORMAL VERTICAL>
                                 ¦ <TOP>
                                 ¦ <CAP>
                                 ¦ <HALF>
                                 ¦ <BASE>
                                 ¦ <BOTTOM>

<filled area attribute elem> ::= <FILL BUNDLE INDEX>
                                   <positive index>
                                 ¦ <INTERIOR STYLE>
                                   <interior style enumerated>
                                 ¦ <FILL COLOUR>
                                   <positive index>
                                 ¦ <HATCH INDEX>
                                   <index>
                                 ¦ <PATTERN INDEX>
                                   <positive index>
                                 ¦ <FILL REFERENCE POINT>
                                   <point>
                                 ¦ <PATTERN SIZE>
                                   <vdc value>(4)

<interior style enumerated>  ::= <HOLLOW>
                                 ¦ <SOLID>
                                 ¦ <PATTERN>
                                 ¦ <HATCH>

<aspect source flags>        ::= <ASPECT SOURCE FLAGS>
                                   <asf pair>+

<asf pair>                   ::= <asf type>
                                 <asf>

<asf type>                   ::= <LINE TYPE ASF>
                                 ¦ <LINE WIDTH ASF>
                                 ¦ <LINE COLOUR ASF>
```

```
                                  | <MARKER TYPE ASF>
                                  | <MARKER SIZE ASF>
                                  | <MARKER COLOUR ASF>
                                  | <TEXT FONT ASF>
                                  | <TEXT PRECISION ASF>
                                  | <TEXT FONT AND PRECISION ASF>
                                  | <CHARACTER EXPANSION FACTOR ASF>
                                  | <CHARACTER SPACING ASF>
                                  | <TEXT COLOUR ASF>
                                  | <INTERIOR STYLE ASF>
                                  | <FILL COLOUR ASF>
                                  | <HATCH INDEX ASF>
                                  | <PATTERN INDEX ASF>

<asf>                             ::= <INDIVIDUAL>
                                    | <BUNDLED>

<pick identifier>                 ::= <PICK IDENTIFIER>
                                      <integer>

<representation element>          ::= <LINE REPRESENTATION>
                                      <positive index>
                                      <index>
                                      <size value>
                                      <positive index index>
                                    | <MARKER REPRESENTATION>
                                      <positive index>
                                      <index>
                                      <size value>
                                      <positive index index>
                                    | <TEXT REPRESENTATION>
                                      <positive index>
                                      <index>  {font}
                                      <text precision enumerated>
                                      <real>   {character spacing}
                                      <real>   {expansion factor}
                                      <positive index index>
                                    | <FILL REPRESENTATION>
                                      <positive index>
                                      <interior style enumerated>
                                      <index>            {hatch index}
                                      <positive index>   {pattern index}
                                      <positive index index>
                                    | <PATTERN TABLE>
                                      <positive index>
                                      <integer>(2)
                                      <local colour precision>
                                      <colour>(integer1*integer2)
                                    {this element has an encoding}
                                    {dependent parameter}
                                    | <COLOUR TABLE>
                                      <starting index>
                                      <red green blue>+

<starting index>                  ::= <colour index>
```

```
<colour index>              ::= <positive index>
```

G.3.7 Escape Elements

```
<escape element>            ::= <ESCAPE>
                                  <identifier>
                                  <data record>

<identifier>                ::= <integer>
```

G.3.8 External Elements

```
<external element>          ::= <MESSAGE>
                                  <action flag>
                                  <string>
                              | <APPLICATION DATA>
                                  <integer>
                                  <data record>

<action flag>               ::= <YES>
                              | <NO>
```

G.3.9 Segment Elements

```
<segment element>           ::= <RENAME SEGMENT>
                                  <old segment name>
                                  <new segment name>
                              | <DELETE SEGMENT>
                                  <segment name>
                              | <REDRAW ALL SEGMENTS>
                              | <SEGMENT TRANSFORM>
                                  <name>
                                  <transformation matrix>
                              | <SEGMENT VISIBILITY>
                                  <name>
                                  <visibility enumerated>
                              | <SEGMENT HIGHLIGHTING>
                                  <name>
                                  <highlighting enumerated>
                              |<SEGMENT PRIORITY>
                                  <name>
                                  <segment priority>
                              |<SEGMENT DETECTABILITY>
                                  <name>
                                  <detectability enumerated>

<old segment name>          ::= <name>

<new segment name>          ::= <name>

<segment name>              ::= <name>
```

```
<transformation matrix>      ::= <real>(6)

<visibility enumerated>      ::= <VISIBLE>
                                 | <INVISIBLE>

<highlighting enumerated>    ::= <NORMAL>
                                 | <HIGHLIGHTED>

<segment priority>           ::=<real>   {0-1}

detectability enumerated     ::= <DETECTABLE>
                                 | <UNDETECTABLE>
```

G.4 Terminal Symbols

The following are the terminals in this grammar.
Their representation is dependent on the encoding scheme used.
In annex A of the subsequent parts of this Standard, these
encoding-dependent symbols are further described.

<element name>
<integer>
<real>
<vdc value>
<string>
<colour index>
<red green blue>
<integer prec value>
<real prec value>
<index prec value>
<colour prec value>
<col index prec value>
<default col prec indicator>
<vdc integer prec value>
<vdc real prec value>
<colour list>
<data record>
<device point>
<name>

The CGM extended opcodes are encoding dependent. A complete list of them can be found in the productions for <element name enumerated> below.

The enumerated types:

<GKSM>
<INTEGER>
<REAL>
<ON>
<OFF>
<INDEXED>
<94 CHAR>
<96 CHAR>
<MULTI-BYTE 94 CHAR>

<MULTI-BYTE 96 CHAR>
<COMPLETE CODE>
<BASIC 7-BIT>
<BASIC 8-BIT>
<EXTENDED 7-BIT>
<EXTENDED 8-BIT>
<ASAP>
<BNIG>
<BNIL>
<ASTI>
<SUPPRESSED>
<ALLOWED>
<POSTPONE>
<PERFORM>
<CONDITIONALLY>
<ALWAYS>
<FINAL>
<INDIVIDUAL>
<BUNDLED>
<HOLLOW>
<SOLID>
<PATTERN>
<HATCH>
<STRING>
<CHARACTER>
<STROKE>
<RIGHT>
<LEFT>
<UP>
<DOWN>
<NORMAL HORIZONTAL>
<CENTRE>
<NORMAL VERTICAL>
<TOP>
<CAP>
<HALF>
<BASE>
<BOTTOM>
<YES>
<NO>
<LINE TYPE ASF>
<LINE WIDTH ASF>
<LINE COLOUR ASF>
<MARKER TYPE ASF>
<MARKER SIZE ASF>
<MARKER COLOUR ASF>
<TEXT FONT AND PRECISION ASF>
<CHARACTER EXPANSION FACTOR ASF>
<CHARACTER SPACING ASF>
<TEXT COLOUR ASF>
<INTERIOR STYLE ASF>
<HATCH INDEX ASF>
<PATTERN INDEX ASF>
<FILL COLOUR ASF>
<VISIBLE>
<INVISIBLE>

<NORMAL>
<HIGHLIGHTED>
<DETECTABLE>
<UNDETECTABLE>

<element name enumerated> ::= <BEGIN METAFILE>
| <END METAFILE>
| <BEGIN PICTURE>
| <BEGIN PICTURE BODY>
| <END PICTURE>
| <BEGIN SEGMENT>
| <END SEGMENT>
| <METAFILE VERSION>
| <METAFILE DESCRIPTION>
| <VDC TYPE>
| <INTEGER PRECISION>
| <REAL PRECISION>
| <INDEX PRECISION>
| <COLOUR PRECISION>
| <COLOUR INDEX PRECISION>
| <MAXIMUM COLOUR INDEX>
| <METAFILE ELEMENT LIST>
| <METAFILE DEFAULTS REPLACEMENT>
| <FONT LIST>
| <CHARACTER SET LIST>
| <CHARACTER CODING ANNOUNCER>
| <VDC EXTENT>
| <BACKGROUND COLOUR>
| <VDC NORMALIZATION>
| <VDC INTEGER PRECISION>
| <VDC REAL PRECISION>
| <CLIP RECTANGLE>
| <CLEAR WORKSTATION>
| <UPDATE WORKSTATION>
| <SET DEFERRAL MODE>
| <DEVICE VIEWPORT>
| <RENAME SEGMENT>
| <DELETE SEGMENT>
| <REDRAW ALL SEGMENTS>
| <POLYLINE>
| <POLYMARKER>
| <TEXT>
| <POLYGON>
| <CELL ARRAY>
| <GDP>
| <LINE BUNDLE INDEX>
| <LINE TYPE>
| <LINE WIDTH>
| <LINE COLOUR>
| <MARKER BUNDLE INDEX>
| <MARKER TYPE>
| <MARKER SIZE>
| <MARKER COLOUR>
| <TEXT BUNDLE INDEX>
| <TEXT FONT AND PRECISION>

<CHARACTER EXPANSION FACTOR>
<CHARACTER SPACING>
<TEXT COLOUR>
<CHARACTER VECTORS>
<TEXT PATH>
<TEXT ALIGNMENT>
<FILL BUNDLE INDEX>
<INTERIOR STYLE>
<FILL COLOUR>
<HATCH INDEX>
<PATTERN INDEX>
<FILL REFERENCE POINT>
<PATTERN TABLE>
<PATTERN SIZE>
<COLOUR TABLE>
<ASPECT SOURCE FLAGS>
<PICK IDENTIFIER>
<LINE REPRESENTATION>
<MARKER REPRESENTATION>
<TEXT REPRESENTATION>
<FILL REPRESENTATION>
<SEGMENT TRANSFORM>
<SEGMENT VISIBILITY>
<SEGMENT HIGHLIGHTING>
<SEGMENT DISPLAY PRIORITY>
<SEGMENT DETECTABILITY>
<ESCAPE>
<MESSAGE>
<APPLICATION DATA>

The following annex forms the new annex H

H Formal Grammar of the Functional Specification of the GKSM0 Category

This will be a subset of Annex G

Information Processing Systems

Computer Graphics

Metafile for the Storage and Transfer of Picture Description Information

Addendum 1

Part 2

Character Encoding

Working Draft November 1986

Page X

Add the following to table 1:

Opcode	7-Bit Coding		8-Bit Coding	
BEGIN SEGMENT opcode	3/0	2/5	03/0	02/5
END SEGMENT opcode	3/0	2/6	03/0	02/6
METAFILE CATEGORY opcode	3/2	3/0	03/2	03/0
VDC NORMALIZATION opcode	3/2	3/1	03/2	03/0
DEVICE VIEWPORT opcode	3/3	2/6	03/3	02/6
DEFERRAL STATE opcode	3/3	2/7	03/3	02/7
CLEAR opcode	3/3	2/8	03/3	02/8
UPDATE opcode	3/3	2/9	03/3	02/9
LINE REPRESENTATION opcode	3/5	2/8	03/5	02/8
MARKER REPRESENTATION opcode	3/5	2/9	03/5	02/9
TEXT FONT AND PRECISION opcode	3/5	3/12	03/5	03/12
CHARACTER VECTORS opcode	3/5	3/13	03/5	03/13
TEXT REPRESENTATION opcode	3/5	3/14	03/5	03/14
FILL REPRESENTATION opcode	3/6	2/13	03/6	02/13
EDGE REPRESENTATION opcode	3/6	2/14	03/6	02/14
PICK ID opcode	3/6	3/2	03/6	03/2
RENAME SEGMENT opcode	3/8	2/0	03/8	02/0
DELETE SEGMENT opcode	3/8	2/1	03/8	02/1
REDRAW ALL SEGMENTS opcode	3/8	2/2	03/8	02/2
SEGMENT TRANSFORM opcode	3/8	2/3	03/8	02/3
SEGMENT VISIBILITY opcode	3/8	2/4	03/8	02/4
SEGMENT HIGHLIGHTING opcode	3/8	2/5	03/8	02/5
SEGMENT PRIORITY opcode	3/8	2/6	03/8	02/6
SEGMENT DETECTABILITY opcode	3/8	2/7	03/8	02/7

Page X

Add the following to the end of sub-clause 5.3:

3/8 for Segment Elements

Page X

The following form sub-clauses 8.1.6 and 8.1.7

8.1.6 BEGIN SEGMENT

<BEGIN-SEGMENT-opcode: 3/0 2/5>
<integer: segment-identifier>

8.1.7 END SEGMENT

<END-SEGMENT-opcode: 3/0 2/6>

Page X

The following form sub-clauses 8.2.16 and 8.2.17

8.2.16 METAFILE CATEGORY

```
<METAFILE-CATEGORY-OPCODE: 3/2 3/0>
<enumerated: metafile category>
<enumerated: metafile category> = <integer: 0> {basic cgm}
                                ¦ <integer: 1> {cgm ext1}
                                ¦ <integer: 2> {gksm}
```

8.2.17 VDC NORMALIZATION

```
<VDC-NORMALIZATION-opcode: 3/2 3/1>
<VDC: low-value>
V<VDC: high-value>
```

Page X

The following form sub-clauses 8.4.7 to 8.4.10

8.4.7 DEVICE VIEWPORT

```
<DEVICE-VIEWPORT-opcode: 3/3 2/6>
<device-point: first corner>   )
<device-point : second corner> )
```

8.4.8 DEFERRAL STATE

```
<DEFERRAL-STATE-opcode: 3/3 2/7>
<enumerated: deferral mode>
<enumerated: implicit regeneration mode>
<enumerated: deferral mode> = <integer: 0> {asap}
                            ¦ <integer: 1> {bnig}
                            ¦ <integer: 2> {bnil}
                            ¦ <integer: 3> {asti}
<enumerated:
implicit-regeneration-mode> = <integer: 0>{suppressed}
                            ¦ <integer: 1>{allowed}
```

8.4.9 CLEAR

```
<CLEAR-opcode 3/3 2/8>
<enumerated: control-flag>
<enumerated: control-flag> = <integer: 0> {conditionally}
                           ¦ <integer: 1> {always}
```

8.4.10 UPDATE

```
<UPDATE-opcode: 3/3 2/9>
<enumerated: update-regeneration-flag>
<enumerated: update-regeneration-flag> = <integer: 0> {perform}
                                       ¦ <integer: 1> {postpone}
```

The following form sub-clauses 8.6.36 to 8.6.43

8.6.36 TEXT FONT AND PRECISION

```
<TEXT-FONT-AND-PRECISION-opcode: 3/5 3/12>
<integer: text-font-index>
<enumerated: text-precision>
<integer: text-font-index>    = <positive index>
<enumerated: text-precision> = <integer: 0> {string}
                              ¦ <integer: 1> {char}
                              ¦ <integer: 2> {stroke}
```

8.6.37 CHARACTER VECTORS

```
<CHARACTER-VECTORS-opcode: 3/5 3/13>
<VDC: x-component-of-height-vector>
<VDC: y-component-of-height-vector>
<VDC: x-component-of-width-vector>
<VDC: y-component-of-width-vector>
```

8.6.38 PICK ID

```
<PICK-ID-opcode: 3/6 3/2>
<integer: pick-id>
```

8.6.39 LINE REPRESENTATION

```
<LINE REPRESENTATION-opcode: 3/5 2/8>
<integer: line-bundle-index>
<index: line-type>
<index: line-type>
<line-width-specifier>
<colour-specifier>

<integer: line-bundle-index> = <positive integer>
<index: line-type>           = <integer: 1> {solid}
                               <integer: 2> {dash}
                               <integer: 3> {dot}
                               <integer: 4> {dash-dot}
                               <integer: 5> {dash-dot-dot}
                               <integer: negative> {private line
                                        type}

<line-width-specifier>       = <real: line width-scale-factor> {if
                               LINE WIDTH SPECIFICATION MODE is
                               'scaled'}
                              ¦ <VDC: line width> {if LINE WIDTH
                               SPECIFICATION MODE IS 'absolute'}
<colour-specifier>           = <integer: colour index> {if COLOUR
                               SELECTION MODE is 'indexed'}
                              ¦ <RGB> {if COLOUR SELECTION MODE IS
                               'absolute'}
<integer: colour-index>      = <non-negative integer>
```

8.6.40 MARKER REPRESENTATION

```
<MARKER-REPRESENTATION-opcode: 3/5 2/9>
<integer: marker-bundle-index>
<index: marker-type>
<index: marker-type>
<marker-size-specifier>
<colour-specifier>

<integer: marker-bundle-index> = <positive integer>
<index: marker-type>           = <integer: 1> {solid}
                                 <integer: 2> {dash}
                                 <integer: 3> {dot}
                                 <integer: 4> {dash-dot}
                                 <integer: 5> {dash-dot-dot}
                                <integer: negative> {private
                                                     marker type}

<marker-size-specifier>        = <real: marker size-scale-factor> {is
                                 MARKER SIZE SPECIFICATION MODE is
                                 'scaled'}
                                | <VDC: marker size> {if MARKER-SIZE
                                 SPECIFICATION MODE IS 'absolute'}
<colour-specifier>             = <integer: colour index> {if COLOUR
                                 SELECTION MODE IS 'indexed'}
                                | <RGB> {if COLOUR SELECTION MODE is
                                 'absolute'}
<integer: colour-index>        = <non-negative integer>
```

8.6.41 TEXT REPRESENTATION

```
<TEXT-REPRESENTATION-opcode: 3/5 3/14>
<integer: text-bundle-index>
<integer: text-font-index>
<enumerated: text-precision>
<real: expansion-factor>
<real: character-spacing>
<colour-specifier>

<integer: text-bundle-index> = <positive integer>
<integer: text-font-index> = <positive integer>
<enumerated: text-precision> =    <integer:0> {string}
                                  <integer:1> {character}
                                  <integer:2> {stroke}
<real: expansion-factor> = <non-negative real>
```

8.6.42 FILL REPRESENTATION

```
<FILL-REPRESENTATION-opcode: 3/6 2/13>
<integer: fill-bundle-index>
<enumerated: interior-style>
<index: hatch-index>
<index: pattern-index>
<colour specifier>
```

```
<integer;fill-bundle-index> = <positive integer>
<enumerated: interior style> =    <integer:0> {hollow
                                ¦ <integer:1> {solid
                                ¦ <integer:2> {pattern
                                ¦ <integer:3> {hatch
                                ¦ <integer:4> {empty
                                ¦ <integer:negative>{private style}

<index: hatch-index>  =     <integer:1> {horizontal}
                        ¦   <integer:2> {vertical}
                        ¦   <integer:3> {positive slope}
                        ¦   <integer:4> {negative slope}
                        ¦   <integer:5> {horizontal/vertical cross}
                        ¦   <integer:6> {positive/negative cross}
                        ¦   <integer:negative> {private styles}

<index; pattern-index> = <positive integer>
<colour specifier> = <integer:colour index> {if COLOUR SELECTION
                                                MODE is 'indexed'
                   ¦ <RGB> {if COLOUR SELECTION MODE is 'direct'
```

8.6.43 EDGE REPRESENTATION

```
<EDGE-REPRESENTATION-opcode: 3/6 2/14>
<integer: edge-bundle-index>
<index: edge-type>
<edge-width-specifier>
<colour-specifier>

<integer: edge-bundle=index> = <positive integer>
<index: edge-type>           = <integer: 1> {solid}
                             ¦ <integer: 2> {dash}
                             ¦ <integer: 3> {dot}
                             ¦ <integer: 4> {dash-dot}
                             ¦ <integer: 5> {dash-dot-dot}
                             ¦ <integer: negative> {private edge
                               type}

<edge-width-specifier>       = <real: edge-width-scale-factor> {if
                               EDGE WIDTH SPECIFICATION MODE is
                               'scaled'}
                             ¦ <VDC: edge width> {if EDGE WIDTH
                               SPECIFICATION MODE is 'absolute'}
<colour-specifier>           = <integer: colour-index> {if COLOUR
                               SELECTION MODE is 'indexed'}
                             ¦ <RGB> {if COLOUR SELECTION MODE is
                               'direct'}
<integer: colour-index>      = <non-negative integer>
```

The following forms sub-clause 8.9

8.9 Segment Elements

8.9.1 RENAME SEGMENT

<RENAME-SEGMENT-opcode: 3/8 2/0>
<integer: old-segment-identifier>
<integer: new-segment-identifier>

8.9.2 DELETE SEGMENT

<DELETE-SEGMENT-opcode: 3/8 2/1>
<integer: segment-identifier>

8.9.3 REDRAW ALL SEGMENTS

<REDRAW-ALL-SEGMENTS-opcode: 3/8 2/2>

8.9.4 SEGMENT TRANSFORMATION

<SEGMENT-TRANSFORMATION-opcode: 3/8 2/3>
<integer: segment-identifier>
<transformation matrix>
<transformation matrix> = <real: a_{11} >
<real: a_{21} >
<real: a_{12} >
<real: a_{22} >
<vdc : a_{13} >
<vdc : a_{23} >

8.9.5 SEGMENT VISIBILITY

<SEGMENT-VISIBILITY-opcode: 3/8 2/4>
<integer: segment-identifier>
<enumerated: segment-visibility>
<enumerated: segment-visibility> = <integer: 0> {invisible}
| <integer: 1> {visibility}

8.9.6 SEGMENT HIGHLIGHTING

<SEGMENT-HIGHLIGHTING-opcode: 3/8 2/5>
<integer: segment-identifier>
<enumerated: segment-highlighting>
<enumerated: segment-highlighting> = <integer: 0> {normal}
| <integer: 1> {highlighted}

8.9.7 SEGMENT PRIORITY

<SEGMENT-PRIORITY-opcode: 3/8 2/6>
<integer: segment-identifier>
<real: segment-priority>
<real: segment-priority> = <0 ≤ real ≤ 1>

8.9.8 SEGMENT DETECTABILITY

<SEGMENT-DETECTABILITY-opcode: 3/8 2/7>
<enumerated:segment-detectability> = <integer: 0> {undetectable}
<integer: segment-identifier> | <integer: 1> {detectable}

Information Processing Systems

Computer Graphics

Metafile for the Storage and Transfer of Picture Description Information

Addendum 1

Part 3

Binary Encoding

Working Draft November 1986

Page X

Add the following to table 2:

8 Segment elements

Page X

Add the following to table 3:

Element Class 0	Element Id	Parameter Type	Parameter List Length	Parameter Range	Default
BEGIN SEGMENT	6	I	BI	IR	n/a
END SEGMENT	7	n/a	0	n/a	n/a

Notes (on table 3)

Code Notes

6 BEGIN SEGMENT: has 1 parameter:
P1: (integer) segment identifier

7 END SEGMENT: has no parameters

Page X

Add the following to table 4:

Element Class 1	Element Id	Parameter Type	Parameter List Length	Parameter Range	Default
METAFILE CATEGORY	16	E	BE	{0,1,2}	0
VDC NORMALIZATION	17	2VDC	2BVDC	VDCR	see note below

Notes (on table 4)

Code Notes

16 METAFILE CATEGORY: has 1 parameter:
P1: (enumerated) category
0 Basic CGM, 1 CGM EXT1, 2 GKSM, 3 GKSMO

17 VDC NORMALIZATION: has 2 parameters:
P1: (VDC) Low value
P2: (VDC) High value

If VDC TYPE is REAL, default VDC NORMALIZATION is 0.0, 1.0. If VDC TYPE is INTEGER, default VDC NORMALIZATION is 0, 1.

Page X

Add the following to table 6

Element Class 3	Element Id	Parameter Type	Parameter List Length	Parameter Range	Default
DEVICE VIEWPORT	8	2DP	2BDP	DPR	See note below
CLEAR	9	E	BE	{0,1}	n/a
UPDATE	10	E	BE	{0,1}	n/a
DEFERRAL STATE	11	E	BE	{0,1,2,3}	n/a

Notes (on table 6)

Code Notes

7 DEVICE VIEWPORT: has two parameters

P1: (device point) first point in decimillimeters
P2: (device point) second point in decimillimeters.

If the entire device view surface is rectangular, then the default DEVICE VIEWPORT is the entire device view surface.

Otherwise the default is set to the largest rectangular subset of the view surface having the desired aspect ratio.

The default is set so that the "first point" is below and to the left of the "second point" as seen by the viewer.

8 DEFERRAL STATE: has two parameters:

P1: (enumerated) Deferral mode
0 ASAP, As soon as possible.
1 BNIG, Before next interaction globally.
2 BNIL, Before next interaction locally.
3 ASTI, At some time.

9 CLEAR: has one parameter:

P1: (enumerated) control flag
0 Conditionally
1 Always

10 UPDATE: has one parameter:

P1: (enumerated) Update regeneration flag
0 Perform
1 Postpone

Page X

Add the following to table 8:

Element Class 5	Element Id	Parameter Type	Parameter List Length	Parameter Range	Default

TEXT FONT AND PRECISION	36	IX,E	BIX+IR	+IXR	(0,1,2), (0,0)
CHARACTER VECTORS	37	4VDC	4BVDC	VDCR	
PICK IDENTIFIER	38	I	BI	IR	n/a
LINE REPRESENTATION	39	2IX,VDC or R,CO	2BIX+BVDC or BFP+BCO	+IX+VDCR or FPR/ +FXR COR	n/a
MARKER REPRESENTATION	40	2IX,VDC or R,CO	2BIX+BVDC or BFP+BCO	+IX+VDCR or FPR +FXR COR	n/a
TEXT REPRESENTATION	41	2IX,E,2R, CO	2BIX+BE+ +2BFP+BCO	+IX+FPR/ +FXR+COR	n/a
FILL REPRESENTATION	42	2IX,CO, 2IX	2BIX,BCO+ 2BIX	+IX+COR	n/a
EDGE REPRESENTATION	43	2IX,VDC or or R,CO	2BIX+BVDC or BR+BCO	+IXR,IXR, ++VDCR or ++RR,COR	n/a

Code Notes

36 TEXT FONT AND PRECISION: has 2 parameters:
- P1: (index) text font index
- P2: (index) text precision: Valid values are:
  - 0 string
  - 1 character
  - 2 stroke

37 CHARACTER VECTORS: has 4 parameters:
- P1: (real) x character height component
- P2: (real) y character height component
- P3: (real) x character base component
- P4: (real) y character base component

38 PICK IDENTIFIER: has 1 parameter
- P1: (integer) pick identifier

39 LINE REPRESENTATION: has 4 parameters
- P1: (index) line bundle index
- P2: (index) line type indicator
- P3: (vdc or real) absolute line width or line width scale factor
- P4: (colour) line colour: its form depends ON COLOUR SELECTION MODE.

40 MARKER REPRESENTATION: has 4 parameters
- P1: (index) marker bundle index
- P2: (index) marker type indicator
- P3: (vdc or real) absolute marker width or line width scale factor
- P4: (colour) marker colour: its form depends ON COLOUR SELECTION MODE.

41 TEXT REPRESENTATION: has 6 parameters
P1: (index) text bundle index
P2: (index) text font index
P3: (index) text precision
P4: (real) character spacing
P5: (real) character expansion factor
P6: (colour) text colour; its form depends ON COLOUR SELECTION MODE

42 FILL REPRESENTATION: has 4 parameters
P1: (index) fill area bundle
P2: (index) interior style: valid values are:
0 hollow
1 solid
2 pattern
3 hatch
4 empty
P3: (colour): fill colour; its form depends on COLOUR SELECTION MODE
P4: (index) pattern index

43 EDGE REPRESENTATION: has 4 parameters
P1: (index) edge bundle index
P2: (index) edge type indicator
P3: (vdc or real) absolute edge width or line width scale factor
P4: (colour) edge colour: its form depends on COLOUR SELECTION MODE.

Page X

The following forms sub-clause 7.10

7.10 Segment Elements

Table 11 Encoding of Segment Elements

Element Class 8	Element Id	Parameter Type	Parameter List Length	Parameter Range	Default
RENAME SEGMENT	1	2I	2BI	IR	n/a
DELETE SEGMENT	2	I	BI	IR	n/a
REDRAW ALL SEGMENTS	3	-	-	-	-
SEGMENT TRANSFORM	4	4R,2VDC	4BFP+2BVDC	FPR,IR	1.,0.,0., 1.,0,0
SEGMENT VISIBILITY	5	I,E	BI+BE	IR,(0,1)	n/a,0
SEGMENT HIGHLIGHTING	6	I,E	BI+BE	IR,(0,1),	n/a,0
SEGMENT DISPLAY PRIORITY	7	I,R	BI+BFP	IR,FPR	n/a,0

SEGMENT DETECTABILITY 8 I,E BI+BE IR,(0,1), n/a,0

Notes (on table 11)

Code Notes

1 RENAME SEGMENT:
P1: (index) old segment name
P2: (index) new segment name

2 DELETE SEGMENT:
P1: (index) segment name

3 REDRAW ALL SEGMENTS:
No parameters

4 SEGMENT TRANSFORM: has 6 parameters
representing a 3X2 matrix of the form:

|P1 P2 P5|
|P3 P4 P6|

where:
P1: (real) x scale component
P2: (real) x rotation component
P3: (real) y scale component
P4: (real) y rotation component
P5: (vdc) x translation component
P6: (vdc) y translation component

5 SEGMENT VISIBILITY:
P1: (index) segment name
P2: (index) segment visibility: valid values are
0 visible
1 invisible

6 SEGMENT HIGHLIGHTING:
P1: (index) segment name
P2: (index) type of highlighting: valid values are
0 normal
1 highlighted

7 SEGMENT DISPLAY:
Priority
P1: (index) segment name
P2: (real) segment display priority

8 SEGMENT DETECTABILITY:
P1: (index) segment name
P2: (index) detectability: valid values are
0 undetectable
1 detectable

Information Processing Systems

Computer Graphics

Metafile for the Storage and Transfer of Picture Description Information

Addendum 1

Part 4

Clear Text Encoding

Working Draft November 1986

Page X

Add the following to the end of sub-clause 5.3.5

```
DPOINTREC ::=          <I>
                       <SEP>
                       <I>

DP        ::=          <DPOINTREC>|<<LEFT PAREN><OPTSEP>
                       <DPOINTREC><OPTSEP><RIGHT PAREN>>

                       {COORDINATE in DC space.  Parentheses
                        are optional.  If they are used,  they
                        must group exactly two integer numbers.
                        The parenthesised form in intended to aid
                        readability of the metafile}

TM        ::=          <R><SEP><R><SEP><VDC><SEP>
                       <R><SEP><R><SEP><VDC>

                       {2*3 real transformation matrix in
                        row-major order}
```

Page X

Add the following to the end of sub-clause 5.4.4

DETECTABLE	DET
DETECTABILITY	DET
DEFERRAL	DEFER
IDENTIFIER	ID
HIGHLIGHTING	HIGHLIGHT
PRIORITY	PRI
REPRESENTATION	REP
SEGMENT	SEG
TRANSFORMATION	TRAN
UNDETECTABLE	UNDET

Page X

Add the following to the end of the table in sub-clause 5.4.5

BEGIN SEGMENT	BEGSEG
END SEGMENT	ENDSEG
METAFILE CATEGORY	MFCATEGORY
VDC NORMALIZATION	VDCNORMALIZATION
DEVICE VIEWPORT	DEVICEVIEWPORT
DEFERRAL STATE	DEFSTATE
CLEAR	CLEAR
UPDATE	UPDATE
TEXT FONT AND PRECISION	TEXTFONTPREC
CHARACTER VECTORS	CHARVECTORS
PICK IDENTIFIER	PICKID
LINE REPRESENTATION	LINEREP

MARKER REPRESENTATION	MARKERREP
TEXT REPRESENTATION	TEXTREP
FILL REPRESENTATION	FILLREP
EDGE REPRESENTATION	EDGEREP
RENAME SEGMENT	RENAMESEG
DELETE SEGMENT	DELETESEG
REDRAW ALL SEGMENTS	REDRAWALLSEG
SEGMENT TRANSFORMATION	SEGTRAN
SEGMENT VISIBILITY	SEGVIS
SEGMENT HIGHLIGHTING	SEGHIGHLIGHT
SEGMENT PRIORITY	SEGPRI
SEGMENT DETECTABILITY	SEGDET

Page X

Add the following to the end of sub-clause 6.2

```
BEGIN SEGMENT                ::= BEGSEG
                                   <SOFTSEP>
                                   <I:SEGID>
                                 <TERM>
END SEGMENT                  ::= ENDSEG
                                 <TERM>
```

Page X

Add the following to the end of sub-clause 6.3

```
METAFILE CATEGORY            ::= MFCATEGORY
                                   <SOFTSEP>
                                   <BASICCGM>
                                   <CGMEXT1>
                                   <GKSM>
                                 <TERM>

VDC NORMALIZATION            ::= VDCNORMALIZATION
                                   <SOFTSEP>
                                   <VDC:LOWVALUE>
                                   <SEP>
                                   <VDC:HIGHVALUE>
                                 <TERM>
```

Page X

Add the following to the end of sub-clause 6.5

```
DEVICE VIEWPORT              ::= DEVICEVIEWPORT
                                   <SOFTSEP>
                                   <DP:FIRSTCORNER>
                                   <SEP>
                                   <DP:SECONDCORNER>
                                 <TERM>

SET DEFERRAL STATE           ::= SETDEFERSTATE
                                   <SOFTSEP>
                                   <ASAP|BNIG|BNIL|ASTI>
```

```
                                <SEP>
                                <SUPPRESSED|ALLOWED>
                              <TERM>

CLEAR                    ::= CLEAR
                                <SOFTSEP>
                                <CONDITIONALLY|ALWAYS>
                              <TERM>

UPDATE                   ::= UPDATE
                                <SOFTSEP>
                                <PERFORM|POSTPONE>
                              <TERM>
```

Page X

Add the following to the end of sub-clause 6.7

```
TEXT FONT AND PRECISION  ::= TEXTFONTPREC
                                <SOFTSEP>
                                <I:FONTINDEX> {=/0}
                                <SEP>
                                <STRING|CHAR|STROKE>
                              <TERM>

CHARACTER VECTORS        ::= CHARVECTORS
                                <SOFTSEP>
                                <DELTAPAIR> {char height vector}
                                <SEP>
                                <DELTAPAIR> {char width vectors}
                              <TERM>

PICK IDENTIFIER          ::= PICKID
                                <SOFTSEP>
                                <I:SEGID>
                              <TERM>

LINE REPRESENTATION      ::= LINEREP
                                <SOFTSEP>
                                <I:BUNDLEINDEX> {positive}
                                <SEP>
                                <I:LINETYPE>
                                              {1=solid, 2=dash
                                              3=dot, 4=dash-dot
                                              5=dash-dot-dot
                                              <0 implement'n dependent}
                                <SEP>
                                <V:LINEWIDTH> {non-negative}
                                <SEP>
                                <K:LINECOLR>
                              <TERM>

MARKER REPRESENTATION    ::= MARKERREP
                                <SOFTSEP>
                                <I:BUNDLEINDEX> {positive}
                                <SEP>
```

```
                    <I:MARKERTYPE>
                                {1=dot, 2=plus
                                3=asterisk, 4=circle
                                5=cross (x)
                                <0 implement'n dependent}
                    <SEP>
                    <V:MARKERSIZE> {non-negative}
                    <SEP>
                    <K:MARKERCOLR>
                  <TERM>

TEXT REPRESENTATION     ::= TEXTREP
                    <SOFTSEP>
                    <I:BUNDLEINDEX> {positive}
                    <SEP>
                    <I:FONTINDEX> {=/0}
                    <SEP>
                    <R:SPACING>
                    <SEP>
                    <R:FACTOR>
                    <SEP>
                    <K:TEXTCOLR>
                  <TERM>

FILL REPRESENTATION     ::= FILLREP
                    <SOFTSEP>
                    <I:BUNDLEINDEX> {positive}
                    <SEP>
                    <HOLLOW|SOLID|PAT|HATCH|EMPTY>
                    <SEP>
                    <I:HATCHINDEX>
                                {1=horizontal,2=vertical
                                3=positive slope
                                4=negative slope
                                5=horiz/vert cross
                                6=+/- slope cross
                                <0 implement. dependent
                    <SEP>
                    <I:PATINDEX> {positive}
                    <SEP>
                    <K:FILLCOLR>
                  <TERM>

EDGE REPRESENTATION     ::= EDGEREP
                    <SOFTSEP>
                    <I:BUNDLEINDEX> {positive}
                    <SEP>
                    <I:EDGETYPE>
                                {1=solid, 2=dash
                                3=dot, 4=dash-dot
                                5=dash-dot-dot
                                <0 implement'n dependent}
                    <SEP>
                    <V:EDGEWIDTH>
                    <SEP>
                    <K:EDGECOLR>
```

Page X

The following forms sub-clause 6.10

```
RENAME SEGMENT          ::= RENAMESEG
                              <SOFTSEP>
                              <I:OLDSEGID>
                              <SEP>
                              <I:NEWSEGID>
                            <TERM>

DELETE SEGMENT          ::= DELETESEG
                              <SOFTSEP>
                              <I:SEGID>
                            <TERM>

REDRAW ALL SEGMENTS     ::= REDRAWALLSEG
                            <TERM>

SEGMENT TRANSFORM       ::= SEGTRAN
                              <SOFTSEP>
                              <I:SEGID>
                              <SEP>
                              <TM:TRANSMATRIX>
                            <TERM>

SEGMENT VISIBILITY      ::= SEGVIS
                              <SOFTSEPT>
                              <I:SEGID>
                              <SEP>
                              <VIS|INVIS>
                            <TERM>

SEGMENT HIGHLIGHTING    ::= SEGHIGHLIGHT
                              <SOFTSEPT>
                              <I:SEGID>
                              <SEP>
                              <NORMAL|HIGHLIGHTED>
                            <TERM>

SEGMENT PRIORITY        ::= <SEGPRI
                              <SOFTSEPT>
                              <I:SEGID>
                              <SEP>
                              <R:PRIORITY>  {0<=priority<=1}
                            <TERM>

SEGMENT DETECTABILITY   ::= <SEGDET>
                              <SOFTSEPT>
                              <I:SEGID>
                              <SEP>
                              <DET|UNDET>
                            <TERM>
```


End

APPENDIX 2

ANSI AND ISO ISSUES FOR LB-49 BALLOT

Accredited Standards Committee
X3, INFORMATION PROCESSING SYSTEMS*

Doc. No.: X3H3/87-48

Date: 13 Feburary 1987
Project: 347M
Ref. Doc.: X3H3/86-187
Reply to: Andrea Frankel
Hewlett-Packard, 61U
16399 W. Bernardo Dr.
San Diego, CA 92127-1899

Subject: **Comments to WG2 on CGEM Working Draft (CGM Addendum 1, November 1986)**

The U.S. comments on the Working Draft of CGM Addendum 1 (CGEM), submitted by X3H3 as TAG for WG2, consist of this letter and the accompanying document:

ANSC X3H3 CGEM Issues Log, Document X3H3/87-46

Editorial comments will be forwarded directly to Anne Mumford, the document editor. We commend the document editor on the excellent job she has done in producing such a polished document in a very short time.

Scope and Goals

We would like to point out, however, that the review of this document was considerably hampered by the lack of a Scope and Goals statement by which to evaluate the document. It is our understanding, based on discussion at the last WG2 meeting in Egham, that the CGEM work is expected and intended to encompass support eventually for both 3D (both GKS and PHIGS) and CGI functionality, and that more than one addendum is planned. It is also our understanding that WG2 has agreed to include such work in this first addendum if resources can be found to do it in a timely fashion, such that support for GKSM is not delayed.

We have not generated issues on these topics because of this understanding. However, we expect to be able to discuss these Scope and Goals questions at Valbonne, and formulate issues then if necessary, before voting on DP registration of this document.

Preliminary Issues Log

We would also like to note that the review of this document was hampered by the lateness of the Preliminary Issues Log, which arrived after the close of domestic balloting on the document itself. We appreciate getting it late rather than not at all, but it would have been preferable to have had it to review along with the Working Draft of the document.

We expect to bring voted U.S. positions on all of these issues, both ISO and ANSI, to the WG2 meeting at Valbonne, along with further editorial contributions. As there has not been time to formulate these positions within X3H3, we are submitting our issues log to ensure that all of the issues are on the agenda for the upcoming meeting.

Additions to Issue CGMA1

New alternative:

> 3. No, but the GKSM0 set is defined as one of the "shorthand" enumeratives for METAFILE ELEMENT LIST.

New arguments:

> c) Pro 3, contra a: GKSM and GKSM0 have no difference in semantics of elements or in the grammar. The difference between them is strictly a matter of which elements are included, and is thus more appropriately done with METAFILE ELEMENT LIST, since that is its purpose. This satisfies the need expressed in (a), and is cleaner.
>
> d) Pro 3, contra b: The concept of "category" should be reserved for cases where the interpreter must treat the metafile differently. These cases include:
>
> - a difference in the parts of the metafile in which an element is permitted to appear,
> - a difference in the order in which elements are permitted or required to appear,
> - a difference in required or prohibited elements,
> - disambiguation of semantics of elements with multiple interpretations.

Note: argument (a) becomes Pro 3 as well.

Accredited Standards Committee
X3, INFORMATION PROCESSING SYSTEMS*

Doc. No.: X3H3/87-47
Date: 27 Feburary 1987
Project: 347M
Ref. Doc.: X3H3/86-187
Reply to: Andrea Frankel
Hewlett-Packard, 61U
16399 W. Bernardo Dr.
San Diego, CA 92127-1899

Subject: Explanation of Some CGEM Concepts

The first draft of the Addendum 1 to the CGM (referred to as the CGEM, or Computer Graphics Extended Metafile), was somewhat lacking in explanation of the new elements. This led to many comments on Letter Ballot 47, some of which challenged the necessity for the new elements or raised questions about how they were intended to work.

The breakout group processing the LB47 comments decided in some cases not to generate issues from some of these comments, but to consider them requests for clarification of the document.

This paper is an attempt to shed some light on two particularly murky areas: the coordinate mapping scheme, and the notion of metafile categories.

If you feel that the discussion to follow does not answer your questions (or is contentious), please generate new issues as part of your response to Letter Ballot 49 on the open CGEM issues.

The Coordinate Mapping Scheme

In the original CGM (ANS X3.122-1986, IS 8632, henceforth simply "CGM"), each picture in the metafile is presumed to be independent of all other pictures. The VDC EXTENT specifies the portion of *VDC space* which is of interest, and this is mapped to the device's viewsurface isotropically. If the SCALING MODE is set to 'metric', the metafile is intended to be displayed at the fixed size obtained by interpreting the VDC EXTENT with the 'scale factor' of SCALING MODE. If the SCALING MODE is set to 'abstract', there is no guidance as to what size to render the picture. In all cases, the CGM contains no information to determine *where* on the device's viewsurface to render the picture.

In GKS, the NDC (Normalized Device Coordinate) space is treated as a virtual device viewsurface. The *Workstation Window* defines the portion of NDC space to be displayed, and the *Workstation Viewport* defines the portion of the device viewsurface to which the Workstation Window is isotropically mapped. When a CGM is created in a GKS environment, the VDC EXTENT element is used as the workstation window, and the DEVICE VIEWPORT function defined in Addendum 1 is used as the workstation viewport.

When a metafile is interpreted as an audit trail in a GKS environment, the interpreter needs to know what NDC space-to-VDC space mapping was assumed by the generator of the metafile, so that the VDC parameters of control, attribute and primitive elements can be properly converted and entered into the GKS state lists and segment store. One solution to this problem would be to assume that the standard NDC space of (0.0,0.0) to (1.0,1.0) was used, and that NDC space and VDC space are identical. The drawback to this approach is that it constrains the metafile to use VDC TYPE 'real', which is significantly less efficient on many systems than the use of integer coordinates. If VDC TYPE 'integer' is used, one might assume that the default VDC EXTENT of (0,0) to (32767,32767) would correspond to NDC space; however, the requirement in GKS to accomodate NDC coordinates within the range ± 7 would create problems for systems based on 16-bit arithmetic. The solution selected was to specify explicitly the region of VDC space which is to correspond to the NDC space of the GKS system. This element (named *VDC Normalization* in the November 1986 Working Draft of CGEM Addendum 1) provides a bounding square which encompasses all of the VDC EXTENTS in that metafile. (The function of this element might be more easily grasped if it is viewed as "MAXIMUM VDC EXTENT".)

Metafile Categories

The CGM provides several mechanisms by which an interpreter can tell if the metafile is one which it is prepared to interpret. The METAFILE VERSION correspond to versions of the standard. The METAFILE ELEMENTS LIST is an upper bound on the list of elements used in that metafile; it can be either an explicitly enumerated list of elements, or one of the "shorthand" enumeration types — DRAWING SET or DRAWING PLUS CONTROL SET.

The METAFILE ELEMENTS LIST mechanism is insufficient in light of the changes to the CGM standard introduced by its use as an audit trail as well as a picture capture mechanism, since it is not a matter of simply adding new functions. The interpreter needs to know which type of metafile it is interpreting, as the "GKSM" type may differ from the "basic CGM" type in several ways:

- The overall structuring of the grammar, and hence the metafile, by the choice of *delimiter elements*.
- Differences in *which elements are required, allowed, or prohibited* (i.e., some combinations of elements may be prohibited, and the appearance of one element may require the appearance of another).
- Differences in *where* in the metafile an element may appear (e.g., an element may be a "picture descriptor element" in CGM but a "picture element" in GKSM).
- Differences in the *order* in which elements may appear.
- Differences in *parameterization* (*syntax*) of a function with the same name.
- Differences in *meaning* (*semantics*) of a function when used in a different category.

Many of these potential differences are the subject of open issues; it is possible that the last two problems will be eliminated by adopting separate and distinct functions when the need arises.

The METAFILE VERSION element could in fact solve this problem, if it is ruled allowable that version "2" correspond to a metafile which incorporates functions or follows the grammar or semantics of Addendum 1 to version 1 of the standard. This issue will need to be addressed on a procedural level within ISO before the issue of whether METAFILE CATEGORY is needed can be resolved on a technical level.

The METAFILE DESCRIPTION is provided but its use is not standardized; it could, however, be used to contain information such as "GKSM0" by agreement between interchanging parties, and such use could be standardized in the Addendum although it has not been done so far.

CGM Addendum 1

Preliminary Issues Log

January 1987

CGMA1 Should there be a category of GKSMO as well as GKSM?

Keywords: GKSM, GKSMO, category.

Description: The category GKSM includes all GKS functionality. It may be useful to indicate to an interpreter that all these functions are not required for a particular metafile.

Alternatives:

1. Yes.

2 No.

3. No, but use Elem. List

Arguments:

a) Pro 1: Useful for interpreters in level zero systems.

b) Pro 1: Easy to define as a subset of the GKSM grammar.

History:

-logged at Frankfurt meeting.
-holder placed in working draft for GKSMO.

CGMA2 How are segments stored?

Keywords: segments.

Description:

Alternatives:

1. In line where they occur.

2. In a separate section of the metafile.

Arguments:

a) Con 2: not needed by current standards work.

b) Pro 2: may be useful for symbol libraries used across pictures.

c) Con 2: opens technical arguments re segments and their nature. The metafile should, perhaps just serve the functional standards.

History:

Alternative 1 chosen for working draft.

CGMA3 How should GDPs be stored in the CGM?

Keywords: GDP.

Description: Some of the GDPs which will be registered are already elements within the CGM. Does this affect the way that the elements are stored?

Alternatives:

1. As GDPs.

2. As CGM elements if these are available.

3. impl. dependent

Arguments:

a) does it matter - is it a language binding problem?

History:

-Logged at Frankfurt meeting.

CGMA4 How should SEGMENT DISPLAY PRIORITY be recorded.

Keywords: segment.

Description:

Alternatives:

1. real 0-1.

2. integer 0-n.

Arguments:

a) Pro 1: as GKS.

b) Pro 2: as CGI.

History:

-Logged at Frankfurt meeting.
-working draft uses Alternative 1.

CGMA5 should DEFERRAL MODE/STATE include implicit regeneration mode?

Keywords: deferral, regeneration.

Description: GKS has a single function for deferral mode and implicit regeneration whereas CGI sets just the deferral mode.

Alternatives:

1. Yes.

2. No.

3.

Arguments:

a) Pro 1: as GKS.

b) Pro 2: CGI function does not include it.

History:

-Logged at Frankfurt meeting.
-working draft includes a GKS-like DEFERRAL STATE.

CGMA6 What order should the transformation matrix for SEGMENT TRANSFORM be in?

Keywords: transformation matrix, segment.

Description: There is an incompatibility between the way this is done in the proposed functional standards.

Alternatives:
1) like CGI
2) " GKS
3) like both
4) make CGI/GKS the same

Arguments:

History:

Logged after Egham meeting.

CGMA7 Do pictures and sessions need to be distinguished?

Keywords: pictures, sessions.

Description:

Alternatives:

1. CGM pictures are used for all divisions of 'snapshot' pictures and for audit draws.

2. Have a concept of 'sessions' as well as 'pictures'.

Arguments:

a) Pro 1: simpler.

History:

Logged at Egham.
working draft to use alternative (1).

CGMA8 How should the transformation from NDC to VDC be accomplished.

Keywords: transformation, NDC, VDC.

Description:

Alternatives:

1. Use scaling mode.

2. Have new element VDC NORMALIZATION to achieve this.

Arguments:

a) Con 1: uses scaling mode in a different way to that envisaged in the CGM.

History:

-Logged at Frankfurt.
-Frankfurt - alternative 2 chosen.
-Egham - straw pole; Alternative 1 (2); Alternative 2 (5); abstentions (2).
-Alternative 2 adopted for working draft

CGMA9 Is the term segment the right one?

Keywords: segment.

Description: The term segment carries a lot of implied meaning which is not necessarily consistent - GKS/CGI.

Alternatives:

1. Use term 'segment' and let the meaning be environment dependent.

2. Use term 'group'.

Arguments:

History:

-Logged at Egham
-Straw pole: 'group' (4), 'segment' (3), abstentions (2)
-working draft uses 'segment'

CGMA10 Where should the formal grammars be located?

Keywords: formal grammars

Description: The formal grammars will include grammars for the different categories which will relate to the various functional standards.

Alternatives:

1. In the CGM Addenda

2. In the standard to which the grammar pertains

Arguments:

a) Pro 1: easy to process

b) Con 2: requires addenda or update for the functional standards

History:

-Logged at Egham
-Straw vote: Alternative 1 (2), Alternative 2 (5), abstentions (2)
-to be left in the working draft

CGMA11 Should there be a super-grammar?

Keywords: formal grammar.

Description: The addition of elements in addenda to support new categories could result in these elements appearing in that category only, or being also included as part of a global grammar.

Alternatives:

1. Yes.

2. No.

Arguments:

History:

-Logged at Egham.
-Straw vote at Egham 7.1.1.
-working draft to have a super grammar.

CGMA12 Should TEXT FONT and PRECISION be added as an element?

Keywords: font, font precision.

Description: GKS has a single element and CGM has 2 for this purpose.

Alternatives:

1. Yes.

2. No.

Arguments:

a) Pro 1: maps GKS exactly.

b) Pro 2: unnecessary.

History:

-Logged at Egham, working draft to adopt Alternative 1.

CGMA13 Should there be an element for HATCH AND PATTERN INDEX.

Keywords: Latch, pattern, fill area.

Description: GKS has a single element for these, whereas CGM splits these.

Alternatives:

1. Yes.

2. No.

Arguments:

a) There are problems with the bundle tables for hatch index and pattern index. In GKS they are allowed to follow different rules.

History:

-Logged at Egham, no technical solution emerged, alternative 2 adopted for working draft.

CGMA14 Should there be a WORKSTATION WINDOW element.

Keywords: workstation, window.

Description: GKS has a WORKSTATION WINDOW function. In the CGM Annex E this has been mapped to VDC EXTENT.

Alternatives:

1. WORKSTATION WINDOW is mapped to VDC EXTENT.

2. Have new WORKSTATION WINDOW element.

Arguments:

a) Con 2: inconsistent with current CGM Annex E.

History:

-Logged at Egham
-working draft to adopt alternative (1).

CGMA15 Does REDRAW ALL SEGMENTS carry with it an implied meaning on interpretation.

Keywords: segments, redraw.

Description: In GKS the workstation display surface is cleared on this call. This is not the case in CGI. Can the same function be used in the metafile, which may have different meanings on interpretation.

Alternatives:

1. The appearance of the element does not imply any particular action on interpretation.

2. Have different elements for the GKS, CGM (and any other) meanings so that the action on interpretation can be guaranteed.

clear
CGI RDAS
GKS RDAS

3 clear CGI RDAS

Arguments:

History:

-Logged at Egham.
-working draft to use alternative (1).

Should the Addendum

CGMA16 ~~Does the Addendum need to~~ include device viewport specification units.

Keywords: viewport.

Description: The CGI has the capability for setting the device viewport specification units to be used. GKS does not have this capability.

Alternatives:

1. Yes.

2. No.

Arguments:

a) Pro 2: can be added later for CGM.

b) Pro 2: not necessary if the default for device viewport specification units is the same as GKS.

History:

-Logged at Egham.
-working draft to use alternative (2).

CGMA17 Should EDGE REPRESENTATION be added with the other representation function?

Keywords: edge representation.

Description: The CGM includes no representation elements so these need to be added in the addendum. GKS does not need the EDGE representations although GKS-3D will need them.

Alternatives:

1. Yes.

2. No.

Arguments:

a) Pro 1: cleaner addition of elements.

b) Pro 1: well defined for GKS-3D.

c) Pro 2: not needed for GKS.

History:

-Logged after Egham.
-working draft uses alternative 1.

CGMA18 Should the addendum work include the definition of item types returned in the functional standard?

Keywords: item types.

Description: Item types returned to the functional standards on metafile input have never been standardized. There is a need for this in order to write standard programs.

Alternatives:

1. No standardization.

2. Item types should appear in functional standards.

3. In Addenda. Annex

Arguments:

History:

-Logged at Egham.
-working draft to adopt (3) but (2) is preferable in long term.

CGMA19 How should item types be defined?

Keywords: Item types.

Description: GKS Annex E defines item types in an arbitrary way. Further item types will be needed for the other CGM elements and for functions in the other standards.

Alternatives:

1. Use GKS Annex E item types and expand where necessary.

2. Use a new definition for item types.

Arguments:

History:

-Logged at Egham.
-working draft adopts alternative (2) and proposes item types should be based on the binary op-codes which allows future extension.

CGMA20 Where should the segment elements appear?

Keywords: Segment.

Description:

Alternatives:

1. In a group on their own.

2. As part of other group - segment control, segment attributes.

Arguments:

a) Pro 1: as GKS.

History:

-Logged after Egham.
-working draft uses (1).

CGMA21 How should a category be defined?

Keywords: category.

Description:

Alternatives:

1. As a single name, eg CGM, GKSM.

2. As a list of keywords, eg CGM, GKSM, 2D, 3D.

Arguments:

a) Pro 2: more general.

b) [illegible] of [illegible] con 2

History:

-Logged at Egham.
-working draft to use alternative (1).

CGMA22 What group of elements should VDC NORMALIZATION fall into.

Keywords: VDC Normalization.

Description:

Alternatives:

1. Picture Descriptor.

2. Control.

3. Metafile Descriptor.

Arguments:

History:

-Logged at Egham.
-Alternative 3 used in working draft.

.CGMA23 Do CLEAR and UPDATE have any implied meaning for the interpreter?

Keywords: clear, update.

Description: The meanings of the workstation control functions differ between the functional standards and the CGI. Do we need more than one element or do we have the GKS meaning for this first addendum.

Alternatives:

1. Leave interpreter to sort out the meaning - it should know what to do for a GKSM category.

2. Have the GKS meaning for the first addendum and new elements for future additions for CGI.

Arguments:

History:

-Logged at Egham.
-working draft to use alternative (1).

Accredited Standards Committee
'3, INFORMATION PROCESSING SYSTEMS*

Doc. No.: X3H3/87-46 (cover)

Date: 13 Feburary 1987
Project: 347M
Ref. Doc.: X3H3/86-46
Reply to: Andrea Frankel
Hewlett-Packard, 61U
16399 W. Bernardo Dr.
San Diego, CA 92127-1899

Subject: ANSI X3H3 CGEM Issues Log

Letter Ballot 47 on the CGEM generated numerous comments. These were correlated, and compared to the ISO issues log, which fortuitously arrived in Ft. Collins in the middle of the meeting (now being distributed as X3H3/87-32). An ANSI issues log was started, containing those issues generated from LB47 (and from subsequent discussion while processing the letter ballot comments) which were not covered by the ISO log. There was not sufficient time to discuss and vote these issues in either the breakout group or in X3H3.3; therefore, there are no recommendations listed.

A number of "major philosophical" issues were identified and discussed in X3H3.3. Not all of these were written up as issues. Here is the resolution of that discussion:

1. *GKSM support:* WG2 has decided that GKSM support is the highest priority; we doubt we could derail that, although we might choose to concentrate our efforts on other aspects of the work. We believe that "support" does not require a 1:1 mapping from GKS functions to CGEM functions, and will continue the approach we took with CGM, to satisfy many application needs, including (but not limited to) GKS's.

2. *3D support (general):* It is our understanding that the work of extending the metafile will entail adding several addenda. The scope and goals for this work includes 3D, but WG2 will only accept including it in this first addendum if it does not delay the (2D) GKSM solution. If we want to push for earlier 3D work, we need to provide a U.S. document editor (volunteers, anyone?) by the Valbonne meeting. The Reference Model work is most appropriate arena for deciding the relationship of CGEM and 3-D, and we should probably do our homework there before lobbying further on this.

3. *PHIGS support:* This could not be addressed now; anyone interested in it should prepare a position paper to be discussed at Tulsa. Again, the Reference Model needs to be addressed.

4. *Segmentation (CGI vs. GKS model):* The consensus was strongly in favor of using the CGI segmentation model, as this was designed both to service GKS and to transcend its limitations. An issue exists, and a position paper will be generated between now and the Tulsa meeting, explaining how to map GKS functions into CGI functions in this area.

5. *Form of the document:* There were several commenters who objected to the "delta document" format. This is not a delta document, it is an *Addendum* to an ISO standard, and must be done in the form you received. The page number references could not be filled in as the ISO version of CGM has not yet been typeset; the Addendum should be less objectionable once those references are supplied, and the "Concepts" sections fleshed out.

6. *Redundant functionality:* Issues were generated for each of the cases. It should be noted that these issues are of greater concern to other ISO members than they are to us. While we have objected to these redundancies in the past and will probably continue to object to them, this is an area where we will probably be more willing to compromise in exchange for other more important issues.

This issues log is in addition to the ISO log (X3H3/87-32); you will need to refer to both logs to see if issues you wish to raise are already covered.

ANSI.1 Should semantics of all elements in the addendum be unambiguously defined?

Keywords: semantics, ambiguity

Description: The CGEM is intended to serve a number of constituencies, either immediately or in the future in additional addenda. Some functions, e.g., CLEAR, have different meanings depending upon the environment and the client using the CGEM. Should the CGEM assign unambiguous meanings to all elements, or are the semantics variable by application or perhaps even undefined? Many issues on individual elements may be answered by the answer to this issue.

Alternatives:

1. CGEM is just a syntactic framework, semantics are by agreement between exchanging parties.
2. Semantics may vary by Metafile Category, CGM addenda will specify the semantics.
3. each element has unique semantics, where different functionality is needed different elements will be included.

Arguments:

a. pro 3, con 1 & 2: If different actions or interpretations are expected, then separate elements must be defined.

b. con 1: Insufficient; any metafile which avoids private items (e.g. ESCAPE, GDP, parameter values) should be able to interpreted unambiguously.

c. pro 1: Consistent with the approach that we are standardizing the metafile, not interpreters.

d. pro 3, con 2: (2) makes it very difficult for other CGEM users (i.e. non-GKS) to determine how to use the metafile and may prevent certain uses.

e. pro 2: Smaller number of elements to be defined; more efficient use of opcode space.

f. pro 2: (2) is equivalent to (3) in GKS environments, using category as an "opcode prefix", leading to fewer elements.

g. con 2: Forces choice of semantics in 'cgmext1' category, when client is not clear.

h. con 2: In category 'cgmext1', not all meanings are available; some are locked out.

i. pro 3: Facilitates coordinating different standards' use of some encoding opcode space (GDS, CGI, CGEM).

History: Logged at 1/87 X3H3 meeting (Ft. Collins), from LB47

References (LB47): Apollo/1, HP/T7, PVI/T9, PVI/C3, SAN/1D, HS/T6, DECUS/6

ANSI.2 Should the contents of the 'drawing set' (shorthand) of Metafile Elements List be changed by the addendum?

Keywords: drawing set, compatibility

Description: The first addendum has changed the meaning of 'drawing set'. This means that an existing CGM interpreter would not properly interpret the meaning of drawing set.

Alternatives:

1. no, devise a new set to include 'drawing set' plus new elements
2. yes, (as is) allow the meaning to change.

Arguments:

a. pro 1, con 2: (2) violates the ISO rules for Addenda.

b. pro 1, con 2: (2) will make existing CGM interpreters malfunction.

History: Logged at 1/87 X3H3 meeting (Ft. Collins), from LB47

References (LB47): BOE/3, PVI/C1, PVI/C2, HS/T1

ANSI.3 What is the appropriate specification of Viewport for this addendum?

Keywords: viewport, data types

Description: The first draft shows DP, which are "meters or other device-dependent units" as in GKS. The CGI (DP9636) has specified a system of viewport specification which supports several styles of DP units — metric scale, abstract device-independent, proportional device units.

Alternatives:

1. Retain viewport specification as is in the first draft of CGEM.
2. Revise viewport specification as per the CGI (DP9636).
3. Define DP units strictly as a fraction of the available device view surface.

Arguments:

a. Pro 2: Use of this method of specification would provide more flexibility for current users and would provide a basis for future extensions.

b. Pro 2: Since most CGI control functions will likely become part of CGM, it would be logical to adopt the CGI approach to the specification of viewports.

c. Con 1, Pro 3: (3) would retain the device independence of CGM.

d. Pro 1, Pro 2: Satisfies GKS.

e. Pro 1: Identical to GKS and is all that is required for current scope.

f. Pro 2: Encompasses all of the other suggestions.

History: Logged at 1/87 X3H3 meeting (Ft. Collins), from LB47

References (LB47): HP/T15, MDC/T5

ANSI.4 What segmentation model should be in CGM addenda?

Keywords: segmentation

Description: The current extensions contain a segmentation model that is adequate to support GKS, but may not adequately support other clients (CGI, PHIGS, etc.) Should another model, capable of supporting other clients in this and in future addenda, be adopted at this point?

Alternatives:

1. leave segmentation as in the first draft of CGEM, supporting only GKS-like systems.
2. adopt the CGI segmentation model.

Arguments:

a. Pro 2: Mapping from GKS to CGM will be identical to mapping from GKS to CGI.

b. Pro 2: CGI model is flexible enough to support a variety of clients.

c. Pro 2: Maintains maximum compatibility with CGI while adequately supporting GKS.

d. Con 2: The CGI segmentation model may not be stable enough.

e. Con 1: If GKS segmentation is used now, much redundancy may be introduced later in expanding to CGI.

f. Pro 1: Simpler and more direct mapping to GKS.

g. Pro 2: GKS implementors' experience with the GKS segmentation model has resulted in the improved model in CGI.

History: Logged at 1/87 X3H3 meeting (Ft. Collins), from LB47

References (LB47): HP/C3, MGI, PUK, BOE/2, MDC/T4, CHIN/8/9/13, HS/T4

ANSI.5 What functions or elements may be included in segments?

Keywords: segmentation

Description: It is unclear in the draft addendum what picture elements (control, attribute, primitive) are allowed to occur between BEGIN/END SEGMENT. Occurrence between BEGIN/END SEGMENT does not necessarily imply that the element (function) goes into segment storage on the client system (just as certain control and workstation functions can occur in CGI, GKS, etc but do not get stored).

Alternatives:

1. any picture elements may be included, and the meaning will be left to the interpreting system.
2. any picture elements may occur, and the meaning will be defined by category.
3. some restricted set of elements, to be defined, will be allowed.

Arguments:

a. Con 1: Portability will be diminished.

b. Con 3, Pro 2: Retain ability to support different environments unambiguously.

c. Con 3: Will probably preclude supporting both CGI and GKS (we have an imperfect crystal ball).

d. Pro 2: Allows more natural mapping between the activities of the client system and the contents of the CGM.

History: Logged at 1/87 X3H3 meeting (Ft. Collins), from LB47

References (LB47): Apollo/15

ANSI.6 Is the "definedness" of a segment limited to the current picture?

Keywords: segmentation

Description: It is implied, but not clearly explained, that segments are defined within a single picture, and that their definition does not exist outside of that picture.

Alternatives:

1. Yes.
2. No.
3. Yes, but segments may be present in Metafile Descriptor, and these may be referenced in all pictures.

Arguments:

a. Pro 1: For pictures to be wholely self-contained and logically independent, all segments referenced within a picture must be defined within that picture.

b. Pro 1: Corresponds directly with the way segments are defined by current clients.

c. Con 2: Metafile is a single driver or workstation. Storage would be an instantiation of WISS, and reference would be an audit of M0, hence metafile is being required to record two workstations.

d. Pro 2: Provides symbol library facility.

e. Pro 2: Reduces file size.

f. Con 2: If WISS is desired, make a new Metafile Category.

g. Pro 3: Serves GKS adequately and provides non-GKS clients with a WISS-like or symbol library facility.

History: Logged at 1/87 X3H3 meeting (Ft. Collins), from LB47

References (LB47): Apollo/2

ANSI.7 Are such functions as UPDATE, SET DEFERRAL STATE appropriate in a metafile standard?

Keywords: interactive

Description: These functions are typically encountered in interactive systems such as GKS, and have been included to provide faithful audit capabilites in GKS applications. It is not clear whether they have a purpose or meaning in a metafile.

Alternatives:

1. Yes, retain the functions.
2. No, delete the functions.

Arguments:

a. Pro 1: Needed for faithful GKS audit.

b. Pro 1: If session restart is a goal, needed to restore system state to point of restart.

c. Pro 2: Simplicity and minimality.

d. Pro 1: Gives the generator of the metafile the ability to batch changes at interpretation time in a manner that avoids unnecessary regeneration.

e. Pro 1: Argument (b) is also applicable if the intention is to backtrack and restart at a previous point in the metafile.

History: Logged at 1/87 X3H3 meeting (Ft. Collins), from LB47

References (LB47): MGI/2

ANSI.8 Should the "interactive" values of the 'deferral mode' parameter of DEFERRAL STATE be in the metafile?

Keywords: interactive

Description: The meaning of these parameter values is not clear in the metafile environment. They have been included to provide faithful audit capabilities for GKS.

Alternatives:

1. Yes, retain the values
2. No, delete the values
3. Yes, but combine to have one value like BNI of CGI.

Arguments:

a. Pro 2: They are meaningless in the absence of input.

b. Pro 2: There should be no interaction or input mixed with picture interpretation.

c. Pro 3: The CGM conceptually corresponds to a single device, so there is no difference between BNIG and BNIL.

d. Pro 1: Required for GKS session restart and state restoration.

History: Logged at 1/87 X3H3 meeting (Ft. Collins), from LB47

References (LB47): HP/T6, CHIN/3

ANSI.9 Should the METAFILE DEFAULTS REPLACEMENT encoding in the Binary Encoding be improved?

Keywords: defaults, binary

Description: The MDR encoding in the Binary (IS 8632/3) is difficult to generate. The other two encodings have broken the element into a BEGIN/END pair. This addendum could add such an encoding to the Binary, while retaining the current method.

Alternatives:

1. Yes, add an alternate method.
2. No, leave as is.

Arguments:

a. Pro 1: Will align binary encoding with other encodings.

b. Pro 1: Much simpler to implement.

History: Logged at 1/87 X3H3 meeting (Ft. Collins), from LB47

References (LB47): MDC/T3

ANSI.10 Is there a need for the redundant specification of certain text attributes — CHARACTER ORIENTATION & CHARACTER VECTORS, and the font and precision elements — or can this redundancy be eliminated?

Keywords: text attributes, redundancy

Description: The addendum has included redundant functionality in order to have the functionality presented in a style that is closer to that of GKS. This improves the fidelity of the audit capabilities for GKS. However it is not clear that the redundancy is justified.

Alternatives:

1. No, eliminate the redundant elements
2. Yes, retain the redundant elements

Arguments:

a. Con 2: These functions violate the design guidelines of minimality, conciseness and orthogonality.

b. Con 2: Elements are entirely redundant and should not be added.

c. Con 2: 8632 is adequate to support the needs of GKS and CGI in this area; elements are not necessary.

d. Con 2: The relationship between the individual aspect source flags is not defined.

e. Pro 2: Maps to GKS exactly.

History: Logged at 1/87 X3H3 meeting (Ft. Collins), from LB47

References (LB47): DECUS/5, HP/7b, HP/T8, BOE/4, BOE/18, PVI/T1, PVI/T2, PVI/T3, CHIN/4, HS/T7, CHIN/5

ANSI.11 Does the meaning of TEXT FONT INDEX change in the addendum?

Keywords: font index, ambiguity, redundancy

Description: In CGM, TEXT FONT INDEX is an index into a table of font designations, which the user may load with private values. The index is positive. In GKS, it is more like an enumerative selector, like LINETYPE, and private values are selected with negative values.

Alternatives:

1. No; if both meanings are needed, invent a new element.
2. Yes, changes by category.

Arguments:

a. Pro 1: Better to have new semantics be a separate element.

b. Pro 2: Avoids unnecessary proliferation of elements, and is more straightforward.

c. Con 2: Ambiguous meaning in category CGMEXT1.

History: Logged at 1/87 X3H3 meeting (Ft. Collins), from LB47

References (LB47): HPT/7a, HS/T15

ANSI.12 Should CGEM coordinate with CGI on opcodes?

Keywords: opcodes, compatibility, encodings

Description: The data stream encodings of the CGEM have not been coordinated with those of the CGI. Should there be a single, unified opcode or name space which includes all functions of CGI, CGM and CGEM?

Alternatives:

1. Any given opcode or function name has exactly the same parameterization in all three standards.
2. Allow a small number of "context-dependent" (i.e. standard dependent) variations, only as needed.
3. No coordination other than all are supersets of CGM.

Arguments:

a. Pro 1, Pro 2: All three are at the same level in the graphics pipeline and it is reasonable to expect products which can interpret more than one of this set.

b. Pro 3: May expedite the progress of each individual standard.

c. Pro 1, Con 2: Confusing to deal with opcodes of similar but not exactly the same meaning.

History: Logged at 1/87 X3H3 meeting (Ft. Collins), from LB47

References (LB47): HP/T14.

ANSI.13 Should the CHARACTER VECTORS be allowed to change within a (compound text) string?

Keywords: compound text, text attributes

Description:

Alternatives:

1. Yes.
2. No.

Arguments:

a. Pro 2: The prohibition on changing the direction of labeling within a compound text primitive is precisely the reason why IS 8632 has the CHARACTER HEIGHT separate from CHARACTER ORIENTATION. Allowing such a change within a string would place an incredible burden on implementors, and for no justifiable reason (complicating text alignment calculation).

b. Pro 2: Since GKS does not contain the concept of compound text, this GKS-ish element (if allowed to remain in CGEM at all) need not be incorporated in compound text.

c. Pro 1: Introduces new capability to label along a curve, as in PostScript, which is useful.

d. Contra c: Creeping functionality!

e. Contra c: Can do this already, without doing it in a compound string. Doing it in compound text only adds the ability to do text alignment on the entire string, which is probably useless in this situation.

History: Logged at 1/87 X3H3 meeting (Ft. Collins), from LB47

References (LB47): HP/T9

ANSI.14 Should CGEM incorporate the Font work of SC18 at the present time?

Keywords: fonts, text attributes

Description: The current text attribute model does not align well with current typographical practice that is reflected in the Font ID and description activities of SC18.

Alternatives:

1. Yes.
2. No.

Arguments:

a. Con 1: This is premature until we have a proposal to evaluate.

b. Con 1, Pro 2: If the models are sufficiently different, it may be cleaner to add TYPOGRAPHICAL TEXT and leave the graphical text model as is.

History: Logged at 1/87 X3H3 meeting (Ft. Collins), from LB47

References (LB47): MDC/T7

ANSI.15 Should the scope of Addendum 1 cover additional stable CGI functionality?

Keywords: scope

Description: For example, CIRCULAR ARC CENTRE BACKWARDS, CLOSED FIGURES, PIXEL ARRAY.

Alternatives:

1. Yes.
2. No.

Arguments:

a. Pro 1: Increases perceived utility of the extended standard.

b. Pro 1: implementors of CGM are already asking for this.

c. Con 1: An issue will have to be generated for each proposed element to examine potential side effects.

History: Logged at 1/87 X3H3 meeting (Ft. Collins), from LB47

References (LB47): MDC/T2, MGI/1

ANSI.16 What is the intended result when SCALING MODE and DEVICE VIEWPORT appear in the same metafile?

Keywords: Scaling mode, Device Viewport

Description: The occurrence of both is possible only in the super-grammar. Priority of one over the other needs to be defined.

Alternatives:

1. Use the last specified.
2. Prohibit any instance of a metafile from having both.
3. Work out an effect of the combination of the two.

Arguments:

a. Con 2: The current philosophy of the super-grammar does not allow this option.

b. Pro 2, contra a: This is simple enough to change. Who said the super-grammar can't have sensible restrictions in it?

c. Pro 1: This is an adequate solution for a well-defined result.

d. Con 3: Any useful effect can be accomplished by one or the other.

e. Pro 3: Some combinations may be valid and need to be explained.

f. Pro 2: Cleanest solution.

g. Con 1, Pro 2: These are two inherently different approaches to adding presentation directives to the metafile, and there is no reason to mix them. It would only increase confusion.

History: Logged at 1/87 X3H3 meeting (Ft. Collins), from LB47

References (LB47): HP/T4

ANSI.17 What should be the data type for segment_id?

Keywords: Segment identifier, data types

Description:

Alternatives:

1. N (Name).
2. I (Integer). (as in November '86 draft)
3. SN (Segment Name).

Arguments:

a. Pro 1: Matches GKS.
b. Pro 2: Matches CGI.
c. Pro 3: More virtual, yet specific to segments.

History: Logged at 1/87 X3H3 meeting (Ft. Collins), from LB47

References (LB47): BOE/20, HS, PVI/T4

ANSI.18 Does the transform matrix need a data type?

Keywords: data types

Description: The components of the transformation matrix are of two different types. Should a data type be assigned to the matrix?

Alternatives:

1. Don't associate a data type with the transformation matrix.
2. Add a data type for the transformation matrix.

Arguments:

a. Pro 1: A new data type does not have to be added.

b. Pro 2: Cleaner.

c. Pro 1: Separate parameters are easier to write into and read from the metafile.

d. Con 2: The matrix is adequately described by the data types of its components.

e. Pro 1: Matches GKS.

History: Logged at 1/87 X3H3 meeting (Ft. Collins), from LB47

References (LB47): BOE/22

ANSI.19 What should be the data type for METAFILE CATEGORY?

Keywords: data types, category

Description:

Alternatives:

1. enumerative
2. index
3. other

Arguments:

a. Con 1: "Enumerative" implies a complete and closed set, and this usage would be inconsistent with that. In some environments (e.g. Pascal) problems can arise if the set changes. "Index", as used for LINETYPE, might be a better choice.

b. Pro 1, contra a: you're fighting an old battle. CGM (an ANSI and ISO standard) defines the enumerative data type as extensible through private and registered values. This usage is entirely appropriate.

c. Pro 2: Fewer problems in language implementations when set is extended.

d. contra c: Language implementations will already have to deal with this, since CGM has other enumeratives which can be extended.

e. Pro 2: Better suited to private values.

f. Pro 1: More informative — "GKSM" conveys more than "2", especially in a Clear Text environment!

History: Logged at 1/87 X3H3 meeting (Ft. Collins), from LB47

References (LB47): Apollo/18b

ANSI.20 Are segments clipped before or after segment transformation?

Keywords: segmentation, clip, transformation.

Description:

Alternatives:

1. Clip before segment transformation.
2. Clip after segment transformation.
3. Unspecified, i.e. implementation-dependent.
4. Category-dependent.

Arguments:

a. Con 1: Does not support GKS.

b. Pro 2: Adequate support for GKS.

c. Con 3: Unacceptable to have unpredictable results.

d. Pro 4: Can support current clients and future clients with non-GKS clipping model.

History: Logged at 1/87 X3H3 meeting (Ft. Collins), from LB47

References (LB47): Apollo/1

ANSI.21 What meaning is attached to "BEGIN SEGMENT (A)" when segment "A" exists?

Keywords: segmentation, formal grammar

Description: A sequence of syntactically correct elements may be in error on interpretation depending on the specific parameter value (segment name). For example,

```
(1)
BEGIN  SEGMENT ( A )
END    SEGMENT
DELETE SEGMENT ( A )
BEGIN  SEGMENT ( A )  --> this is ok

(2)
BEGIN  SEGMENT ( A )
END    SEGMENT
BEGIN  SEGMENT ( B )  --> this is ok

(3)
BEGIN  SEGMENT (A)
END    SEGMENT
BEGIN  SEGMENT (A)     --> this is the problem
```

Alternatives:

1. Syntax error. (e.g. page 42 CGM)
2. Equivalent to APPEND_SEGMENT function.
3. Unspecified, i.e. implementation-dependent.
4. Category-dependent.

Arguments:

a. Pro 3: CGM standardizes the metafile, not the interpreter. This question addresses interpretation and is in the scope of the client.

b. Contra a, Pro 4: This can be addressed as a question of syntax.

c. Contra b, Pro 3: Our formal grammar has no conditional constructs to handle this! This is NOT a question of syntax. You are asking to put run-time resource management into a formal grammar, where it doesn't belong.

History: Logged at 1/87 X3H3 meeting (Ft. Collins), from LB47

References (LB47): Apollo/1

ANSI.22 What data type should the PICK IDENTIFER be?

Keywords: PICK IDENTIFIER, data types

Description: In other standards, identifiers are defined as INTEGER data. In the interest of consistency, it might be best to define the PICK IDENTIFIER as INTEGER, thus eliminating the need for a separate NAME data type.

Alternatives:

1. Retain data type 'N' for PICK IDENTIFIER.
2. Change to data type INTEGER.

Arguments:

a. Pro 2: Con 1: is not consistent with other identifiers.

b. Pro 1, Con 2: inconsistent with GKS.

c. Contra b: SEGMENT IDENTIFIER might become NAME data type.

History: Logged at 1/87 X3H3 meeting (Ft. Collins), from LB47

References (LB47): HS/T7, PVI/4

ANSI.23 Should a PICK PRIORITY element be added, separate from the SEGMENT DISPLAY PRIORITY element?

Keywords: PICK PRIORITY, SEGMENT DISPLAY PRIORITY

Description: PICK PRIORITY does not appear in clause 5 and needs to be defined as either a unique element, or as part of the SEGMENT DISPLAY PRIORITY element (as implied in clause 4).

Alternatives:

1. Yes.
2. No.

Arguments:

a. Pro 1: like CGI.

b. Pro 1: more flexible and can handle CGI, GKS and other future clients.

c. Pro 1: is orthogonal.

d. Con 1: is less efficient for GKS.

e. Pro 2: like GKS.

History: Logged at 1/87 X3H3 meeting (Ft. Collins), from LB47

References (LB47):

ANSI.24 Should a value of two (2) be used for the Metafile Version element in metafiles that contain items from, or follow grammars of, CGM Addendum 1?

Keywords: Metafile Version, compatibility

Description: The presence of items in a metafile from Addendum 1 would adversely affect current CGM implementations that attempt to process it.

Alternatives:

1. Yes.
2. No.

Arguments:

a. Pro 1: prevents problems with using extended metafiles with current interpreter implementations.

b. Con 1: may violate the ISO rules governing the use of the Metafile Version element (i.e. is this an addition or a change?).

c. Contra b: If this is a problem, we can certainly find a solution, such as adding the line "However, metafiles using grammars or elements defined in addendum 1 are to include METAFILE VERSION with value '2'" after the current line describing the version as '1'.

d. Pro 1: this is exactly what the METAFILE VERSION was intended for, namely, to let an interpreter know that it had encountered a metafile using constructs not known at the time the interpreter was written.

History: Logged at 1/87 X3H3 meeting (Ft. Collins), from LB47

References (LB47):

ANSI.25 Should the discussion of conformance include references to generators and interpreters?

Keywords: generator, interpreter, conformance

Description:

Alternatives:

1. Yes.
2. No.

Arguments:

a. Con 1: CGM explicitly excludes this and the Addendum should be consistent.

b. Pro 2: this is currently being handled by client Application Profiles.

c. Con 1: if in a standard, it belongs in the client standard of the relevant category (i.e. GKSM).

History: Logged at 1/87 X3H3 meeting (Ft. Collins), from LB47

References (LB47): NBS, MDC/T8

ANSI.26 What is the relationship between CGM Addendum 1 and GKSM? i.e. is the Addendum meant to conform to GKSM with a strict 1-to-1 mapping, or on a more general level with a well defined mapping?

Keywords: conformance, GKSM, compatibility

Description: In some areas it appears the Addendum is attempting to provide direct function-for-function support for GKSM. In other areas, however, it appears that the Addendum has been generalized to encompass other standards, such as the emerging CGI specification.

Alternatives:

1. direct 1-to-1 GKSM support.
2. More generalized support with well defined mapping to GKS.

Arguments:

a. Pro 2: as there will be future addenda for other clients, ease of extension should be considered.

b. Pro 1: quicker.

c. Pro 2, Contra b: some generality should be achievable without undue delay.

History: Logged at 1/87 X3H3 meeting (Ft. Collins), from LB47

References (LB47): MDC/T8

ANSI.27 How is conformance with CGM Addendum 1 to be defined?

Keywords: conformance

Description:

Alternatives:

1. Levels.
2. Categories.

Arguments:

a. Pro 2: conformance by category is easily defined and tested.

b. Pro 2: categories can be accurately tailored to a particular constituency class.

c. Con 1: levels are only meaningful if all desired variations have a strict superset/subset relationship.

d. Pro 1: matches GKS.

History: Logged at 1/87 X3H3 meeting (Ft. Collins), from LB47

References (LB47): NBS, MDC/T8

ANSI.28 Should Deferral Mode have a default listed in clause 6?

Keywords: Deferral Mode, default

Description: Since the default CGM category is 'Basic', it may be invalid to list a default for an element that does not exist in the default category.

Alternatives:

1. Yes.
2. No.
3. All defaults are defined but only apply to the category in which they appear.

Arguments:

a. Pro 1: CGM currently has no deferral mode, so the metafile category 'basic CGM' should not contain this element.

b. Contra a: It is not a problem, as long as the default deferral mode matches IS8632 (ASTI, by inference).

c. Pro 2: All modal elements require a default setting by definition.

d. Con 1, Pro 3: clause 6 is intended to be a complete list of defaults (for reference purposes).

e. Con 1: the DEFERRAL MODE element needs to have a default.

History: Logged at 1/87 X3H3 meeting (Ft. Collins), from LB47

References (LB47): Chin/11

Accredited Standards Committee
X3, INFORMATION PROCESSING SYSTEMS*

Doc. No.: X3H3/87-65
Date: 25 Feburary 1987
Project: 347M
Ref. Doc.: X3H3/86-187
Reply to: Lofton Henderson
Henderson Software
P.O. Box 4036
Boulder, CO 80306
— or —
Andrea Frankel
Hewlett-Packard, Bldg. 61
16399 W. Bernardo Drive
San Diego, CA 92127

Subject: **U.S. Editorial Comments on CGM Addendum 1 (Working Draft)**

Comments on Part 1

1. An introduction stating the scope, need and purpose of the addendum is needed.
2. The need for VDC NORMALIZATION is not made clear. The justification given in recent working meetings involves interpretation of metafiles in GKS environments. The document should explain the element's relation to other elements affecting the transform (VDC EXTENT, DEVICE VIEWPORT) as well as to terms such as "the NDC of the graphics system" and "physical device units". Restructuring (or at least renaming) the functionality as MAXIMUM VDC EXTENT might make the explanation easier, since it would then describe what information the element conveys rather than what GKS does with that information.
3. There are references throughout to "extended metafile". This should not go in the text of the addendum, as the addendum becomes part of the standard identified as ISO 8632. That is, there will be no distinction between the CGM standard and the "extended CGM standard" when the work is complete. Any discussion that needs to refer to the contents of this addendum as opposed to the original version of ISO 8632 should specifically refer to Addendum 1.
4. While all parts reflect a basically uniform direction, there is much inconsistency between them (for example, the encodings of SEGMENT TRANSFORMATION in parts 2, 3, 4 are all different). This should be corrected in the text submitted for DP registration.
5. The need for METAFILE CATEGORY is not made clear. Its meaning and relationship to other elements such as METAFILE ELEMENT LIST should be clarified. Some of the functionality should be sorted out as well — e.g., the elements allowable in a category can be handled by the METAFILE ELEMENT LIST. It should be made clear what can vary across categories — element lists? grammar? parameter ranges? semantics? Material is needed in the document explaining what does vary with category and how — it is only in the annexes now. (See the U.S. comments on issue CGMA1, submitted to WG2 in document X3H3/87-48.)

6. The important task of specifying which elements are required for particular categories of metafiles, which are prohibited, which are allowed, and in which order they are constrained to appear, must not be left to an annex which is not part of the standard (and it definitely should not be buried in the formal grammar, which tends to make many persons' eyes glaze over when they try to read it). The exact organization and contents of metafiles tailored to perform as GKSMs of various types should be detailed as early in the document as possible; it would appear to fit nicely as a new subsection of 4.10 where the "Conceptual State Diagram" is presented.

7. Even though we have always claimed that the CGM standardizes the metafile, not the interpreter, it was relatively straightforward for an implementor to read IS 8632 and determine what was expected (if not actually required) of his product. With the introduction of more than one type of metafile, this is more difficult to determine.

   The document would benefit from a discussion of CGM interpreters vs. CGEM interpreters, preferably accompanying the specification of the different types of metafiles proposed for Clause 4 (Concepts), but at least in Annex D if nowhere else. The present D.5 should be expanded to include capability lists for CGM interpreters (what is currently in IS 8632), as well as for the different GKS relationships; a chart similar to the CGI's new Constituency Profiles (in DP9636) might work well here.

8. part 1, p.1 — a clear definition of "session capture" is lacking. More material is needed in clauses 0.5 and clause 1, paragraph 3.

9. part 1, p.1 — a) "This picture description"...we should be talking about a file format, not a picture description; b) "includes...session capture requirements". How can "requirements" be included in a picture description or file format?

10. part 1, p.1, last line — change "elements" to "blocks of elements".

11. part 1, clause 3 — glossary entries are needed for picture, session, session capture, dynamic picture regeneration.

12. part 1, p.2, 4.2 — we infer from this that segments must be contained wholly within a picture body. Some more discussion and clarification is needed to show how segments fit in with the other delimited parts of the metafile (metafile descriptor, picture descriptor, picture body).

13. part 1, p.2, 4.3 — Last sentence — replace through ";" with "There is a single set of defaults applicable to all categories, specified in Clause 6 of this part;"

14. part 1, p.3, 4.3.2.3 — Change "conforming to" to "necessary to support".

15. part 1, p.3, 4.3.2.3 — BEGIN/END SEGMENT do not belong in the GKSM0 set. Nor do CHARACTER SET LIST or CHARACTER CODING ANNOUNCER. We presume this is editorial oversight.

16. part 1, 4.3.2.3 — state correct GKS reference to "GKS IS 7942 Level 1a".

17. part 1, p.4, 4.3.2.4 — Revise to refer to GKSM0 and just list the additional elements.

18. part 1, p.4, 4.3.2.4 — CHARACTER SET LIST and CHARACTER CODING ANNOUNCER do not belong in the GKSM set.

19. part 1, p.3, 4.3.2.3 and p. 4, 4.3.2.4 — FONT LIST may need to be deleted from the GKSM and GKSM0 sets as well, but this hinges on the resolution of ANSI issue 11.

20. part 1, p.5 — 3rd line should read DEFERRAL STATE.

21. part 1, p.6 — second item, there is confusion between deferral state and deferral mode. Check usage throughout for correctness.

22. part 1, p.9 — segments may also be renamed.

23. part 1, p.9 — usual CGM style for a section titled "xxx Elements" is to talk about the elements. This section does little of that.

24. part 1, p.9 — why are BEGIN SEGMENT and END SEGMENT not mentioned in this section?

25. part 1, p.10 — there are now four shorthand names.

26. part 1, p.11 — The element descriptions (5.2.6, 5.2.7) should include a statement of where the elements may appear.

27. part 1, p.11, 5.3.16 — the descriptive style is GKS-like. For CGM, "..informs the metafile interpreter.." would be better.

28. part 1, p.11, 5.3.16 — 'cgm' is the usage here, whereas 'basic cgm' is used elsewhere.

29. part 1, figure 12 (Conceptual State Diagram) is out of date. It needs to include segments.

30. part 1, p.11 — the meaning of CGMEXT1 is never explained.

31. part 1, p.13 — why are the expected actions of CLEAR and UPDATE not specified?

32. part 1, p.15, 5.7.9 — What happens if the ASFs are mixed? This is mainly a problem of explanation.

33. part 1, p.16 — first item, "second paragraph" should be "second sentence".

34. part 1, p.16 — references to "structure" should be to "data structure" instead, to avoid confusion with PHIGS structures.

35. part 1, p.16, 5.7.38 — The last paragraph is inappropriate to a metafile standard and should be deleted.

36. part 1, p.22 (5.10.5) and p.23 (5.10.8) — Picking is not done in the metafile, to a metafile, or by a metafile; it is a function of the software using the metafile. All references to picking and pickability should therefore be removed from the metafile, as the subject is treated completely in GKS (and in any event, is outside the scope of the metafile standard, as is any discussion of the use to which metafile elements may be put). For example, the NOTE in 5.10.5 is inappropriate to a metafile standard and should be deleted.

37. part 1, p.23, Clause 6 — Delete RENAME SEGMENT through REDRAW ALL SEGMENTS.

38. part 1, p.24, — the description of SEGMENT TRANSFORM is appropriate for a procedural standard, not a file format standard. More generally, this comment applies to all of the text taken from the GKS standard. Rephrasing for a file format standard is needed throughout.

39. part 1, clause 6 — all functions whose default is listed as "n/a" should simply be deleted. The concept of a default is not applicable to an element such as RENAME SEGMENT any more than it is to a primitive such as POLYLINE, which you will notice is not included in this list in IS 8632.

40. part 1, clause 7 — text needs to be added requiring that the elements in a metafile correspond to the category.

41. part 1, Annex E — Annex E will need much more added to it, as the relationship of GKS to CGM is different from the relationship of GKS to metafiles corresponding to 'GKSM', 'GKSM0', or 'CGMEXT1'.

42. The brain boggles at the proliferation of formal specification included in this draft. We see no reason for both Annex F and Annex G.

    The formal specification of the new functions should be added to the appropriate places in Annex A, just as their functional specifications are to be added to the appropriate places in Clause 5. Redundancy is dangerous, as it creates the possibility of conflicting versions in different places, not to mention leading to a waste of trees!

    In integrating the formal grammar into Annex A, try inserting a new root production:

    <generalized metafile> ::= <metafile>
    | <gksm>
    | <gksm0>
    | <extended cgm>

    where <metafile> is the root production of IS 8632's Annex A.

43. part 1, annex A — add a note at the beginning stating that this is the grammar for category CGM.

44. part 1, p.26 — move <identifier>::=<string> to follow <metafile> on p.25.

45. part 1, p.26 — move <metafile category> after <metafile description>.

46. part 1, p.26 — the fourth line of <identification> is indented too far, so that it looks like a parameter of the previous. See also annex G.

47. part 1, p.26 — 'gksm0' is omitted from <category enumerated>.

48. part 1, p.31 — inconsistency, "primitive attribute elements" or "attribute elements"?

49. part 1, p.31 — include <pick identifier> in the "Attribute Elements" section.

50. part 1, p.35 — <identifier> is duplicated, but this time defined as <integer>.

51. part 1, p.36 — "<>" is missing from the last production of this section. See also annex G.

52. part 1, p.37 — "colour" and "precision", abbreviated as "col" and "prec", are fully spelled out in the description of the formal grammar.

53. part 1, p.38 — move <VISIBLE> and <INVISIBLE> after <EXTENDED 8_BIT> and insert <CLOSE_VISIBLE> and <CLOSE_INVISIBLE>.

54. part 1, p.39 — add <COLOUR VALUE EXTENT> after <MAXIMUM COLOUR INDEX>.

55. part 1, p.43 — <category> can't be optional for the grammar of a GKSM metafile, otherwise it is not possible to announce that this is GKSM.

56. part 1, throughout — In text lifted from CGI and GKS the word "function" occurs frequently. Change to "element".

57. part 1, p.6, 4.3.4 — Reword: "VDC NORMALIZATION defines an isotropic change of coordinates. It can be used, for example, to define the correspondence between a sub-range of the metafile's VDC range and the normalized coordinate space of a recipient graphics system." Repeat this for p.11, 5.3.17.

58. part 1, p.6, 4.5.3, 2nd paragraph — "deferred" should be "deferral".

59. part 1, p.6, 4.5.3, BNIG — add "on the interpreting system" after "device".

60. part 1, p.7, top of page — add "In table 1, 'may not be bundled' column, add CHARACTER VECTORS at the end".

61. part 1, p.9, 4.12, 2nd paragraph — Change "functions stored inside" to "elements within".

62. part 1, p.17, 5.7.39 — The paragraph for "colour" is missing. See the other representation setting elements for missing text.

63. part 1, p.24, E.7 — Change "will be" to "is".

64. part 1, p.25, F.3.1, 4th line — There should be a plus, "+", after <metafile contents>.

65. part 1, p.26, F.3.2 — Add GKSM0 to <category enumerated>.

66. part 1, p.26, F.3.2 — "*" should be "+" after <element name>.

67. part 1, p.36, F.3.9 — <name> is nowhere. It is <integer>, not a terminal.

68. part 1, p.36, F.3.9 — <transformation matrix> is <real>(4) <vdc>(2).

Comments on Part 2

69. part 2, p.2, 8.2.16 — Add GKSM0.

70. part 2, p.2, 8.2.17 — Typo in "V<VDC>".

71. part 2, p.1, beginning — Datatype "Device Point" is not defined anywhere.

72. part 2, p.4, 8.6.40 — The comments, {..}, are for line, not marker.

73. part 2, p.6, 8.9.5 — "visibility" should be "visible".

74. part 2, p.6, 8.9.7 — Is this SEGMENT PRIORITY or SEGMENT DISPLAY PRIORITY?

Comments on Part 3

75. part 3, p.all — All references to FP and FPR are incorrect. The references should be to R and RR.

76. part 3, p.1, beginning — Table 1 needs to be updated to define DP, Device Point.

77. part 3, p.2, code 7 — The entire discussion is inconsistent with Part 1, clauses 5 and 6.

78. part 3, p.3, table — The 'length' and 'range' columns are wrong for all of the representation elements.

79. part 3, p.5, code 4 — It is incorrect that the components of the 2x2 submatrix can be called scale or rotate. All 4 reflect rotation. Also, the order is wrong and should be made to match part 2.

Comments on Part 4

80. part 4, p.1, 5.3.5 — "COORDINATE" in DP definition should be "POINT".

81. part 4, p.1, 5.3.5 — The order of entries in TM definition is wrong and should be made to match part 2. Also "row major" should be "column major".

82. part 4, p.1, 5.4.4 and 5.4.5 are inconsistent — "DEFERRAL" is shown abbreviated as "DEFER", so DEFERRAL STATE should be "DEFERSTATE", not "DEFSTATE" as shown.

83. part 4, p. 1 — for the Clear Text Encoding of the new elements' names, we suggest abbreviating a few more of the words:

DEVICE	→	DEV
HIGHLIGHT	→	HIGHLT or HILT
VIEWPORT	→	VIEWPT

The "List of Approved Abbreviations for Bindings" (TC97/SC21/WG2 N349) shows that no abbreviation has yet been assigned for "device" or "viewport". The abbreviation for "highlighting" is "highlight", which is a grammatical mapping rather than an abbreviation per se.

84. part 4, p.2, 6.3 — Add 'GKSM0' to match Part 1 (the list is incomplete). Also, use the "|" character to indicate that only one of the types is specified in the element.

85. part 4, p.2, 6.5 — The verb "SET" mysteriously crept into the encoding of DEFERRAL STATE; it should be deleted. (Note: CGM does not use "SET" for any of its attributes or controls, as the CGM contains elements rather than functions.)

86. part 4, p.3, 6.7 — What is {=/0}?

87. part 4, p. 5, 6.10 — There is an extraneous "<" before the production of SEGMENT PRIORITY.

APPENDIX 3

RESULTS OF ISSUE PROCESSING AT TULSA

Table 1: Results of Vote on ISO Issues

ISO Issue Number	Alternatives 1	2	3	No Preference	New Alt.
1	17	6	21	10	1
2	45	2	-	6	2
3	11	22	(*)	18	4
4	22	26	-	7	0
5	20	(18	*)	13	4
6	-	-	-	46	9
7	40	4	-	10	1
8	3	33	-	15	4
9	21	17	-	13	4
10	24	13	-	17	1
11	27	4	-	24	0
12	(19)	30	-	5	1
13	7	37	-	10	1
14	40	3	-	10	2
15	27	(14	*)	12	2
16	(13)	27	-	14	1
17	44	3	-	8	0
18	4	26	(11)	14	0
19	6	30	-	16	3
20	33	4	-	14	4
21	26	11	-	16	2
22	1	6	20	25	3
23	25	9	-	17	4

U.S. position on ISSUES:

1) Majority position of advisory ballot unless circled, in which case the circled alternative is the endorsement of X3H3, as per decisions at Tulsa X3H3 meeting.

2) (*); means new alternative selected.

3) (n m); means one of the two alternatives without strong preference.

Table 2: Results of Vote on ANSI Issues

ANSI Issue Number	Alternatives 1	2	3	4	No Preference	New Alt.
1	7	12	24	-	12	0
2	40	1	-	-	14	0
3	2	40	1	-	11	1
4	8	36	-	-	10	1
5	4	37	1	-	12	1
6	20	3	(15)	-	16	1
7	28	12	-	-	14	1
8	19	12	4	-	19	1
9	33	2	-	-	19	1
10	41	7	-	-	6	1
11	22	8	-	-	21	4
12	36	8	2	-	9	0
13	6	41	-	-	8	0
14	3	40	-	-	12	0
15	37	6	-	-	12	0
16	17	11	5	-	21	1
17	8	25	9	-	13	0
18	29	8	-	-	17	1
19	29	3	0	-	22	1
20	0	24	0	14	15	2
21	(10)	1	21	6	15	2
22	7	33	-	-	14	1
23	35	7	-	-	13	0
24	35	2	-	-	18	0
25	5	33	-	-	17	0
26	2	38	-	-	15	0
27	2	34	-	-	19	0
28	7	4	23	-	21	0

APPENDIX 4

MINUTES OF THE VALBONNE MEETING

Minutes of CGEM Rapporteurs Meeting
Sophia-Antipolis
18 May 1987, Afternoon
Attending: E. Moeller (Rapporteur), A. Mumford (UK), B. Trocherie (France)
F. Dawson (US), L. Henderson (US), P. Egloff (Germany),
H.G. Bufka (Germany), A. Moltedo (Italy)

E. Moeller opened the meeting and welcomed delegates. Thank were given to Anne Mumford for a fine job on the draft Addendem 1 and the preliminary issues log.

The Egham minutes were approved unanimously without change.

The following documents, distributed since Egham, were reviewed:

1. Working draft CGM Addendum number 1 -- N1403;
2. CGM Addendum Initial Issues Log -- N529;
3. Minutes of the Egham meeting -- N534;
4. US Comments on N1403;
5. Comments on N1403 from France, UK, Austria -- N558;
6. CGM Addendum #2 proposal from UK -- N531.

New documents were given temporary numbers:

1. SA1 -- summary and classification of N1403 comments and issues;
2. SA2 -- AFNOR comments on N1403;
3. SA3 -- Guidelines for Implementing GKS Segments with CGI (US);
4. SA4 -- 3D Graphics Metafiles, E.Moeller;
5. SA5 -- PHIGS/GKS 3D metafile, PHI-GKS-M.

Eckhardt called for a volunteer to be issues librarian. No one volunteered yet. Mumford thinks it possible that she may be able to continue as document editor for the 3D addendum, but this is uncertain at this point.

Liaison meetings with other Rapporteur groups were announced -- CGI for Tuesday morning, 19 May; 3D for Friday morning, 22 May.

Technical discussions began with a discussion, initiated by Dawson, of perceived critical needs in CGM for drawing control functions and other functions to support constituents in design and engineering and in mixed document applications. It was pointed out that our scope was limited on Addendum 1; Mumford reported a UK position that we should not keep spawning new work (items), but stick with our scope and finish that work. Dawson and Henderson pointed out that potential constituencies of CGM will be lost if their needs are not addressed.

The group decided to: take the issue back to national delegations for guidance; and discuss the topic later in the meeting and come up with a Rap group position on what our role should be and how we should proceed.

Mumford pointed out that UK and France had both identified the "item types" issue as an important and asked if it shouldn't be put on the agenda. It was pointed out that it will have to be discussed because it is an open issue.

Discussion turned to ANSI.1 in the issues: variable or fixed semantics for elements. US feels semantics should be uniquely defined. UK feels alternative semantics should depend on category. Borufka – category is an indication that a particular subset is to be used, in some limited way. Major problem with sematic-free metafile is in global grammar. Some discussion of over-use of opcode space – not problem in binary but could be in character. It was remarked that 'category' isn't needed at all if don't vary semantics.

Straw vote on alternatives:

Alt 1 – CGEM is just syntactic framework – 0
Alt 2 – semantics depend on category – 4
Alt 3 – semantics uniquely defined – 1

abstaining, 1; missing, 1.

Discussion turned to whether our scope is limited to just GKSM at this point. US feels additional stable CGI functionality should be included. UK questions what defines "stable".

Discussion of ANSI.26: must there be a 1-to-1 mapping GKS to CGEM or is "close support" adequate. There seemed to be a general feeling in the group that "close support" was adequate.

Discussion of BSI 2.1: should CGEM allow for nested segment structure, and should BEGIN/END SEGMENT be delimiter elements or some other class. UK was concerned about implications by being Delimiter Elements. There was a weak consensus to leave things as they are.

Discussion of ANSI.6: there seemed to be much interest in a symbol library facility. SOme discussion of whether the segments should be in the MD or in some other block, whether they had immediate visibility for all pictures, etc. It was agreed that a tentative technical proposal should be completed, the group should look for technical problems, and the impact on the schedule should be examined (delaying GKSM would be bad).

Discussion of problems with anisotropic COPY transformation: it causes difficulty with the bundled attributes of 'absolute' linewidth and such. Moeller explained the problem. It was agreed that either the COPY function had to be included or transformation as a primitive attribute must be included. The problem was stacked for later discussion.

It was announced that Reference model experts, and possibly 3D experts, might join the Tuesday CGI liaison meeting.

The meeting was adjourned for the day.

Minutes of CGEM Rapporteurs Meeting
Sophia-Antipolis
19 May 1987
Attending: E. Moeller (Rapporteur), A. Mumford (UK), B. Trocherie (France)
F. Dawson (US), L. Henderson (US), P. Egloff (Germany),
H.G. Beruſka (Germany), A. Moltedo (Italy) and CGI Rapporteur Group Attendees

9:20a - 10:30a Liaison meeting with CGI Rapporteur Group

The meeting began with a discussion of a reference model question, "where does the CGM exist in relation to the CGI?". It was noted that previously the CGM existed below the CGI. The U.S. expressed its position that the reference model is still valid. In addition, the CGI could be used to produce either an arbitrary CGM or a limited CGM.

The discussion continued concerning the mechanism for creating a CGM from a CGI client service. One view was that each CGM element could be created from a corresponding CGI function. Another view was that the CGI functions may map to a series of CGM elements.

The U.S. returned the discussion to the reference model by remarking that the CGM may be at the level of the CGI, but the CGEM (i.e., GKSM) is below the level of the application programming interface (API) standards, acting as an audit trail. This brings up a question of whether the CGEM and the CGM are intended to operate at two different levels in the reference model. While this view might present certain problems, utilizing the functionality of the CGI to support the CGEM would facilitate the dual role. DIN expressed the view that the CGEM functionality should be as close as possible to the CGI. However, this desire should not compromise the ability of information available by the GKSM generator from being passed on to the GKSM interpreter. A BSI delegate indicated that while one may want CGEM to be interpreted below the API, at the same time the application program may like to interpret individual elements in the metafile. A solution to the two methods of interpretation of a CGEM may be needed.

The U.S. wanted to determine whether there was concensus that not every element of the CGM need be interpreted by GKS. No vote on concensus was taked. However, there appeared to be noone willing to voice disagreement.

The view that the GKS audit trail metafile is at the level of the workstation interface was voiced by several individuals. It was stated that the audit trail is a mapping of the dialogue to an individual workstation. The U.S. commented that this view is added argument for the mapping of the CGEM functions to CGI functionality rather than attempting a 1:1 mapping of the CGEM to GKS API functionality.

The U.S. expressed the opinion that two issues need resolution prior to progression of the CGEM. First, it must be determined what the GKSM audit trail ought to be and what must be done to support this in the CGEM. Second, it must be determined how much more CGI functionality than that needed to GKSM should be added to the CGEM. In support of the first issue, the U.S. offered reference to a paper on GKS segment model mapping to the CGI segment model.

The meeting fell into a digression on the question of whether the CGEM should explicitly specify associated semantics of the individual elements or whether the category of the CGEM would be used to determine the semantics of the individual elements.

The BSI expressed the view that there are components of the GKS standard that are incorrect

and should be identified by the CGEM effort and transmitted to the GKS Maintenance work effort.

The whole question of the charter of work for the CGEM and GKS Maintenance work efforts was discussed. Peter Bono indicated that a Head of Delegation (HOD) meeting, scheduled for the afternoon of 20 May, was planned for discussion of these items. He expressed the opinion that it was up to the CGI and CGEM Rapporteur Groups to define a draft statement of work or else the HOD would be forced to define it. In addition, the limits to the work effort need to be identified.

The U.S. brought to the attention of the group the additional needs for technical illustration and publishing and the danger of ignoring the constituencies with these requirements. Ignoring the needs would force these constituencies to make use of proprietary page description languages (PDL's) or product data exchange standards (i.e., IGES, PDES, or STEP).

The meeting briefly discussed the need for explicit definition of GKS Item Types such that CGEM elements could be created with unambiguous meaning so that a GKS application could interpret the CGEM. It was mentioned that this may require the GKSM interpreter to do, "a little extra amount of work" when it is reading a CGEM.

10:30a - 11:05a Coffee Break

11:05a - 1:00p Continuation of liaison meeting with CGI Rapporteur Group

A lengthly discussion was held on the differences between GKS and the CGM or CGI. The intext of the discussion was to determine whether a GKSM could actually be generated and interpreted using the current CGI functionality. The following differences were noted.

1. CGM font index and precision versus GKSM text font and precision
2. CGM pattern and hatch index versus GKSM interior style index
3. CGM character height and orientation versus GKSM character width and height vectors
4. CGI prepare viewsurface versus GKS clear workstation
5. CGI redraw all segments versus GKS redraw all segments
6. CGI deferral mode and segment regeneration mode versus GKS set deferral state
7. CGI pick priority and segment display priority versus GKS set segment priority
8. CGI segment model versus GKS segment model
9. CGI states versus CGEM states

A series of straw votes were taken to determine concensus of the participants.

- Are the positive text indices sufficient for GKSM support?
    1. Yes
    2. No

    Vote (18-0,2)

- Are separate font index and font precision elements sufficent for GKSM support?
    1. Yes
    2. No

    Vote (11-6,3)

- Are the CGM mechanisms sufficient for GKSM support?
    1. Yes
    2. No, Add index

    Vote (19-0,3)

In the CGM the character height is specified separately than the character orientation (i.e., character up and base vectors). This permits append text to change the height without changing the orientation. GKSM specifies character height and width vector. If the CGI functionality were to be used for GKSM support, then the character height and the character orientation elements would have to be mandated as being output together.

- Should the CGM mechanism for character height and character orientation be used to support the GKSM?
    1. Yes, But through the proposal in the preceeding paragraph
    2. No, Add separate elements

    Vote (18-1,3)

A lengthly discussion followed that attempted to resolve the mapping from a GKSM function to CGI functionality. This work was finally decided to be left to a separate working group.

- If it is determined that to use CGEM as a GKSM that additional functions are needed; will CGI add these functions?
    1. Yes, Identically
    2. Yes, An explicit mapping
    3. No

    Vote (1-18-1,2)

- Once, at this meeting, the CGI Rapporteur Group identifies stable functionality, should the CGEM Rapporteur Group endeavorto use this capability?
    1. No
    2. Yes, For GKSM support only
    3. Yes, In general

    Vote (11-10,1)

The U.S. stated its concern that the work on the Addendum 1 for CGEM support should not prevent or restrict further enhancement to the CGM.

Most attendees voiced concern that attempts to add additional functionality to the CGEM

should not adversely effect the time table for the CGEM. It was requested that the draft of the CGEM sent for letter ballot include a comment soliciting member body suggestions of stable CGI functions that should be added to Addendum 1. Some members indicated that additional functionality should be prerequisite on being directed at some agreed upon scope for the added functionality.

Before adjourning the liaison meeting, it was decided to meet again on Thursday morning, at 11:00a, after the morning coffee break.

1:00p - 2:00p Lunch Break

2:00p - 5:30p CGEM Rapporteur Group Meeting

The open issues summarised in temporary document SA1 were addressed with an attempt to resolve those that did not appear to be controversial.

ANSI.1: SHOULD SEMANTICS OF ALL ELEMENTS IN THE ADDENDUM BE UNAMBIGROUSLY DEFINED?

This issue remained unresolved, even after much more discussion.

E. Moeller stated that he believed the CGI Rapporteur Group had the responsibility to prove whether the CGI could support GKSM. In addition, the rapporteur group must prove that the mapping back to GKS from the CGEM did not lose any information.

H.G. Berufka stated that he believed that the CGI segment model could be used in the CGEM. E. Moeller was concerned that the redrafting of the CGI segment model was not commonly known by attendees and it would be difficult to discuss this further. H.G. Berufka brought in a member of the CGI Rapporteur Group to explain the new segment model of CGI. The temporary document SA8 was used as a support document for the discussion.

The group spent a considerable amount of time discussing where the CGEM would reside in the CGI segment pipeline. It was decided that the CGEM would have to reside to the left of the point in the pipeline where bundled attributes were associated. Probably, the location would be the caption "primitives" appears in the picture on page 11 of SA8.

CGMA1.9: IS THE TERM 'SEGMENT' THE RIGHT ONE?

The group resolved that it was the right term.

ANSI.4: WHAT SEGMENTATION MODEL SHOULD BE IN CGM ADDENDA?

The group resolved that the CGI model was appropriate, but not all of the functions might be included in the Addendum 1 work.

BSI.2.1: SHOULD THE CGM ADDENDA ALLOW FOR NESTED SEGMENT STRUCTURE?

The statement of this issue was did not reflect the actual issue. The group resolved that the Begin_Segment should be left as a delimiter element.

ANSI.5: WHAT ELEMENTS MAY BE INCLUDED IN SEGMENTS?

The group resolved that most picture elements, with the exception to be determined, can occur in segments. This has no implication for contents of segment store for the interpreter of a CGEM. In category "GKSM", the "clear" function can not appear in the segment.

BSI.2.1 (sic): SHOULD THE MEANING OF ELEMENTS WITHIN SEGMENTS BE DEFINED BY CATEGORY?

Resolution of this issue will be based on whether the group is to adopt CGI semantics and the possible outcome on ANSI.1 and ANSI.4.

CGMA1.20: WHERE SHOULD THE SEGMENT ELEMENTS APPEAR?

The wording of this issue was clarified to mean where in the CGEM document should the various segment elements appear. The group resolved that the segment elements should appear where they are currently located, in the segment section.

CGMA1.2: HOW ARE SEGMENTS STORED?

The group felt that the issue was more appropriately described by the next issue.

ANSI.6: IS THE "DEFINEDNESS" OF A SEGMENT LIMITED TO THE CURRENT PICTURE?

There was general agreement that a "symbol library" capability is needed. Technical work will be addressed later this week.

CGMA1.1: SHOULD THERE BE A CATEGORY "GKSMO" AS WELL AS "GKSM" (SEE ALSO ADDITIONS TO ISSUE CGMA1 BY ANSI)?

The group resolved that alternative 3, "No, but the GKSMO set is defined as one of the SHORTHAND enumeratives for Metafile_Element_List" should be selected since the grammars are the same or at least a set:subset relationship. Vote (1-0-4,2).

BSI.3.3: SHOULD ANNEX A OF IS 8632 BE ADDED TO ALLOW FOR THE METAFILE CATEGORY ELEMENT?

The group resolved that Metafile_Category element is not a valid element for IS 8632.

BSI.3.5: SHOULD THE RELATIONSHIP BETWEEN CATEGORIES BE DESCRIBED?

Discussion on this issue was postponed.

ANSI.7: ARE SUCH ELEMENTS AS UPDATE, SET DEFERRAL STATE APPROPRIATE IN A METAFILE STANDARD?

The group resolved that they were, and the functions should be retained.

ANSI.8: SHOULD INTERACTIVE VALUES OF THE "DEFERRAL MODE" PARAMETER OF DEFERRAL STATE BE IN THE METAFILE?

The group resolved that they were, and the values should be retained.

BSI.2.3: SHOULD VARIOUS STATEMENTS RELATING TO WORKSTATION BEHAVIOR BE MOVED TO AN ANNEX CONCERNED WITH INTERPRETATION EFFECTS?

The group resolved that these statements should be moved to an annex.

ANSI.2: SHOULD THE CONTENTS OF THE "DRAWING SET" (SHORTHAND) OF METAFILE ELEMENT LIST BE CHANGED BY THE ADDENDUM?

The group resolved that it would not support to change the content.

ANSI.12: SHOULD CGEM COORDINATE WITH CGI ON OPCODES?

The group resolved that any given opcode or function name has exactly the same parameterization in all three standards, but that coordination was needed with the CGI Rapporteur Group and appropriate standards bodies outside of SC 21/WG 2.

ANSI.24: SHOULD A VALUE OF TWO (2) BE USED FOR THE METAFILE VERSION ELEMENT IN METAFILES THAT CONTAIN ITEMS FROM, OR FOLLOW GRAMMARS OF, CGM ADDENDUM 1?

The group agreed that the CGEM would have a version of 2.

The meeting was ajourned.

Minutes of CGEM Rapporteurs Meeting
Sophia-Antipolis
20 May 1987, Afternoon
Attending: E. Moeller (Rapporteur), A. Mumford (UK), B. Trocherie (France)
F. Dawson (US), L. Henderson (US), P. Egloff (Germany),
H.G. Borufka (Germany), L. Moltedo (Italy am only)
P. Bono (US pm only), Satoru Kawai (Japan)

Moeller noted the need to find an issues librarian.

Mumford re-affirmed the possibility that she would be interested in editing the 3d metafile addendum

The priority for GKS support was noted.

There is pressure on many national groups to extend the primitives attributes etc in the graphics standards. A major problem is the slow rate on standard development.

There was interest from clients of the graphics standards for more support for office documents and for raster. Ansi had reviewed the work of SC18 who are developing SPDL. Any future work on extending primitives and attributes would require some liaison with SC18. There was interest in the group in working on further 2D functionality rather than 3D extensions. The BSI are likely to be more interested in a GKS-3D metafile. This was in contradiction to most other views.

It was agreed that the first addendum should procede with highest priority and that this might include CGI functionality when functions were deemed to be stable by the CGI group as the work developed.

The group then turned to a discussion of the issues.

ITEM TYPES (BSI2.6)

These are used to support the functional standards. There is a problem with the algorithm based on the binary op-codes which was proposed at Egham because there is unlikely to be a 1-1 mapping between the elements of the CGM and the functional standards. It was agreed that the GKS item types are logical data items and that these can be mapped to physical elements in the metafile. The agreed solution was that the functional standards should provide the logical item types and that the metafile group should then provide a mapping to the CGEM elements. If the GKS Annex E item types are sufficient these could be the ones used for GKS. This might help implementors who have used GKS Annex E already. There may be some benefit in having items defined for the elements in an encoding independent way but it was agreed that this would be an implementation decision.

MAPPING OF THE GKS FUNCTIONS WITH THE CGI FUNCTIONS

1. Text Font and Precision

The discussion centred round the use of the CGM elements TEXT FONT INDEX, TEXT PRECISION and FONT LIST to represent the TEXT FONT AND PRECISION of GKS.

The problem is that the CGM uses positive indices indicating the font required from the font list. In GKS positive values indicate registered fonts (currently only 1) and negative values are private fonts. One suggestion is to map the GKS negative values to positive indices by listing them in

the FONT LIST e.g. "GKS-1".

The CGM FONT LIST is a metafile descriptor and therefore has to appear at the head of the metafile. In GKS we do not know which fonts are needed until they are requested. It is not possible to dump all known fonts at the start of the metafile as the GKS application may require more fonts than those on the generating system. One alternative is to carry out a double pass on the metafile and to record those fonts used at the end of the GKS session. This was considered to be unacceptable.

There is a requirement for GKS to update the FONT LIST within the picture body. It is possible that CGI may also need such a function. The suggestion to
be made to the CGI group is to add a new MODIFY FONT LIST element which can then be used in the picture body of the CGEM to add to the FONT LIST (default or explicitly set).

There may also be a requirement for a MODIFY CHARACTER SET LIST.

2. Pattern and Hatch

The CGM elements are PATTERN INDEX and HATCH INDEX. These need to be mapped in both directions if possible to the GKS function SET FILL AREA STYLE INDEX.

The mapping was agreed as follows:

GKS FILL AREA STYLE INDEX maps to CGEM PATTERN INDEX and HATCH INDEX where the index in GKS has a value greater than zero. Where the value is less than zero then only HATCH PATTERN is written as anything else would be illegal.

The reverse mapping is that the appearance of either of the CGEM elements causes a SET FILL AREA STYLE INDEX.

3. Character Vectors

In the initial draft a new element CHARACTER VECTORS was added to the list of elements as GKS Annex E stores the height and base vectors as a pair.

A mapping to the existing CGM elements for height and orientation was considered to be beneficial. It was agreed that the GKSM category should write the CHARACTER HEIGHT and CHARACTER ORIENTATION as a pair in the CGEM in that order. On interpretation this would return the character vector information back to the GKS application.

(CH,CO) maps to CH*Char Up Vect, CH*Char Base Vect

$$\frac{}{|\text{Char Up Vect}|} \quad \frac{}{|\text{Char Up Vect}|}$$

4. Clear Workstation

A mapping was agreed as follows:

CLEAR WORKSTATION in GKS will map the new CGEM elements (taken from the CGI):

MAKE PICTURE CURRENT; PREPARE VIEW SURFACE; DELETE ALL SEGMENTS.

5. Redraw All Segments

This will map to the new elements: MAKE PICTURE CURRENT; PREPARE VIEW SURFACE (conditional clear); REDRAW ALL SEGS.

CGI ISSUE: why does PREPARE VIEW SURFACE only do a conditional clear for hardcopy not all devices?

6. Update

The group considered an MO to be IMM.

A problem was found with this as the mapping needs to be different for whether the Update flag is perform or postpone.

For perform the mapping could be: MAKE PICTURE CURRENT; PREPARE VIEW SURFACE; REDRAW ALL SEGS.

For postpone the mapping might be just MAKE PICTURE CURRENT.

It was noted that it was not possible to distinguish between REDRAW ALL SEGS and UPDATE with perform. The reverse mapping causes problems because in GKS the behaviour of the two functions is different. This is because of the different behaviour dependent on the flag 'new frame necessary at update'. For MO this is set to NO.

We could map to the effect of update with perform but this is not a true audit then. One possibility is a flag on REDRAW ALL SEGS. This possibility and the nature of the problem was presented to CGI.

7. Set Deferral State

This will be mapped to new elements taken from the CGI. The mapping will be to DEFERRAL MODE; IMPLICIT SEGMENT REGENERATION MODE. If the regen mode is allowed then REDRAW ALL SEGS (conditionally). There is a need to map BNIG and BNIL to BNI when generating the metafile and the reverse mapping should be to BNIG.

During the afternoon the Rapporteur attended the HOD/RAP meeting. he reported that the Addendum for GKSM was considered to be a high priority in WG2. Most of our efforts are to be given to this project. Beyond that all other work has a low priority. The HOD/RAP group agreed that the need for extra functionality as addressed by the group should be given considreation at some time and one possibility is to ask member bodies to start drawing together a wish list. this will be done at the WG2 level at this time. The timescale for the first addendum looked to be similar to CGI

Minutes of the CGEM Rapporteurs Meeting

Sophia Antipolis
21 May 1987

Attending: E. Moeller (Rapporteur), A. Mumford (UK), B. Trocherie (France)
F. Dawson (US), L. Henderson (US), H.G. Borufka (Germany),
P. Egloff (Germany), Kawai Satoru (Japan)

The meeting began with a discussion of the pick priority data type. Three alternatives were identified:

1. integer (as it is in CGI)

2. integer on the range of values (lower limit, upper limit)

3. real (as it is in GKS)

The result of the discussion was to define a new function PICK AND DISPLAY PRIORITY EXTENT, which sets the lower and upper limit (alternative 2) and adds it to the CGEM functionality as a metafile description element.

The discussion continued with the new segmentation model of CGI, which was introduced by a paper of D. Vanderschel. The group was of the same opinion as the day before that this model fulfills the needs of GKS.

After a short discussion of the guidance given by the HOD and rapporteurs group concerning the segment problem with respect to the possibility of a location of segments outside pictures the group agreed to discuss those problems if enough time was left.

The discussion returned to the segment model and the mapping of related GKS functions to CGI functions. In particular the GKS functions INSERT_SEGMENT and ASSOCIATE_SEGMENT were investigated. Karla Vechiet of CGI group helped to diminish some confusion whether or not the functions COPY_S, REOPEN_S and INHERITANCE_FILTER are needed to support the GKS functionality. The CGEM group agreed that these three functions are not necessary for the GKS support and therefore are not part of the category GKSM, but they were added to the CGEM.

In connection with the INHERITANCE_FILTER the discussion returned to the reference model problem of CGEM and CGI. Especially the relation of the GKS workstation independence segment storage (WISS) to the CGI was adressed. The CGEM group had the general feeling that the WISS conceptually is situated outside (that means above) the CGI.

After a coffee break two points were adressed:

- The laison meeting with the 3D group scheduled for Friday should be a more general one on an informal basis.

- Due to the working draft status of Addendum 1 of CGM it is not necessary to produce a response document containing the answers to the discussed issues.

Then the CGEM group's discussion focussed on the issues list. The issues listed below were briefly discussed under the aspect of an unambiguously defined semantics and reduction of redundancy and resolved unanimously (otherwise marked):

* CGMA 1.12: Should TEXT FONT and PRECISION be added as an element?
Alternative 2: No

* CGMA 1.13: Should there be an element for HATCH and PATTERN INDEX?
Alternative 2: No

* CGMA 1.14: Should there be a WORKSTATION WINDOW element?
Alternative 1: Workstation window is mapped to VDC extent.

* ANSI.10 : Is there a need for the redundant specification of certain text attributes - ....?
The first part of this issue is solved by alternative 1: No, eliminate the redundant elements. The second part was discussed in CGMA 1.12 and is solved.

* ANSI.11 : Does the meaning of TEXT FONT INDEX change in the addendum?
Alternative 1: No
In addition, the new function MODIFY_FONT_LIST was invented as an attribute element with index and font name as parameters. This function must ensure that the adding of new entries in the font list is possible. In parallel the function MODIFY_CHARACTER_SET_LIST with the index and a list of pairs of character set type / designation sequence tail was introduced.

* ANSI.13 : Should the CHARACTER VECTORS be allowed to change within a (compound text) string?
Alternative 2: No, there are no more character vectors in this element.

* ANSI.15 : Should the scope of Addendum 1 cover the additional stable CGI functionality?
Alternative 1: Yes
Alternative 2: No

The CGEM group took a vote on this issue with the following result as regards alternative 1: (5, 1, 2). There was an agreement that this issue was solved by alternative 1.

* CGMA 1.17: Should EDGE REPRESENTATION be added together with the other representation function?
Alternative 1: Yes, as it is in the document.

*ANSI.16 : What is the intended result when SCALING_MODE and DEVICE_VIEWPORT appear in the same metafile?
Alternative 1: Use the last specified.

Note: If none is specified in the metafile, the DEVICE_VIEWPORT has precedence in categories where both exist!

* CGMA 1.7: Do pictures and sessions need to be distinguished?
This issue was skipped.

* ANSI.9 : Should the METAFILE DEFAULTS REPLACEMENT encoding in the Binary Encoding be improved?

New Alternative 3: No, but a text should clarify this problem in Addendum 1 in the context of subclause 5.3.12. Multiple occurance of the DEFAULTS REPLACEMENT element is legal and the effect is concatenation.

The CGEM group meeting was adjourned at 1.00 p.m. due to the social event which took place in the afternoon.

Minutes of CGEM Rapporteur Group Meeting
Sophia-Antipolis
22 May 1987

Attending: E. Moeller (Rapporteur), A. Mumford (UK), B. Trocherie (France)
F. Dawson (US), L. Henderson (US), P. Egloff (Germany),
H.G. Borufka (Germany), Satoru Kawai (Japan)

AGENDA:

1. Font List, Text Font & Precision (mapping GKS - CGM)
2. Resolving rest of the issues
3. Segmentation model (inheritance filter, attribute binding)
4. Segments outside pictures
5. Drafting
6. Additional, stable CGI functionality

Item 4:

It was decided that in the afternoon (during the mid-term plenary) a break out group should prepare a proposal on segments outside pictures as a basis for discussion on Sunday.

Item 1:

H.G. Borufka presented a model describing a mapping of the different font selection mechanisms in CGM/CGEM and GKS. The model requires a new metafile control element MODIFY FONT LIST (starting index , list of font names). It is based on a table of registered fonts residing on both the MO and MI workstation. It assumes that GKS text font numbers will be assigned to registered fonts by the Registration Authority. At every occurance of the GKS TEXT FONT AND PRECISION function the registered or implementation dependent (negative) font number generates an entry in a table of fonts in use if it does not exist already. This table corresponds to the CGM FONT LIST element. The new element MODIFY FONT LIST allows the font list to be build up sequentially and to be dense.

It was decided to add a note to the description of the CGM FONT LIST element recommending to generate a *dense* font list (note, that the registered GKS font numbers

may contain gaps or an application uses only a few fonts from a long list of registered fonts).

Whereas registered font names within ISO presumably will consist of the initial characters "ISO " the model proposes a GKS font introducer "GKS " for all fonts available in a GKS implementation, independent of whether they are registered fonts outside ISO (i.e. having a positive font number assigned) or implementation dependent fonts (negative font numbers). E.g. CGM font name "GKS -3" would be assigned to GKS font number -3. The CGEM group recommends CGI to include the new element.

Item 6:

The group then discussed the list of additional, stable CGI functions (document number V034, respectively CGEM document number SA10) as candidates for inclusion in CGEM. It was decided to add

DEVICE VIEWPORT MAPPING	to category 'cgmext1'.
DEVICE VIEWPORT SPECIFICATION MODE	(previously "UNITS") to category 'cgmext1' if the default would correspond to GKS (which is not the case in the current CGI draft DP 9636), otherwise include it also in the category 'gksm'.
SET DEFERRAL MODE	to category 'gksm'. CGI's BNI corresponds to GKS's BNIL. It was not resolved whether GKS BNIG should be allowed. Also the difference in the defaults was pointed out.
MAKE PICTURE CURRENT	to category 'gksm'.
PREPARE VIEW SURFACE	to category 'gksm'.
BEGIN FIGURE	to category 'cgmext1'.
END FIGURE	to category 'cgmext1'.
NEW REGION	to category 'cgmext1'.
IMPLICIT EDGE VISIBILITY	to category 'cgmext1'.
CIRCULAR ARC CENTRE BACKWARDS	to category 'cgmext1'.
SAVE PRIMITIVE ATTRIBUTES	to category 'cgmext1'.

RESTORE PRIMITIVE ATTRIBUTES to category 'cgmext1'.

Note, that the last two elements are closely related to the segmentation model which will be discussed later.

The role of the CGI function END PAGE was then discussed. It was not clear why this function behaves differently for softcopy and hardcopy devices. CGI experts should be consulted. However, it was decided not to include this function in the CGM Addendum 1.

The discussion then focussed on the two elements DRAWING MODE and PIXEL ARRAY from part 6 of CGI:

- how does DRAWING MODE apply to output?
- shall PIXEL ARRAY be included, if yes to which group of elements does it belong?
- it was noted to consider the device dependence of PIXEL ARRAY
- PIXEL ARRAY is to be ignored on non-raster devices
- consider addressability of raster devices
- shall there be a shorthand for CGI functions from part 6 in case they are included?
- should encoding technics go beyond the current ones, e.g. including compression?

Whether or not these elements should be included was left open at this point.

Karla Vecchiet (CGI) was consulted. She said, that for the DRAWING MODE function a raster device is needed but no bitmaps. However, it was not clear which value of DRAWING MODE would apply for CGI implementations not including part 6 (raster functions). This must be resolved in order to possibly include DRAWING MODE in the set of CGM/CGEM attribute elements. The group would like to have further arguments for including DRAWING MODE in CGEM.

Item 3:

Karla Vecchiet (CGI) explained the CGI segmentation model, in particular the attribute binding and inheritance filter. The discussion focussed on the question:

- Do we need the inheritance filter to support GKS?

The group came to the conclusion that GKS implementations with only one output device could make use of the inheritance filter and the copy function to emulate a WISS below the CGI.

It was decided to include these two functions in the category 'cgmext1' but not in the category 'gksm'.

The group would like to know what is the *minimum* CGI functionality to support GKS (which may be different from the GKS profiles!).

The meeting was adjourned for the day.

Minutes of CGEM Rapporteurs Meeting
Sophia-Antipolis
24 May 1987
Attending: E. Moeller (Rapporteur), A. Mumford (UK), B. Trocherie (France)
F. Dawson (US), L. Henderson (US), P. Egloff (Germany),
H.G. Berufka (Germany), and S. Kawai (Japan)

Documents Introduced

1. SA13 — Liaison Report-ISO TC 97/SC 18/WG 5 and ISO TC 97/SC 21/WG 2 (DIN - A. Schiller)
2. SA14 — Location of Segments in CGEM (US - L. Henderson)
3. SA15 — How to Interpret a Metafile Containing a Segment Library (DIN - H.G. Berufka)

9:15a - 10:45a CGEM Rapporteur Group Meeting

E.M. began meeting by addressing the question of whether anyone felt that the CGEM 3D (Addendum 2) work effort should be addressed in an interim meeting prior to the June 1988 "SC 24" meeting. A discussion proceeded on whether it was needed and when/where to hold it. The discussion ended with the question of an interim meeting being tabled.

A.M. expressed the view that since the BSI had circulated a expert opinion paper on CGEM 3D, at least one day should be allocated for such work. Per the recommendations out of the Egham meeting, this work should be restricted to GKS-3D Metafile support. The work should focus on developing a draft document.

E.M. indicated that it was his opinion that this document need not be technically complete. But, that the document could include sections that were left open or incomplete.

Christian Egelhaaf (DIN - CGI Rapporteur Group Member) came into the meeting to informally bring up CGI reference model concerns with the previous day's work on the CGEM "Global Segment Definitions".

E.M. indicated to the CGI Rapporteur Group Member that it was the responsibility of the CGEM Rapporteur Group to map to the CGI reference model. In addition, it was up to the CGEM Rapporteur Group to assist the CGI Rapporteur Group in developing a reference model for CGI.

Andrea Frankel (US - CGI Rapporteur Group Member) came into the meeting to informally clarify the CGI position on several questions that were liaised to the CGI Rapporteur Group on the previous day. (1) WHY HAVE INHERITANCE FILTER ELEMENT IN THE GKS PROFILE? A.F. indicated that the CGI Rapporteur Group developed the GKS Profile with the basic assumption that functionality would be added to support GKS plus, additionally, add functionality that would (a) aid implementors and (b) encourage use of CGI functionality in such implementations. (2) IF PART 6 FUNCTIONS ARE NOT INCLUDED IN THE CGEM, WHAT IS THE DEFAULT DRAWING MODE? A.F. indicated that the default would be "replacement". (3) WHAT WERE THE ARGUMENTS FOR INCLUDING DRAWING MODE FUNCTION IN THE SET OF STABLE CGI FUNCTIONALITY? A.F. indicated that the CGI Rapporteur Group felt that this was an attribute of primitives, was a stable function, and that while modes in addition to the basic 16 boolean operations were unstable at this time a mechanism would be added to permit extension to added operations when they become stable. (4) WHAT IS THE

RATIONALE FOR THE END PAGE FUNCTION? A.F. indicated, after much distracted discussion, that "blind interchange", to devices that can not effectively inquire device characteristics (i.e., is it a soft or hard copy device) was sufficient rationale.

The discussion returned to the agenda for CGEM Addendum 1. The meeting focused on resolved outstanding issues summarised in document SA1.

ANSI.23: SHOULD A PICK PRIORITY ELEMENT BE ADDED, SEPARATE FROM SEGMENT DISPLAY PRIORITY ELEMENT?

The group resolved that yes, it would be added.

BSI.2.4: SHOULD THE DESCRIPTION OF PICK IDENTIFIER BE LESS DETAILED?

The group resolved that yes, we should be sure that it does not contain too much information about the interpretation of this function.

CGMA1.18: DOES THE ADDENDUM NEED TO INCLUDE DEVICE VIEWPORT SPECIFICATION UNITS?

The group resolved that yes it did.

ANSI.3: WHAT IS THE APPROPRIATE SPECIFICATION OF VIEWPORT FOR THIS ADDENDUM?

The group resolved to revise the viewport specification as per the CGI (ISO/DP 9636).

CGMA1.4: HOW SHOULD SEGMENT DISPLAY PRIORITY BE RECORDED?

The group resolved that it be integer [0,n]. However, several members feel an "extent" function is needed. A new issue will need to be added. The group did not formulate one. An attempt was made to manipulate a certain member of the group into becoming the issues member.

ANSI.22: WHAT DATA TYPE SHOULD THE PICK IDENTIFIER BE?

The group resolved that the data type should be integer.

ANSI.17: WHAT SHOULD BE THE DATA TYPE FOR SEGMENT IDENTIFIER?

The group resolved that the data type should be integer.

BSI.4.2.1: SHOULD SEGMENT IDENTIFIER PARAMETER BE OF DATA TYPE N?

The group resolved that this issue was a rewording of ANSI.17 and should be resolved the same as that issue.

DIN.3.1.2: SHOULD SEGEMENT IDENTIFIER AND PICK IDENTIFIER BE OF THE SAME DATA TYPE?

The group resolved that yes, the identifiers should both be integer.

ANSI.18: DOES THE TRANSFORM MATRIX NEED A DATA TYPE?

The group resolved to not associate a data type with the transformation matrix.

CGMA1.21: HOW SHOULD A CATEGORY BE DEFINED?

The group resolved that categories should be defined as a single name (e.g., CGM, GKSM, etc.).

ANSI.19: WHAT SHOULD BE THE DATA TYPE FOR METAFILE CATEGORY?

The group resolved that the data type should be enumerative.

BSI.3.2: SHOULD A MAPPING OF THE GKS FUNCTIONS TO THE EXTENDED CGM BE ADDED TO THE CURRENT ANNEX E OF IS 8632?

The group resolved that the mappings would appear as part of the CGM Addendum 1. However, the mapping will be in a new annex that WILL be a part of the IS 8632, CGM Standard.

10:45a - 11:15a Coffee Break

11:15a - 12:30p Continuation of the CGEM Rapporteur Group meeting

L.H. presented the previous day's work effort on defining a reference model for location of segments in the CGEM. In particular, the presentation focused on the concept of "globally defined segments" versus "locally defined segments". Both capabilities are to be supported in the CGEM. The discussion focused on several issues that L.H. raised for group resolution:

CAN METAFILE DESCRIPTOR ELEMENTS SUCH AS THE VARIOUS PRECISION SPECIFICATION ELEMENTS APPEAR WITHIN STATE "GLOBAL SEGMENT OPEN"?

The group resolved that no Metafile Descriptor elements can appear in state Global Segment Open.

CAN A DELETE FUNCTION FOR THE CURRENTLY OPEN GLOBAL SEGMENT OR THE DELETE ALL SEGMENTS FUNCTION APPEAR WITHIN STATE "GLOBAL SEGMENT OPEN"?

The group resolved that this would not be allowed.

CAN DELETE SEGMENT, REOPEN SEGMENT, AND SEGMENT ATTRIBUTE ELEMENTS APPEAR IN THE METAFILE DESCRIPTOR BETWEEN GLOBAL SEGMENT DEFINITIONS?

The group resolved that no segment or segment attribute elements other than BEGIN SEGMENT can appear in state Metafile Descriptor.

The concepts in SA14 were approved unanimously by the group.

The subject of relationship of the CGEM to the CGI was brought up. This question surfaced through the rapporteur group meeting. Someone in the group asked, "IS THE CGEM A CGI SESSION CAPTURE MECHANSIM?". The group referred to the 19 May liaison meeting and concluded that by the current state of the CGI reference model the answer was NO. Someone else asked, "CAN EVERY CGEM BE CREATED THROUGH THE CGI?". The group again referred to the 19 May liaison meeting and concluded that while some way want such functionality the current reference model for CGI indicated that it was possible to generate some CGEMs from a CGI but that every CGEM could NOT be generated from a CGI.

12:30p - 2:00p Lunch

2:00p = 4:00p CGEM 3D Discussions

The agenda for the discussions included:

1. Limited general discussion until 4p
2. Reference model with respect to GKS-3D and PHIGS
3. Scope and basic concepts for CGEM Addendum 2

Reference Documents included:

1. BSI Proposal — SC 21/WG 2 N531
2. DIN Expert Contribution
3. Summary of a joint meeting of 2D and 3D DIN experts

Attendees included:

E. Moeller (Germany), R Gnatz (Germany), W. Brandenburg (Germany), S. Kawai (Japan), A. Mumford (UK), L. Henderson (US), F. Dawson (US), B. Trocherie (France)

A.M. began the discussion by presenting the BSI position paper. The BSI position included the reuse of the existing CGM opcode space with the addition of Z coordinates to current 2D primitives and control elements. The BSI position is that the CGEM 3D would reside in the same place in pipeline as the CGM.

E.M. commented that since the Egham rapporteur group meeting, the CGEM has focused on CGI functionality. However, there is no 3D CGI work and no indication has been offered as to when 3D extensions will be added to the CGI draft.

A.M. felt that most of the problems would result from the addition of 3D coordinates. The CGM 3D work was initiated under the assumption that it would follow GKS-3D.

W.B. presented the DIN expert contribution. The DIN position was expressed as being based on the assumption that 3D extensions to CGEM should not preclude or invent problems that would

have to be resolved to utilize the CGEM 3D capability in a PHIGS environment. The paper divided the possible 3D activity into 3 categories:

1. CGM with extensions added for 3D primitives. No new operation codes invented.
2. PHIGS picture metafile support. Things would be added to the CGM for PHIGS such as NAME SETS.
3. PHIGS Archive metafile. This capability would include functionality needed to generate after the Modeling Transformation before the Viewing Transformation. This would be like a CGM with only a Metafile Descriptor component made up of structure definitions.

It was noted that some elements of the DIN proposal would not be useable by a GKS-3D capability.

A.M. commented that the differences between the DIN and BSI proposals were that BSI feels that unresolved issues prevent work on a PHIGS capability, but that GKS-3D is all but complete. Hence, the 3D work for CGEM ought to be GKS-3D based. A.M. feels that the GKS-3D extension to CGEM would be compatible with the future PHIGS extensions.

A.M. commented on how the two papers were similar even though developed independently. Just how many elements should be added now for future PHIGS addenda was questioned.

F.D. commented that if it meant a significant delivery schedule impact for resolving whether to added PHIGS or GKS-3D support, he would prefer a minimal 3D extension to CGEM with just 3D primitives.

Considerable discussion continued on both GKS, CGI, PHIGS, and "hybrid" semantic differences.

Most parties informally agreed that any CGEM 3D work should not restrict future addenda effort to support PHIGS capabilities above the Addendum 2, GKS-3D Metafile support.

E.M. commented that if PHIGS capability could be built on a GKS-3D type CGEM by merely replacing the segment model with a structure model, then he was in favor of current extension for GKS-3D and later adding PHIGS.

W.B. felt that the PHIGS structure model was not too different than the GKS-3D segment model.

A.M. expressed her desire that the PHIGS rapporteur group will look at the CGEM Global Segment Definition paper (SA14) and review it with respect to how PHIGS capability could make use of this approach to global segments outside of pictures.

Summary of liaison meeting activities:

1. The meeting did not resolve the question of scope.
2. Discussion focused on some basic concepts; (a) 3D CGM, (b) GKS-3D CGM, and (c) PHIGS Archive type file with structures only.
3. Any CGEM 3D enhancements should be based on the CGEM Addendum 1 working draft.
4. An interim CGEM Rapporteur Group Meeting for 3D (Addendum 2) work is needed

4:40p - 5:30p Continuation of CGEM Rapporteur Group Meeting

The meeting reconvened with futher resolution of issues concerning Addendum 1.

CGMA1.3: HOW SHOULD GDPs BE STORED IN THE CGM?

The group resolved that they should be stored in an implementation dependent manner. L.H. commented that it was the domain of application profile and registration procedures to specify how the GDPs would be stored.

CGMA1.18: SHOULD THE ADDENDUM WORK INCLUDE THE DEFINITION OF ITEM TYPES RETURNED IN THE FUNCTIONAL STANDARD?

The group resolved that, in general, the Item Type definitions should appear in the functional standards. However, the GKS Item Types will be specified in the Addendum 1. Item Types are the responsibility of functional standards to define. It is up to the CGEM to define the mapping from the CGEM elements to the functional standard item types.

CGMA1.19: HOW SHOULD ITEM TYPES BE DEFINED?

The group resolved that item types would be defined as in GKS Annex E and expansion would be made as necessary.

CGMA1.22: WHAT GROUP OF ELEMENTS SHOULD VDC NORMALIZATION FALL INTO?

The group resolved that the element's name would be changed to MAXIMUM VDC EXTENT and that it would be a Metafile Descriptor element.

ANSI.14: SHOULD CGEM INCORPORATE THE FONT WORK OF SC18 AT THE PRESENT TIME?

The group resolved that the font work was premature to base technical work on. The font work would be watched for future incorporation.

BSI.3.4: SHOULD THE CONCEPT OF A "SESSION" BE DEFINED?

The group resolved that wording would be changed to "graphical session" and appropriate explaination of the term would be added.

ANSI.20: ARE SEGMENTS CLIPPED BEFORE OR AFTER SEGMENT TRANSFORMATION?

The group resolved that the clipped would be after the segment transformation, like the other standards.

CGMA1.8: WHAT ORDER SHOULD THE TRANSFORMATION MATRIX FOR SEGMENT TRANSFORMATION BE IN?

The group resolved that it would be in the same order as the CGI draft.

CGMA1.8: HOW SHOULD THE TRANSFORMATION FROM NDC TO VDC BE ACCOMPLISHED?

The group resolved that the MAXIMUM VDC EXTENT would be utilised by the GKSM to provide such functionality.

DIN.2.1: SHOULD GEOMETRIC ATTRIBUTES THAT MAY BE BUNDLED, SUCH AS LINE WIDTH WITH LINE WIDTH SPECIFICATION MODE "ABSOLUTE", BE TRANSFORMED UNDER SEGMENT TRANSFORMATIONS?

DIN.2.2: HOW DO THE CIRCULAR ELEMENTS AND THE RECTANGULAR ELEMENT TRANSFORM UNDER SEGMENT AND VDC-to-DC TRANFORMATIONS?

DIN.2.3: SHOULD THE EFFECT OF DIFFERENT SCALING IN X AND Y ON LINE WIDTH, MARKER SIZE, AND EDGE WIDTH WITH THE CORRESPONDING SPECIFICATION MODES SET TO "ABSOLUTE", AS WELL AS ON THE RADIUS PARAMETER OF CIRCULAR ELEMENTS BE SPECIFIED IN THE CGM ADDENDUM 1?

The group resolved that all three of these issues are similar and would follow how the CGI resolved these issues based its pipeline.

ANSI.21: WHAT MEANING IS ATTACHED TO "BEGIN SEGMENT (A)" WHEN SEGMENT "A" EXISTS?

The group resolved that this achieves no use, so it is invalid syntax.

DIN.3.1.11: IS THE SEGMENT PRODUCTION RULE INCORRECT FOR THE CATEGORIES GKSM/GKSM0 WITH RESPECT TO THE CLEAR ELEMENT?

The group resolved that the grammar will be fixed.

CGMA1.11: SHOULD THERE BE A SUPER-GRAMMAR?

The group resolved that there should be a formal grammar.

CGMA1.10: WHERE SHOUD THE FORMAL GRAMMARS BE LOCATED?

The group resolved that the formal grammar should be located within the CGM Addendum.

ANSI.26: WHAT IS THE RELATIONSHIP BETWEEN CGM ADDENDUM 1 AND GKSM? (I.E., IS THE ADDENDUM MEANT TO CONFORM TO GKSM WITH A STRICT 1-to-1 MAPPING, OR ON A MORE GENERAL LEVEL WITH A WELL DEFINED MAPPING?

The group resolved that a more generalized support with well defined mapping to GKS was appropriate.

ANSI.25: SHOULD THE DISCUSSION OF CONFORMANCE INCLUDE REFERENCES TO GENERATORS AND INTERPRETERS?

The group resolved that the discussion of conformance should not include references to generators and interpreters. However, annex information can be used to make clarifications or recommendations.

BSI.2.2: SHOULD CONFORMANCE BE RELATED TO CATEGORIES?

The group resolved that it would.

ANSI.27: HOW IS CONFORMANCE WITH CGM ADDENDUM 1 TO BE DEFINED?

The group resolved that conformance would be defined by category.

ANSI.1: SHOULD SEMANTICS OF ALL ELEMENTS IN THE ADDENDUM BE UNAMBIGUOUSLY DEFINED?

The group resolved that each element has unique semantics, where differnt functionality is needed, different element will be included.

CGMA1.23: DO CLEAR AND UPDATE HAVE ANY IMPLIED MEANING FOR THE INTERPRETER?

The group resolved that by virtue of the unique semantics of the CGEM that this issue was moot.

BSI.2.3: ARE THE ROLES OF UPDATE AND CLEAR WELL DEFINED AND SUITABLE FOR A WIDE RANGE OF CATEGORIES?
CXGMA1.15: DOES REDRAW ALL SEGMENTS CARRY WITH IT AN IMPLIED MEANING ON INTERPRETATION?

The group resolved that these issues should be withdrawn.

BSI.3.1: HOW SHOULD THE EFFECT OF THE REPRESENTATION ELEMENTS ON INTERPRETATION OF THE METAFILE BE DESCRIBED?

The group resolved that the description is left to the CGEM annex that maps the CGEM opcodes to the client Item Types. The actual effect is intended to be dynamic. In the case of the Color Table and Pattern Table the annex will clarify the relationship from that that is poorly annotated in the CGM.

The meeting was adjourned.

Minutes of CGEM Rapporteur Meeting
Sophia-Antipolis
25 May 1987
Attending:E. Moeller (Rapporteur), A. Mumford (UK),F. Dawson (US), L Henderson (US)Satoru Kawai (Japan)

AGENDA FOR THE REST OF THE MEETING

The work needing to be drafted was considered. This includes:

1. Recommendations
2. Work on Clauses 0-4
3. Clause 5
4. Encodings
5. Mappings of the CGEM elements to the GKS items
6. Font mappings CGEM/GKS and guidelines to implementors
7. Annex for mappings
8. Formal grammar

CGI LIAISON

1. It was noted thaT THE UPDATE issue had not been solved for CGI
2. CGI have asked us to incorporate DELETE ATTRIBUTE SET - agreed
3. CGI Rap group are split on whether metafiles should be generated via the CGI
4. In wording of IMPLICIT REGEN MODE it is believed that when spuuressed is allowed the regeneration will occur (not may)
5. CGI to add the regen pending flag to the CGI state

ITEM TYPES

The group felt that item types should not be a part of the CGEM but of the functional standard to which they pertain. In the case of GKS the item types may be a part of an Annex to the first addendum.

RECOMMENDATIONS

It is believed that we can produce the mark up of the first addendum out of this meeting. The addendum will reference the CGI text where appropriate rather than taking the text from the first DP. The text will be produced in August. The next draft -hopefully DIS- will come out of the next SC24 meeting and the IS out of the next.

For Addendum 2 there will be no text out of the meeting. It was agreed that we neede to have a metafile for GKS 3D. It was decided to have a meeting in Sept/Dec timescale to produce a working draft for comment and hopefully progression to Draft Addendum at the next SC24 meeting.

APPENDIX 5

SYMBOL LIBRARY PROPOSAL

SPR 11
US24

Location of Segments in CGEM
Valbonne, 24 May 1987

1. Introduction

Segments are being added to the CGEM. The CGEM Experts at the Sophia-Antipolis meeting, in consultation with the CGI experts, have chosen the CGI segmentation model for CGEM. This is believed to be adequate to support GKSM (with the resolution of a couple of minor deficiencies, now being considered by CGI).

One open issue (both in the initial ISO issues log and raised again by ANSI in the US review of the draft CGEM) is where segments may appear. It is agreed that they may appear in pictures and have a definition that exists only within the the picture (these will be called Local Segments). The question arose whether segments should be definable in one other place as well, such as the metafile descriptor, and be referencable from anywhere in the metafile (these will be called Global Segments).

The CGEM experts feel that this is valuable functionality, that it can be included without inventing new elements or concepts, and it should be included if possible. This paper describes how the CGI/CGEM segmentation model provides such capability.

2. Goals and Design Criteria

The segment model of CGEM is to meet the following criteria:

1. Local Segments should be provided;
2. Global Segments (globally referencable segments) should be provided;
3. The mechanisms (functions and elements) to implement the two should be identical -- no new functions should be added;
4. Random access to pictures and logical independence of pictures should be preserved;
5. The segmentation model including both Local and Global segments should maintain a clean state model of CGEM;
6. The model must be implementable.

3. Technical Description

The CGEM sub-group working on this problem defined the solution by addressing a number of issues. The solution is described by the resolution of these issues.

3.1 Segments or Macros?

Does the CGI segmentation model, and the desired global segment capability, defined a macro facility or a segmentation facility?

Certain functions. particularly some control functions, may occur in CGI/CGEM between BEGIN SEGMENT and END SEGMENT. These functions are executed (interpreter for metafile), but in a segmentation facility do not go into segment storage. In other words, a COPY function would not cause the functions to be executed again. If a macro facility were being defined, then the effect of these functions would happen again at invocation time (e.g., during a COPY).

It was agreed that the CGI model is a segmentation facility for both local and global segments. No attempt will be made to provide a macro facility.

3.2 Where Are Global Segments Defined

Are global segments defined in the Metafile Descriptor, or in a separate block (either in the MD or following)?

Whereas the existence of a separate segment definition block is somewhat cleaner in concept, it would require new elements to be added and would lose some functionality that is achievable through allowing the segment definitions to occur within the Metafile Descriptor. Allowing a segment definition anywhere within the MD is the chosen solution. This allows useful interaction with the MD elements (including Metafile Defaults Replacement). A clean state model which includes a Global Segment Definition state is still preserved (GSD state is entered by BEGIN SEGMENT while within the MD state, and exited by END SEGMENT).

3.3 How are Segments Accessed from Pictures

Are global segments automatically known/visible from within pictures, or must they be specifically invoked?

Global segments (those defined in the Metafile Descriptor) must be referenced by a COPY element from within a picture for their primitives to become visible withing the picture.

3.4 May a Local Segment and Global Segment Have the Same Name

May segments defined locally and globally share the same set of names, or must the names be unique?

The names must be unique.

3.5 How Are Primitive Attributes Defined and Bound

The modally applied elements comprised of Primitive Attributes, Control Elements (e.g., clipping), and Picture Descriptor elements are modally bound to primitives in CGEM/CGI. How do these apply to globally defined segments?

At the occurance of BEGIN METAFILE, and at every point in the metafile thereafter, all modal elements have a well-known and well-defined value. These are the values that are bound to primitives at the occurance of the primitives between BEGIN SEGMENT and END SEGMENT.

The metafile descriptor is processed sequentially. Conceptually, there is a temporary state list of all modal values (temporary because the state list is reinitialised to default values at the start of every picture), and this "temporary state list" of modal values is updated as expected as modal elements are encountered. Segments may be defined throughout the Descriptor, and segment definitions may be interleaved with Metafile Defaults Replacement.

Note that Picture Descriptor elements (e.g., LINE WIDTH SPECIFICATION MODE) are bound like all others, and that a bound value could be in conflict with the value set in the Picture Descriptor of a picture referencing (COPYing) the segment. This is an implementation detail for interpreters to pay attention to, and is implementable (it exists in CGI now), but presents no conceptual problems.

3.6 What Changes to Global Segments May Occur Within Pictures

Several functions would change the definition of segments, or their default appearance. These include REOPEN, DELETE, and all of the segment attributes (highlighting, transform, etc). Are these allowed within pictures when referring to a globally defined segment?

No, they must not be allowed. If they were, then the metafile would no longer be randomly accessible. If segment attributes are to be manipulated, then the global segment should be COPYed to a local segment, and the local segment should be manipulated. COPY is the only

allowable function referencing a global segment.

3.7 What Elements May Occur Between BEGIN/END of a Global Segment

Certain functions like Redraw All Segments (RDAS) may occur between the BEGIN/END of a local segment. The are executed there, but not stored in segment storage. They make no sense in the Metafile Descriptor. Should they be allowed and defined as No-ops, or not allowed.

They should not be allowed. It is cleaner not have to specify the effect of a function by state. Specifying allowability by state can be achieved with a formal grammar, on the other hand. Therefore, RDAS and a few more functions (to be defined) are not allowable in GSD state.

3.8 Is COPY Allowed in GSD State

May the COPY function occur between BEGIN SEGMENT and END SEGMENT of a global segment definition?

Yes, this is useful functionality and valuable for data compression.

3.9 May DELETE and REOPEN Occur in GSD state

May DELETE of another global segment occur in the Metafile Descriptor, either in GSD state (within a segment) or in MD state (outside of segment definition).

As long as it is understood that the metafile is processed sequentially, there is no problem with allowing DELETE. The same sequence may occur in the CGI, and implies certain implementation constraints. REOPEN only makes sense outside of GSD state, i.e., in MD state. Once again there is no problem in allowing it. An issue that has not yet been addressed is whether these functions should be allowed in MD state (as opposed to GSD state).

4. Summary

The CGEM experts considering the "global segment" question were unanimous in believing that this is a useful feature and should be included in CGEM. No new elements are needed – the capability can be provided with the set of elements already provided by CGI/CGEM segmentation. The capability is provided utilising only the current attribute binding models of CGI/CGM – no modifications or new concepts are required.

APPENDIX 6

MINUTES FROM CGM WORKSHOP

Minutes from CGM Workshop 9/15/87

Attendees:
Mark Skall
Dan Benigni
Sue Quinn
Steve Carson
Anne Mumford
Peter Bono
Steve Jepsen
Ted Reed
Glenn Davison
Charles Tucker
Chris Osland
Kevin Hardman
Lofton Henderson
Jane Pink
Sharon Kemmerer
Brian Rossin
Joe Collica
John Stoll
Gary Silverman
Richard Carr

9:10 Mark Skall gave introduction explaining the history of the conference and the relationship with the Europeans. He also gave the logistics of the conference: each speaker presents his/her paper and the audience proposes issues. We are looking for conclusions to the issues that are proposed, and these will be included in a book being produced by Eurographics.

9:20 Anne Mumford explained how the output of the conference will be used. There will be an NBS Report produced at the close of this conference, and there is a Eurographics Seminar Series. Both parts will be included in the Eurographics book. Eurographics gets the copyright and will get all of the royalties. All patricipants will get one copy of the book.

9:25 Mark identified the chairs of each session:
Anne Mumford will chair the papers on Performance.
Mark will chair the papers on Testing.
Sharon Kemmerer will chair the papers on Cals.
Peter Bono will chair the papers on Implementations.

9:30 Introductions were made around the table.

9:40 Anne Mumford chaired the panel on performance.
Glenn Davidson presented his paper titled "Protocol Comparisons, CGM and Others." He implemented character and clear text encodings in VAX/GKS and did a comparison of the two encodings. Peter Bono asked regarding the character and clear text encodings: are

the generators intelligent enough not to pick up superfulous changes? The response was negative. Ted Reed inquired how a binary metafile would compare in this study. Glenn didn't implement a binary encoding so he did not have the experience to answer that question. Lofton stated that the clear text encoding is the most difficult to implement with all of the variations. He wanted to know what differences were found between the binary and character encodings. For example, the control information in the encodings; if you don't use the incremental (?). Steve Jepsen said that the displacement in character encodings is basically equivalent to the incremental approach. (***See Mark's notes on Glenn Davison***)

Chris Osland was the next speaker to present his paper titled "Bridge that GAP" where GAP stands for Graphics Application Protocol. He wanted to bridge the gap between Crays and workstations using metafiles. He stated that the CGM doesn't use all of the power that is available to the machine and wanted to bridge the gap between the application and the workstation in a way to use the all of the capability of the workstation; with the workstation he wanted to be able to SELECT and DRAW. He wanted to be able to ship application profile data, not just graphics. Instead of using the CGM, he used ISO 8632; parts 2,3,4 represent the encodings he used. It worked efficiently, but it is a complicated standard to use. He was able to send n elements of an item of one groupp. It allows full use of the power of a workstation. Ted Reed inquired if Chris had looked at the remote procedure call to get the same results. Chris replied that it was not available at his site and he needed a solution immediately because he was transmitting an entire database.

10:10 Charlie Tucker presented his paper next on "The Implementation of CGM as a GKS Metafile." He informed us that there was a book available on CGI/CGM from Nova that could be obtained through him. Some of the differences that he pointed out were that GKSM is not part of the GKS standard; CGM is static data - picture capture whereas GKSM is an audit trail. The GKSM is weakly specified such that there are not specificiations at the bit level; the CGM is well specified in this way. The GKSM has not picture frames, and the CGM has no segment capability - it must be simulated by a GKS driver which is inefficient. The output mappings are suitable, with the exception of text, and most functions are present for generation. Regarding interpretation of the CMG by a GKSM: GKSM thinks the level of visibility that the application profile should have is low. The implementor needs to

map the CGM opcodes to GKSM item types. Peter Bono suggested that Annex E of the CGM standard only addresses the level 0 case, and says nothing of segments. Lofton inquired about the simulation of bundles and predefined bundles in generation. Charlie stated that they are set up in the workstation description table. Lofton also pointed out that the GKS pattern and colour tables are dynamic and unspecified in CGM. Charlie stated that he used CGI to implement the interface between the CGM and GKSM. Chris Osland stated that workstation specific functions caused ommission of certain functions and items in CGM. GKSM uses the metafile as a picture, whereas CGM is able to modify its contents. Peter stated that CGM elements will be eventually put together in a way that can be used by GKSM.

Anne Mumford identified the following issues from her group:

1. From Glenn's paper:
   a. the difference between the encodings and the reasons for choice are the ones in the standard collect;
   b. differential chain coding needs to be looked at;
   c. sharing metafiels and the demand for encodings.
2. From Chris' paper:
   a. input to ISO work on addenda
3. From Charlie's paper:
   a. using CMG in the GKS environment: do we need consistent usage?
   b. Mark Skall: item types - addendum.

10:55 BREAK

11:10 Sharon chaired the CALS session

Steve Carson was the first speaker in this session to present his paper titled "Extending the CGM for Publishing and Technical Drawing Exchange." He did a study of the CGM to determine its suitability in the CALS arena. He decided that it was suitable, but extenstions to the standard were necessary. He suggested the registration procedures to do this. The criteria he used for selecting extensions were compactness, ease of generation/interpretation, device independence, and consistency with other standards. The extensions he suggested were linetypes, symbols, curves, text, images, and def. and instancing objects. He used IGES, PDES, and Postscript to get his ideas for the extensions. Regarding images, Steve suggested that we need raster input primitives to accept input from

scanners and files stored on disk. Lofton stated that TPM is an interface to printers, and not a document exchange protocol. The current CGM text model is not a typographical model. He suggested that we leave it in and add to it.

Peter Bono presented his paper "Raster-to-Vector Conversion: A State-of-the-Art Assessment." He presented the idea that CALS needs drawings in several areas: engineering databases, bid packages, technical documentation, and training. Mark Skall suggested that education of raster-to-vector implementors about CGM is needed; also if it was possible to get the source code from vendors and modify algorithms to intercept data and produce CGM files. Peter stated that vendors are either liberal with the code or state that it is strictly proprietary information. He suggested two companies that Mark might contact who would probably be willing to provide their source code: Optigraphics and Anatech. Peter also stated that suppliers should supply CGM as an output format for vector and CAD data, but enhancements to CGM are necessary. He also stated that a user requirements document could be obtained from the workshop and be sent to ANSI and ISO which would be beneficial to all involved. He also stated that they are currently looking at an IGES to CGM translator. Steve Carson stated that using IGES for engineering drawings instead of CGM results in a loss of 95% of the data that is being transferred - CGM is much more efficient. He also stated that for raster-to-vector images, there is a need to compress them for storage. Lofton stated that if you are preserving documents, either form is appropriate. If you want to modify the image later, then it should be stored in vector format. Chris Osland asked about 3D reconstruction, and was answered that there is no demand in the market area yet. Lofton stated that the many enhancements that were suggested by Steve Carson and Peter Bono have been added to CGEM, but others were not added due to a strict time schedule for addendums. He agreed with Peter Bono that a user-requirements manual generated from this workshop would be beneficial. He also suggested that the US take the lead on Addendum 3.

Lofton Henderson presented his paper on "The CGM Application Profile for CALS: Current Specification and Major Issues." He stated that the reason for application profiles is that the CGM syntax in the standard is complete and unambiguous, but the semanticas are left incomplete to make them endependent. Due to this, generators/interpreters behavior is not standardized. The result is that there is no single correct interpretation, which is essential

in the CALS environment where predictibility is reequired. Also, there is no way of testing generators/interpreters. The application profile is a resolution of ambiguous semantics and places specific requirements on generators/interpreters. It deals with functional extensions of the standard. The CALS Application Profile specifically is the same as or is a superset of the TOP Application Profile. Some of the objectives and criteria used for the CALS AP were that the CALS CGM must be a legal CGM, the picture specifications must be unambiguous, the behavior for generators/interpreters must be predictable, the data formats for interchanges must be specific, and no functional extensions were used until they have been approved for Registration. They are currently working on ensuring that the TOP AP and the CALS AP are compatible. The first revision of the TOP AP will probably be equivalent to the CALS AP. The binary encoding was used for both application profiles. Two generator requirements and issues were identified: mapping out-of-range attributes, and data structure support and maximum primitive lengths (-1024 poly elements, -256 color tables). Two conformance levels were identified for interpreter conformance requirements: publication quality (do it correctly- the entire picture), and preview quality (map color to black/white, scaling mode, transparency and aux color, <skewed> cell array, attributes per annex d defaults, etc.). They chose to wait to use enhancements until they were officially registered because too many changes take place before the registration process is over. The CALS proposal will go without extensions for now, and in a year they will be added in as revisals. The question regarding using all three encodings in an application profile was brought up. Lofton stated that the character encoding could be useful for communication, but the difficulty increases having to handle three encodings. A study needs to be done to see how useful the encodings are. He also stated that 3D needs to be looked at. Steve Carson brought up the issue of the proprietary nature of fonts and suggested that the government come up with a good public font for general use, which would save them money. He suggested upgrading the Hershey fonts from 1968.

Mark Skall identified several key issues in this area:

1. Levels of engineering drawings
   a. conceptual and developmental design > use CGM
   b. prototype and limited production > instead
   c. production > of or
      > in addition
      > to IGES
2. IGES is used for simulation - could CGM be used?
3. reliability and maintainibility

4. Can CGM be used in reports?
5. to Lofton: it would be useful to know which extensions have been done away with and which are being used?
6. what do we do now at the interim for the CALS AP and the revisions in one year (how do you handle the proposed items for registration that belong in the ap)?

Mark Skall chaired the session on TESTING

Mark pointed out that testing does not get enough attention during development of the standards.

Richard Carr presented his paper on "An Overview of ODA and ODA Conformance Testing." ODA stands for (electronic) Office Document Architecture standard, and is in its 2nd DIS stage in ISO. ODA is an interchange format that is used for processable documents. It can include future architectures based on the way it is set up. An application profile has been developed for the U.K. He discussed the ODA model (section 3 of his paper). He pointed out that there are several document classes, categorized by the common characteristics they share. As far as conformance testing is concerned, there are three document architecture classes with various levels in each class. The emphasis is not on application profiles, and not on the levels (which may eventually disappear). Conformance testing in the ODA standard area is also concerned with compatibility with other standards. Within the application profiles, a superclass of objects is defined. He went over the ODA testing environment, which is on the last page of his paper, but used a newer environment on his viewgraph, which includes value-added testing that is not required by the standard. To relate the ODA testing environment to that of CGM, he said all that was necessary was to change the document analyzer to a CGM content analyzer. He stated that DAPs will be registered by the registration procedures.
In the discussion that followed, Richard stated that the ODA Application Profile was written with the TOP Application Profile in mind. Mark was asked toget a copy of the "NBS/ODA Implementor's Agreement" from Fran Neilson. Peter suggested looking at the NBS/ODA Implementor's Agreement and the ODA/TOP AP to see how closely is is associated with the TOP and CALS APs. Peter suggested that the following should go in the users requirements manual: how an ap should look; what should they use.

Jane Pink presented her paper on "Testing of the Computer Graphics Metafile." She informed us that this was an "ideas only" paper and that she was looking for

feedback from both implementors and users. She presented us with a brief history of NCC and described the types of testing they do: conforming, non-conforming, and capacity programs. For CGM, she suggested that we would want to check for correct storage of format, correct generation, and correct interpretation. However, the standard is not concerned witht he performance of interpreters and generators.
There are two levels of conformance that we need to be concerned with: full conformance (using one of the three encodings) and functional conformance (which could use a private encoding). Conformance checking for the CGM is limited to full conformance and syntax checking only. However, it would be useful to do evaluation testing which would test interpreters and generators. TO test for conformance, a testing lab would use a syntax checker which must be able to deal with three encodings, and this would have to be done in an automatic fashion. To do evaluation testing of generators, which is outside the scope of the standards but may be within the scope of an application profile, the testing lab would need to provide sample programs to the client for him to generate, or provide pictures to the client as a backup procedure. This would result inmanual checking of displays. To do evaluation testing of interpreters, the Implementation Under Test (IUT) would generate metafiles and the lab would need to look at the displays once again, on site. At this point certificates would be issued if appropriate. The reference implementation would have to be an encoder/decoder which could handle all three encodings correctly, and perhaps incorrectly. The testing lab would need a comprehensive database of metafiles testing all features of the CGM (the GKS suite could perhaps be used as a starting point). No pass/fail criteria could be established for testing interpreters and generators as this is only value-added testing. The problems with this method is that automatic testing is limited and it is manually intensive. Also, true remote testing is not possible as the testing lab would need to travel to the site being tested.
In the discussion that followed Jane's paper, Peter Bono pointed out that there is an immediate need for using these test files in NCGA, and that implementors would be willing togenerate these test programs that would cover the full CGM. Gary Silverman stated that prior to testing the syntax of a CGM the ambiguities in the standard would need to be removed. Peter pointed out that a mechanism is needed where implementors and users can ask questions regarding the interpretation of the standard and receive an answer in a reasonable time frame. He stated that users and implementors need to be educated in knowing that there is a CGM Control Board out there and that answers are available. Peter

also mentioned that in the latest ANSI mailing, document X3H3.87 was a paper by Dave VAndershell on CGI/CGM relationss. This proposal on the CGI binding to cover the CGM binding would allow you to pass an application program...(?). John Stoll informed us that he has programs which do evaluation testing of CGMs, but he needs a reference point. Steve Carson suggested a way to handle private encodings in the functional conformance area. He suggested building a parser to test the metafile itself that is table drive. Peter knows someone who is doing this now with an interpreter (see Mark's notes). Anne Mumford suggested that Gary Silverman's experience would be useful here. She stated that difference in this model and the OSE model is that in the OSI testing layer 4, you know what comes above it, but with the CGM you don't (you could put a functional standard above it).

Sharon Kemmerer presented her paper on "A National Bureau of Standard Conformance Testing Program, Ideaas and Procedures for Graphics Testing." In it she proposes a process on how to develop a test program for any area in computer graphics. The FIPS for GKS, CGM, and eventually PHIGS would be affected by this. Her goal is to eventually produce a FIPS that is a step-by-step procedure on how to become certified.

**********Sharon Kemmerer took notes here**********

Peter Bono chaired the session on Implementation.

Anne Mumford presented her paper on "The Use of the Computer Graphics Metafile in the UK University Community." She stated that there was much confusion in the university environment regarding software, hardware, and terminals. The reason she started using the CGM was that the turnaround time was too long for a job to be completed. Now there are many choices when she needs something done. She used the character encoding. At plotting time, she is able to use any plotter she desires, which provides her with more flexibility and the ability to add different devices to create any possible configuration desired. She encouraged other universities to use the CGM as well as to share their resources and newtworking. Her CGM project was a character encoding with a FORTRAN interface to GKS-UK and a common packaage. She needed to write the software to do the job, and to persuade others to use the CGM. Peter Bono asked if her development work included writing a generator and an interpreter, and she replied yes. He alsc asked why micros were excluded from her configuration, and why

she did not use the binary encoding. Anne replied that the micros were excluded by omission, and that the initial intention was to use the character encoding and the binary encoding will eventually be implemented. Lofton asked if the CGM can stand alone. Anne replied that it is a subroutine library. Lofton pointed out that the initial action on the part of the US was to implement the binary encoding, whereas the initial action in the UK is to implement the character encoding. Anne said that this was due to their initial reading of the standard, and now that they have access to efficient networking they are going to go back and implement the binary encoding. Chris Osland stated that the binary encoding was impossible with the given networking facilities in the UK.

John Stoll presented his paper on "The CGM Implementation at McDonnell Douglas." He stated that one of the most useful applications is the ability to transfer compound documents. MDD needed a single corporate metafile to handle in-house and factory uses. The corporate plot file was implemented as the CGM. On page 3 of his paper, he points out that compound documents have both content and structure. There are two standards, namely SGML and ODA/ODIF, which are emerging to handle the structure. SGML leaves much up to the implementor and he therefore looked at ODA. His CGM interpreter is really an element parser. He also stated that he is now maintaining the CGM Vendors/Implementors List, which Peter suggested may be transferred to NCGA for maintenance.

Ted Reed presented his paper entitled "After Ten Years of Metafiles - Where Does the CGM Fit?" He works at Los Alamos, which is a diverse and stable environment. Their system has been optimized for metafile support. He stated the following six issues are necessary as software support for the CGM:

(1) efficient generation of the CGM,
(2) consistency of displayed images on different devices (of generators and interpreters on the same device),
(3) rapid random access to any CGM image,
(4) CGM software compatibility across many environments,
(5) translators between the CGM and other graphical formats, and
(6) CGM based graphics editors.

He stated that during the transfer of the current system at Los Alamos they will need to maintain both systems. This will take several years to complete the process. Gary Silverman pointed out regarding item (2)

above that the standardizaton of fonts is needed, as well as a standard default color table, and the need to know which application profile is being used.

Brian Rossin of Wang presented his paper on "The Ramifications of Adopting the CGM as THE IMAGE FILE TRANSFER MECHANISM." He stated that there were not enough hatches define in the CGM standard. The Wang CGM (WCGM) defines 125 hatches in their implementation. However, the problem they meet with such a large number of hatches is that when a new device is added to a configuration it is difficult to keep the hatches consistent. Brian stated the reason for moving to the CGM was corporate acceptance. Chris Osland pointed out that the ISO 20.22 conventions could be used and inquired whether Brian was going to use this or something else. Brian replied that he was going to use both because there was a need to back support the existing Wang implementation. Steve Carson pointed out the number of hatches necessary is large, and Lofton Henderson suggested that the generator could take the burden and produce a user-defined hatch. Peter brought up the issue of extensibility.
The issues that Peter Bono pointed out from his position as chair were:

1. gather user requirements
   a. real end-user
   b. application developers
   c. business graphics
2. education
   a. corporate
   b. external users
   c. Nova's book
3. need and value of guidelines
   a. when to use certain features
   b. other categories where recommendations would be useful
4. user requirements and barriers to acceptance
   a. recommendations to new application profiles to get around the current barriers due to current status of standards.

Kern Hardman spoke on the MAP/TOP Application Profile. He stated that version 3.0 is open for comments and changes. He stated that the purpose of the TOP CGM AP was to promote interoperability and to define what is outside the scope of the standard. The user is able to preset defaults and set limitation is he knows he is using a TOP CGM, but he is able to still explicitly set other parameters. He listed the following events as demostrations of the TOP AP:

1. NCGA '88 in March

2. ENE '88i from June 6-8 at the Baltimore Convention Center.

Meeting was adjourned and the group split into smaller subgroups to discuss their issues.

APPENDIX 7

PROPOSAL FOR CGM ADDENDUM 3

X3H3/87-150

Proposal for CGM Addendum 3

9 October 1987

Purpose

The purpose of this addendum is to extend the CGM to effectively fulfill the picture transfer requirements of:

1) Engineering drawing and technical illustration
2) Graphic arts quality pictures, including geometric graphics, raster images, and text
3) Technical publishing

An additional intent is to keep pace with the graphics requirements of office systems, especially ODA requirements.

Scope

This addendum comprises a set of elements which will extend the capabilities of CGM as needed to meet additional user requirements in engineering drawing graphic arts and technical publishing. The set of elements should include all elements necessary to meet those requirements. It should be the minimal set sufficient to meet those requirements effectively.

The following preliminary list of capabilities is identified as necessary to meet these requirements.

1) Advanced 2D graphics, to include:
   - Bezier curves
   - Rational B-splines
   - Parametric spline curves
   - Line attributes of line cap, miter and join
   - Composite line primitive
   - User-defined line types
   - User-defined hatch styles
   - Additional standardized hatch styles
   - Arbitrary text path
   - Conics and conic arcs

2) Text and font model of ISO 9541, Information Processing--Font and Character Information Interchange

3) Picture composition and control to include:
   - Arbitrary clipping boundary (general closed curve)
   - Shielding
   - Alignment

4) Additional color models beyond RGB
   - CIE
   - CMYB
   - Named colours

5) Additional raster graphics (scanned image) capabilities

6) Symbols: external reference to "standard" libraries of named symbols

The scope of this addendum assumes that the capabilities of CGM Addendum 1 and Addendum 2 are available.

Justification

CGM users have found that in some application areas the present standard provides a general framework that is suitable but lacks some functionality required by these applications. These application areas include engineering drawing, the preparation of graphic arts quality presentation materials, and technical publishing.

A recent workshop sponsored jointly by the NBS and Eurographics, entitled "The CGM in the Real World", examined this issue and concluded that the CGM lacked capabilities to effectively meet some advanced user needs. As an example, for engineering drawings, it is difficult if not impossible to effectively represent some higher-level constructs, e.g. splines and curves. Though such constructs can be simulated with simpler primitives in the CGM at present, it is impossible to maintain accuracy and visual continuity and still retain device-independence.

In all cases, there is a requirement to expand CGM text to include font definition capabilities that are consistent with the ISO DP9541 font standard. The font definition as it exists in ISO 8632 (CGM) is too general for practical use. Even though several fonts are identified in the TOP V3.0 CGM Application Profile, these fonts are not adequate for publishing and graphics arts applications.

Many publishing and graphic arts systems use color models other than RGB. For efficiency and ease of implementation in these areas, for example, additional color models are needed. It also became apparent at the workshop that the Cell Array primitive in the CGM is not adequate for most applications that use raster graphics. Thus, this addendum will also provide additional raster graphics capabilities.

Program of Work

The following schedule is proposed for CGM Addendum 3:

December 1987 - US Proposal at Berlin SC24 meeting
December 1987 - Initial Draft (ID) available
January 1988 - Joint ANSI/ISO meeting produces Working Draft (WD)
July 1988 - WD comments processed at SC24
February 1989 - DP comments processed; 2nd DP produced
August 1989 - DIS text produced
August 1990 - Final IS text; publication of IS

CGM PRESENTATION USER REQUIREMENTS

USER ROMT	FUTURE NEED	CALS ENG. DRWG	CALS TECH. MAN.	APPLICATION BUSINESS GRAPHICS	APPLICATION OFFICE SYSTEMS	APPLICATION PUBLISH-ING
ADV. 2-D GRAPHICS	CURVES/BEZIER	X	X	X	X	X
	PATH	X	X	X	-	X
	PEN	-	?	-	-	?
	CLOSED FIGURES	X	X	X	X	X
	ARBITRARY CLIPPING	X	X	-	-	X
	SPLINES (CONVEX/B-)	X	-	-	-	-
	USER DEFINED:	//////////	//////////	//////////	//////////	//////////
	LINE TYPES	X	X	X	X	X
	CAP,JOIN, MITRE	X	X	X	-	X
TEXT/ FONTS	IMPLEMENT ISO DP 95401	-	X	X	X	X
	COMPATIBLE TEXT & FONTS	-	X	X	X	X
	ARBITRARY PATH	-	X	X	-	X
COMPOUND DOCUMENT	ARBITRARY CLIPPING	X	X	-	-	X
	ALIGNMENT	X	X	-	X	?
	SHIELDING	X	X	X	-	X
COLOR	INTERPOLATED FILL	-	X	X	-	X
	MODULES:	//////////	//////////	//////////	//////////	//////////
	CYMX	-	-	-	-	X
	NAME THE COLORS	-	-	X	-	X

CGM PRESENTATION USER REQUIREMENTS

USER ROMT	FUTURE NEED	APPLICATION: CALS ENG. DRWG	CALS TECH. MAN.	BUSINESS GRAPHICS	OFFICE SYSTEMS	PUBLISHING
COLOR(cont.)	CIE	-	-	-	-	X
COMPACTNESS(1)	STORAGE	X	X	-	X	-
	TRANSFER	X	X	X	-	-
EDITABILITY	APPLICATION STRUCTURE	X	X	X	X	X
	SEGMENTATION	X	X	X	X	X
	RANDOM ACCESS	X	X	X	-	X
	MACROs	X	X	X	X	X
IMAGE	RASTER ATTRIBUTES	X	X	X	X	X
	DEVICE-INDEPENDENT RASTER DATA	X	X	X	X	X
SYMBOLS (LIBRARY)	EXTERNAL INTERFACE	X	X	X	X	X
	USER-DEFINED	X	X	X	X	X
PATTERNS/ HATCHES	USER-DEFINED	X	X	X	X	X
	DR. CARSON'S RECOMMENDATIONS(2)	X	X	portions	portions	x

KEY

X = REQUIRES - DOES NOT REQUIRE OR IS NOT HIGH PRIORITY
? = GROUP WAS UNSURE OF REQUIREMENT
Portions = SELECTED PATTERNS/HATCHES FROM RECOMMENDATION

NOTES:
(1) "Compactness" was recognized as a general need for the applications designated, not necesarily falling under the scope of user requirements needing future additions to CGM.

(2) Carson, George S.,"Extending the CGM for Publishing and Technical Drawing Exchange," GSC Associates, Inc., 7 August, 1987.

FINAL REPORT

CALS SOW TASK 2.2.3.3.3

A REFERENCE IMPLEMENTATION FOR CGM: FUNCTIONAL REQUIREMENTS AND CONCEPTUAL DESIGN

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

CALS CGM Reference Implementation

I. PURPOSE

Plan for development of additional CGM conformance tests needed to validate software that generates and interprets (reads) metafiles (CALS SOW Task 2.2.3.3.3). The approach taken in defining these tests has been to develop a plan for a reference implementation for metafile generators and interpreters, or a piece of software capable of generating any legal metafile and capable of interpreting any legal CGM (Computer Graphics Metafile), including testing for the CALS Application Profile. In particular, this report provides a functional specification and conceptual design for this reference implementation, as well as how it might be used as a basis for CGM testing tools or as a model for a CGM test service.

II. BACKGROUND AND CONCEPTS

1.0 The Development of a Standard for a Graphics Metafile

1.1 A Brief History

Many computer graphics applications have a requirement for both the output of a picture onto a device and for the storage of the pictures in some way. This storage may be for a number of reasons including:

- long term storage of pictures;
- transfer of pictures to another machine;
- off-line spooled plotting where the picture files are queued;

The requirement for data storage of graphical images has been seen as a requirement during the development of graphics standards. The functional standards which have been developed (GKS, GKS-3D, PHIGS) all have the capability for the storage of graphical data and the subsequent inclusion of stored data into an application. The file in which the graphical data is stored has become known as a metafile. A metafile is created, or generated, by an application. It is then read back, or interpreted, into another application.

The functional standards recognize the need for the storage of graphical data in various environments. This is realized through the storage of the data in a metafile (for example, the GKS

Metafile, GKSM). The functional standards have the concept of workstations for Metafile Output (MO) and Metafile Input (MI) and they supply the functions providing access to, and interpretation of, metafiles. The functional standards do not, however, define a metafile format as part of the standard. Annex E of both GKS and GKS-3D suggest a format suitable for the storage of metafiles from the GKS and GKS-3D environments but these annexes are not a part of the standard.

Rather than develop the proposal of the Annex to the GKS standard, separate work was initiated in the area of a metafile for computer graphics. At the time this work was initiated in the standards arena there were requirements for a standard metafile which could not be met by the proposed metafile for GKS which is now found in Annex E to GKS. It was felt that a metafile had applications outside the GKS environment and that this constituency needed to be satisfied by the production of a standard in this area. This has resulted in the current standard for the Computer Graphics Metafile.

The GKS concept is of an audit trail metafile where the entire process of generating a picture is stored for future replay. While a picture is being interpreted GKS anticipates that the application may or may not choose to interrupt the replay with some further graphical output or input. In contrast, a metafile to the CGM standard captures a snapshot of the graphical image. Any elements which imply dynamic change to the image are not incorporated into the CGM standard. This was an intentional philosophical decision but can cause problems for GKS applications wishing to write metafiles to the CGM standard. The relationship between the CGM and GKS is considered further by Brodlie, Henderson and Mumford (1987).

It should be remembered that the CGM is only concerned with the storage of graphical data. It does not store information about the structure of the picture which it comprises. There are no possibilities in the standard for reconstructing the way that the picture was built. The CGM does not store any other information concerning the picture such as product data. Such information may be stored as APPLICATION DATA, but has to be done in a non-standard way.

1.2 The CGM Standard

1.2.1 The Roles of the Standard

The CGM standard has two distinct roles. The first is to define the functions which need to appear in the metafile and to lay down rules as to the structure of the metafile and the order and position of the various elements. This part of the metafile

standard is defined in Part 1 of the CGM standard. The second role is to define the way that these functions are recorded in the metafile. This is known as the encoding of the elements defined in the functional part. Parts 2, 3 and 4 of the CGM standard are concerned with the encoding of the elements whose abstract functionality is described in Part 1. The CGM also contains a formal grammar in an annex to Part 1 which describes in detail the behavior of the elements.

Some details of the structure and encoding of the CGM standard which are necessary to understand the proposals made for the Reference Implementation are discussed below.

1.2.2 The Functional Specification

A metafile is a collection of elements. These elements may be the graphical primitives such as polyline, polygon or attributes, such as line color, which describe the graphical image, or may be information to the interpreter about how to interpret a particular metafile or a particular picture. The CGM standard specifies which elements are allowed to occur in which positions in a metafile.

The CGM standard defines the following classes of elements:

- **Delimiter Elements**, which delimit significant structures in the metafile.

- **Metafile Descriptor Elements**, which describe the functional content, default conditions, identification, and characteristics of the CGM.

- **Picture Descriptor Elements**, which set the interpretation modes of attribute elements for each picture.

- **Control Elements**, which allow picture boundaries and coordinate representation to be modified.

- **Graphical Primitive Elements**, which describe the visual components of a picture in the CGM.

- **Attribute Elements**, which describe the visual components of a picture in the CGM.

- **Escape Elements**, which describe device- or system-dependent elements used to construct a picture.

- **External Elements**, which communicate information not directly related to the generation of a graphical image.

A metafile conforming to the CGM standard is a collection of elements from the sets in the list above. The permissible relative positions of the elements follow rules defined in the abstract syntax. These relative positions can be indicated via the use of states which are defined in the standard. The states which are recognized are:

- **Metafile Closed State** which is prior to any elements being written
- **Metafile Description State** in which Metafile Descriptor elements may appear
- **Picture Description State** in which Picture Descriptor elements may appear
- **Picture Closed State** which is prior to beginning a picture
- **Picture Open State** which follows the opening of a picture and in which Control. Graphical Primitive and Attribute elements may appear in any order
- Partial Text State which is between calls to text primitives where strings are incomplete between calls

Escape and External elements can appear in any state.

These states comprise a useful concept for defining where elements may appear. These states will be used later in this report where the conceptual design is considered.

1.2.3 The Encoding of the CGM

Requirements of the Encodings

There are different requirements for data storage and transfer. This is not necessarily specific to the graphics community. These requirements include:

- minimal file size;
- ease of transfer across networks;
- the speed with which the data can be generated and interpreted;
- the readability of the stored files.

It is not possible to give equal weight to these requirements, and the choice must depend on the application. For some environments it may not be necessary to transfer the stored data to other machines; in another environment it may be advantageous to be able to edit the graphical data which is stored, in which case readability becomes important. To address these different requirements the CGM defines three encodings, namely the character, binary and clear text encodings contained in Parts 2, 3 and 4 of the CGM standard, respectively. These encodings are described briefly in this section.

The Character Encoding

The character encoding is found in Part 2 of the CGM standard. The main concerns of this encoding are to ensure that the encoding is compact, and to guarantee ease of transfer across networks between machines. To achieve this second aim, the character encoding is made up of only the Ascii printing characters. Each element is coded as an op-code followed by the data associated with the element.

The Binary Encoding

The binary encoding is found in Part 3 of the CGM standard. The emphasis of this part of the standard is on ease of generation and interpretation of the CGM. For this reason the storage of the graphical data is in a form which is easily written and translated on most computer systems. Although compactness was not the primary consideration when this encoding was being developed, the encoding should not be seen as inefficient in its storage. Many applications have found that there is not a significant storage overhead in using this encoding rather than the character encoding. The binary encoding does use a format which may cause difficulties when transferring the data between machines which do not adhere to the developing networking standards. To date though, this has been the most popular encoding, and it has been adopted for the CALS and MAP/TOP Profiles.

The Clear Text Encoding

The clear text encoding is found in Part 4 of the CGM standard. The data which are stored in a clear text-encoded CGM are human readable. This allows editing of the metafile, which may be useful in environments where editing is more beneficial than minimizing file size. A translation of a metafile in one of the other encodings into the clear text encoding may facilitate debugging of an invalid metafile.

Private Encodings

The CGM standard specifies the abstract functions independently

of the encodings. This means that a CGM can be written in a private encoding while adhering to the principles laid out in Part 1 of the CGM standard. This may have limited application, since there may be insufficient interpreters for such an encoding.

1.3 Conformance of the CGM

The conformance statements in the CGM standard relate to the conformance of a metafile. They do not refer to the conformance of the generator or interpreter. There can be no expectation that a metafile sent to an unknown interpreter will be understood by that interpreter. Groups of users, such as CALS and MAP/TOP users, are concerned about this, and are trying to reduce the problem.

The CGM standard defines two levels of conformance: full conformance and functional conformance. Full conformance occurs when a metafile conforms to the abstract functional specification of Part 1 of the CGM standard, and also uses one of the three standard encodings. Functional conformance of a metafile occurs when a metafile conforms to the abstract functional specification, but a private encoding is used.

Thus, the standard is very limited in its conformance requirements. This should be remembered when CGM software is purchased, since there can be no guarantee of minimum support by generator or interpreter software.

2.0 Using the CGM

2.1 A Flexible Tool for Graphical Data Storage

The CGM standard offers flexibility for the software supplier wishing to use the standard as a means of storing graphical data. The range of choices implies that there is some danger of the CGM being little better than a proprietary product, depending on the options chosen. For this reason it is advantageous if groups of people wishing to transfer metafiles all use the same subset of the CGM options to guarantee the successful exchange of graphical data. It is for this reason that the MAP/TOP Application Profile and then more recently the CALS Application Profile have been developed. These Application Profiles state the elements which are legal for that Profile and the element ranges to which all software will adhere. These Application Profiles specify a maximum requirement for generator software and a minimum for interpreter software. They also allow the possible use of registered and private Escapes, GDPs and other registerable items.

The CALS and MAP/TOP Application Profiles are now close to final approval; however, there are other possible groupings of elements which may emerge. The various possibilities for application profiles are discussed below.

2.2 Application Profiles

2.2.1 MAP/TOP

The CGM is to be included in MAP/TOP V3.0 due to be published during 1987, and it defines the most appropriate specification of the CGM for the MAP and TOP community. The Profile chooses the binary encoding, and restricts the encoding to the long form for command headers and strings. The defaults chosen are mostly in line with those specified in Parts 1 and 3 of the CGM.

2.2.2 CALS

The CALS Application Profile has been developed under a separate NBS task. It considered the MAP/TOP Profile and made changes based on the CALS requirements. Fonts, line styles and hatch patterns are important areas where a profile designed for another community may be insufficient, and these are defined in the Profile for the CALS projects. The work on the Profile also suggested that consideration needs to be given to allowing coding techniques beyond those of the MAP/TOP Profile. This could involve an implementation burden for writers of interpreter software and testing software. In the first instance it is recommended that the binary encoding should be the only one specified. The Reference Implementation described in this report is, however, designed with the possibility of being used as part of other software to write and interpret metafiles. This may lessen, or at least share, the burden. The user facilities described in this report and the testing software are applications on the underlying core of generator and interpreter software.

2.2.3 Other Profiles

ODA/ODIF

A standard for the Office Document Architecture (ODA) and for its transfer format the Office Document Interchange Format (ODIF) is being developed in the standards arena (ISO DIS 8613). The standard recognizes the need for mixing text and graphics, and the CGM is used for graphical data storage in a document. The graphical storage is described in Part 8 of the standard, and is

known as the Geometric Graphics Content Architecture (GGCA). It uses the binary encoding of the CGM with limited modification of the default rules. However, the majority of the defaults chosen are those detailed in the CGM standard. The metafiles are complete, but only contain a single picture, and do not use the Escape or External elements of the CGM. The relationship between the CGM virtual device space and the ODA basic layout object is also described.

Metafile Categories

The CGM is now being extended via addenda being developed within the standards arena for further 2D and 3D support. The addenda are seen as defining categories which limit the elements available, the behavior of the elements, and may limit the parameters available to a category. In effect, these too are application profiles for use in particular environments, for example, a GKS environment.

CGM Shorthands and Defaults

The CGM currently includes two shorthands for groups of elements which can be used in the list of elements which are in a particular metafile. There are also defaults both for the abstract functional description of part 1 and for the three encodings. These could also be seen as being profiles, and it may be useful to know if an implementation includes the defaults as a minimum, or whether it supports the shorthands described in the standard.

2.3 Summary

Application profiles of one kind or another are certainly gaining momentum. These may be formal ones such as the CALS Application Profile or may be just groups of useful elements or defaults. As metafile categories emerge in the addenda, so the concept will expand. The concept is also used in the emerging Computer Graphics Interface (CGI) standard. The incorporation of the concept of application profiles into the Reference Implementation for the CGM and any associated applications and utilities is vital. This report makes considerable use of the application profile concept. The ideas included may be extensible to other graphics standards which include profiles, such as the CGI.

III. DISCUSSION

1.0 Content and Structure

This report is consideration of the requirements for, and the implementation of, a Reference Implementation of the current standard for the Computer Graphics Metafile (CGM) (ISO 8632, ANS X3.122-1986, and FIPS 128). The report looks at the functional specification of such an implementation and the conceptual design of the software, and considers the way that this could be used for testing purposes.

This section of the report contains two parts:

1. The functional specification of a Reference Implementation for the CGM (Section 2 below);

2. The conceptual design of this Reference Implementation (Section 3 below).

2.0 Functional Requirements of a CGM Reference Implementation

2.1 The Need for a Reference Implementation

The CGM standard offers a useful method for the storage of graphical images. It defines a wide range of options which can be used by the generating software. These options include, for example, the precision of the data which is stored and the way that color is defined within the metafile. There is, however, no guarantee that the interpreting software will be able to make any sense of the metafile. The standard does not specify the behavior of generators and interpreters, and this makes it difficult for the purchaser of CGM software to guarantee that the software for generating and interpreting metafiles is what is required. Application profiles, such as the one designed for CALS, attempt to limit the use of the standard. Those parties who use the CALS Application Profile should be able to predict the behavior of the generator and interpreter software. The metafiles written by one CALS application can thus be guaranteed to be understood when transferred within the CALS environment.

It is important to ensure that the CALS Application Profile is adhered to by software purchased for the CALS effort. Therefore, it is necessary to offer some form of testing service for software to ensure that software does conform to the standard and to the CALS Application Profile.

In order to accomplish this, it is necessary to develop a CGM implementation capable of generating and interpreting metafiles

to the CALS Application Profile. Such software can be used in the testing of CGM implementations. In the long term, however, it may be too limiting to develop software solely for the current CALS Application Profile for a number of reasons which include:

- The CALS Application Profile may be extended to take into account the future needs of the CALS community.

- The Profile may be extended in the future to include those elements which are now under development within ANSI and ISO to extend the CGM for further 2D and, eventually, 3D support.

- There may already be requirements in CALS to use other profiles where appropriate. This might include the MAP/TOP profile. Also, there may be a need to adopt the limits imposed by ODA.

To limit the CALS implementation to the CALS profile is therefore too great a restriction both today and for the future.

There is also a requirement in the United States and elsewhere for a general CGM testing tool. This was discussed at a meeting in Disley, UK in March 1987. Representatives from the National Bureau of Standards attended that meeting. CGM testing was also a major discussion area at the NBS/Eurographics workshop on "The CGM in the Real World" held at NBS in September 1987. The ideas from those meetings have been incorporated and developed in this report where appropriate.

The needs of CALS and the general requirement for CGM testing means that this project should not be too limited. Many of the decisions being made for the CALS implementation are also relevant for an implementation with wider use. Therefore, the software discussed in this report will take a broad view. It discusses a reference implementation which is capable of generating and interpreting any metafile. The software also takes into account the need to test application profiles, and will need to be extensible to allow the inclusion of other application profiles and further standard metafile developments.

The **Reference Generator** - referred to in the rest of the text as the **Generator** - must be capable of creating any legal metafile. This will allow the user of the Generator to put together a metafile suitable for use on a particular implementation of CGM interpreter software. The Generator will be able to restrict a metafile to conform to a particular application profile.

The Reference Interpreter - referred to as the Interpreter in the text below - will be capable of understanding any legal metafile. It will be able to draw the results via a suitable graphical interface. The Interpreter will also be able to attempt recovery

from an incorrect metafile and to continue processing the rest of a metafile following an error. The Interpreter will be written to allow tracing of the metafile content and possible consideration of efficiency of storage. The Interpreter will also be capable of being restricted to an application profile to check adherence of generator software in a system under test to known profiles. The mechanism used for testing profiles will be extensible to allow further profiles to be added in the future.

The development of the Generator and Interpreter software described above form the basis for developing a testing service for the CGM. Test metafiles to a particular specification can be interpreted on systems under test. Similarly, metafiles from the system under test can be interpreted and analysed by the Interpreter. If there is a requirement for conformance to a particular application profile, this can also be tested. Creating the CGM software as a reference implementation will provide a flexible tool for building a test utility.

2.2 Some General Requirements for the Reference Implementation Which Apply to the Generator and Interpreter Software

2.2.1 The General Structure of the Software

The core software of the Reference Implementation will be implemented as a series of routines/procedures which may be accessed via an application. This software will give access to the CGM elements for both the Generator and Interpreter, and will also include other procedures as necessary to handle the encodings.

Applications will be developed above the Reference Implementation for simple interactive generation and interpretation of metafiles. These applications will enable metafiles to be generated and interpreted to particular specifications. Such specifications include generating and interpreting a metafile within the definition of an application profile such as the CALS Application Profile. A further application for a testing environment is discussed in the Recommendations section of this report.

This layering of the software into a core of software with access at the CGM element level presents a useful, general model for the implementation. This part of the report considers the functional requirements, from the viewpoint of the user, of the CGM generator and interpreter applications. This approach is taken because these requirements must be reflected in the conceptual design of the software in section 3 below.

2.2.2 Some General Considerations for the User Interface

Applications should be developed to allow straightforward access to the Reference Implementation. The user interface should be consistent between the Generator and Interpreter software. To allow experienced, inexperienced and casual users easy access to the Reference Implementation, there needs to be a number of ways that the software can be used:

1. as an interactive session with prompts where needed;

2. as a command driven session for the experienced user;

3. driven by a script created by an editor or by a previous session;

4. as (1) or (2) above but creating a script to feed into (3).

It is beyond the scope of this report to consider the implementation of the user interface in detail. This report discusses the information which comes from this user interface layer, and specifies the user interface in very general terms. It is suggested that satisfying user interface requirements by one of the user interface managements systems (UIMS) available on the market at the time should be investigated. The use of an UIMS could save development effort and would result in a consistent interface across testing tools for the CGM.

2.2.3 Error Handling

The error mechanism designed must be usable from both the Reference Implementation core software and from any applications developed using the Reference Implementation. The applications must also be able to control the level and output of the error information. The behavior of the application on the receipt of an error also needs to be controlled.

2.3 Detailed Requirements for the Reference Generator

2.3.1 Level of Application Access

The Reference Generator will be capable of generating any legal metafile in any of the three encodings. This section of the report considers the functionality needed within the software to achieve this. The way that these requirements will be implemented is considered in section 3 below. The procedures must be written in a way which is compatible with access at the CGM element level. Access also needs to be available to those options required to specify a particular encoding.

The Generator must be able to account for the following factors when creating a metafile:

1. any limitations imposed by the choice of an application profile;

2. the choice of encoding, assuming this is not limited by (1);

3. compulsory elements in the CGM;

4. elements required by the user creating the metafile;

5. the ability to pack the data as a faithful audit trail of the user requests as an alternative to efficient buffering of the graphical data.

These requirements are considered in more detail in the next section.

2.3.2 Generator Options in Choosing an Application Profile

The generator code will be configured to allow tailoring of the software to suit a particular application profile. Such a mechanism must at least be capable of dealing with the current MAP/TOP and CALS profiles, and it will be an extensible one.

When using the generator utility to create a metafile, the user will not be allowed to create a metafile which does not conform to the requirements of the application profile selected.

Other choices for application profiles which could be incorporated are the ODA/ODIF defaults and the CGM default situation. The user must also be allowed to simply create a conforming metafile which complies with the CGM standard.

2.3.3 Generator Options in Choosing an Encoding

General Considerations

The user of the generator utility will be able to select the encoding if not restricted by the application profile. In the case of the MAP/TOP Profile, only the binary encoding is allowed, and currently, the use of that encoding is further restricted. For this profile there are no choices to be made with regard to the encoding. In other cases, such as for full CGM conformance, all three encodings need to be selectable.

This section considers the encoding-dependent information which needs to be collected in order to generate a metafile. This does not include elements which are common to all encodings.

Character Encoding

There are three pieces of information which need to be gathered in order to generate a metafile in the character encoding.

1. It is necessary to know whether the metafile is allowed to use both incremental and displacement modes to store point list. Displacement mode should only be selectable by the user, since it is possible that the incremental mode will not have been implemented at all sites.

2. Information regarding the use of character substitution is also required. This is stored in the metafile descriptor and allows certain characters to be substituted in the metafile to make file transfer easier. These characters may occur in strings and are the non-printing characters, space, tilde and delete. The substitution is of a 2-byte sequence in place of the single byte.

3. The format of the color lists is also needed. Color lists may take a number of forms and any limits imposed by the application profile must be known to the Reference Implementation. The formats are: normal; bitstream; runlength; runlength bitstream.

This information should be obtained by the generator utility once the character encoding has been chosen.

Binary Encoding

Again, assuming that there is any flexibility given by the application profile, there is information to be collected which is pertinent to this encoding. This information includes:

1. Whether the application profile limits the encoding of the command headers and strings to the long or short form;

2. Whether the CELL ARRAY color values can be stored as run length representation and packed representation.

Clear Text Encoding

For the clear text encoding it is necessary to know whether the application profile restricts the following:

1. Whether both UNDERSCORE and DOLLAR characters can be used as null characters;

2. Which format effectors are allowed from the total list (BACKSPACE, CARRIAGE RETURN, LINEFEED, NEWLINE, HORIZONTAL TAB, VERTICAL TAB, FORMFEED);

3. Whether both SEMICOLON and SLASH can be used to delimit elements;

4. Which SOFTSEP characters are allowed;

5. Which HARDSEP characters are allowed;

6. Any limits on the bases of integers;

7. Whether reals can be written as explicit point numbers, scaled real numbers and decimal integers;

8. Whether both single and double quotes are allowed to delimit strings;

9. Whether both absolute and incremental point lists are allowed.

2.3.4 Elements in the CGM

These will be accessed by applications at the CGM element level. The application should be forced to select elements appropriate to the current state of the CGM. For example, Metafile Descriptor Elements cannot appear in the state Picture Open. These CGM states are defined in the CGM standard. Any attempt to cause the Generator to create an illegal metafile will cause the error mechanism to come into play.

2.4 Detailed Requirements for the Reference Interpreter

2.4.1 General Considerations

The Reference Interpreter must be able to interpret any legal metafile. The output from this interpretation may take a number of forms, including:

- graphical output;
- trace output;
- evaluation output.

The requirements for these different forms of output are discussed in this section.

The interface to the Reference Implementation will be independent of any application, and will allow the application to look for the next element in the metafile, and to interpret it or skip it. It is proposed that the model for this be based on the GKS metafile functions to get, read and interpret metafile elements.

2.4.2 Graphical Output

The main purpose of a metafile is to store pictures. It is important, therefore, that an interpreter facility should be able to draw the picture which has been stored. This facility should also allow the user to select a number of options, including:

1. Choice of the output device. The user will need to select the output device required. To allow a range of devices to be used, the software must ensure that the devices can be extensible. This can be done by fitting the application on top of a graphics package and designing it to ensure that it is at least suitable for the range of functional standards in the graphics standards arena.

2. Choice of the viewport in which the picture will appear when drawn.

3. Choice of the level at which the picture is to be rendered. The user may have different requirements for rendering at different times. These can be classified into two groups, called preview and publication quality output.

   The user of the Interpreter may not always be concerned to get the size, colors, font types, etc., correct all the time. The user may simply wish to examine the picture in general terms to ensure that the rough outline of the picture is there and to draw it quickly. But when the picture is drawn for actual use in a document, it is necessary to ensure that the rendering is correct. The size of the final output will need to be correct, either to that specified in the metafile or to a user defined scaling of the virtual coordinate space. Fonts, line styles and other attributes will also need to be correct. For this reason two levels of quality should be available to the user interpreting a metafile and producing graphical output. The capability, or otherwise, of the interpreter to render the picture to publication quality will be handled by the error control mechanism. This is important for CALS where accurate representation of the final output picture is essential in many cases.

2.4.3 Trace Analysis of the Metafile

This comprises text output of the metafile which describes the content of the metafile to the user. Details of the metafile can be output at different levels chosen by the user. The user will have control over viewing the metafile at the following levels:

1. As a directory structure which lists the metafile delimiter elements and the pictures contained within the metafile;

2. As a whole metafile, on a picture-by-picture basis, or as a trace of the descriptor sections (These options are not mutually exclusive.);

3. As a trace of the elements within sections defined by (2);

4. As a trace of the elements and parameters within the sections defined by (2).

The analysis should ensure that the syntax is correct at the level being considered. There should also be the capability within the design for checking that the metafile conforms to a selected application profile. The trace should include information concerning any errors at the place where these have occurred, following the error model designed for the Reference Implementation.

The output must be easy to read. A suitable form of output is an extended form of the clear text encoding. This would need to be extended, since there are elements required by the other encodings and also encoding-dependent parameters.

2.4.4 Evaluation of the Metafile

The graphical output and analysis of the metafile described above are concerned with ensuring that the metafile conforms to the CGM standard and, where appropriate, to an application profile. Clearly, this is necessary, both for rendering the metafile and for testing purposes. Also, there is other information which can be considered evaluation rather than conformance testing. Examples of such evaluation might be to consider how an implementation has made use of compaction features of the metafile. While software may conform to the standard, there are efficiencies which can be gained by the implementer. This is not as important for CGM software compared with GKS implementations, but it is a valid consideration.

A simple example of where efficiencies can be gained is in the storage of the attributes. This relates to whether the software gives a faithful audit trail or whether attributes are buffered and only output when necessary. The following sequence does not

need to be stored in full, for example:

- line color red;
- line color green;
- line color blue;
- polyline.

Clearly, it is only necessary to store the color blue. Many implementations write the attributes when required by the primitives. Checking for this is a useful evaluation of the software as considerable savings in space can be made by buffering attributes. Other evaluation might include:

- The use of the Metafile Elements List and a comparison of the list with the actual elements used;
- The picture sizes within the metafile;
- The lengths of the variable length elements, for example POLYLINE;
- The element point-to-point displacements stored on the metafile;
- The distribution of the elements that have been used;
- The ESCAPE identifiers which have been used;
- The GDP identifiers which have been used;
- Any encoding dependent information, such as the use of the long/short form command headers and string in the binary encoding.

2.5 Summary

This part of the report has considered the functional requirements for the Reference Implementation and some associated applications for generating and interpreting metafiles to the CGM standard. The use of application profiles, such as the CALS Application Profile, means that the Reference Implementation forms the basis for a number of applications, including testing to the standard and to application profiles. It is recommended that the software take a broad view and not be restricted to the current CALS Application Profile. Both current and future requirements for CALS make this undesirable.

3.0 The Conceptual Design of the Reference Implementation

3.1 Introduction

This part of the report is concerned with the way in which the Reference Implementation should be realized. The design is at a general level and does not indicate the specific way that such software should be written. However, all general design criteria are covered. The production of a reference implementation provides a potentially flexible tool for many applications. It is the aim of this design to ensure that future applications and uses of the CGM will not be overly restricted by the proposals contained in this report.

Since there are a number of general design criteria which apply to both the generator and the interpreter parts of the Reference Implementation, these are considered first. The discussion below then turns to the specific requirements of the generator and interpreter software.

3.2 General Design Criteria

3.2.1 The Nature of the Software

A major concern of the Reference Implementation design is to ensure that applications can be developed which can sit on top of the implementation. These applications will include utilities for generating, interpreting and testing metafiles to the CGM standard and to application profiles. A further, desirable design criteria is not to preclude the inclusion of encodings beyond those specified in the standard. Testing has been carried out at a number of sites for other test requirements, such as for languages and GKS testing. This gives a further requirement, namely, for the writing of the Reference Implementation in a portable way.

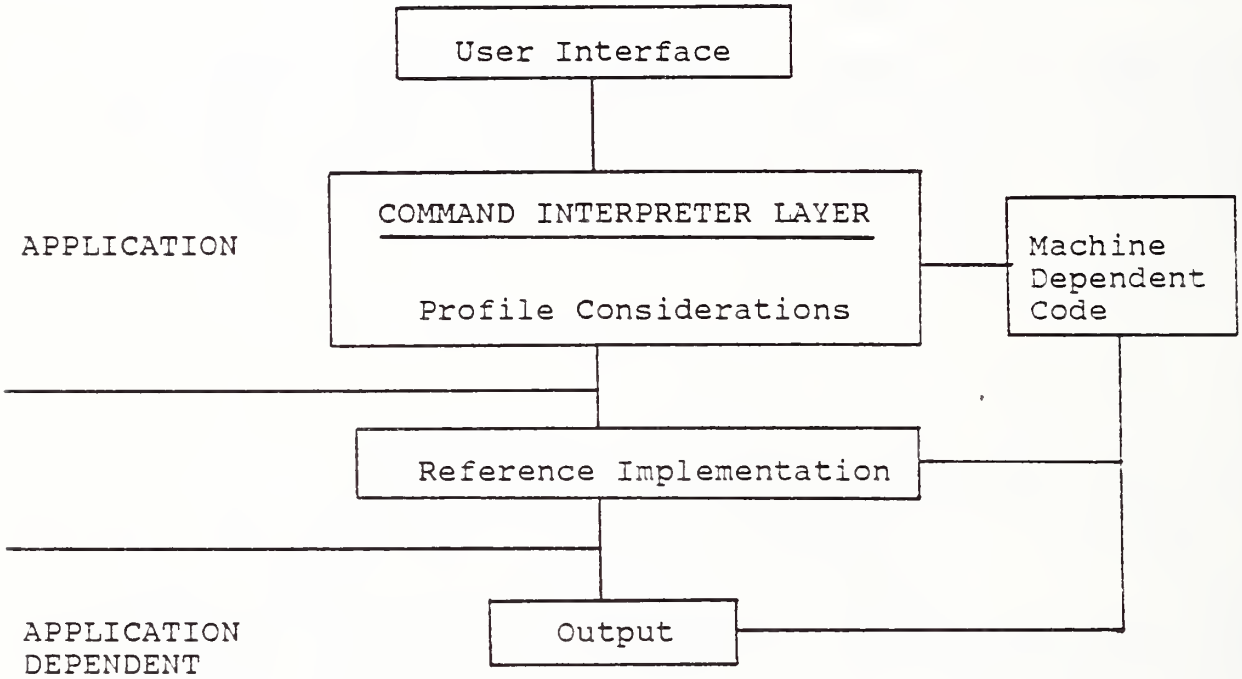
These design criteria will be met by having an implementation which is modular and has a layered design. There must be clean breaks between the layers with the data flowing between the layers clearly defined. It is recommended that the software be written in Fortran77, since this is considered to be the most widely applicable language in the environments where the Reference Implementation and testing tools will be used.

The design specification presented in this report will define the layers of the software which are required for the Reference Implementation.

3.2.2 General Structure of the Software

The design of both the generator and interpreter software have similarities in terms of their structure. This can be most easily examined via the diagram shown in Figure 3.1.

Figure 3.1: The Structure of the Generator and Interpreter in General Terms



The structure outlined in Figure 3.1 will be used as a general model for both the generator and interpreter parts of the Reference Implementation and associated applications. The applications sit above the Reference Implementation and interact with the user via the User Interface Layer where this is required by the application. The Command Interpreter Layer includes a profile consistency check. This division is made to allow other applications to use this layer of code, and such general usage is a design objective. The machine-dependent code is clearly split from the rest of the code to enable portability of the Reference Implementation and any applications, and is available to all layers.

3.2.3 The User Interface

The functional requirements for this task imply that the Reference Implementation must be an application-independent implementation, so that applications can be developed to use it. It is recommended that a detailed design of these applications should consider using a UIMS available on the market at that time. It is beneficial to ensure that all CGM applications developed for the CALS requirements have the same user interface. Other applications in the CALS environment might also benefit from standardizing the user interface. Any further consideration of this aspect is beyond the scope of this report.

3.2.4 Using Application Profiles

The use of application profiles is vital to most of the software proposed in this report. However, application profiles should be envisaged as a subset of elements and parameters used for a particular application. Examples of application profiles important to particular communities are the CALS and MAP/TOP profiles. If a suitable mechanism is used, the concept of application profiles can be expanded to give a more general capability. A more general facility would ensure that: future application profiles could be developed and incorporated; testing could be carried out on a particular implementer's profile; and testing could be carried out to see if the CGM defaults are handled by a particular implementation.

To ensure that this is an extensible mechanism, it is necessary to build the information outside the main body of the code. There are two methods for doing this. The first is to build a data file containing the information required - for example, which elements are allowed, in what order, what are the defaults, and so on. The second method is to have a routine in the software containing the information (rather than a database). This routine might set up data values in the generator and interpreter code. Such code would look like a typical graphics device driver, where only those entries available within the application profile are set, and the rest are indicated to be unavailable.

It should be ensured that such data files and code can be extended in the future and new ones created. There should be a utility to create the database or code for a new application profile. For this reason the database option has been chosen, sice it is easier to create.

Therefore, an important feature of the Reference Implementation is a configuration database which contains all the information necessary to deal with the concept of the use of limited subsets

of the CGM. These subsets may be accepted application profiles or may be other subsets - for example, a subset used for testing purposes. This data file needs to contain the following information:

- application profile name - as described in the metafile;
- encodings allowed;
- encoding dependent information;
- which elements are allowed to appear in the metafile;
- where in the metafile these elements may occur;
- limits on the parameter values;
- limits on the size of the variable length elements (e.g. polyline);
- which registered items may appear in the metafile (ESCAPEs. GDPs).

The data file should be able to be created via a utility. However, it should be readable and thus able to be edited. Since abbreviations exist for the CGM elements via the clear text encoding, they should be used as the basis for defining the data file format. This will need to be extended to allow for encoding options, ranges of values and state information. But it does give a useful and currently defined basis from which to work. The configuration database must be extensible to accommodate new application profiles and further developments in metafile definition via the CGEM work. These can be accommodated by mandating that any elements appearing in the database indicate whether they are allowed. Those which do not appear are assumed to be illegal for that application profile.

It is outside the scope of this task to define the precise nature of the database for application profiles. Table 3.1 shows a suitable format which could be adopted. This shows that there are two forms of data: first, a general application descriptor; and second, an element-by-element list. The form of this should be similar to any script which is used as input to the Command Interpreter Layer by the user interface. The inclusion of state information allows the database to be extensible as further states are added (for example, segment and figure states) and as elements move from one state to another between application profiles. This means that the model proposed here will be extensible for the CGEM work under development.

Table 3.1: A Proposal for the Structure of the Application Profile Database

```
BEGIN AP HEADER
     AP NAME
          (name)
     AP ENCODING
          (encodings allowed)
          (encoding dependent parameters)
END AP HEADER
BEGIN AP ELEMENTS
     ELEMENT  (name)
          STATUS
               (allowed/not allowed)
          STATES
               (states in which element is allowed)
          RANGE
               (ranges or permissible values)
          LENGTH
               (length if this is a variable length element)

     .......other elements.........

END AP ELEMENTS
```

This configuration database will use the element names and any other abbreviations which have been defined for the clear text encoding.

3.2.5 Error Handling

The actual errors which may be produced by the system are a feature of the detailed design of the software. The discussion in this report is concerned with the general error model to be used.

The current (December 1986) draft of the CGI document gives a useful basis for the error model. It is recommended that the detailed design should use Parts 1 and 2 of the CGI as an input to the design process. The model discussed below is based on the current CGI proposal.

The Reference Implementation will maintain reports of errors that have occurred in a last in, first out (LIFO) data structure known as the error stack. The application may read and remove errors from the stack. The application may also clear the stack.

To fulfill the functional requirement that the error model must apply to the Reference Implementation and to any applications, the error stack needs to be made available to applications. A routine must be available to allow an application to write to the stack. This routine should be modelled on the CGI function for extracting errors from the stack. Thus, the application has full control of the level of error reporting which is appropriate for that application. A testing environment may choose to report more error information than other environments.

The CGI error classes are appropriate for adopting into the error model for the Reference Implementation. To prevent duplication of work and to give consistency with future CGI implementations, it is suggested that the CGI document be used as the basis for the definition of error messages and values when the detailed design is carried out. To allow for changes and to permit extensibility, it is recommended that the errors should be parameterized within the software.

3.2.6 Data Structures

The following data structures will be required by the Reference Implementation for both the generator and interpreter software.

- the application profile configuration database;
- the error stack plus information as to the current error state and processing directions;
- error message database;
- state information as to the current state and the requested state;
- element op-code tables and mappings to an encoding independent form together with the state information for each element;
- encoding dependent information-one structure per encoding containing the information discussed earlier in this report;
- operand and I/O buffer space.

The generator also requires information on whether the metafile is being written in audit or buffered-attribute mode. It also needs a parameter-contents table for each element to allow computation of the element length for the binary encoding.

The interpreter needs to store information concerning the type of output being carried out and the quality of that output if it is

graphical. It is also necessary to keep a statistics table if evaluation is to be carried out.

3.2.7 Machine Dependent Routines

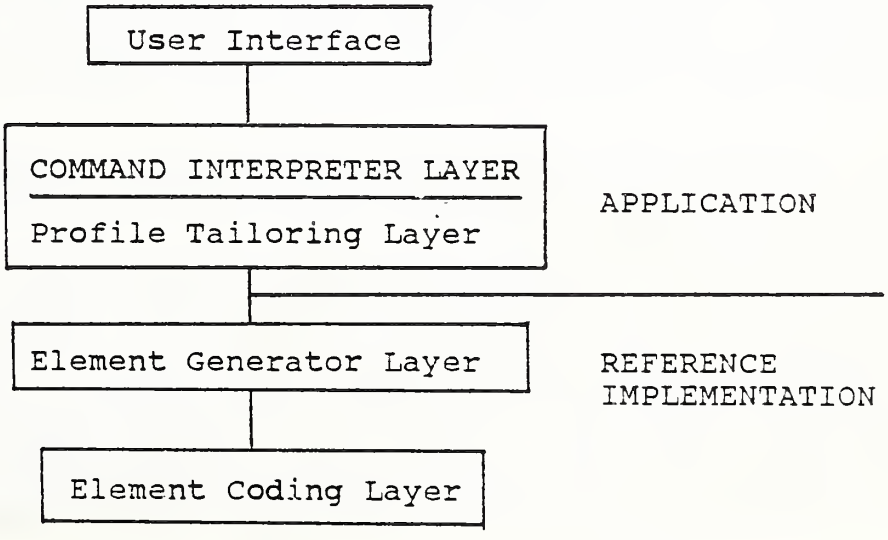
The code will be split into machine-dependent and machine-independent layers. To ensure portability of the code, the distinction between these two layers will be clear and well documented. The number of entries into the machine-dependent layer will be kept to a minimum. These entries will handle: files, where necessary beyond the Fortran77 standard; system information, where needed by the application layers; and utility routines, for bit shifting, setting, extraction and comparison. These routines will be available to all layers of an implementation; that is, both the application layer and the Reference Implementation. They will be common across both the generator and interpreter code. The machine dependent routines are not shown on the diagrams in the next sections below, which consider the Generator and Interpreter in more detail.

3.3 The Reference Generator

3.3.1 General Considerations

The structure of the Reference Implementation and the application required for generating metafiles is shown in Figure 3.2.

Figure 3.2. The Structure of the Generator Reference Implementaion and its Associated Application



The user interface and the general method used for handling application profiles has been discussed earlier. This section will consider the concepts involved in this structure which are applicable to the generation of metafiles.

3.3.2 Command Interpreter Layer

The Command Interpreter Layer is a part of the application above the Reference Implementation. This layer handles the application information which is required and requests information from the User Interface Layer relevant for the particular application. Any data conversion - for example, for coordinates, from the application units to metafile units, must be done in this layer. The Reference Implementation only accepts these units to be stored on the metafile.

An important sub-layer is the Profile Tailoring Layer. This ensures that the data are valid within the constraints of a particular profile. The advantage of separating this as a sub-layer is that other applications could make use of this layer, since it is at the level of the CGM elements and encoding dependent information. Elements which are allowed within a profile are passed through to the Reference Implementation below. The error handler is used to deal with elements outside the profile and appropriate action is taken to inform the layer above of the error.

A role which could be played by the Profile Tailoring Layer is to simulate requests from the user into CGM elements which conform to the application profile and are within the limits of the system. Suppose, for example, that the maximum length for a polyline is 1000 points. A user requesting a 2000 point polyline could have this stored as two polylines. This would ensure that the application profile was adhered to while allowing the user to transfer the picture required. Although this tailoring is useful, it is suggested that its implementation should be secondary to that of the core of the Reference Implementation.

3.3.3 The Reference Implementation for the Generator

Element Generator Layer

This layer is independent of any encoding which may be used. The reason for taking this approach is that a private encoding could be added if it was construed as useful in the future. This layer will contain one routine per CGM element, and will include all elements specified in Part 1 of the CGM plus any others required for the encodings (for example DOMAIN RING). This layer is envisaged as having a similar level of access to the emerging CGI

language bindings. When the detailed design is carried out, the CGI language bindings must be considered as input to the design work. However, it is essential that the binding should allow partitioning of the elements, since there are elements with potentially a large amount of data (such as cell array).

This layer also maintains all the state lists, attribute tables and color tables that are required by the Generator. These have been discussed above under the section on data structures. It also handles the use of the audit or buffered mode for output of attributes to the metafile.

Since the proposed language for this implementation is Fortran77, it is necessary to consider the constraints of passing parameters to the encoding layer below. The parameter types will vary depending on the descriptor elements selected. To make the code simpler it is proposed to convert the parameters to a canonical form. The parameters will be passed to the layer below as real and character data. There will only be a limited number of entries to the encoding layer below.

The binary encoding also requires the element length to be passed to the Element Coding Layer. This information will be obtained in this layer for both fixed and variable length parameter lists via a function.

Element Coding Layer

This layer converts the data passed from the Element Generator Layer into the appropriate encoding and writes the CGM. This layer will have entries for the following types of data to be output to the metafile:

- op-codes which handle the type of element and any encoding dependent information which needs to be stored for the op-codes (e.g. data length in the binary encoding);
- scalars which handle most of the elements with fixed lengths;
- lists for elements which have variable length lists and where the encoding may vary from scalars;
- strings which require particular handling.

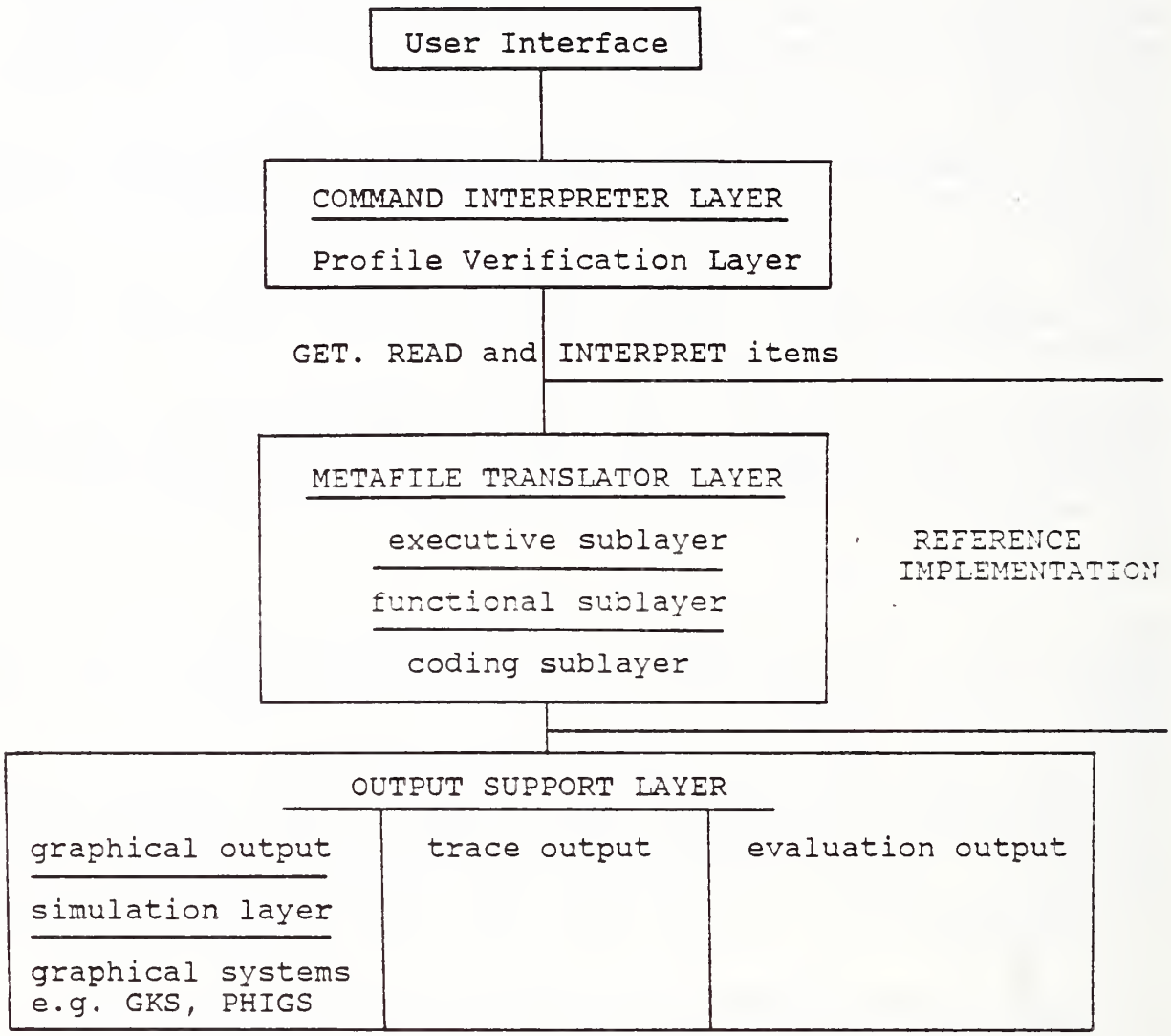
3.4 The Reference Interpreter

3.4.1 General Considerations

The interpreter has a similar structure to the generator which

has been discussed above. The Reference Implementation for the Interpreter and the general design of the interpreter application is shown in Figure 3.3.

Figure 3.3: The Structure of the Reference Implementation Interpreter and and Interpreter Application Layer



The requirements of the Interpreter will be obtained from the User Interface Level to ensure that all necessary data has been collected prior to the processing of the metafile by the interpreter code.

The overall organization of the code is a division into three major parts whose requirements are served by the user interface

above. These parts are:

- Command Interpreter Layer;
- Metafile Translator Layer;
- Output Support Layer.

The Reference Implementation is contained in the Metafile Translator Layer, is application-independent, and also independent of the type of output. These layers are considered in more detail below.

3.4.2 Command Interpreter Layer

This layer performs a number of tasks using the information provided by the user interface layer:

- sets up the Output Support Layer as requested by the user;
- sets up the data structures as appropriate for the application profile specified;
- sets up the error handler as requested by the user.

This layer deals with the level of interpretation required by the user. The user may have decided to only look at the structure of the metafile and not the individual elements and data. Alternatively, the whole metafile may be of interest. This layer has the ability to go through a metafile and only interpret the data which is required. To do this, all the elements are known to this level together with information concerning the state of the metafile in which they occur. These are the states which were discussed in the Background section above. The requested actions to get, read and interpret metafile elements may relate to the whole metafile, to a selected picture or to a state within a single picture.

The Profile Verification Layer crosschecks the elements which have been returned to this layer with the application profile configuration database. Where elements are not allowed within the profile in a particular state or when parameters have gone outside the permitted ranges, an error will be generated on the stack.

3.4.3 Metafile Translator Layer - Functional Responsibility

This layer is the Reference Implementation for the Interpreter. It is driven by the Command Interpreter Layer, and is concerned with initializing and terminating translation, and with passing

the elements interpreted down to the Output Support Layer. This layer also manages the internal state list and tables that are required by the lower layers.

3.4.4 Metafile Translator Layer - Organization

Executive Sub-layer

This layer is driven by the Command Interpreter Layer. It handles the administrative tasks outlined above. It determines the nature of the action required and calls the appropriate code in the lower layers to get, read or interpret items in the metafile.

Functional Sub-layer

This layer performs the get and read functions while the Output Support Layer deals with the interpretation. The get function requests the next op-code from the coding layer below, and is returned in an encoding-independent form. It is suggested that the detailed design should consider the use of the binary class and element id to indicate the item type returned, in order to provide an extensible mechanism. The read function obtains the information about the parameters in the metafile. Both these actions are carried out by the Coding Sub-layer below.

This layer also maintains the state lists (as described in the section on data structures), and checks the syntax of the data being read and interpreted.

Coding Sub-layer

This layer will handle the different encodings. It will have two entry tasks:

- return the op-code of the next element in the metafile and the length of the element data;
- return the data in the buffer space provided by the application.

It also needs to set any error flags as required, which will be tested by the executive layer of the Metafile translator Layer above. The data will be returned in a canonical form as reals and character data to allow the other layers to be encoding independent.

3.4.5 Output Support Layer - General Role

This layer will handle the output of the interpretation, which

may take a number of forms as discussed in the functional specification. To summarize, there may be requirements for:

- graphical output;
- trace output of the elements and data found in the metafile;
- statistical counts relating to the data.

These are envisaged as interpretation tasks which need to be handled in a coding-independent way. At this level there will be a number of drivers, one for each of the output tasks. On entry this driver will be given the information as to the op-code and the data required by that op-code. The driver can then use this information in a way appropriate to the task at hand. The data will be passed in canonical form as real and character data. The entries into the Output Support Layer drivers will be based on an encoding-independent numbering system. It is recommended that the detailed design considers the binary encoding element class combined with the element id as the method of specifying the element values.

The initial implementation should at least have two drivers:

1. a driver for GKS;
2. a driver which will print out a trace of the contents of the metafile.

This will allow the metafile to be interpreted graphically, and also provide a trace of the metafile contents.

3.4.6 Output Support Layer - Graphical Output

The metafile being interpreted may contain elements which cannot be directly drawn by the graphical output system chosen. GKS, for example, does not have a circle primitive. Another example can occur where an application requires a 200 point polyline but the system maximum is 1000 points. Thus, it is desirable that this layer include a simulation layer to allow the picture to be represented in a way which is compatible with the underlying graphics system. Since this layer will involve a considerable amount of software development, it may be appropriate to approach this as a second phase of the software development.

A functional requirement for this graphical output is to allow the selection of preview or publication quality output. In the preview quality it is possible to fall back to the suggestions made in Annex D of the CGM standard - for example, mapping color to black and white. For publication quality this is not permitted. The drivers for the Output Support Level need to

decide whether the picture can be rendered to the required level of quality.

3.5 Summary

This part of the report has carried out a conceptual design for the Reference Implementation of the CGM and some associated applications and output modules. The software designed includes:

1. A Reference Implementation for generating metafiles. Access to it is from routines at the CGM element level. There are also routines for handling encoding-dependent information.

2. A Reference Implementation for interpreting metafiles, which also gives access to the metafile element level through get, read and interpret item functions. The interpretation depends on the Output Support Layer drivers.

3. Output Support Layer Drivers for GKS and a metafile trace.

4. An Output Support Layer driver for metafile evaluation.

5. The Command Interpreter Layer for both the generator and interpreter which includes profile checking.

6. Interactive application programs which make use of the Reference Implementation for simple generation and interpretation of metafiles.

7. A utility for creating the application profile configuration database.

8. A simulation layer for both the generator and interpreter to allow pictures to be stored and interpreted as requested, but in a way which is compatible with the application profile and with the system being used.

This list of the software required is presented in the order in which development could take place.

IV. IMPACTS AND RECOMMENDATIONS

1.0 Introduction

This part of the report considers how the Reference Implementation can be used in the creation of testing tools and a testing service for the CGM.

In the Background section of this report the CGM conformance statements were reviewed. The CGM defines two levels of conformance: full conformance, where the metafile conforms to the abstract functionality and uses one of the standard encodings; and functional conformance, where a private encoding of the abstract functionality is used. The only conformance requirements relate to actual metafiles and do not relate to the generating and interpreting software. There are no constraints placed on the software.

Application profiles, such as the CALS Application Profile, result in the need for testing beyond the conformance statements of the CGM. It is necessary to test that metafiles produced by software in the CALS environment do conform to the CALS Application Profile as well as to the CGM standard. The Reference Implementation and the Command Interpreter Layer of the software described above allow the configuration of the generator and interpreter software for a particular application profile, and this is precisely what is required for the testing of application profiles.

It is important for CALS applications to test not just the generator software but also the interpreter software as well. Confidence in the rendering of pictures by interpreter software needs to be established. This functionality is outside the CGM conformance statements, but is necessary for particular environments such as CALS.

This part of the report will review possible testing strategies and will then go on to look at setting up a test facility using the Reference Implementation and associated applications which were discussed earlier in the report.

2.0 Review of the Possible Testing Strategies

2.1 General Guidelines

Testing is important for ensuring that implementations do conform to the standards. Using existing testing methods it is impossible to guarantee that there are no errors in a product. The testing strategy usually adopted attempts to show the

presence of errors in the product. If a suitably large number of test cases is used, then confidence can be built in a product which handles these tests.

Existing validation suites adopt this philosophy and use a black box approach to testing; that is, the external specification or interface specifications of the product are examined and test cases generated, but no information is required about the internal workings of the implementation being tested.

Exhaustive testing is ideal but may be uneconomic to achieve. The best which can be achieved is to select a wide variety of test cases to exercise the implementation under test as fully as possible. It is important to ensure that the test cases generated have a high probability of detecting any errors in the implementation. Different approaches have been taken to testing software implementations, and these are briefly considered below.

2.2 Compiler Testing

Testing for compilers involves running a large number of test programs. These test the compiler in a given operating environment. The programs which make up the test suites are fully portable and include provision for implementation dependent parameters. The test suites include the following types of test cases:

- Correct (conforming) test programs for which the implementation under test should generate a specific result;
- Incorrect (non-conforming) test programs for which the implementation should generate an error (If specific errors are specified in the standard, then these can be checked.);
- Test programs which provide information on the implementation - for example, precisions and limits (These may be testing beyond the standard but are a useful evaluation.).

2.3 GKS Testing

GKS provides a basic graphics system for the display and manipulation of pictures in two dimensions. A GKS test suite has been developed in Europe with support of the Commission of the European Communities. The test suite adopts a similar approach to the compiler testing described above, namely that test programs to uncover errors have been devised. The tests are at two different interfaces:

- The application programmer interface, where tests are carried out of the data structures and the error mechanism;

- The operator interface, where the tester manually checks pictures drawn by the GKS implementation by comparing the output with a script and example pictures.

A problem with the test suite is that testing is a very manual process. In addition, the test suite is currently available only in Fortran77, although language bindings to a number of languages exist for GKS.

2.4 OSI Testing

Open Systems Interconnection (OSI) testing is concerned with checking the transfer and processing of data between two systems. The aim is to check for the correct transfer of data between the same layer of the 7 layer model in two different systems. One of the systems is the implementation under test and the other is the test center.

The National Computing Centre Ltd in the United Kingdom has developed a model for OSI testing and have implemented it for the Transport Layer. This involves black box testing, with the interfaces above and below the Transport layer being examined by a test responder implemented on the system under test. This test responder is available in a number of languages to assist in the testing. In this scheme, test data are sent from the test center to the system under test. The data are created using a reference implementation of the relevant layer and are tailored to the system being tested. Incorrect data can also be generated by an exception generator.

2.5 The Application of Testing Strategies to the CGM

From the brief examination of the various strategies above, it is apparent that a number of points needs to be considered when developing a CGM testing tool:

- black box testing is the practical solution;

- an extensive range of test cases should be designed within economic constraints;

- self checking and automated test should be designed wherever possible;

- the manual checking of graphical data should be minimized.

Testing the CGM is more like testing OSI implementations than it is compiler and GKS testing, since the concern is for testing of data flow between systems. The definition in the CGM standard is that of the data storage. Therefore, there is no application programmer interface to CGM.

However, there is a major difference between the OSI tests described above and testing the CGM. The difference is that the CGM is at the top layers of the OSI 7 layer model. The abstract functionality is at the Application Layer and the encodings offer Presentation Layer transfer syntaxes. It is possible to examine the data flow coming out of a generator and to send CGMs to an interpreter, but this only tests one side of the black box. There is no standard interface for the other side.

In practical terms this means different problems for testing generators and interpreters. On the generation testing side, it is necessary to define the metafiles which have to be produced in some independent way. On the interpreter side, it is necessary to define how the interpreter is to be tested, and what the output of such tests should be. GKS testing has shown that checking pictures is possible, but very time consuming.

These problems are discussed below when models for testing the generator and interpreter software are considered. Recommendations for a potential test service are also made in light of the availability of the Reference Implementation discussed earlier in this report.

3.0 The Use of the Reference Implementation

The Reference Implementation described above in this report give access to applications at the CGM element level. Applications are to be developed above this for the generation and interpretation of metafiles. A layer of this application, for both the generator and interpreter, involves profile considerations. It is this layer which ensures that metafiles can be configured on generation to conform to an application profile. This layer also checks that an metafile which is being interpreted conforms to an application profile. Therefore, this layer is essential for setting up a test service.

When considering testing it is necessary to envisage application profiles in their widest sense. The application profile can be any legal combination of CGM elements which are deemed appropriate for a particular environment. This means that the way an implementation has been carried out and the options selected is, in effect, a private application profile. The CALS Application Profile and the MAP/TOP Application Profile are important examples of this general concept.

On the interpretation side, the testing service requires drivers within the Output Support Layer. The three drivers which were recognized earlier in this report are required. These are the graphical, trace and evaluation drivers. Drivers beyond these three are considered below in some more detail.

4.0 Testing the CGM Cenerator

4.1 A Simple Model

Testing metafiles produced by an implementation is, in effect, a test of the generator software. A simple model for testing generator software is shown in Figure 4.1.

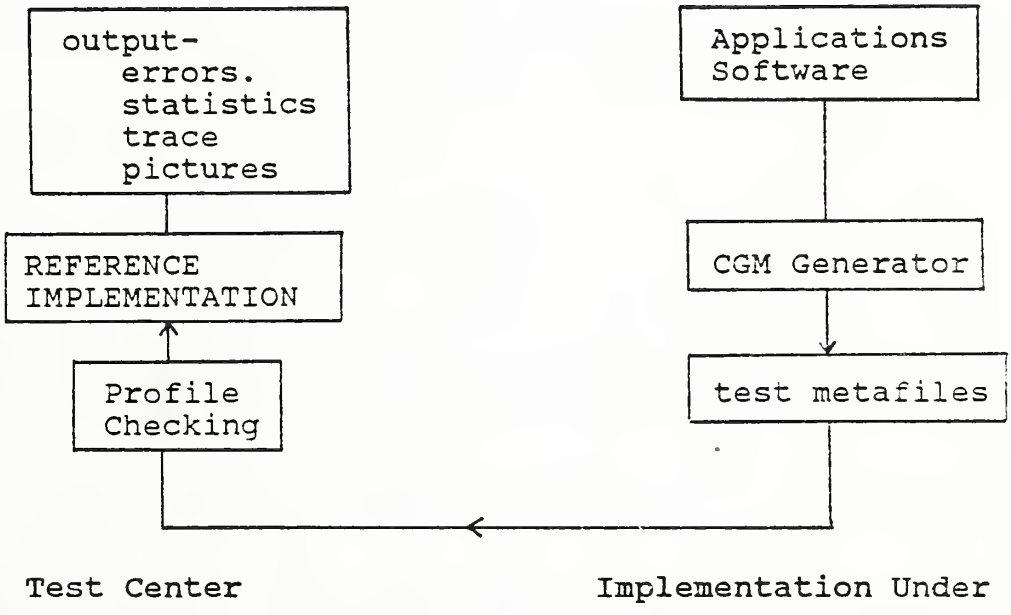


Figure 4.1: A Simple Model of a CGM Generator Test Service

This model shows the implementation under test generating metafiles from an application. These metafiles are then sent, via some form of file transfer, to the test center. The test center then checks them for validity and conformance to the specification of the implementation. This specification may be to the CALS Application Profile.

4.2 Information About the Implementation Under Test

In order to carry out tests it is necessary to obtain information

about the nature of the implementation. The majority of this information has been discussed in this report when the application profile configuration database was considered, and this is the information which needs to be collected from the site to be tested. This will allow the test center to build up a configuration database for the implementation under test using the utility for preparing this file. Also, information needs to be collected regarding any simulation which might exist in the software. An implementation may, for example, store polygons as polylines to simplify the metafile and allow it to be interpreted at a wide range of sites. This information also needs to be collected on an element-by-element basis (along with the rest of the information described earlier), since it is required for the application profile configuration database.

4.3 Specifying the Metafile to be Generated

Metafiles created by the generator can be tested for conformance at the test center. The real problem is how to define the test metafiles to be generated. The choices are:

1. let the site being tested choose the test metafiles;

2. offer sample programs using GKS, resulting in metafiles where a metafile device driver is available;

3. give the site being tested some sample pictures and ask them to create metafiles corresponding to the pictures;

4. describe a picture more formally, for example using the clear text encoding of the CGM.

The first option is not a good independent test since it does not allow impartial selection and thus is more prone to error detection. However, it could be used in conjunction with metafiles specified by the test center. Then the metafiles chosen by the site being tested could be oriented towards their particular applications.

The second choice is more reasonable as an independent selection of graphical output to be stored on the metafile. This method gained considerable support at the Disley, UK meeting on CGM testing. However, there are problems associated with this choice, too, since there is no requirement for the site being tested to have a GKS implementation. Even if the site does have an implementation, there may be problems if it is not a validated implementation. Also, there is no certainty which CGM elements would be written by the GKS program. Although a recommended mapping appears in the CGM, this is only a recommendation. It would be very difficult, if not impossible, to automate testing of the generated metafile with this method of specifying test

metafiles to be produced.

The third option involves a great deal of effort on the part of the site being tested, and for that reason is not useful. The fourth option is probably the best, particularly while no mapping from GKS to CGM exists as a standard specification. The clear text encoding is also available within the Reference Implementation, and this also makes the fourth option more economic.

As noted earlier in this section, these problems all arise because the CGM is at the top of the OSI model. This discussion is attempting to fit an OSI-type testing model to the CGM. Should this work continue, it will be necessary to consider the current effort in standardizing language bindings for CGI. Currently, there are proposals to extend this effort for the CGM. This may not be accepted, but the effort on it should be input to any decisions made for the work decribed here.

4.4 The Application Module for Generating Test Metafile Descriptions

This section considers the way that the test metafile descriptions should be generated by the test site which has a Reference implementation.

The application for generating test metafile descriptions will sit above the Profile Tailoring Layer described above in the Discussion section. This layer has information regarding the implementation being tested, since it exists in the application profile configuration database. This application will only generate metafiles which can be understood by the implementation being tested.

The application will also have access to a database of partial metafiles. These metafiles will incorporate primitives and attributes which can be encoded in a manner which can be understood by the site being tested. Considerable work went into building the operator test suite pictures for GKS and in producing the evaluators manual. It is recommended that the primitives and attributes used for the GKS test pictures be used in the partial metafiles database. These can be selected where appropriate. This will not be an easy task, since the GKS test programs contain control and inquiry functions. However, the use of the static pictures for CGM definitions should be attempted.

A major goal for the design of this application module is the automatic generation of clear text encoded metafiles, which will be used as specifications at the test site. The application needs to select a range of options within the supported values. Any extra information for the test site, for example encoding

information, will be given as comments in the metafile.

The design of the Reference Implementation lends itself to this automatic generation of clear text encoded specifications. It is also suggested that the site being tested be given the option of producing metafiles from its own applications for conformance testing.

4.5 Testing the Generated Metafiles

These generated metafiles can be tested for their conformance to the CGM standard by going through the metafile with no Output Support Layer driver, but just checking for error conditions. This application will sit above the Profile Verification Layer in the Interpreter design.

This may be sufficient for many applications. There is, however, no guarantee that the correct information that was requested has been stored. It is important to consider whether any further automatic checking can be carried out without resorting to picture checking.

Since the generator is only being asked to generate metafiles to a level which it is capable, there should be no simulation of elements. Therefore, it is recommended that the test center require a further Output Support Layer driver for automatic metafile analysis. This driver will have the same form as the other drivers; that is, it will have entries at the CGM element level. The driver will use the clear text encoded specification to give information on what should have been generated by the implementation under test in the form of a database for the analysis. This will then be used as a checklist for what should be stored in the metafile. There is no requirement that any order be maintained by the implementation under test, but all elements in the database should appear somewhere in the correct CGM state. There are some problems concerning precise comparison of the values, and the analysis may have to be more general than the precision at which the values are stored. But the economic benefits of automatic testing should outweigh any problem of lost accuracy in such testing.

4.6 Testing the Interpreter

A simple model for the testing of the interpreter is shown in Figure 4.2.

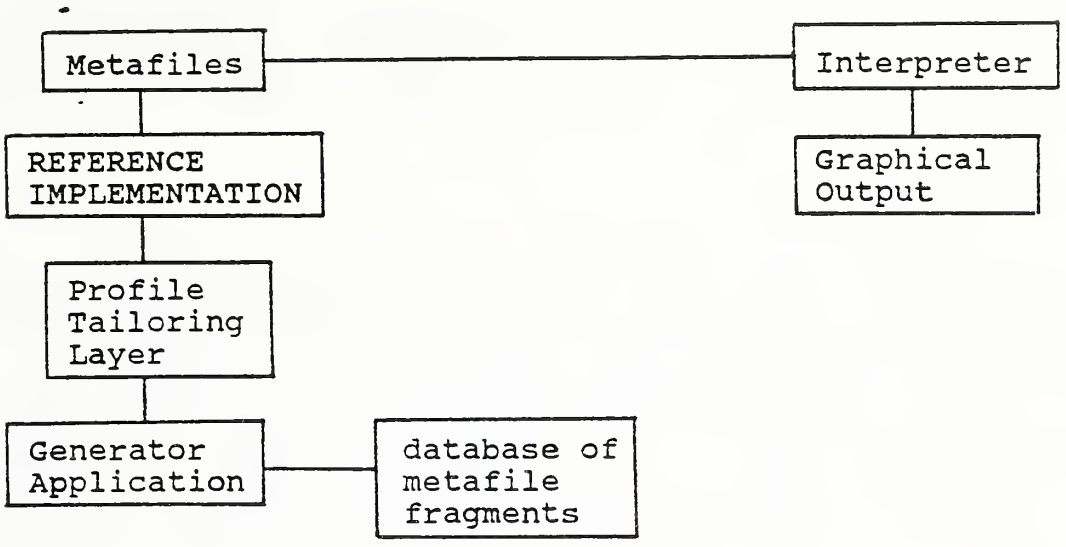


Figure 4.2: A Simple Model for the Testing of Interpreter Software

In this model the test center creates test metafiles which comply with the application profile configuration file set up for the test. Then these are sent to the test system. This model is exactly analogous to the creation of the metafile specifications for the generator testing. The same application module can be used with an extension to allow all three encodings.

The only form of testing for the interpreter software appears to be testing the graphical output from the interpretation. The production of test metafiles may be useful to implementors while they are writing their software. They may be prepared to put the time in to checking the pictures with the script and test pictures. The production of test metafiles may be a useful (and profitable?) role for the test center where the software can be configured to suit any implementation.

For CALS it may be necessary to carry out formal testing of implementer software. On-site testing of the graphical output is necessary for this. It is recommended that the GKS test pictures be used as the basis for developing a suite of test metafile descriptions. This will save effort in developing ideas, test pictures and descriptions.

It may be useful to test the implementation for its behavior on the receipt of metafiles outside its application profile and also

for receipt of corrupt metafiles. Using the scheme proposed for the Reference Implementation it is straightforward to produce metafiles outside the profile in a controlled way. It is suggested that to allow the latter test there should be a utility capable of corrupting test metafiles in a specified way. This utility could be an automatic one or could be an interactive program.

4.7 Summary

The adoption of the Reference Implementation structure allows the production of test applications for testing both generators and interpreters. The use of the application profile configuration database means that the specification of metafiles and their subsequent analysis can be automatic. There may be some loss of testing accuracy, but it is considered minimal compared with the economic savings. Generating metafiles for testing on an interpreter also falls readily into the scheme. Manual checking of graphical output appears to be unavoidable for this stage of testing. It is recommended that effort be put into the area of picture comparison testing since it is a problem across the range of graphics standards.

V. SUMMARY AND CONCLUSIONS

The Computer Graphics Metafile (CGM) standard, FIPS 128, specifies the syntax and semantics of a standard file format for storing and communicating computer graphics pictures. It does not specify the behavior of the software that generates and interprets CGMs. This makes behavior of CGM software somewhat unpredictable. Worse yet, it means that there is no basis for testing and certifying CGM products. This situation is unacceptable for the CALS effort, which is adopting the CGM into its family of standard interface specifications. There are two components to resolving this situation.

First, the specifications of CGM must be augmented so that the syntax and semantics of generators and interpreters is unambiguous, and so that the expected behavior of generators and interpreters is clearly stated. In other words, the specification must be "testable." This is one function of an Application Profile, one of which has been produced for the CALS environment.

Second, a testing methodology must be devised. As part of this testing methodology implementations ("Reference Implementations") of CGM software are needed.

This report is concerned with designing a reference implementation for the CGM. The report then considers the role of such a Reference Implementation in a testing environment.

First, the concepts of CGM which are important for the design of the Reference Implementation are considered. It is noted that many implementations will adopt one of the developing application profiles, such as the CALS Application Profile. This will help to ensure that the metafiles generated can be interpreted on a wide range of other systems.

Next, the report looks at:

1. The functional requirements for the Reference Implementation. The Reference Implementation will allow the generation and interpretation of any legal metafile. It is also important that the software can be configured to application profiles. This needs to be a general configuration tool to cater for current and potential requirements of CALS. It is important to ensure that the Reference Implementation can be used from a wide range of applications. These applications will include testing but there may be wider uses of the software in the CALS environment.

2. The conceptual design of the Reference Implementation. The design is a layered structure and is modular with access at

the CGM element level. Access is also given for encoding dependent elements. The Reference Implementation will be concerned with CGMs written to conform with the CGM standard. A layer for tailoring the software for application profiles sits above the Reference Implementation. This will also be available to other applications. The software is tailored to fit an application profile via a configuration database.

Finally, the use of the Reference Implementation as a basis for testing tools and a model for a test service is considered. The layered design of the implementation and the use of a configuration database is a useful design for testing. The site to be tested has to supply the information for the configuration database to indicate the nature of the implementation. The test center will have a number of standard databases for application profiles, such as CALS and MAP/TOP which are in wide use.

Using this database it is proposed that the test center creates a definition of the metafiles which the implementation under test has to create. This specification will be given in the CGM Clear Text encoding which can be created by the Reference Implementation. The test center will have a number of metafile fragments which can be configured to a particular implementation. These fragments will be based on the GKS operator test pictures. This clear text encoding will be used for automatic analysis of the metafiles to be tested.

Interpreter testing requires manual on-site testing of the output created by the test system following the interpretation of a range of test metafiles. The report suggests that research on automatic picture checking is needed if testing of graphics standards in the future is to be economic.

The implementation of the software designed in this report will require a detailed design and costing to be carried out prior to implementation.

VI. REFERENCES

Graphical Kernel System (GKS); ANSI X.3.124-1985

Computer Graphics Metafile (CGM); ANS X.3.!22-!986

Computer Graphics Interface (CGI); working draft American standard; ANSI X3H3/85-47

Programmers Hierarchical Graphics System (PHIGS); dpANS X3.144-1987

ISO DIS 8805 Information processing systems - Computer graphics - Graphical Kernel System for three dimensions (GKS-3D) functional description, 1987

ISO DIS 8613 Information processing systems - Text and office systems - Office Document Architecture (ODA) and Interchange Format - Part 8: Geometric Graphics Content Architectures (GGCA), 1987

ISO IS 8632 Information processing systems - Computer graphics - Metafile for the storage and transfer of picture description information (CGM), Addendum 1, working draft ISO TC97/SC21/N1403, 1986

Manufacturing Automation Protocol (MAP) Specification, Version 2.2, January 1986, Society of Manufacturing Engineers

Technical Office Protocol (TOP) Specification, Version 1.0, November 1985, Boeing Computer Services, Seattle, WA.

K.W. Brodlie, L.R. Henderson, A.M. Mumford, "The CGM - A Metafile for GKS?", Computer Graphics Forum, 1987, Vol 6, No 2.

L.R. Henderson, M. Journey, C.D. Osland, "The Computer Graphics Metafile", IEEE Computer Graphics and Applications, Vol 6, No 8, 1986

A.M. Mumford, "Open Systems Opportunity Study - The Computer Graphics Metafile", 1987, Loughborough University Computer Centre, UK

A.M. Mumford, "Why Care About the Computer Graphics Metafile?", Computer Aided Design, Vol 19, No 8, 1987

FINAL REPORT

CALS SOW TASK 2.2.2.2.1

IGES TO CGM TRANSLATOR
DESIGN SPECIFICATION

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

I. PURPOSE

Develop a link between IGES data files and CGM picture files (CALS SOW Task 2.2.2.2.1). CAD/CAM packages, which produce IGES files, provide input to both automated technical manual and engineering data repository systems. To minimize storage and processing overhead and maintain required system performance, CGM is the protocol which has been chosen for CALS as the mechanism for the transfer of graphical pictures within and across these systems. An approach must be developed to transfer data from IGES format to CGM format. To meet this requirement, this report comprises a detailed design specification for a piece of software whose function is to translate from certain IGES product data files to CGM picture files.

Not all IGES files are suitable for translation; indeed, many IGES entities are not directly representable as components of pictures. Instead, IGES files conforming to the IGES application subset for Technical Illustrations [described in Appendix A of DOD-D-(28000), an interface standard, TS401, which is included in the CALS Core Requirements (phase 1.0) document] has been selected for implementation.

The specification herein contains sufficient detail for a programmer familiar with both IGES and CGM to code and test the program.

II. BACKGROUND

1.0 CALS REQUIREMENTS

1.1 Review of CALS-Related Requirements for Standards

References 1 and 2 contain analyses of CALS requirements for graphics-related standards in the areas of engineering design, technical publishing, procurement support, and interactive delivery systems.

This report focusses on the picture interchange requirements of CALS when applied to the task of technical manual publishing and illustration.

1.2 Relevant Standards

1.2.1 The Computer Graphics Metafile

The CGM provides a file format suitable for the storage and retrieval of picture description information. The file format consists of an ordered set of elements that can be used to describe pictures in a completely device-independent way. One or more pictures can be stored in a single metafile, and the

metafile is defined in such a way that, in addition to sequential access to the whole metafile, random access to individual pictures is well defined. That is, the pictures are completely independent, one from another: their appearance does not depend upon the order in which they are accessed or displayed.

In addition to a functional specification, the CGM standard documents three standard encodings of the metafile semantics. The Character encoding requires minimum metafile size and is suitable for transmission across networks of heterogenous systems but is expensive to encode and decode. The Binary encoding requires minimum effort to generate and interpret but is not well-suited for exchange between computers of different arithmetic data types. It is nearly as efficiently coded as the Character encoding. The Clear-text encoding provides maximum readability and editability for ease of use by humans (e.g., for debugging purposes) but, generally, pays a heavy penalty in size and performance. The size is much larger because English and other natural languages contain a lot of redundancy. The performance is worse because parsing and recognizing text strings and converting text strings to internal numbers for use by a graphics subsystem is expensive in its use of CPU cycles.

In reference 1, the standardized CGM elements are listed by type. The ESCAPE and APPLICATION DATA elements have been provided to support uses of the CGM in ways that go beyond the exchange of pictures. Nongraphical data and graphical elements not yet standardized can be incorporated into metafiles in a regular way. When these extended metafiles are exchanged by cooperating processes, standard commercial products can be used to handle the standard metafile elements, and new code need be written only for the special, non-standardized elements. Large groups of users of extended metafiles can get together and agree upon a set of extensions--just like MAP and TOP users have agreed upon guidelines to the implementation of the OSI standards. For example, the elements of a business chart--like legend entries, tick marks, and axis labels--or the elements of a project schedule--like PERT chart symbols, milestone markers, or title--could be marked in the metafile. An editing program could be written to read such metafiles and allow modifications to them before rendering the chart on a hardcopy device or including it in a report or manual.

In the absence of any facsimile standard capable of handling multicolor images (i.e., those with more than one bit per pixel), a CGM employing only the CELL ARRAY primitive could be used. Images expressed with either indexed and direct color specifications can be represented. In the Character-Coded and Binary encodings, run-length encoding may be used to reduce the size of the resulting CGM files.

The CGM was approved as ANSI/X3.122 in 1986. It has also been approved as FIPS 128.

1.2.2 The Initial Graphics Exchange Specification

The Initial Graphics Exchange Specification (IGES) is a mature standard, first published in 1981, for the digital exchange of database information among present-day CAD systems. Now in its third version, engineering drawings, 3D wireframe and surfaced part models, printed wiring product descriptions, finite element mesh descriptions, and process instrumentation diagrams are application usages addressed by IGES.

IGES information, including drawings and 3D wireframe product models, is intended for human interpretation at the receiving site. However, IGES is often used to attempt interchange between CAD databases and to feed external geometric data into a CAD system, where the data are expected to be processed automatically by computer as well as being worked on by human operators. Consequently, when used for this kind of interchange--a purpose it was not originally designed for, IGES files are often restricted in the kinds of entities used.

The complete IGES standard describes well over 50 entities, some with very elaborate and numerous variations. DOD-D-(28000), Appendix A, reference 5, specifies a subset of IGES suitable for the interchange of technical illustrations. Seventeen entities, including curves, arcs, closed areas, splines, and text, are included in this subset. In addition, drawings comprising several views of objects, whose component parts may be positioned via transformation matrices, can be specified. Subfigures occurring several times in the drawing can be defined once and then instanced.

III. DISCUSSION

1.0 Abstract Model

1.1 IGES Model

The description contained in this section and in the remainder of this report relates to that subset of IGES known as the Application Subset for Technical Illustration (see reference 5).

1.1.1 IGES Geometry

The products that can be illustrated are represented as wire-frame models. All components (or objects) are two-dimensional and lie in one IGES layer. Geometric objects can be comprised of circular and conic arcs, linear curves, lines, parametric spline and rational B-spline curves, and composites of such curves. Geometric objects can also be comprised of simple closed areas and sectioned (i.e., cross-hatched) areas. General annotation may also be supplied. Points and their generalization to instances of marker symbols, whose definitions are supplied separately, can also be used to form illustrations.

1.1.2 IGES Structures

Products can be described by multiple instancing of defined IGES structures, known as subfigures. The subfigures themselves are made of multiple occurrences of the basic IGES entities. Generally speaking, IGES subfigures are defined in their own local coordinate system.

1.1.3 IGES Drawings and Views

A DRAWING entity allows a set of IGES VIEWS to be identified and arranged for human presentation. Each view is a representation of a selected subset of the geometric model, together with non-geometric information such as text. The VIEW entity controls such representations, providing information for orientation, clipping, line removal, and other characteristics associated with individual views rather than with the model itself.

The view and drawing entities contain only the rules and parameters for extracting drawings from the geometric model. The actual product definition is not duplicated in various views.

1.1.4 IGES Coordinate Systems

The IGES Technical Illustration Subset model space is a

two-dimensional Euclidean space, the space in which the model (or product) resides. The model space coordinate system is fixed relative to the model.

In addition to the model space, IGES has the notion of a **definition space**, which is also a two-dimensional Euclidean space. In contrast to the model space where a single fixed coordinate system exists, the definition space coordinate system may vary from entity to entity. The origin of a definition space coordinate system may be any point in model space, and the orientation of definition space may be arbitrary with respect to model space. However, it is assumed that the unit of length is always the same in both the model space and the definition space coordinate systems.

The definition space concept allows the use of a temporary coordinate system in positioning certain geometric entities into model space. Use of definition space entails initially describing an entity in definition space, and then converting this to a model space description.

A TRANSFORMATION MATRIX entity is used to specify the rotation and translation components of the mathematical transformation that maps definition space coordinates into model space coordinates. Indeed, the complete definition of a geometry entity, with respect to model space, involves the Transformation Matrix entity. However, when the Transformation Matrix is exactly comprised of the identity rotation and zero translation, a special convention is provided in IGES to signal this situation and prevent unnecessary processing.

1.1.5 IGES Attributes

Attributes such as line weight, line style, and color are specified separately for each instance of each IGES entity in an IGES file.

1.2 CGM Model

1.2.1 CGM Geometry

CGM pictures can be represented by lines, circles, circular and elliptical arcs, markers, text, and filled areas of various appearance. No parabolic and hyperbolic curves are supported, nor are splines of any sort.

1.2.2 CGM Structures

Within each CGM picture, there is no substructure corresponding to IGES subfigures.

1.2.3 CGM Drawings and Views

Within the CGM, each picture is a complete drawing. There is no notion of a view, with its associated Transformation Matrix.

1.2.4 CGM Coordinates

There is only one coordinate system in the CGM--Virtual Device Coordinates (VDC), a two-dimensional Euclidean space. No transformation matrix can be specified, even to map an element in VDC space into another position or at another orientation in VDC space.

1.2.5 CGM Attributes

CGM attributes are modal. Once set, they apply to all subsequent elements until explicitly changed.

1.3 The Translation Task

1.3.1 Geometry

As described in detail in section 3 below, several IGES geometric entities do not map directly to CGM elements. Consequently, these entities must be rendered as simpler constructs known to CGM; e.g., parabolas, hyperbolas, and splines as CGM POLYLINES. This creates a problem for the translation task, because the resolution of the CGM is virtually infinite; therefore, it is impossible to calculate how many line segments each portion of a curve should be divided into before being placed in the CGM.

The only solution is that the translator task be informed by some sort of external input--perhaps stored in a configuration file. The assumed resolution of the ultimate target device for the CGM must be provided. By default, one might write the translator to generate simulated curves assuming 400 dots per inch across a display surface of 6" by 8."

The curved-line-to-straight-line algorithms built into the translator task should be parametrically configured, so that the algorithms work correctly regardless of the targetted output resolution.

1.3.2 Structures

During translation, each instance of an IGES subfigure must be fully expanded and all the geometric information about each instance written as separate CGM elements.

1.3.3 Drawings and Views

The whole illustration represented by a DRAWING entity and its related VIEW entities must be created by the translator task as a single "snapshot" and stored in a single picture of the metafile.

1.3.4 Coordinates

For the purposes of the translation task, IGES model space can be made equivalent to CGM VDC space. One of the tasks of the IGES-to-CGM translator is to specify a VDC EXTENT that encloses all elements in the drawing, without introducing too much "white space" around the border.

When generating CGM elements from IGES entities, the coordinates of the IGES entity must be subjected to all transformations implied by the TRANSFORMATION MATRIX, VIEW, and DRAWING entities.

The translator task must keep a stack for saving and later restoring transformation matrices. Whether the matrix is composed with previous transformations or replaces the current matrix is controlled by the status field in the Directory Entry. The processing required by the IGES standard is very complex and is described in great detail on pages 31-33 of Reference 7.

The parameters defining all IGES entities should be transformed to model space coordinates before any required emulation of geometry is performed.

1.3.5 Attributes

When the translator encounters an IGES entity, the entity attributes should not be written immediately to their CGM equivalents. Instead, a set of local CGM attributes is maintained for each type of CGM primitive (line, marker, text, and filled area). With each attribute set is a set of flags that record whether the current attribute value stored in the local variables is different from the current attribute value written to the metafile.

The attributes associated with each IGES entity are used to set the local attribute variables and to set the associated flags

accordingly. Then, when a CGM primitive element is written to the metafile, only those local attributes that have changed since the last CGM primitive of the same type was encountered need be written to the metafile before the primitive element itself is written.

The attribute mapping process is further complicated when processing subfigures, because the Hierarchy Status bits in Directory Entry field 9 affect whether attributes are inherited by the subfigure instances or not.

Some IGES attributes map directly (e.g., line types 1 and 2), some have CGM equivalents (e.g., IGES color indices 0 through 8 can be mapped to CGM color indices 1 through 9, but not in the same order), and some have no equivalent (e.g., IGES line types 3 and 4).

The translator task must write the default IGES color table explicitly in each metafile. For IGES line weight (known as line width in CGM), IGES global section parameters 16 and 17 can be used to compute an absolute line width in model space coordinates, which can then be mapped to VDC for the CGM. The CGM line width and edge width specification mode is always set to absolute.

2.0 Top-Level Design

2.1 Data Structures

2.1.1 From the Global Section

Global Data

GD-FP	GD Index 7-11 and Index 19	Used to set CGM REAL PRECISION and VDC REAL PRECISION.
GD-PI	GD Index 12	Used for the BEGIN METAFILE string.
GD-MS	GD Index 13	Used for metric scale factor in the SCALING MODE.
GD-LW1 GD-LW2	GD Index 16 GD Index 17	Used to translate Line Weight information into CGM absolute Line Width VDC values.
GD-XT	GD Index 20	Can be used to derive the initial values to be used with the CGM VDC EXTENT element.

The remaining information in the Global Section is not needed for proper CGM generation, although the information in Indices 1 and 2 is needed for proper parsing of the IGES input file.

2.1.2 From the Directory Entry Section

The Directory Entry (DE) section has one DE for each entity in the file. The purposes of the DE section are to provide an index for the file and to contain attribute information for each entity. Each DE points to the first line of the Parameter Data (PD) record. The order of the DE entities within the DE section is arbitrary, with the exception that a definition entry must precede all of its instances.

Blank fields are considered to be specified with the corresponding default values. Default values for each field depend upon the entity type and are specified in the IGES standard.

Entity Data

There is one instance of a DE data record for each PD entity in the IGES file. This complex record type is structured as an array of records. The translator task builds a table that associates the array index with the original IGES entity pointer value.

DE-ET	DE Field 1	Entity type number.
DE-ST	DE Field 3	Pointer to a structure definition entity (type 308).
DE-LF	DE Field 4	Value from 1 to 4, specifying the line font pattern to be applied to the rendering of the object.
DE-VW	DE Field 6	If zero, the entity should be displayed in all views. Otherwise, a pointer to the DE entry of a VIEW entity to be used for a single view.
DE-TR	DE Field 7	If zero, the identity transformation matrix and the zero translation vector is implied. Otherwise, a pointer to the DE entry of a transformation matrix (type 124).

DE-ST1	DE Field 9 digits 1-2	Blank Status. If 00, entity is visible; if 01, entity is not visible.
DE-ST2	DE Field 9 digits 3-4	Subordinate Entity Switch. Legal values are **independent, physically dependent, logically dependent,** and **both physically and logically dependent.**
DE-ST3	DE Field 9 digits 5-6	Entity Use Flag. Legal values are **geometry, annotation, definition, other.**
DE-ST4	DE Field 9 digits 7-8	Hierarchy Flag. Legal values are **all attributes inherited, no attributes inherited, some attributes inherited.**
DE-LW	DE Field 12	System display thickness. Can be converted to CGM absolute line width by multiplying by GD-LW2 and dividing by GD-LW1.
DE-CO	DE Field 13	IGES color number, which is mapped to a CGM color index. IGES color 0 maps to CGM color 1; IGES 1 to CGM 8; IGES 8 to CGM 9; and IGES 2-7 to CGM 2-7.
DE-FO	DE Field 15	(Integer) form number. Represents a subcategorization of entity types.

2.2 Program Flow

2.2.1 External View

The IGES-to-CGM-Translator Task (hereafter called **translator**) will be a stand-alone, non-interactive program. It will take as input a single IGES file, which it may assume conforms to the IGES Technical Illustration Subset (if found to be non-conforming, **translator** outputs suitable error messages). **Translator** shall produce as output a single CGM file, whose contents, when interpreted and displayed on a graphics device, shall represent an image similar to that which would be produced were the IGES file to be directly interpreted and displayed on the same graphics device.

Translator shall also search for an optional configuration file, which contains additional information concerning the nature of the CGM file to be created (see section 2.3). The contents of the configuration file specify, for example, the encoding format to be used for the CGM, and are described later in this section. If the configuration file is not present or contains partial information, standard defaults for all missing information are assumed.

An error file listing all errors of syntax or semantics encountered while interpreting the IGES file is produced, if any errors indeed occur. These messages may also be directed to the operator's console. Upon completion of the task, a message indicating success or failure is displayed on the operator's console.

The exact syntax of the command to invoke translator and to specify the file names of the various files is not specified. The syntax should be designed according to the conventions of the operating system to be used. PC-DOS conventions are different from UNIX and from VMS, for example.

Figure 2-1 summarizes the external view of the IGES-to-CGM translator task and its relationship to its environment.

2.2.2 Internal View

Figure 2-2 provides an overview of the internal processing of the translator task.

After checking for the presence of the specified input and configuration files, translator makes a pass through the IGES input file, reading in most of the information contained in the Global and Directory Entry (DE) sections. The initial Flag section, if present, tells translator whether the file is in Binary Form or Compressed ASCII form. The Start section simply contains comments that could be displayed by translator upon the operator console (if desired by the operator) and echoed to the error file to provide some context for any errors that are logged.

In section 2.1 above, we have indicated which pieces of information are retained in the local data structures as they are read from the Global and Directory Entry Sections. Furthermore, any constraints placed on the file by the Technical Illustrations Subset (reference 5) are checked at this time and errors recorded if present.

As the Parameter Data (PD) Section is scanned, all back pointers to directory entries are checked for consistency. A table correlating IGES pointers with internal data structure indices is

created at this time. Other constraints as represented by notes 2, 3, 5-10, and 12 of Table I of reference 5 are checked and errors recorded.

If any syntactic errors are encountered, the task is terminated at this time. The error file gives the operator sufficient information concerning the invalid information in the file to repair the file, if desired.

The configuration file contents, if present, are read and loaded into local storage, replacing the default values already stored in the corresponding variables. A new, but temporary file for the metafile is opened. On some systems, its type (e.g., formatted or unformatted) will need to vary with the CGM encoding used: Character-coded, Binary, or Clear-text.

Information in the Global Section, perhaps overridden by some elements in the configuration file, specify the precision of the integer and floating point data to be written to the CGM. In fact, the CGM may be restricted to storing integer data only. In this case, the configuration file must indicate the range of integer data that the CGM is allowed to handle. During later processing, if a coordinate is generated that lies outside this range, an error is recorded and further translation of the IGES file is halted. The metafile is closed by writing END PICTURE and END METAFILE elements to the CGM.

Next, the BEGIN METAFILE element is written to the CGM, using information obtained from the Global Section. Metafile Descriptor elements are then written to the CGM. Next, the BEGIN PICTURE element is written, again using information obtained from the Global Section. Picture Descriptor elements are then written to the CGM and, finally, the BEGIN PICTURE BODY element and several Control elements are written to the CGM. These elements are all described in section 3 below.

The input file is rewound and positioned at the first PD entry. The following logic is then repeated for each entry in the PD section (see Figure 2-3):

1. The PD entry is read and the coordinates saved. The DE index is looked up.

2. All structure, drawing, and view calculations are applied to the coordinates, resulting in the conversion of the coordinates from definition space to model space within the IGES file.

3. The attributes specified in the associated DE are processed to include considering whether the entity inherits some or all of its attributes from a parent entity.

4. The entity is mapped to its CGM equivalent(s), as described in section 3. The minimum and maximum X and Y VDC coordinates generated while processing the whole IGES file are saved in local storage. At the end, these values will be used to compute good values for the CGM VDC EXTENT element.

5. The resulting CGM primitive element(s) are output, sometimes preceded by a sequence of CGM attribute elements.

If a group of parameters is defined at the end of the PD entry, these parameters must contain pointers to general notes (type 212). If present, an additional step to translate these notes into their corresponding CGM elements must be accomplished, as discussed in section 3.

When translator has finished processing every PD entry, the final version of the metafile can be written. The temporary version is copied to the permanent version, with actual VDC extent information (perhaps rounded up to whole number values giving an approximate 5% border) replacing the default VDC information placed in the VDC EXTENT element when it was first written to the metafile. END PICTURE and END METAFILE elements are written to the end of the permanent CGM file.

No GDP, Escape or External CGM elements are written to the metafile, because there are no IGES elements that map to these classes of CGM elements.

2.3 Configuration File

A **configuration file** is a disk file, which, if present, is read by the translator task prior to commencing translation of the IGES input file. The configuration file represents a source of information, external to the IGES file itself, that provides direction to the translator task.

2.3.1 Format

The configuration file should be set up as an ASCII file, so that it can be created by any standard text editor. The file consists of a series of groups of lines, each group starting with a line containing only a keyword (the **keyword line**). The remaining lines of the group (the **parameter lines**) contain the parameters (if any) associated with the keyword. The format of a parameter line is free form: parameters are expressed as integers or strings separated by one or more delimiters (space, colon, semicolon, and comma should all be legal delimiters). Keywords groups need not be in any particular order in the configuration file.

As specified in detail throughout this section and section 3.0, the configuration file contains a miscellany of information. These items are collected and described in the following paragraphs.

2.3.2 Content

Target Device Resolution

Keyword: RESOLUTION

Parameters: number of addressable points in X;
number of addressable points in Y;
length of dimension X in mm;
length of dimension Y in mm

CGM Encoding Format

Keyword: ENCODING

Parameter: CHARACTER or BINARY or CLEAR.

Metafile Description

Keyword: DESCRIPTION

Parameter: Arbitrary string.

VDC Type

Keyword: VDC_TYPE

Parameter: INTEGER or REAL.

CGM Integer Precision

Keyword: INTEGER_PRECISION

Parameter: A legal CGM INTEGER PRECISION value.

CGM Real Precision

Keyword: REAL_PRECISION

Parameters: A legal CGM REAL PRECISION set of three values.

CGM Index Precision

Keyword: INDEX_PRECISION

Parameter: A legal CGM INDEX PRECISION value.

CGM Background Color

Keyword: BACKGROUND_COLOR

Parameters: Three integers, representing the RGB components of the color to be associated with the color index 0.

CGM VDC Integer Precision

Keyword: VDC_INTEGER_PRECISION

Parameter: A legal VDC INTEGER PRECISION value.

CGM VDC Real Precision

Keyword: VDC_REAL_PRECISION

Parameters: A legal VDC REAL PRECISION set of three values.

CGM Auxiliary Color Index

Keyword: AUXILIARY_COLOR

Parameter: An integer between 0 and 8, inclusive, representing a color index value.

CGM Transparency

Keyword: TRANSPARENCY

Parameter: PRESENT or ABSENT.

Curve Approximation Guidelines

Keyword: MINIMUM_LINE_LENGTH

Parameter: An integer representing the minimum length of a line segment (in tenths of VDC units) used to approximate curves that are not directly representable in the CGM.

Text Approximation Guidelines

Keyword: MINIMUM_TEXT_LINE_LENGTH

Parameter: An integer representing the minimum length of a line segment (in tenths of VDC units) used to approximate text characters when they are stroked out and approximated by CGM POLYLINE elements.

Point Coincidence Guidelines

Keyword: **TOLERANCE**

Parameter: An integer representing the maximum distance (in tenths of VDC units) that two (X,Y) coordinate pairs, that is, points, are allowed to be separate, while still considering the points to be coincident.

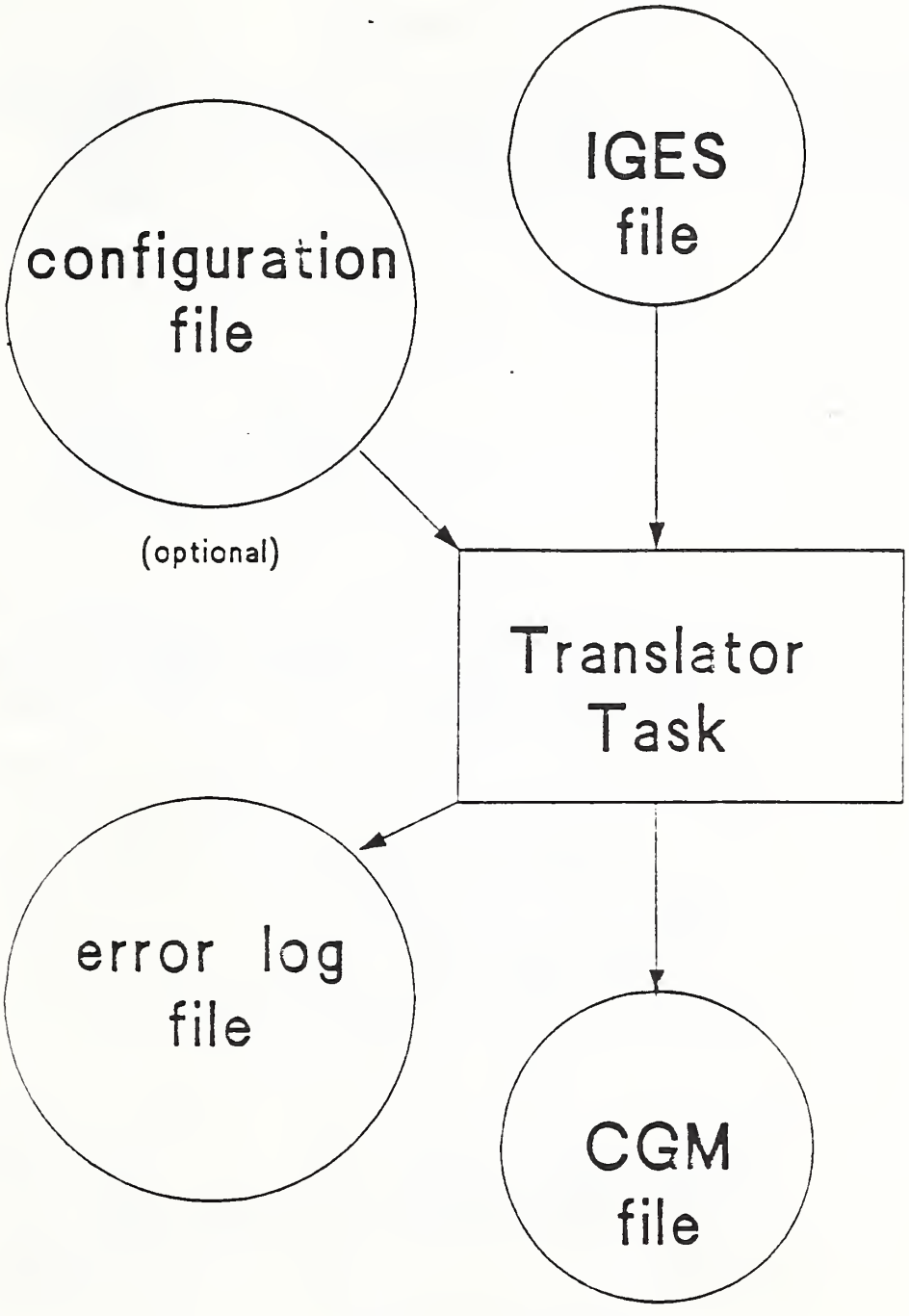


Figure 2-1. External View.

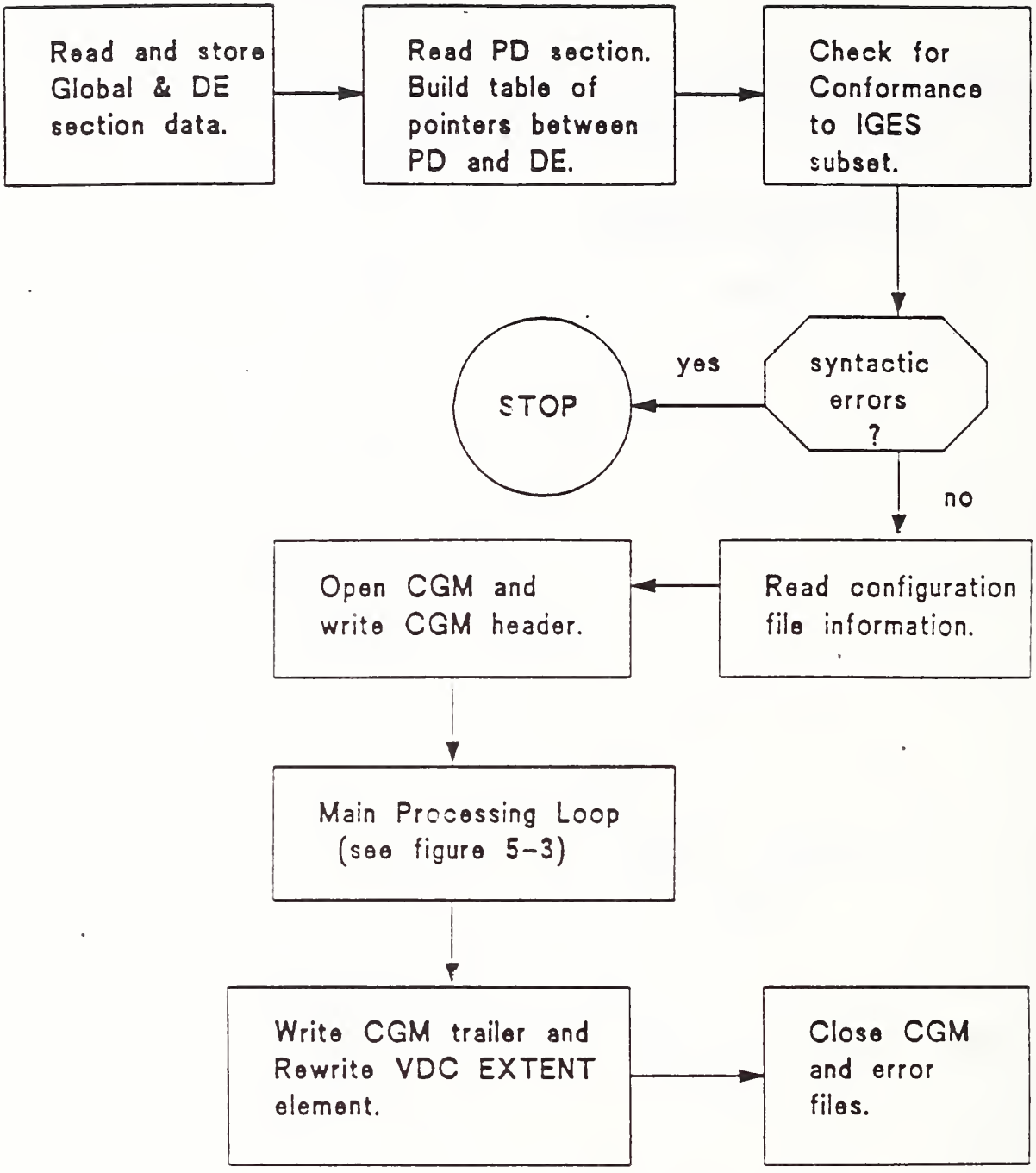


Figure 2 -2. Internal Logic Flow.

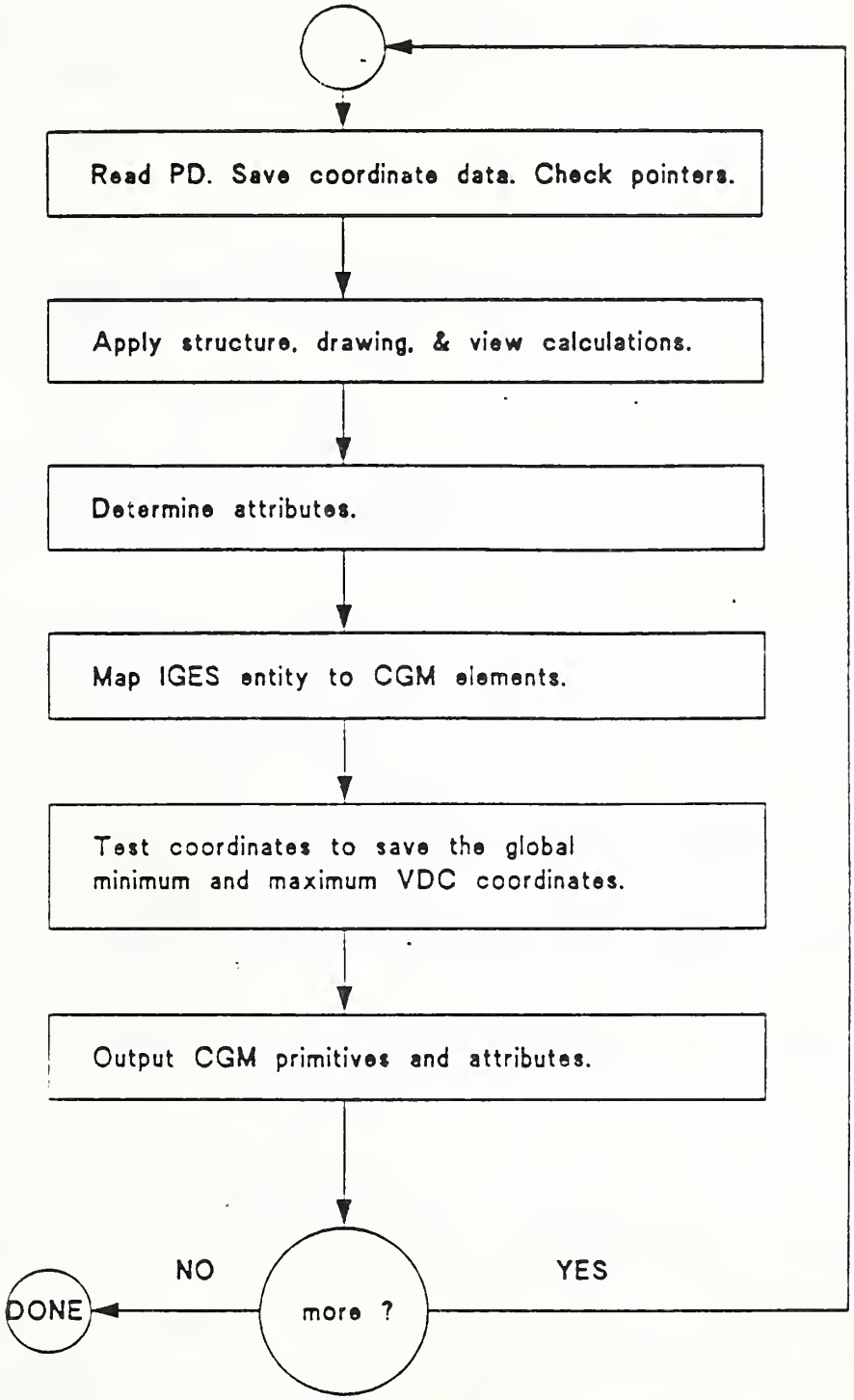


Figure 2-3. Main Processing Loop.

3.0 Mapping Between IGES and CGM

3.1 Header Information

3.1.1 Metafile Descriptor

A metafile descriptor consisting of one instance of each of the following elements is written to the CGM following the BEGIN METAFILE element.

METAFILE VERSION. Version 1 is the current CGM version.

METAFILE DESCRIPTION. Write the contents of IGES Global Section indices 21 (author), 22 (organization), and 18 (date and time of file generation) in this CGM element. If present, append a string provided in the configuration file.

VDC TYPE. Unless overridden by a specification of VDC TYPE integer in the configuration file, declare VDC TYPE to be real.

INTEGER PRECISION. Use the value provided in the configuration file, if present. Otherwise, set INTEGER PRECISION to 16 (bits).

REAL PRECISION. Use a value stored in GD-FP, which has been derived from Global Section indices 7-11 and 19, unless overridden by a specification in the configuration file.

INDEX PRECISION. Use the same value as for INTEGER PRECISION, unless overridden by a value provided in the configuration file.

COLOUR PRECISION. This element should not be placed in the metafile, because color specification mode **direct** is never used in IGES; consequently, the information contained in this element would never be used.

COLOUR INDEX PRECISION. Use the value equivalent to 4-bits (16 varieties) of color, because in the Technical Illustration Subset, only colors 0 through 8 (9 varieties) may be specified and no new indices can be set.

MAXIMUM COLOUR INDEX. Set this value to 8 (the maximum index, not the number of varieties).

METAFILE ELEMENT LIST. Set this value exactly to the list of elements that translator can in fact generate as a result of processing an IGES file.

METAFILE DEFAULTS REPLACEMENT. Follow the CALS application profile (AP) recommendations for CGM.

FONT LIST. Follow the CALS AP recommendations for CGM.

CHARACTER SET LIST. Do not generate this element.

CHARACTER CODING ANNOUNCER. Do not generate this element.

3.1.2 Picture Descriptor

A picture descriptor consisting of one instance of each of the following elements is written to the CGM following the BEGIN PICTURE element.

SCALING MODE. Scaling mode is always **metric**. The metric scale factor is taken from GD-MS.

COLOUR SELECTION MODE. Colour selection mode is always **indexed**. Because this is the CGM default, this element need not be generated.

LINE WIDTH SPECIFICATION MODE. This value is always **absolute**.

MARKER SIZE SPECIFICATION MODE. This value is always **absolute**.

EDGE WIDTH SPECIFICATION MODE. This value is always **absolute**.

VDC EXTENT. Initially use the values (-GD-XT, -GD-XT) and (+GD-XT, +GD-XT) as the two corners. During processing, the actual range of VDC coordinates will be monitored. During final copying of the CGM, the corner values in this element will be replaced by the actual values, rounded to convenient whole numbers and allowing for a border of approximately 5% of the total picture size.

BACKGROUND COLOUR. This element should not be generated, unless an explicit value is provided in the configuration file.

3.1.3 Control Elements

Following the BEGIN PICTURE BODY element, the following elements should be written at most once to the CGM.

VDC INTEGER PRECISION. This element is written only when VDC TYPE is **integer**. Use the value specified in the configuration file, if present; otherwise, use the value used for INTEGER PRECISION.

VDC REAL PRECISION. This element is written only when VDC TYPE is **real**. Use the value specified in the configuration file, if present; otherwise, use the value used for REAL PRECISION.

AUXILIARY COLOUR. This element is written only when the configuration file specifies a value--an index selected from the range 0 through 8.

TRANSPARENCY. This element is written--always with the value off--only when the configuration file specifies that transparency control is present.

CLIP RECTANGLE. Do not write this element.

CLIP INDICATOR. Do not write this element.

3.2 Geometric Entities

Entity 100 -- Circular Arc. Generally speaking, this maps either to CGM CIRCLE or CGM CIRCULAR ARC CENTRE, depending upon whether the start point and the end point are coincident (GD index 19 provides the measure of granularity to determine coincidence).

In both instances, the center point (X1,Y1) is directly available and the radius R must be calculated as SQRT[(X2-X1)**2 + (Y2-Y1)**2].

With CIRCULAR ARC CENTRE, the four additional VDC values needed by the CGM element are (X2-X1), (Y2-Y1), (X3-X1), and (Y3-Y1).

Entity 102 -- Composite Curve. A composite curve is a connected curve that results from the grouping of certain individual constituent entities into a logical unit. It is defined as an ordered list of entities of the following types: point (type 116), line (110), circular arc (100), conic arc (104), parametric spline (112), and rational B-spline (126).

Each constituent entity has its own transformation matrix and display attributes. Each constituent entity may have text associated with it. Each constituent entity may inherit attributes from a parent structure. Consequently, translator can process a composite curve by repetitively calling independent modules to process the more primitive entity types that comprise the composite curve.

Entity 104 -- Conic Arc. A conic arc is a bounded connected portion of a parent conic curve, which consists of more than one point. The parent conic curve is either an ellipse, a parabola, or a hyperbola.

The conic curve is defined by the six coefficients in the following equation:

A*X**2 + B*X*Y + C*Y**2 + D*X + E*Y + F = 0.

Field 15 of the associated DE is a form number. If DE-FO is non-zero, the form number specifies what kind of conic the parameter data represents. However, if DE-FO is zero, the form of the parent curve must be determined from the general equation.

In section 3.4.4 of reference 7, three quantities Q1, Q2, and Q3 can be computed from the six coefficients. These quantities are used as described in reference 7 to deduce the form of the parent curve.

If the parent curve is an ellipse, use the CGM ELLIPSE or ELLIPTICAL ARC element, depending upon whether the starting point and the ending point are coincident. Then, calculate the centerpoint and a set of conjugate diameter pairs using the parametric equations given in section 3.4 of reference 7.

If the parent curve is a parabola or hyperbola, then the CGM has no direct corresponding primitive element and a simulation routine must be called to generate a POLYLINE element. The number of line segments generated to approximate the parent curve is based on information obtained from the configuration file.

Entity 106 Form 11 -- Copious Data: Linear Planar Curve.

The Interpretation Flag should always be 1 and the Common Z Displacement should always be zero (0) for the Technical Illustration Subset. The remaining parameters map directly into the data needed for the CGM POLYLINE element.

Entity 106 Form 63 -- Copious Data: Simple Closed Area.

The Interpretation Flag should always be 1 and the Common Z Displacement should always be zero (0) for the Technical Illustration Subset. An additional syntax check on the IGES file may be performed to verify that X1=XN and Y1=YN. The remaining parameters map directly into the data needed for the CGM POLYGON element, with XN and YN being discarded for the CGM. An error message should be written to the error file if the IGES file fails the syntax check and the coordinate (XN,YN) should not be discarded.

Entity 110 -- Line. This element maps directly to CGM POLYLINE, where the number of points equals 2.

Entity 112 -- Parametric Spline Curve. The CGM lacks any kind of spline primitive. Consequently, translator must provide a full simulation, mapping each cubic polynomial segment into CGM

elements. Only if the specified segment is in fact a circular arc or elliptical arc can the corresponding CGM primitives be used. Otherwise, CGM POLYLINEs must be used. The length of the straight line segments generated by the curve drawing algorithm is controlled by a parameter in the configuration file.

Much of the mathematics needed for the approximation algorithm is found in reference 7, pages 132-137 and Appendix D.

In the Technical Illustration Subset, the splines are always planar.

Software to convert between parametric spline curves and the corresponding rational B-spline curves is available from the IGES office at the National Bureau of Standards. Materials provided include a magnetic tape of Pascal source code, a listing of the code, and accompanying documentation. **Translator** could use this software to convert from one spline representation to the other and support only one set of spline-drawing algorithms.

Entity 116 -- Point. The point entity is used to provide a position for general notes to be attached and for subfigures to be located. Other than picking up the coordinates of the point, no special processing is required. If the first PD value is 0, no processing is required unless there is a general note attached (see the discussion of Entity 212 below). If non-zero, the fourth PD value is a pointer to the DE of a subfigure instance (see the discussion of Entity 408 below).

The CGM POLYMARKER element cannot be used when translating IGES Technical Subset files to CGM files.

Entity 126 -- Rational B-Spline Curve. The CGM lacks any kind of spline primitive. Consequently, translator must provide a full simulation, mapping each cubic polynomial segment into CGM elements.

The form number associated with entity 126 indicates the nature of the spline. If form 1 (line), CGM POLYLINE can be used directly. If form 2 (circular arc), see the discussion of entity 100 above. If form 3, 4, or 5 (elliptical arc, parabolic arc, or hyperbolic arc), see the discussion of entity 104 above.

If the form number is 0, the form of the curve must be determined from the rational B-spline parameters. Only if the spline turns out to be, in fact, a circular arc or elliptical arc can the corresponding CGM primitives be used. Otherwise, CGM POLYLINEs must be used. The length of the straight line segments generated by the curve drawing algorithm is controlled by a parameter in the configuration file.

Much of the mathematics needed for the approximation algorithm is found in reference 7, pages 169-171 and Appendix D.

As noted before, software to convert between rational B-spline curves and the corresponding parametric spline curves is available from the IGES office at the National Bureau of Standards.

Entity 230 -- Sectioned Area. A sectioned area is a portion of a design that is to be filled with a pattern of lines. In addition to specifying the general pattern of lines (18 of them are available in IGES version 3.0--see figure 3-1), the IGES entity may also contain information concerning distance between the lines as well as the angle between the lines and the X-axis of the definition space. Sectioned areas may also contain islands (that is, smaller areas wholly contained within the sectioned area entity that are **not** to be filled with the specified line pattern).

Although not explicitly stated in the IGES standard, it is assumed that the definition of a sectioned area includes drawing its boundary.

In the CGM, the closest comparable element is POLYGON SET. If not for other problems, CGM POLYGON could be used if there were no islands.

In the general case, a complete simulation of Sectioned Area as POLYGONS and DISJOINT POLYLINES must be performed. POLYGON is used for the boundaries of the area and any islands; DISJOINT POLYLINE is used for each of the hatch pattern lines filling the interior. The CGM FILL INTERIOR STYLE should be empty, with the CGM EDGE VISIBILITY **on** and CGM edge attributes set according to the IGES line attributes set in the Directory Entry or inherited from any parent structures.

There are several complex aspects to the simulation algorithm:

1. Because the Sectioned Area boundary and enclosed islands can be curves (that is, represented by entities 100, 102, 104, and 106--perhaps even 112 and 126) not just polygons, the simulation logic must first convert the boundary curve and any island boundaries into their polygonal representations, using the same logic required for translating the basic entities, as described above.

2. Because the angle between the lines and the X-axis is in terms of definition space, this angle must be modified to reflect any rotation contained in the Transformation Matrix specifying the relationship between definition space and model space.

3.3 Geometric Attributes

Line Font Pattern. The IGES line font pattern, provided in DE field 4, indicates a display pattern to be used to display a geometric entity. In the IGES Technical Illustration Subset, four fixed line font patterns are defined: solid (1), dashed (2), phantom (3), and centerline (4). The first two patterns correspond directly to CGM's LINE (and EDGE) TYPEs 1 and 2. There are no CGM equivalents to the IGES phantom and centerline line font patterns. These must be emulated by using the CGM DISJOINT POLYLINE element.

Line Weight Number. The IGES line weight number, provided in DE field 12, denotes the thickness with which an entity should be displayed. A specific series of possible thicknesses are specified by GD-LW1 (global parameter 16) and GD-LW2 (global parameter 17). The largest thickness possible is that specified in GD-LW2 and is denoted by setting this value equal to the value in GD-LW1. The smallest thickness possible is equal to the result of dividing GD-LW2 by GD-LW1 and is denoted by setting this value equal to 1. Thicknesses between the smallest and largest thickness are available in increments equal to the smallest possible thickness and are denoted by setting this value equal to the integer number of (adjacent) increments required.

Thus, display thickness is:

(eq. 3-1) Line weight number * GD-LW2/GD-LW1.

A value of 0 indicates that the default line weight display of the receiving system is to be used.

As mentioned in section 3.1.2 above, CGM LINE WIDTH SPECIFICATION MODE and EDGE WIDTH SPECIFICATION MODE are always set to absolute. Consequently, the VDC value for LINE and EDGE WIDTH can be calculated by applying equation 3-1 above.

Color Number. IGES entities may use color indices 0 through 8. Color number 0 is not assigned any particular color and presumably corresponds to the natural background color of the receiving system's display. Color number 1 corresponds to black (RGB values: 0.0,0.0,0.0) while color number 8 corresponds to white (RGB values: 1.0,1.0,1.0). Colors 2 through 7 correspond to red, green, blue, yellow, magenta, and cyan respectively.

At the beginning of each picture, translator should put a CGM COLOR TABLE element that loads the appropriate RGB values into CGM color indices 1 through 8. Color index 0 should not be loaded, so that references to color index 0 can be resolved to the normal background color of the device onto which the picture will be imaged.

Hatch Patterns. As discussed under the Sectioned Area entity, in general, the CGM interior style hatch attribute setting cannot be used by **translator.** This is because the IGES entity may specify rotated patterns, the distance between lines, and variations in the angle of the pattern with respect to the X-axis of the display. However, when the distance and angles specified are standard, there is a chance to use CGM INTERIOR STYLE hatch. IGES section line pattern 1 corresponds to CGM hatch style 3, pattern 16 to hatch style 6, and pattern 18 to hatch style 5.

If it were expected that most IGES entities would not be specified with the distance and angle of the section line patterns altered, it might be worth defining the other 15 IGES section line patterns using the CGM PATTERN TABLE element. However, this approach will have limited value because many CGM interpreters do not yet support fully the pattern element capabilities of CGM.

3.4 Annotation

Entity 212 -- General Note. A general note entity consists of one or more text strings. Each text string contains text, a starting point, text size, text slant, and mirroring and angle of rotation information, either explicitly or implicitly. A single font number applies to the whole note and incorporates the separate concepts of type face (appearance of the characters; e.g., bold Helvetica, italic Futura) and character set (shape of the characters; e.g., ASCII, German National Set, Math, Greek). Only 7-bit character codes are supported.

The form number is used to select from 12 different layouts of the (possibly multiple) text strings. These layouts specify whether embedded text font changes need to be made, what the justification (left, center, right) of the strings within the specified text rectangle is, and whether there are subscripts, superscripts, and fractions.

Although, in general, the CGM text attribute elements are sufficiently powerful to represent all desired IGES general note facilities, two aspects of IGES notes--the symbol fonts and text slant--cannot be realized by CGM elements. Consequently, a faithful rendering of IGES general notes requires a complete stroke text simulation facility. The rules to be followed by such a facility are found on pages 223-237 of reference 7.

Implementation of a simulation facility should be divided into two stages. First, from the form number, the box location, the box height and box width, and the box rotation angle, the position of each individual text string can be found. Second, the individual characters are stroked out, based on the font number, slant angle, mirror flag, and rotate-internal-text flag

specified with the entity. Of course, any description space to model space transformations must be applied to all position and size calculations during the simulation.

Text Font. The IGES Technical Illustration Subset requires the support of three fonts: numbers 1, 1001, and 1002. In both IGES and CGM, font 1 is used to refer to any font that can represent the 7-bit ASCII character codes. The appearance of the font (e.g., stick figures, filled characters, bold weight) is not specified by the standards.

IGES fonts 1001 and 1002 are called "symbol fonts." The numerals, uppercase letters, and punctuation marks all correspond to ASCII. However, the 26 lowercase letters are mapped to 26 different symbols (see figures 3-2 and 3-3). These symbols correspond to no registered 7-bit character code and consequently are unlikely to be supported in any graphics device. Therefore, simulation of these characters by stroking out the shape is the only approach for translation.

Other Text Attributes. Only if the slant angle is pi/4 and if the font number is 1 could CGM TEXT elements be used, instead of POLYLINEs that approximate the stroked text. CGM TEXT and APPEND TEXT in conjunction with the CGM attributes of TEXT PRECISION, TEXT COLOUR, TEXT HEIGHT, TEXT ALIGNMENT, CHARACTER EXPANSION FACTOR, and CHARACTER SPACING, take care of the stage 1 considerations. CGM TEXT FONT, TEXT PATH, and CHARACTER UP VECTOR permit mirroring of the text sometimes needed in stage 2 to be accomplished.

3.5 Structures

CGM has no hierarchical structuring and instancing capability. Consequently, none of the IGES structuring facilities can be directly represented in the CGM. Instead, each instance of an object must be fully expanded and stored explicitly in the CGM. Any mappings from definition space to model space to drawing space must be performed on coordinates in order to derive the CGM VDC coordinates (VDC corresponds to IGES drawing coordinates).

The simulation capabilities required of translator are extensive and complex, especially when dealing with the nesting and stacking of the transformations, which are based on the settings of the status digits in DE field 9.

Entity 124 -- Transformation Matrix. The IGES Technical Illustration Subset permits only rotations and translations of 2D objects.

Transformation Matrices (TM) are used in only two situations in the Technical Illustration Subset:

1. Mapping definition space coordinates into model space coordinates: DE field 7 contains a pointer to the TM entity, and

2. Mapping model space coordinates into viewing space coordinates.

Successive coordinate system changes are specified by allowing a TM entity to reference another TM entity through its DE field 7. Such matrices are composed by applying the second matrix to the first using left multiplication of the matrices.

In the IGES Technical Illustration Subset, **translator** should see only Form number 0 matrices, because left- or right-handedness is irrelevant when dealing solely with 2D objects and because IGES node entities (type 134) are not included in the Subset.

Entity 404 -- Drawing. A drawing is a collection of **annotation entities** (i.e., any entity with use flag--DE field 9, digits 5-6--set to 01) defined in drawing space, and **views** (i.e., projections of model space data in view space), which together constitute a single representation of a part, in the sense that an engineering drawing constitutes a single representation of a part in standard drafting practice. Views are specified by referring to a View Entity--see discussion below.

Drawings are located in drawing space as illustrated in figure 3-4, with sides coincident with the drawing coordinate system axes, and with the lower left corner at the drawing space origin. The drawing space coordinate system is a special 2D system used for view origin locations in the Drawing Entity and for annotation entities referenced by the Drawing Entity.

Annotation entities can be defined in drawing space and be referenced by the Drawing Entity directly or can be defined in model space and appear in individual views. When defined in drawing space, the subordinate entity switch should be set to **physically dependent** (01). The subordinate entity switch for a View Entity referenced by a Drawing Entity should be set to **logically dependent** (02).

The transformation of a view from view space to drawing space is controlled by the view scale factor, specified in the View Entity, and the view drawing locations, specified in the Drawing Entity.

In the Technical Illustration Subset, the drawing units will be the same as the model units, and the size of the drawing may not be communicated to the receiving system.

The following values are given in drawing units:

- view origin drawing locations

- coordinates of annotation entities referenced directly.

Entity 410 -- View. The View Entity specifies the view matrix and the translation vector by use of a pointer to a Transformation Matrix in field 7 of the Directory Entry.

In the Technical Illustration Subset, the scale factor is always 1.0 and the view volume pointers are always 0, which implies that no clipping of the objects in the view takes place as the objects are mapped to the drawing space.

Note that the value of field 6 of the DE for an entity controls whether that entity is displayed in a particular view.

Entity 308 -- Subfigure Definition. The subfigure definition entity supports the concept of a subpicture, if you equate drawing creation with graphics picture processing. This entity permits a single definition of a detail to be utilized in multiple instances in the creation of the whole picture.

Subpictures can be nested: the contents of the subfigure definition may include a set of pointers to any combination of entities and other subfigures. The depth of subfigure nesting is provided in the first parameter data field.

Translator expands subfigures by calling procedures that expand each entity in the definition, applying the proper coordinate transformations and attribute inheritance rules at each stage.

Entity 408 -- Subfigure Instance. This entity defines the occurrence of a single instance of the specified subfigure. The first parameter data field points to the subfigure definition. The remaining data specifies scale and translation information to be applied as the subfigure definition is expanded and the entities comprising the subfigure are placed in the CGM.

In a drawing, subfigures are displayed in all views; they cannot be constrained to appear only in some views.

Entity 412 -- Rectangular Array Subfigure Instance. The rectangular array produces copies of an object called the base entity, arranging them in equally spaced rows and columns. The following types of entities from the Technical Illustration Subset are valid for use as a base entity: point (type 116), line (110), circular arc (100), conic arc (104), parametric spline curve (112), rational B-spline curve (126), any annotation entity (212, et. al.), rectangular array instance (412), circular array instance (414), or subfigure definition (408).

The number of rows and columns of the rectangular array, together with their respective horizontal and vertical displacements are given in the parameter data. Also, the coordinates of the lower left hand corner for the entire array is indicated in the parameter data. This is where the first entity in the reproduction process is placed and is called position 1. The successive positions are counted vertically up the first column, then vertically up the second column, and so on.

The whole array of instance locations for the base entity can be specified as being rotated about the line through the lower left hand point. However, the instances of the base entity are not rotated from their original orientation.

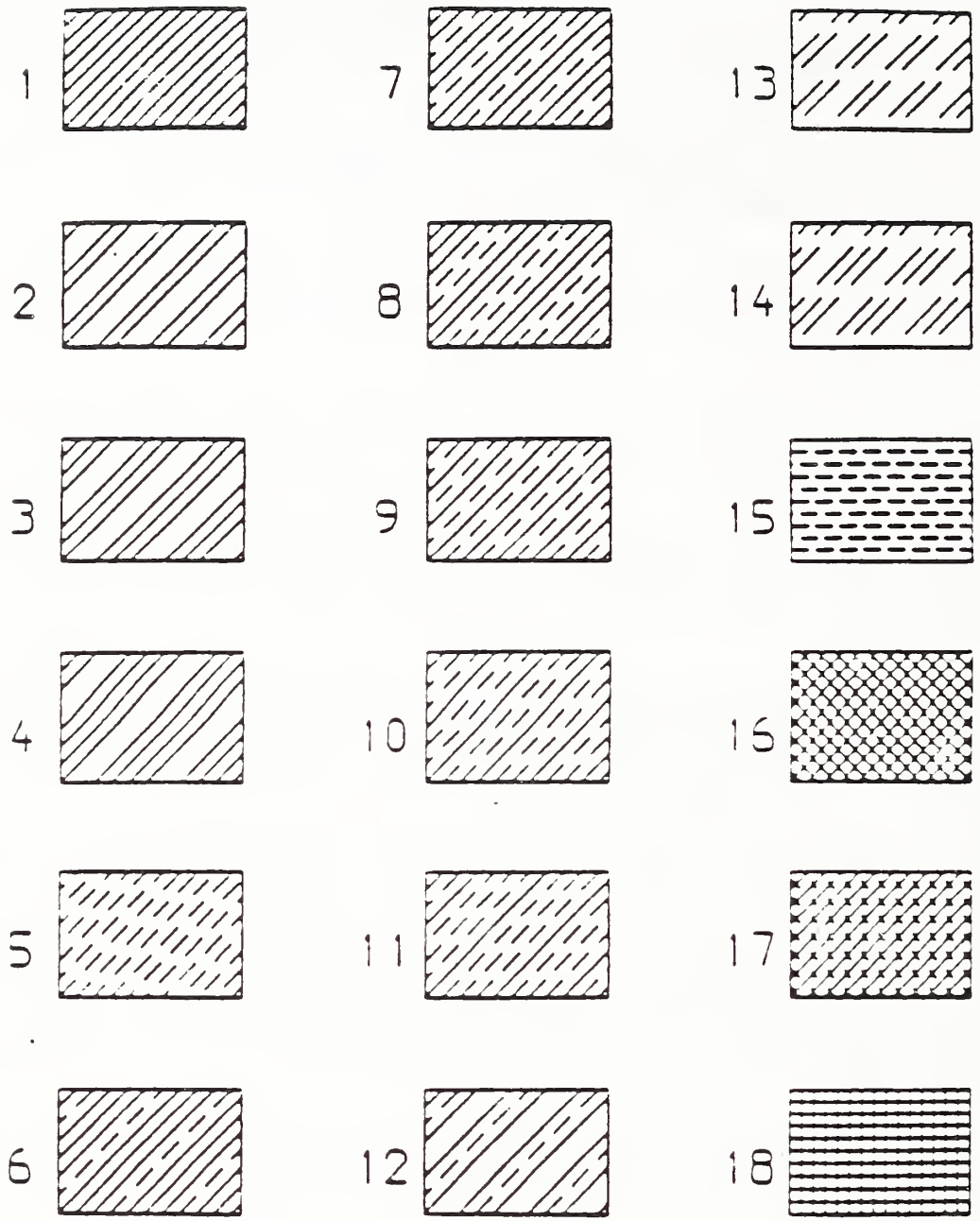
A special flag associated with the parameter data enables one to display only a portion of the instance array. You may display either the first n instances or the last m instances.

The algorithm needed for subfigure expansion is straightforward if tedious. Repeated subroutine calls, stacking and applying of transformations, and changing of individual attributes is required.

Entity 414 -- Circular Array Subfigure Instance. The circular array subfigure instance entity produces copies of the base entity, arranging them around the edge of an imaginary circle whose center and radius are specified. The same types of entities may be used with entity 414 as with entity 412.

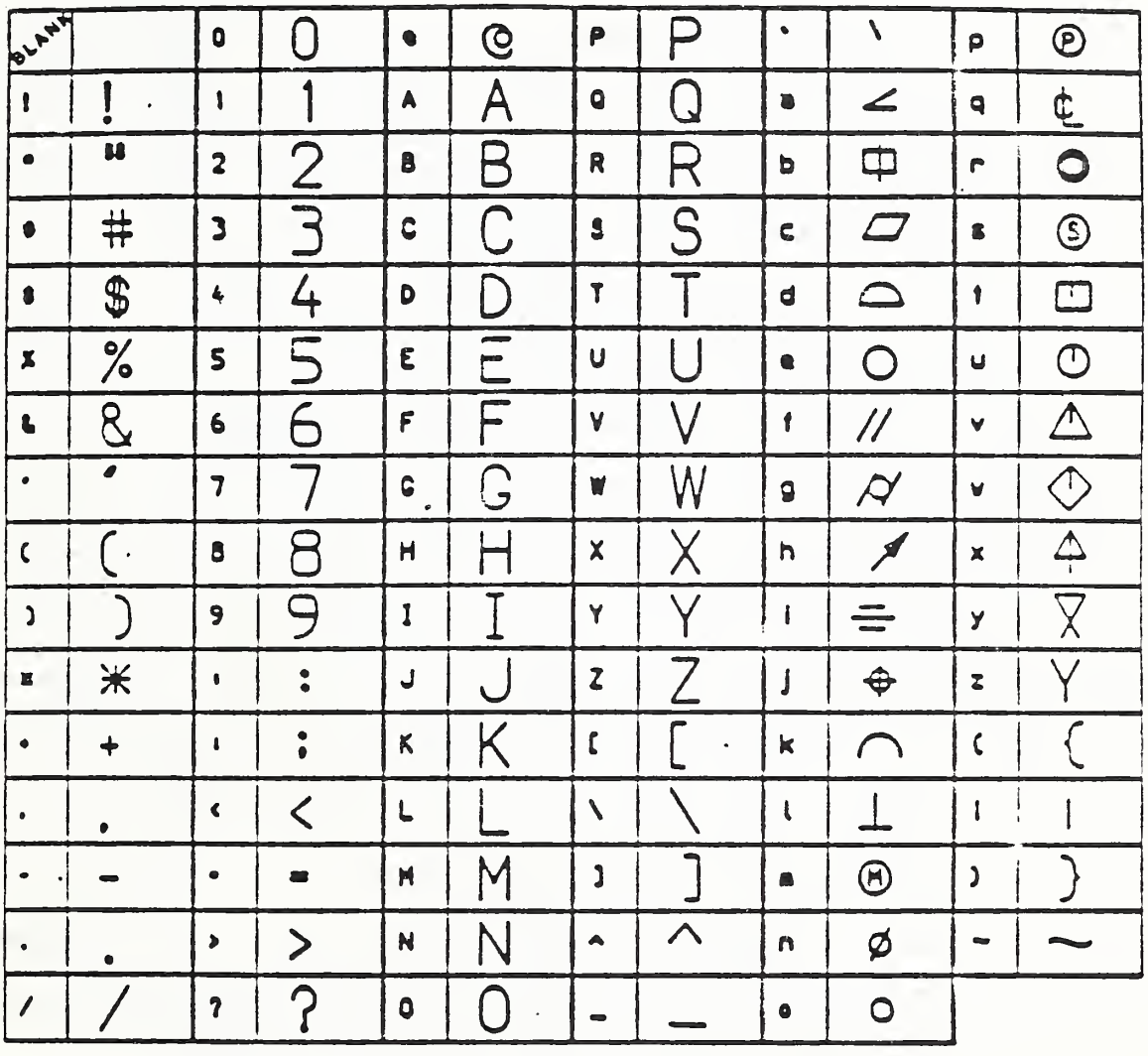
The number of possible instance locations for the base entity is specified and the location of the first instance position is specified in terms of a radius and a start angle measured positive, counterclockwise in radius from the line through the center parallel to the X-axis. The successive positions follow a counterclockwise direction around the imaginary circle and are distributed according to the delta angle specified in parameter data field 8.

As with the previous entity, a special flag associated with the parameter data enables one to display only a portion of the instance array. You may display either the first n instances or the last m instances.



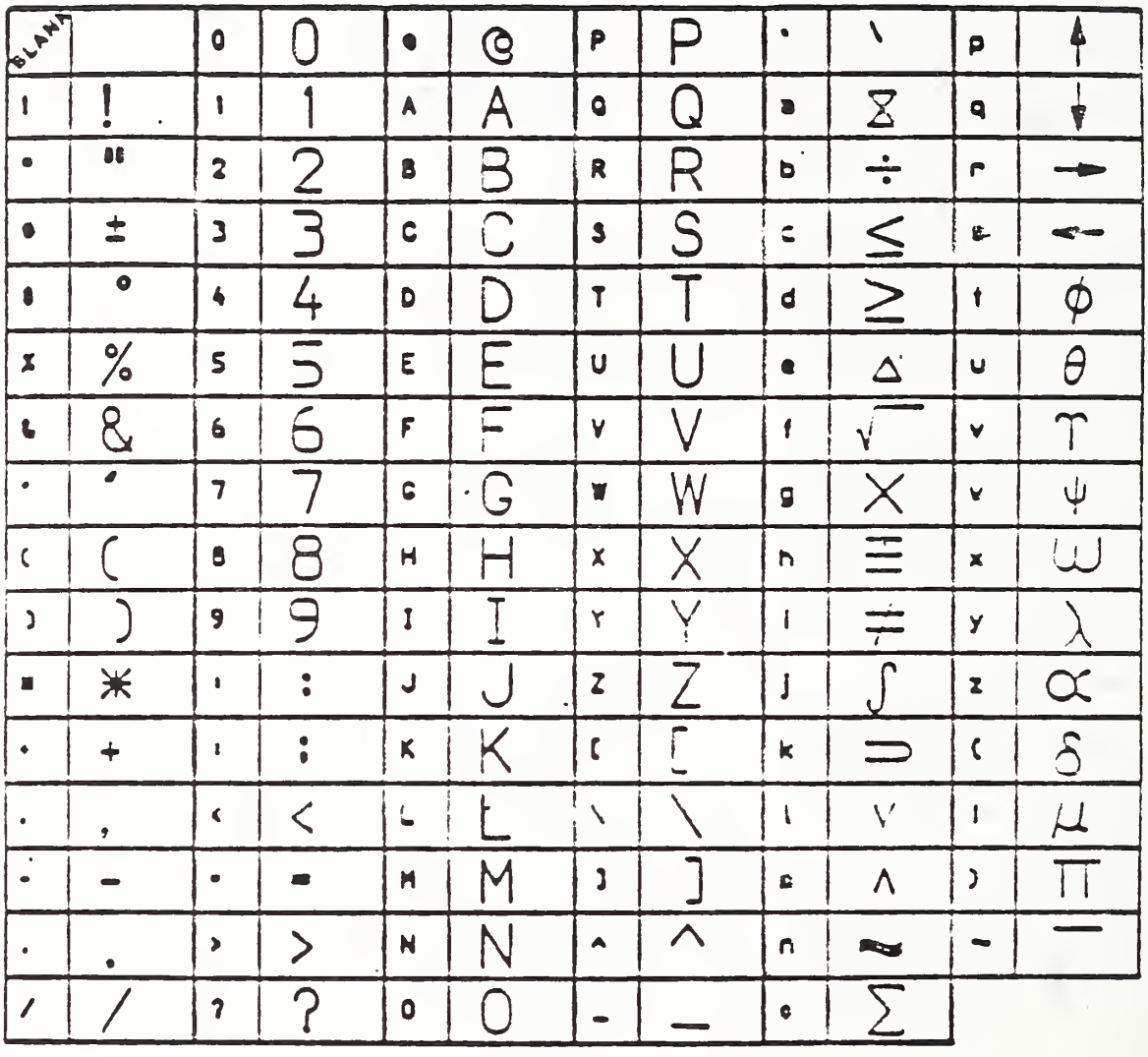
LINE PATTERN CODES

FIGURE 3-1 SECTION LINE PATTERNS



FONT CODE 1001

FIGURE 3-2



FONT CODE 1002

FIGURE 3-3

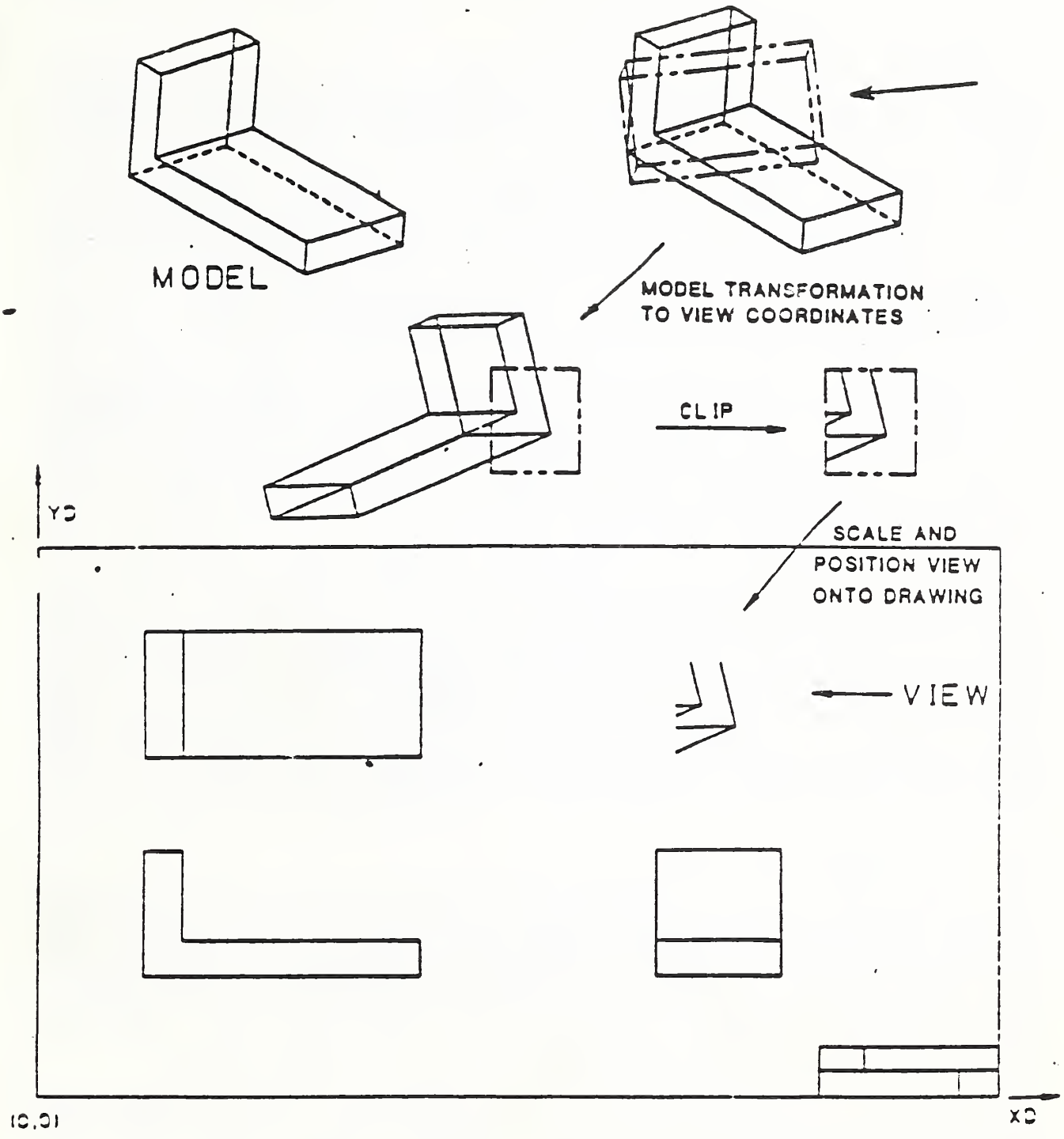


FIGURE 3-4 DRAWING ENTITY

IV. IMPLEMENTATION RECOMMENDATIONS

1.0 Implementation Alternatives

To encourage the development of IGES to CGM translators, the Government has three principal alternatives, plus a derivative alternative strategy if it already owns IGES translator software.

Alternative 1. Contract for Translator. Based on this report, the Government could solicit bids by private developers to develop an IGES-to-CGM translator. The deliverable would be the source code for the program, owned by the Government, and made available to all organizations via such a mechanism as NTIS. The Government would incur the entire development cost, but the developer would retain no further rights in the software.

Alternative 2. Wait for Private Development. The Government could publicize the need for an IGES-to-CGM translator, disseminate this report--and others relating to CALS requirements, and wait for private industry to develop the product. The developer would incur the entire development cost, but the government would have to pay for each copy of the translator it needed.

Alternative 3. Issue an RFQ/RFP for Volume Purchase. Based on this report, the Government could indicate its need for a large number of copies of such a translator and request that industry respond with a bid. The exact form of the purchase could be on a site-license basis or volume-purchase basis. In either of these cases, all Government agencies needing the translator would be able to get the product inexpensively and promptly, but the developer would retain the rights to sell the software to private organizations and to other, non-US government agencies.

2.0 Discussion

Alternative 1 is likely to be most costly in the short run (say, 1 year), but it is likely to provide the product most quickly and with the greatest probability that the Government's requirements are met in full. It might be less expensive than Alternative 2 and, perhaps, Alternative 3 in the long run (say 5 years). However, Alternative 1 requires the Government to take on maintenance responsibilities over the life of the product.

Alternative 2 is least costly in the short run, but it might not even come to pass in a timely fashion without some economic encouragement by the Government.

Alternative 3 is the most likely to provide a cost-effective solution in a timely manner. Government and industry share the

risk and the cost. The Government provides a guaranteed initial market, while industry provides the initial capital investment and shoulders the maintenance responsibility.

Alternative 4. Develop from Existing IGES Software. A fourth alternative does exist if the Government already owns software that provides a quick-view capability for IGES files. Under contract to private parties or developed in-house, the Government could modify such software by replacing its calls to graphics drawing primitives to calls to generate equivalent CGM elements. Most drawing packages are less rich than the CGM in their collection of graphics primitives and attributes, so the replacement process would be a relatively straightforward one.

Overall, the translator task consists of about 60% IGES interpretation, 25% IGES-to-CGM element mapping, and 15% CGM generation. The complexity lies about 40% with IGES, 40% with the IGES-to-CGM element mapping (because of all the emulation required), and 20% with CGM. The cost of developing from scratch a full CGM generation capability, tested and reliable, is probably about $75K-$100K. A similar interpretation capability for the IGES Technical Illustration Subset would cost over $250K.

Consequently, NBS estimates that the cost to provide an IGES to CGM translator in source code form to the Government for distribution as public-domain software is about $350K.

Following Alternative 3, the Government might be able to get a volume purchase license for such software for about $150K-$200K from suppliers that either already had developed the CGM capability or the IGES translation capability. The Government would have to agree not to make copies available to its Contractors as GFE, because the suppliers would be looking at the Contractors as the target market where it would make up the remaining development costs and pay for maintenance and enhancement of the software.

Of course, the details of the terms and conditions that might be required are too extensive and complex to be discussed in a report of this type. Furthermore, lessons might be learned from the Government's purchase of other, popular software like word processing and spreadsheet software for the PC.

VI. SUMMARY AND CONCLUSIONS

Previous NBS CALS reports have demonstrated that the CGM is the most appropriate and cost-effective way to store and transmit device-independent picture descriptions for use in Technical Illustration and Technical Publishing application environments. The CGM is also the basis for importing graphics into "compound" documents, as specified in forthcoming ISO standards for document architecture (ODA/ODIF).

These previous reports argue for the ability to convert technical drawings represented as IGES product definition files into pure picture descriptions represented as Computer Graphic Metafiles (CGMs).

This report develops the design specifications for an IGES-to-CGM translator utility program. In particular, the detailed mapping of the IGES Technical Illustration Subset to the CGM standard (ANSI X3.122-1986) is described and explained. In addition, the program structure and principal internal processing steps for such a program are outlined.

Finally, procurement alternatives are proposed for implementing the IGES-to-CGM translator utility program. These alternatives include:

1. Contract for Translator;

2. Wait for Private Development;

3. Issue an RFQ/RFP for Volume Purchase;

4. Develop from Existing IGES Software.

Pros and cons for each alternative are discussed. NBS would like to develop this translator for CALS. However, the funding needed for such an effort ($350K), makes this task very uncertain. NBS will await feedback from DOD CALS as to the cost benefits and projected use of such a translator in relation to the high cost of its proposed development.

GLOSSARY

character set	The set of displayable symbols mapped to individual character codes in a text string. A character set is independent of the font or typeface.
color	In the context of this report, in addition to its ordinary meaning, the word color includes bi-level black-and-white (so called, monochrome) systems and multilevel gray-scale systems.
color table	A table for use in mapping from a color index to the corresponding color.
control elements	Metafile elements that specify metafile delimiters, address space, clipping boundaries, picture delimiters, and format descriptions of the metafile elements.
descriptor elements	Metafile elements that describe the functional content, format, default conditions, identification, and characteristics of a metafile.
device-dependent	A system or portion of a system that contains logic, algorithms, or data that are consistent with the behavior of a specific graphical device.
device-independent	A system or portion of a system that contains logic, algorithms, or data that do not require nor represent knowledge about the behavior of any particular graphical device.
device coordinates	The coordinates native to a device; device-dependent coordinates; physical device coordinates.
direct color	A color selection scheme in which the color values are specified directly, without requiring an intermediate mapping via a color table.
escape functions	Graphical functions that describe device-dependent or system-dependent elements used to construct a picture, but that are otherwise not standardized.

external functions — Functions present in some graphics standards that communicate information not directly related to the generation of a graphical image.

IGES — Initial Graphics Exchange Specification. A mechanism for exchanging product data information in human-readable form.

metafile — A mechanism for retaining and transporting graphical data and control information. This information contains a device-independent description of one or more pictures.

metafile generator — The process or equipment that creates a metafile. Also, **CGM generator.**

metafile interpreter — The process or equipment that reads a metafile and interprets the contents to produce again the picture represented in the metafile. Also, **CGM interpreter.**

model space — A two-dimensional Euclidean space in which an IGES product model is described.

normalized device coordinates (NDC) — Coordinates specified in a device-independent coordinate system, normalized to some range (typically 0 to 1).

pixel — The smallest element of a display surface that can be independently assigned color.

prior agreement — A process whereby the creator of a metafile and the intended recipient of the metafile come to some understanding regarding the content or format of the metafile, that understanding not being recorded in the metafile itself. In a blind interchange environment, prior agreement can be used to overcome limitations of exchange standards.

segment — A collection of graphical functions that can be manipulated as a unit. Once functions are grouped into segments, they are referred to as segment elements.

translator	In the context of this report, a utility program that converts a picture represented as an IGES file, conforming to the IGES Technical Illustration Subset, to a picture represented as a conforming Computer Graphics Metafile.
world coordinates	Coordinates specified in a device-independent coordinate system, whose units are selected by and are meaningful to the client.

VII. REFERENCES

1. "CALS FY-86 Final Progress Report," National Bureau of Standards Report, November 1986.

2. "Raster-to-Vector Conversion: A State-of-the-Art Assessment," Final Report to OSD, CALS SOW Task 2.2.2.2.2, September 1987.

3. "CALS Core Requirements (Phase 1.0)," Coordination Draft, 24 March 1987.

4. "Automated Interchange of Technical Information," MIL-STD-1840A, Draft, 24 March 1987.

5. "IGES Application Subset for Technical Illustration," DOD-D-(28000), Appendix A.

6. "Information Processing Systems--Computer Graphics--Metafile for the Storage and Transfer of Picture Description Information," ANSI/X3.122-1986 and ISO 8632-1987.

7. "IGES Version 3.0," National Bureau of Standards Report, NBSIR 86-3359, April 1986.

NBS-114A (REV. 2-8C)

U.S. DEPT. OF COMM. BIBLIOGRAPHIC DATA SHEET (See instructions)	1. PUBLICATION OR REPORT NO. NBSIR 88-3727	2. Performing Organ. Report No.	3. Publication Date MARCH 1988

4. TITLE AND SUBTITLE

A Collection of Technical Studies Completed for the Computer-aided Acquisition & Logistic Support (CALS) Program - Fiscal Year 1987 (Vol. 2 of 4)

5. AUTHOR(S)

Edited by Sharon J. Kemmerer

6. PERFORMING ORGANIZATION (If joint or other than NBS, see instructions)

NATIONAL BUREAU OF STANDARDS
U.S. DEPARTMENT OF COMMERCE
GAITHERSBURG, MD 20899

7. Contract/Grant No.

8. Type of Report & Period Covered
NBSIR, 10/86 through 9/87

9. SPONSORING ORGANIZATION NAME AND COMPLETE ADDRESS (Street, City, State, ZIP)

Office of Assistant Secretary of Defense (A&P)/WSIG
Department of Defense, Pentagon
Washington, DC 20301-8000

10. SUPPLEMENTARY NOTES

☐ Document describes a computer program; SF-185, FIPS Software Summary, is attached.

11. ABSTRACT (A 200-word or less factual summary of most significant information. If document includes a significant bibliography or literature survey, mention it here)

The overall objective of the Department of Defense Computer-aided Acquisition & Logistic Support (CALS) Program is to integrate the design, manufacturing, and logistic functions through the efficient application of computer technology. The National Bureau of Standards has been funded since Spring 1986 to recommend a suite of industry standards for system integration and digital data transfer, and to accelerate their implementation. A major FY87 thrust was the completion of initial documentation of the high-priority standards required in the CALS environment. This volume contains a collection of the final graphics reports presented to the CALS Policy Office. The other three volumes contain additional graphics reports as well as technical reports on text, product data, data management, raster compression, media, and conformance testing.

12. KEY WORDS (Six to twelve entries; alphabetical order; capitalize only proper names; and separate key words by semicolons)

application profile; CALS; CGEM; CGM; DoD: graphics; IGES; raster; vector

13. AVAILABILITY

☒ Unlimited

☐ For Official Distribution. Do Not Release to NTIS

☐ Order From Superintendent of Documents, U.S. Government Printing Office, Washington, D.C. 20402.

☒ Order From National Technical Information Service (NTIS), Springfield, VA. 22161

14. NO. OF PRINTED PAGES
474

15. Price
$36.95

USCOMM-DC 6043-P80


AMERICAN BINDERY · EAST
1993