NISTIR 4806

Procedures Manual for Testing CGM Generator Products That Claim Conformance to FIPS 128 and MIL-D-28003

Daniel R. Benigni
Editor

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

NIST

NISTIR 4806

Procedures Manual for Testing CGM Generator Products That Claim Conformance to FIPS 128 and MIL-D-28003

Daniel R. Benigni
Editor

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

March 1992

U.S. DEPARTMENT OF COMMERCE
Barbara Hackman Franklin, Secretary

TECHNOLOGY ADMINISTRATION
Robert M. White, Under Secretary for Technology

NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY
John W. Lyons, Director

Procedures Manual For Testing CGM Generator Products That Claim Conformance to FIPS 128 and MIL-D-28003

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

EXECUTIVE SUMMARY

PURPOSE

This report was prepared by NIST in support of the Computer-aided Acquisition and Logistics Support (CALS) initiative. It represents a particular FY91 contract deliverable task to develop a tool to determine conformance of a CGM (Computer Graphics Metafile) Generator to the CALS Application Profile, military specification MIL-D-28003.

BACKGROUND

It was known from the beginning of the CALS work, back in 1986, that the CGM standard offered no conformance statements concerning either writers (generators) or readers (interpreters) of metafiles. An international workshop on CGM Certification held in the United Kingdom in March of 1987 concluded that "a CGM Testing Architecture must include testing for CGM generators and interpreters."

Although work has concentrated on developing the Application Profile for CGM in CALS (namely MIL-D-28003), over the last three years the National Institute of Standards and Technology (NIST) has developed a testing methodology to test metafiles to both the CGM Federal Information Processing Standard (FIPS 128) and the MIL-D-28003. A test tool has been developed, and a conformance testing service for metafiles recently (May 1991) began. This is the first of three parts of a total CGM system conformance test suite. The other two parts necessary for a complete CGM system conformance test suite entail: verifying that a CGM generator produces conforming metafiles which accurately and correctly define the intended picture, and ensuring that a CGM interpreter can correctly and completely parse any CGM file and produce the intended picture.

MIL-D-28003 specifies that a CGM Application Profile contains three parts: the metafile; the generator; and the interpreter. Now that testing metafiles has begun, the next logical phase of this work involves developing procedures for testing conformance for generators both to the CGM and to MIL-D-28003. The Procedures Manual provides the details necessary for performing this function.

DISCUSSION

The objective of the generator testing program is to determine whether a given product, in this case a CGM generator, always produces picture interchange files (Computer Graphic Metafiles--CGMs) that conform to the FIPS 128 CGM standard and

the DOD CALS CGM Application Profile (MIL-D-28003). The tradeoff has been to be able to perform this job with a reasonable likelihood of correctness while at the same time at a cost commensurate with the value.

Correctness has two aspects: syntactic correctness and semantic correctness. Syntactic correctness can be determined by examining each CGM produced by the tested product and verifying that it meets all the rules including such considerations as the order of the elements, how each element is coded, and whether required elements are present.

Semantic correctness is more difficult to assess. The tester must compare the picture produced from the coded CGM with the original picture that the tested product intended to store in the CGM. Differences between the pictures either indicate a semantic error in the CGM generator or can be explained as variations across implementations--variations that are allowed by the standard.

Generally speaking, it is impossible to test "all possible" CGMs any given generator is capable of producing. Therefore, one must test an appropriate sample (subset) of all possible CGMs and look for errors in syntax or semantics. Determining and specifying this "appropriate subset" of CGMs is the main challenge of the test service.

The CGM Testing Methodology is based on the concept of falsification testing. This concept refers to a process that proceeds as follows:

(a) Determine the characteristics of the product being tested.

(b) Specify a certain set of initial conditions (and inputs), each of which causes the generation of one particular result (output).

(c) Examine each output for correctness--first syntactic correctness, then semantic correctness.

(d) If no discrepancies are noted, declare the product "correct."

Falsification testing has known weaknesses. The most important of these is that the absence of errors does not guarantee conformance. But for many testing situations, it is the only method that is practical and whose costs are commensurate with the value to be obtained by testing. For testing CGM generators for conformance to FIPS 128 standard and the CALS MIL-D-28003 application profile, it is the only practical method that has been identified.

CALS USE/IMPACT

This Procedures Manual provides a complete methodology for how to test generators for conformance to MIL-D-28003. However, due to the release of the revision of MIL-D-28003, known as MIL-D-28003A, this manual must be updated to account for the additional functionality contained in the new revision.

The procedures manual does:

- identify the appropriate number, both in quantity and type, of metafiles necessary to verify correctness;
- specify the different metafiles needed for testing; and
- specify guidelines for the tester and implementor to determine correctness of the imaged files.

RECOMMENDATIONS

The NIST Computer Systems Laboratory (CSL) recommends strongly that a full testing program for MIL-D-28003 requires testing of CGM generators, and that testing must be done to up-to-date versions of both the CGM standard and MIL-D-28003. As new versions of the CGM standard and MIL-D-28003 are released, a maintenance effort is required to update the Procedures Manual, test suite, and supporting test tools to be of use to CALS in the future.

NIST/CSL also recommends that this Procedures Manual be used in an actual testing environment to determine its effectiveness in testing CGM generators.

ACKNOWLEDGMENT

The editor would like to acknowledge the major technical contributor to this report, Dr. Peter R. Bono of Peter R. Bono Associates, Inc.

NOTES

The CGM Formal Description explained in Chapter 2 does not represent certain relationships between CGM elements. Inquiring about these special cases is not covered by the Questionnaire. Therefore, no special instructions are included to require sponsors of generators-under-test to produce CGM files that show the special relationships. These special cases, not covered by these testing procedures, are:

(a) Specifications of two VDC corner points in VDC EXTENT and CLIP RECTANGLE, where the relationship of the first point to the second point (i.e., below and to the left, above and to the left, above and to the right, below and to the right) is at issue.

(b) Special meanings of characters with a string (e.g., formatting characters).

(c) Specifications of circular arc 3 point close with various degenerate conditions:

  (c1) Only one distinct point is given.
  (c2) Two distinct points and the intermediate point coincides with the starting point.
  (c3) Two distinct points and the intermediate point coincides with the ending point.
  (c4) Two distinct points and the starting point coincides with the ending point.
  (c5) Three collinear points and the intermediate point lies between the starting point and the ending point.
  (c6) Three collinear points and the intermediate point does not lie between the starting point and the ending point.

(d) ELLIPSE elements where the start ray and end ray coincide.

(e) CHARACTER ORIENTATION elements where the character up vector and the character base vector are not at right angles to each other.

(f) COLOUR elements whose default is a function of the "device dependent background colour" or "device dependent foreground colour."

(g) Attribute elements (MARKER SIZE, LINE WIDTH, CHARACTER HEIGHT, EDGE WIDTH, FILL REFERENCE POINT, and PATTERN SIZE) whose defaults are a function of VDC EXTENT.

1. Testing Methodology: Overview and Definitions

1.1 Testing Philosophy

1.1.1 Objective

The objective of this testing program is to determine, within a reasonable likelihood of correctness and at a cost commensurate with the value, whether a given product always produces picture interchange files (Computer Graphic Metafiles--CGMs) that conform to FIPS 128 and/or the DOD CALS CGM Application Profile (MIL-D-28003).

1.1.2 Basic Concepts

Correctness has two aspects: syntactic correctness and semantic correctness. Syntactic correctness can be determined by examining each CGM produced by the tested product and verifying that it meets all the rules including such considerations as the order of the elements, how each element is coded, and whether required elements are present.

Semantic correctness is more difficult to assess. The tester must compare the picture produced from the coded CGM with the original picture that the tested product intended to store in the CGM. Differences between the pictures either indicate a semantic error in the CGM generator or can be explained as variations across implementations--variations that are allowed by the standard.

1.1.3 ISO 10641

ISO 10641, "Conformance Testing of Implementations of Graphics Standards," is now at the Draft International Standard (DIS) stage. It provides some general guidance regarding the aims and objectives of a Graphics Test Suite and a Graphics Test Service. It also suggests procedures and guidelines for the establishment and operation of such a Test Service.

1.2 Definitions

Words or phrases defined in this section are set in boldface type wherever they are used in the remainder of this manual.

1.2.1 CGM Generator

A **CGM generator** is a program, process, or product that can write CGM (FIPS 128) files, which represent a graphical picture assembled from information gathered or calculated by the program, process, or product. Some **CGM generators** always operate in **CALS mode** or can be made to operate in **CALS mode.**

1.2.2 CALS Mode

When a **CGM generator** produces CGMs that always conform to the additional constraints of the CALS MIL-D-28003 CGM application profile, the **CGM generator** is said to be operating in **CALS mode.**

1.2.3 CGM-Under-Test

In the context of this test service, a **CGM-under-test** is a CGM file that is being examined to determine whether it indeed satisfies the requirements of FIPS 128 alone or the requirements of both FIPS 128 and MIL-D-28003.

1.2.4 Generator-Under-Test

In the context of this test service, a **generator-under-test** is a product being tested to determine if all CGMs it is capable of writing in fact conform to FIPS 128 or, when it operates in **CALS mode**, conform to MIL-D-28003.

1.2.5 CGM Suite-Under-Test

The **CGM suite-under-test** is the total collection of individual CGM files that will be examined in order to determine the correctness of the **generator-under-test.**

1.2.6 Native Suite

The **native suite** is a collection of files, generally in a format proprietary to the specific application which contains the CGM generation capability. The **native suite** is that collection of files which the CGM **generator-under-test** uses to produce the **CGM suite-under-test.**

1.2.7 CGM Hardcopy

To determine the semantic correctness of a **CGM-under-test**, the test service uses a **reference interpreter** to read the **CGM-under-test** and produce a pictorial realization of the **CGM-under-test.** This picture will be called the **CGM hardcopy**--although it may appear only transitorily on a display screen.

1.2.8 Native Hardcopy

The **CGM hardcopy** is compared with the picture intended to be interchanged by the **generator-under-test.** This picture, known as the **native hardcopy**, is produced by the **generator-under-test** and may be truly in hardcopy form or be visible only on a display screen. There is one **native hardcopy** for each file in the **native suite.** There is one **CGM hardcopy** for each file in the **CGM**

suite-under-test. There is a one-to-one correspondence between the set of **native hardcopies** and the set of **CGM hardcopies**.

1.2.9 Reference Interpreter

In the context of this test service, a **reference interpreter** is a program that reads **CGMs-under-test** and produces correct pictorial realizations of their contents. Ideally, the **reference interpreter** has been designed to satisfy the assertions of a test requirements document, which itself has been derived directly from the standard document. **Reference interpreters** should have been thoroughly checked by the developers and testers and also should have endured considerable field use.

1.2.10 Test Suite Description

The verbal specifications that outline the requirements to be satisfied by the CGM **suite-under-test** are known collectively as the **test suite description**. The testing process requires that the CGM **suite-under-test** adhere to the specifications in the **test suite description**.

1.2.11 Test Image

To supplement the **test suite description**, the Test Service has developed a suite of **test images** (see Appendix D). The testing process asks that the CGM **suite-under-test** contain as many of these **test images** as possible, matching the **test image** appearance and content as closely as possible. An absolute requirement cannot be set because not all **generators-under-test** are capable of creating all **test images**.

1.3 Testing Methodology

Generally speaking, it is impossible to test "all possible" CGMs any given product is capable of producing. Therefore, one must test an appropriate sample (subset) of all possible CGMs and look for errors in syntax or semantics. Determining and specifying this "appropriate subset" of CGMs is the main challenge of the test service. Once the **CGM suite-under-test** is selected and **native hardcopies** obtained, it is a relatively straightforward procedure to check for syntactic correctness (see section 1.3.2.1) and semantic correctness (see section 1.3.2.2).

1.3.1 Falsification Testing

The CGM Testing Methodology is based on the concept of falsification testing. This concept refers to a process that proceeds as follows:

(a) Determine the characteristics of the product being tested.

(b) Specify a certain set of initial conditions (and inputs), each of which causes the generation of one particular result (output).

(c) Examine each output for correctness--first syntactic correctness, then semantic correctness.

(d) If no discrepancies are noted, declare the product "correct."

Falsification testing has known weaknesses, but, for many testing situations, it is the only method that is practical and whose costs are commensurate with the value to be obtained by testing. For testing **CGM generators** for conformance to FIPS 128 standard and the CALS MIL-D-28003 application profile, it is the only practical method that has been identified.

1.3.2 Visual Inspection

To determine semantic correctness of each **CGM-under-test**, it is necessary to visually compare the **CGM hardcopy** corresponding to the **CGM-under-test**, which is produced by the **reference interpreter**, with the **native hardcopy**, which was produced by the **CGM generator-under-test**. Certain differences (coming from differences in hardware capabilities or implementation-dependencies permitted by the standard) are acceptable, while most are not. Guidelines should be available to the Test Service operator, indicating where the boundary between allowable differences and errors lies.

1.4 The CGM Standard, FIPS 128, and MIL-D-28003

1.4.1 CGM Binary Encoding (ISO 8632/3--FIPS 128) Essentials

A CGM file conforming to FIPS 128 is coded as a stream of octets. A sequence of octets comprise a CGM element. Each element consists of a header followed by its parameter data.

The header contains the opcode (class and id) of the element plus length information. The length value is the number of octets of parameter data that belong to this element. The length values must be exact, according to the requirements of the standard. It is not permissable to specify longer lengths and pad the parameter data with nulls or other data not specified in the standard. Each element must align on an even octet count, so it might be necessary to insert a null character occasionally within the CGM file. It is not allowed that this pad character be included in the length count of the preceding element.

Elements come in short form (the element with its data fits within 31 octets) and long form (all other elements). Furthermore, elements may be partitioned or not.

Strings are an important data type in the FIPS 128. A string is coded as an octet containing the string length followed by the string characters themselves. This data type is used for GDP, ESCAPE, and APPLICATION DATA elements. It is a common error to omit the string length information as the first octet of the parameter data. Strings also can be coded in short form (for up to 30 characters) and long form (for all longer strings).

1.4.2 CALS CGM Application Profile (MIL-D-28003) Essentials

Generally speaking, the CALS CGM application profile adds constraints beyond those specified in FIPS 128. All MIL-D-28003 conforming CGM files are, by definition, FIPS 128 conforming, but the converse is not true.

The purpose of the CALS CGM application profile is to set "maximum allowable limits" for the behavior of CGM generators and specify "minimum requirements" for the behavior of CGM interpreters. When and only when these requirements are compatible is unambiguous interchange possible. Otherwise, a CGM generator could use a given element (say, MARKER TYPE) or a given attribute value (say, MARKER TYPE 3) and try to present the file to a CGM interpreter that did not recognize the MARKER TYPE element or did not know what to do if MARKER TYPE 3 were encountered.

In sum, the CALS CGM application profile places constraints on generators (e.g., limiting the numbers of points in primitives or the index values used in attribute elements) and specifies minimum requirements for interpreters (e.g., that all elements be able to be interpreted).

1.4.3 Required vs Optional Elements and Behavior

A correct CGM need only contain a few specific elements. These are the delimiter elements, plus METAFILE VERSION and METAFILE ELEMENT LIST. A CGM conforming to MIL-D-28003 must also contain a METAFILE DESCRIPTION element that includes a specific text substring as part of the parameter data.

From a CGM generator point of view, all other elements are optional. That is, when producing CGMs, there is no specific requirement that any given CGM element (e.g., RECTANGLE) or CGM parameter (e.g., TEXT PRECISION character) be able to be written to a CGM file.

This situation presents problems for a Testing Service because it is not possible to demand that any given set of CGM files be able

to be produced by the **generator-under-test**. Consequently, the characteristics of the **generator-under-test** must be determined (see section 1.6) before the instructions for creating the **CGM suite-under-test** can be produced (these instructions are known as the **test suite description**).

1.5 Available Testing Tools

1.5.1 CGM/MIL-D-28003 Requirements Document

The CGM/MIL-D-28003 Requirements Document was published by NIST in May 1990 as document NISTIR 4329. Next to the standards themselves (ISO 8632 and MIL-D-28003), the Requirements Document is the most authoritative source for determining what constitutes a correct CGM.

1.5.2 Reference Interpreters

As yet, the Testing Service has not certified any reference interpreters for CGM. Consequently, the testing service will use testing tools that are candidates for certification, but which are not yet certified.

Two testing tools are used in implementing the CGM Test Service:

- MetaCheck with the CALS Option (see section 1.5.2.1) to check syntactic correctness; and
- CGM-View (see section 1.5.2.2) to check for semantic correctness.

Because neither tool has been certified as correct, any discrepancies between the output provided by the generator-under-test and the testing tool will be thoroughly investigated by manual methods to verify that the testing tool is indeed accurate in its reporting.

1.5.2.1 CTS/MetaCheck with the CALS Option

CTS/MetaCheck version 2 with the CALS Option (known as "MetaCALS" for short) is a stand-alone utility (i.e., computer program) that analyzes the **syntactic correctness** of an individual CGM file. It produces a conformance report listing all errors found. Deviations from the basic ISO 8632 (CGM) standard as well as deviations from the CALS MIL-D-28003 application profile are reported in separate sections. MetaCALS can show a trace of all the elements in a CGM; error messages are interspersed with the listing of the file. MetaCALS also can show a frequency of occurrence table for all the elements in a CGM.

MetaCALS was developed by CGM Technology Software (PO Box 648, Gales Ferry, CT 06335) for commercial sale. It has been licensed

for redistribution and use by the National Institute of Standards and Technology. It runs on PC-DOS machines and on a variety of workstations (e.g., VAX/VMS, Unix of various flavors, IBM RS6000).

MetaCALS was developed to the CALS CGM/MIL-D-28003 Test Requirements Document (see section 1.5.1 above), derived from ISO 8632, Parts 1 and 3, and from MIL-D-28003. MetaCALS is not infallible (although, at present, no errors are known), but it is widely used and has been available for over two years in some form or another.

1.5.2.2 CGM-View

CGM-View is a stand-alone utility (i.e., computer program) that permits one to see on a graphics display screen or on a graphics hardcopy device a representation of the content of a CGM file. It serves the Testing Service as a **semantic correctness** testing tool.

CGM-View was developed by Advanced Technology Center (22982 Mill Creek Drive, Laguna Hills, CA 92653). It runs only on workstations (e.g., VAX/VMS, Unix of various flavors). It supports a wide variety of color and black-and-white displays and hardcopy output devices.

1.6 Overview of the Testing Process

1.6.1 Characteristics of the Generator-under-Test

In Chapter 2, how the Test Service can determine the specific characteristics of the generator-under-test is explained. A questionnaire is available, the answers to which indicate the collection of CGMs that this product is capable of generating. A software tool has been developed to assist the test service in building a questionnaire (Appendix B) from a formal description of the requirements of the standard and the CALS profile (Appendix A).

1.6.2 Specifying the CGM Suite-under-Test

In Chapter 3, a methodology for generating a set of specifications describing the desired CGM suite-under-test is depicted. The sponsor of the generator-under-test uses these specifications to prepare his/her native suite and native hardcopy, both of which are delivered along with the CGM generator-under-test for use by the CGM Generator Test Service.

1.6.3 Testing the Completeness of the CGM Suite-under-test

Chapter 4 describes how certain features of MetaCALS are used to verify that the CGM suite-under-test in fact contains all the CGM files and elements that were asked for in the test specifications.

During this stage, the CGM suite-under-test is produced from the native suite provided by the client sponsoring the generator-under-test.

1.6.4 Testing for Syntactic Correctness

Chapter 5 explains how to use MetaCALS to test the syntactic correctness of each of the files in the CGM suite-under-test.

1.6.5 Verifying Semantic Correctness

Chapter 6 explains how to use CGM-View and the native hardcopy to test the semantic correctness of the CGM suite-under-test.

1.6.6 Assembling the Test Report

Finally, in Chapter 7, the results of the various tests are assembled into a Test Report.

2. Gathering Data About the Generator-Under-Test

2.1 Determining What can be Produced by the Generator

The Binary Encoding of the CGM standard (ISO 8632, Part 3) describes the syntax of some 91 elements that can be used by **CGM generators** to describe pictures in a device-independent format. Only a few of these elements are required in all conforming CALS CGMs that contain a visible image; namely, BEGIN METAFILE, END METAFILE, BEGIN PICTURE, BEGIN PICTURE BODY, END PICTURE, METAFILE VERSION, and METAFILE ELEMENT LIST. METAFILE DESCRIPTION is required for CALS (MIL-D-28003) conformance. In addition, for CALS conformance, at least one graphical primitive element must also appear between each BEGIN PICTURE BODY and END PICTURE.

Conversely, a conforming CALS CGM may not include any GDP or APPLICATION DATA elements and only a certain number of well-defined ESCAPE elements. Furthermore, a conforming CALS CGM may have constraints on some of the values of some of the parameters of the other elements (see section 3.1.1 for a complete checklist).

Most of the other elements and parameter combinations are permitted in a conforming FIPS 128 or CALS CGM, yet many CGM generators do not take advantage of the full capabilities of the CGM standard or the CALS application profile. For example, a generator-under-test may restrict itself to 16-bit integer VDCs or to color indices in the range of 0 to 63.

In order to properly and completely verify the conformance of a generator-under-test, it is necessary to know the range of CGM elements and parameter settings of those elements that may appear in any CGM produced (i.e., written) by the generator-under-test. To gather that information from the sponsor of the generator-under-test, a software utility has been developed to aid in this process. This is described in subsequent sections of this chapter.

2.1.1 Formal Definition of the FIPS 128 Standard

Appendix A provides a formal definition of the FIPS 128 (CGM) standard annotated with additional descriptive information. The language used to express this definition is an extension of the familiar BNF notation and is described in Appendix B.

The CGM formal description file is a simple ASCII text file and may be modified using a text editor. Currently, the CGM formal description file is named cgm.prf.

2.1.2 Generating the Questionnaire

A software utility, called cpc, has been developed to read the CGM formal description and produces a questionnaire that must be filled out by the sponsor of the CGM generator-under-test. The questionnaire is represented by a ASCII TeX file, which must be processed by a TeX compiler to produce an intermediate, device-independent representation of the questionnaire. Then a TeX driver is used to produce the final, device-dependent form of the questionnaire. The version of TeX (known as LaTeX) in current use runs on PCs. Several drivers are available for the PC version of LaTeX. The HP Laserjet driver is used for hardcopy, and the PC screen driver for previewing. Appendix C shows the Questionnaire as it is generated on an HP LaserJet III printer.

The entire sequence of operations is shown in the following.

(Step 1) From the CGM formal description, generate an ASCII TeX file containing the text and formatting instructions for the questionnaire:

cpc -q cgm.prf > cgm.tex

(Step 2) Then invoke LaTeX to convert the TeX source to a device-independent intermediate form:

latex cgm.tex

This causes the file cgm.dvi to be created.

(Step 3a) Now you may preview the questionnaire on the PC screen:

v cgm.dvi

In this program, **q** will exit; + and - are used for zooming in and out on the page; **PgUp** and **PgDn** for previous and next page, respectively; and the **arrow keys** for panning over the layed-out page.

(Step 3b) Produce hardcopy on a HP LaserJet 300 dpi laser printer:

dvihplj @lj.cnf /po=prn cgm.dvi

The printed questionnaire derived from the CGM formal description (listed in Appendix A) is shown in Appendix C. **cpc** is written in ANSI C; it was developed on a PC using Borland Turbo C.

2.1.3 Filling Out the Questionnaire

The questionnaire is sent to the client, who must answer every question based on his knowledge of what his product (the **CGM generator-under-test**) is capable of doing.

The answers to the questions drive the specification of the **native suite** and the corresponding **CGM suite-under-test**. This process is described in the next chapter.

3. Defining the Client-Specific Test Suite

3.1 Using the Questionnaire Results

The client's written responses are used as input to the next stage in the process; namely, producing specifications for the sponsor to follow in assembling his native suite of files formatted in an application-specific way.

3.1.1 Checking the Answers to the Questionnaire

Before proceeding further with the testing process, the Test Service should check that the version number in METAFILE VERSION is correct (currently, it should be version number 1) and that the required elements are present (see section 1.4.3). No other manual checks are worthwhile at this stage if the Test Service is verifying FIPS 128 Conformance only.

However, if CALS conformance is being tested for, before proceeding further with the testing process, the Test Service should review the answers to the questionnaire, looking for obvious violations of MIL-D-28003, the CALS CGM profile. A checklist of such items follows:

(a) Verify that the generator-under-test can generate the CGM Binary Encoding.

(b) Check that the maximum number of NO-OPs does not exceed 32767.

(c) Check that the METAFILE DESCRIPTION is used and that the proper CALS profile description string (currently, "MIL-D-28003/BASIC-1") is contained within the metafile description string.

(d) Check that only 16-bit INTEGER PRECISION is used.

(e) Check that only 32-bit fixed point or 32-bit REAL PRECISION is used.

(f) Check that only 16-bit INDEX PRECISION is used.

(g) Check that only 8-bit and 16-bit COLOUR PRECISION is used.

(h) Check that only 8-bit and 16-bit COLOUR INDEX PRECISION is used.

(i) Verify that the font names are drawn from the names listed in the CALS CGM Profile and that no more than four font names can appear in the font list simultaneously.

(j) Check that the CHARACTER SET LIST contains only a two-element list; namely, {(0,4/2),(1,4/1)} where 4/2 denotes the hexidecimal value 42 (the ASCII character "B") and 4/1 denotes the hexidecimal value 41 (the ASCII character "A").

(k) Verify that CHARACTER CODING ANNOUNCER is either 0 or 1, if this element is used.

(l) Verify that VDC INTEGER PRECISION is either 16-bit or 32-bit precision, if integer VDCs are used.

(m) Verify that VDC REAL PRECISION is either 32-bit floating point or 32-bit fixed point, if real VDCs are used.

(n) Verify that TRANSPARENCY is always 1, if this element is used.

(o) Verify that LINE, MARKER, FILL, and EDGE BUNDLE INDEX values always lie between 1 and 5.

(p) Verify that the LINE TYPE values are restricted to 1-5 plus the special CALS values.

(q) Verify that the MARKER TYPE values are restricted to 1-5.

(r) Verify that TEXT BUNDLE INDEX values are either 1 or 2.

(s) Verify that the TEXT FONT INDEX values are restricted to 1-4.

(t) Verify that the CHARACTER SET INDEX and ALTERNATE CHARACTER SET values are restricted to 1 or 2.

(u) Verify that the HATCH INDEX values are restricted to 1-6 plus the special CALS values.

(v) Verify that the EDGE TYPE values are restricted to 1-5.

(w) Verify that the PATTERN TABLE starting index is restricted to 1-8 and that the number of rows and columns are restricted to 1-16.

(x) Verify that the COLOUR TABLE starting index value is restricted to the range 0-255.

(y) Verify that the action required flag of the MESSAGE element always has the value "no action required."

(z) Verify that the number of colour values that can appear in a colour array or colour list parameter does not exceed 1048576 for CELL ARRAY, 2048 for PATTERN TABLE, or 256 for COLOUR TABLE.

(aa) Verify that the number of points and VDCs that can appear in parameters for any of the elements cannot exceed 1024.

(bb) Verify that all string elements are limited to a maximum of 254 characters, except for data records, which are limited to 32767 characters.

(cc) Verify that no GDP elements can appear.

(dd) Verify that only the CALS-specified ESCAPE elements can appear.

3.1.2 Interacting with the Client

If the results of the manual analysis of the answers to the questionnaire show violations of the CGM standard or deviations from the CALS CGM application profile, the client should be informed in writing and no further testing should be performed until the client fixes the errors and submits a new filled-out questionnaire.

3.2 Semi-automatic Generation of the Test Suite Description

Appendix D contains a listing of an ASCII file that has been assembled to assist in the specification of requirements on the **native suite** and the corresponding **CGM suite-under-test.** This specification is known as the **test suite description.** It is expected that, depending upon the answers to the questionnaire, one could use a text editor to cut-and-paste the appropriate paragraphs from this file into a new file, which, when completed, will contain the specifications needed by the supplier of the **generator-under-test.**

This file has the PC-DOS file name of **questans.**

3.3 Manual Tuning of the Test Suite Description

The verbal specifications will need to be edited into a coherent whole and supplemented with requests to generate certain pictures (called **test images**), as specified in section 3.4.2 below. However, only certain **generators-under-test** will indeed have the operator controls required to be able to cause the desired CGM files corresponding to the specified **test images** to be generated.

3.4 Client Instructions

3.4.1 The Test Suite Description

The supplier of the **CGM generator-under-test** ("client" for short) is given the **test suite description** prepared from his/her answers to the questionnaire. He/she is required to develop a **native suite** that, when processed by the **generator-under-test**, produces a **CGM suite-under-test** that meets the requirements stated in the **test suite description.**

For each requirement in the test suite description, the client should be instructed to specify which file or files in the native suite meet the requirement. This will allow the Test Service to more easily determine whether the CGM suite-under-test corresponding to the native suite indeed meets the requirements of the test suite description.

3.4.2 Test Images

Appendix E contains a collection of **test images.** These images check many (but clearly not all) of the myriad possible combinations of primitive elements and attribute elements.

Vendors should be instructed to try to create native files that approximate the specified **test images** as closely as possible using the intended CGM primitive and attribute element combinations suggested by the labelling. If the **generator-under-test** does not use certain primitives or attributes, vendors should still try to produce any of the combinations that the generator does use.

The Test Service is aware that some **generators-under-test** might not be able to operate in a mode that makes it easy or convenient to produce images that correspond to the **test images.** In these cases, the client should try to supply a representative set of native files, which, when exported as CGM files by the **generator-under-test**, have primitive and attribute combinations similar to those requested by the test images.

4. Checking the Completeness of the Client Package

The items that should be provided by the client sponsoring the CGM **generator-under-test** are listed below.

(1) The client product (**generator-under-test**) and corresponding users manual or a filled-out form that authorizes on-site testing by the Test Service.

(2) The native files from which the **CGM suite-under-test** can be generated (the **native suite**).

(3) Hardcopy of the expected images corresponding to the native files from which the CGMs were generated (the set of **native hardcopies**).

(4) The CGMs meeting the test suite description (the **CGM suite-under-test**). This last item is redundant if the client product is provided, because the **CGM suite-under-test** can be produced from the **native suite**, which the client must supply. However, because it is time-consuming to produce the **CGM suite-under-test**, it is useful to have the client do this work in advance.

This chapter discusses how to verify that the client has supplied a **native suite** that meets the requirements of the **test suite description**. We call this stage "completeness testing" to distinguish it from syntactic and semantic "correctness testing."

Once a CGM **suite-under-test** has been assembled (see section 4.1 below), it is possible to begin to analyze the files with regard to whether they meet the requirements stated in the **test suite description**. However, due to the labor-intensive nature of completeness testing, it might be prudent to perform the correctness testing described in chapters 5 and 6 before performing the remaining actions documented in this chapter.

4.1 Using the Client's Product to Generate the CGM Suite-Under-Test from the Native Suite

The first step is to produce the **CGM suite-under-test**. If the set of native files is not too large and the client **generator-under-test** is available, the **CGM suite-under-test** should be generated anew, rather than using the **CGM suite-under-test** provided by the client. This guarantees that the **CGM suite-under-test** corresponds exactly to the version of the **generator-under-test** and that no manual "adjustments" were made to the CGM files after they were produced by the **generator-under-test**.

If the **native suite** is too large (e.g., hundreds of files), it would be possible for the Test Service to generate a subset of the **CGM suite-under-test** from a random sample of the files comprising the **native suite**. The resulting CGM files should be identical to their corresponding file in the **CGM suite-under-test** supplied by the client. If no discrepancies are found after sampling, say 20% of the total, it is probably justified to proceed and use the **CGM suite-under-test** provided by the client rather than to continue to generate the **CGM suite-under-test** anew. The Test Service will have to make this judgment.

4.2 Using the MetaCheck Frequency Information

Many requirements reduce to the statement, "At least x but not more than y percent of the files shall contain an instance of z element." Consequently, element frequency count data is needed for each file in the **CGM suite-under-test**.

This information is provided by MetaCheck running at level 1 trace (i.e., with the **-l1** switch on the command line invoking MetaCheck). When the **-s** switch is also used, the resulting element frequency count information is routed to a file, whose name is given with the switch.

By looking at the element frequency count information for each requirement in the **test suite description**, a test service operator can determine whether the files listed by the client indeed meet that requirement.

For some requirements, it might be more efficient to combine all the frequency count information for all the files in the **CGM suite-under-test** into a single file. Then, simple utilities (e.g., using **sed** and **grep** under UNIX) can be used to count the number of instances of each element over the whole **CGM suite-under-test**.

4.3 Using the MetaCheck Tracing Facility for Verifying The Inclusion of Required Metafile Descriptor, Picture Descriptor, and Control Element Parameter Combinations

Other requirements in the **test suite description** require examination of the parameter settings of certain CGM elements. A MetaCheck level 3 trace provides sufficient information to determine whether the files indicated by the client indeed contain the various element parameter data required by the **test suite description**. If the **-t** switch is used in addition to the **-l3** switch, the trace information is routed into a separate file, where the data can be analyzed by other programs, custom-written by the test service to look for certain items of information. Again, UNIX tools like **sed** and **grep** are probably adequate for this purpose.

A final set of requirements in the test suite description specify that a certain number of points, color indices, and the like appear

in specified elements. In this case, it is necessary to examine the element length counts in the element header and calculate whether the proper number of parameters are present. Alternatively, one can use MetaCheck at trace level 4; MetaCheck then counts the parameters for you. However, this last approach can result in very large trace files.

4.4 Using the MetaCheck Tracing Facility for Verifying The Inclusion of Required Primitive and Attribute Element Parameter Combinations

The files from the **CGM suite-under-test** corresponding to the test images should be analyzed with MetaCheck using the level 3 to verify that the expected elements are present and that the expected attribute settings are used at the appropriate times. This is labor intensive work that might be alleviated by writing an appropriate set of tools that would scan the trace output looking for the proper sequence of elements and parameter settings.

5. Checking the Syntactic Correctness of the Client-Supplied CGMs

Once the completeness of the CGM suite-under-test is established, the test service can proceed to determine if the CGMs are correct. This chapter focuses on syntactic correctness; the next chapter on semantic correctness.

In fact, it is possible to defer the completeness testing documented in Chapter 4 until after it is determined that all the files in the CGM suite-under-test are both syntactically and semantically correct. This is a decision for the test service. Eventually completeness testing will have to be accomplished, but due to its labor-intensive nature, it might be prudent to perform completeness testing only on CGM suites-under-test that are known to be syntactically and semantically correct.

5.1 Running CTS/MetaCheck

MetaCheck is invoked from a command line. By default (at trace level 0) a conformance report is routed to the standard output device. This report can be redirected to a file and examined by software utility programs.

The CTS/MetaCheck version 2 user's manual gives complete operating instructions.

5.1.1 Checking for FIPS 128 Syntactic Correctness

By default, MetaCheck looks only for FIPS 128 conformance, so there is no special process to be followed to obtain this result.

5.1.2 Checking for CALS MIL-D-28003 Syntactic Correctness

By including the **-rcals** switch on the command line, MetaCheck is directed to look for conformance to the CALS MIL-D-28003 application profile. The conformance report produced distinguishes between FIPS 128 (CGM) errors and CALS application profile errors.

5.2 Understanding the Error Reports

The error reports are self-explanatory and fully explained in the CTS/MetaCheck users manual. Providing these reports directly to the client should be sufficient for documenting the test service results.

5.3 Understanding the Warning Advisories and Bulletins

CTS/MetaCheck has the ability to look for "unusual" usage of the CGM. Although not syntactically incorrect, these elements might

indicate a semantic problem (e.g., POLYLINES with all the points the same) with the generator-under-test. To direct MetaCheck to look for these kinds of potential problem areas, use the **-w** switch on the command line.

However, remember that these warnings and bulletins are not syntactic errors! They are merely indicators that there could be a problem with the semantics of the CGM-under-test. This warning reporting feature of MetaCheck must be used with care by any test service. In general, the warnings and bulletins should not be included in a test report. Instead, if the warning or bulletin is indicative of a real semantic problem, then that problem should be described directly, without alluding to the warning signal provided by MetaCheck.

5.4 Documenting Discrepancies

It should be sufficient to provide the client with a copy of the level 0 trace output for each file in the CGM suite-under-test that contains a syntactic error. The test service could also provide a level 4 trace of the element or group of elements that were in error.

6. Checking the Semantic Correctness of the Client-Supplied CGMs

Once the **CGM suite-under-test** has been determined to be syntactically correct, the Test Service can go on to check for semantic correctness.

As explained in Chapter 1, the general approach is to compare the view of the native file as displayed by the client application with the view of the corresponding CGM file as displayed by the utility program CGM-View.

6.1 Comparing Native Hardcopy with CGM Hardcopy

Ideally, both the client application and CGM-View can be installed on the same workstation, which is able to support multitasking and a windowed user interface. The client application should then be run in one window, while CGM-View runs in a second window. Initially, the windows should be placed so that they do not overlap. Then, the client application is made to display a native file from the **native suite** (producing the **native hardcopy**) while CGM-View displays the corresponding CGM file from the **CGM suite-under-test** (producing the **CGM hardcopy**). The visual appearances of the two displays are then compared visually by Test Service personnel.

In less than these ideal conditions, other configurations are also acceptable. These are described in the following paragraphs:

(A) Problem: The client application cannot run on the CGM-View workstation (e.g., the application is a PC product only).
Solution: The client application is run on a different machine, whose monitor is placed side-by-side with the CGM-View workstation monitor. It is important to try to have the general graphics capabilities of the application machine and the CGM-View workstation be as similar as possible (especially in their color palette capabilities, resolution, and monitor size).

(B) Problem: The client application has no softcopy (screen display) capability.
Solution: Use the **native hardcopy** provided by the client to substitute for the real-time generation of **native hardcopy** used in the ideal testing situation.

(C) Problem: The client application runs on a machine that is not available to the Test Service.
Solution: Perform semantic testing at the client's site. If the client has a machine on which CGM-View can be run,

this situation resembles alternative (A) above. If not, this situation resembles alternative (B), but in this case, the **native hardcopy** can be displayed in real-time, while the **CGM hardcopy** must be prepared in advance by the Test Service and brought to the client site.

6.2 Understanding Visual Differences

Clearly, the effectiveness of this testing depends upon the Test Service operator's ability to notice differences between the corresponding **native hardcopy** and **CGM hardcopy**. These differences can be attributed to four major sources:

(a) Errors of the CGM generator-under-test. These need to be documented and reported to the client as part of the Test Report (see section 6.4);

(b) Errors in CGM-View. These need to be documented and reported to ATC, the developer of CGM-View. In these cases, alternative candidate **reference interpreters** can be used (if available) to provide the **CGM hardcopy**. Also, it might happen that CGM-View only makes an error on its screen display, but that other display paths (e.g., to a color printer) work correctly. In these cases, the **CGM hardcopy** used by the Test Service need not be the screen display but any other form of hardcopy correctly produced by CGM-View.

(c) Documented inadequacies of CGM-View. In these cases, as with case (b), it might be possible to find alternative candidate **reference interpreters** which do handle the CGMs-under-test in question. However, if no candidates can be found, either the Test Service has to skip the test or it can attempt to use the full trace capability of MetaCheck to view the contents of the CGM file and use "intellectual" methods to determine whether the given CGM file indeed appears to reflect the intended picture appropriate for the corresponding native file.

(d) Differences in hardware environment. A picture intended for a color prepress application might use a color table with several hundreds of entries. Both the client application and CGM-View might be running on systems with the ability to display only, say, 64 or 256 colors simultaneously. In this case, neither the **native hardcopy** nor the **CGM hardcopy** will reflect exactly the information contained in the native file or in the corresponding CGM file. Furthermore, the algorithms used by the client application and CGM-View to map the large number of colors requested to the smaller number of colors available are probably not identical. Other

possible sources of allowed visual discrepancies are listed in section 6.3, below.

6.3 Allowed Visual Discrepancies

The sources of allowed visual discrepancies are varied. Annex D of FIPS 128 contains a fairly complete discussion of the issues. A summary list of the topics covered in Annex D is provided in the following paragraphs:

(a) Handling of geometrically degenerate primitives. These include line elements of zero length, filled-area elements of zero area (degenerating either to a line or to a point), and CELL ARRAY elements of zero area.

(b) Mapping of out-of-range indices.

(c) Support for direct color on hardware that is inherently color-mapped (that is, uses indexed color).

(d) Mapping from the specified color precision in the CGM to the available color precision on the Test Service hardware.

(e) Whether BEGIN PICTURE BODY causes the view surface to be cleared to the background color.

(f) How AUXILIARY COLOUR is interpreted if this capability is not supported by the Test Service hardware.

(g) How clipping is implemented when any part of the CLIP RECTANGLE is outside the VDC EXTENT.

(h) Whether kerned fonts lie entirely within the RESTRICTED TEXT bounding parallelogram.

(i) How CIRCULAR ARC and ELLIPTICAL ARC primitives are rendered when the start ray and end ray coincide.

(j) The exact appearance of the standardized LINE TYPEs might vary both due to the algorithm used and to differing resolutions used for **native hardcopy** and **CGM hardcopy**.

(k) The exact appearance at joins (where line segments meet) in a POLYLINE element.

(l) The realization of LINE WIDTH.

(m) The exact appearance of the standardized MARKER TYPEs.

(n) The realization of MARKER SIZE.

(o) The handling of TEXT PRECISION. It is permissable to approximate both string and character text by stroke text. This could lead to quite different appearances.

(p) The exact realization of other text attributes like TEXT ALIGNMENT, CHARACTER EXPANSION FACTOR, CHARACTER SPACING, CHARACTER ORIENTATION, and CHARACTER HEIGHT is affected by the resolution of the output device.

(q) The mapping between the font names placed in the CGM file (if any) and the font names available through CGM-View or any other reference interpreter. It is likely that there will not be a match either in font metrics or in typeface appearance.

(r) The exact realization of the standardized hatch indices.

(s) The exact appearance of the standardized EDGE TYPEs might vary both due to the algorithm used and to differing resolutions used for native hardcopy and **CGM hardcopy**.

(t) The exact appearance at joins (where edge segments meet) in a POLYGON or POLYGON SET element with edge visibility on.

(u) The realization of EDGE WIDTH.

(v) Any realization of GDP, ESCAPE, and APPLICATION DATA elements.

(w) Any appearance and behavior of the MESSAGE element.

6.4 Documenting Discrepancies

All discrepancies should be noted and documented, although only type (A) discrepancies should appear in the Test Report.

Ideally, discrepancies should be documented by saving physically permanent versions of the **native hardcopy** and the corresponding **CGM hardcopy**, along with the native file and CGM file. This would require that the Test Service have access to a reasonably high-quality color output device, capable of rendering at least 256 colors simultaneously on standard and medium-sized paper (A and B or C) and with sufficient resolution to show small differences between plots.

In addition, because CGM-View is not yet a certified **reference interpreter**, it will be necessary to investigate all apparent discrepancies to determine which category they fall under. The principal methods available are two:

(1) Use MetaCheck to get a full trace of the file. Examine the trace file and "hand simulate" the behavior of an ideal CGM **reference interpreter**. Compare the results with the **native hardcopy**. If they differ, there is an error in the **CGM generator-under-test**. Warning: For some client applications, even the **native hardcopy** might not represent accurately the application user's intention! In these cases, it is extremely labor intensive to become fully conversant with the total capabilities of the client application and to determine where the source of the visual differences is coming from.

(2) Acquire one or more other CGM **reference interpreters** and compare their results with the **CGM hardcopy**. If all the **reference interpreters** agree, it is likely that the fault lies with the client application (the **generator-under-test**). This method is known as the "Delphi method."

Depending upon the subtlety of the error made by the **generator-under-test**, it can be extremely costly to find its actual source.

7. Assembling the Test Report

The Test Report will be designed by NIST in accordance with the experience with GKS and PHIGS testing and based on NIST accepted practice. The following sections provide an overview of the major testing results that need to be included in the Test Report. These results are in addition to supporting documentation, such as a description of the client application (the **CGM generator-under-test**), a copy of the filled-out client questionnaire, a copy of the **test suite description**, and the client's documentation of how his submitted **native suite** satisfies the requirements of the **test suite description.**

7.1 Package Completeness

The results of the testing described in Chapter 4 is documented in this section. The major resources are the frequency reports and element traces produced by MetaCheck.

7.2 Syntactic Correctness

The results of the testing described in Chapter 5 is documented in this section. The major resources are the conformance reports produced by MetaCheck.

7.3 Semantic Correctness

The results of the testing described in Chapter 6 is documented in this section. The major resources are the native hardcopies and CGM hardcopies produced by the client application and CGM-View, respectively.

7.4 Registration of Report by NIST

The drafting, review, finalization, and registration of the report by NIST will follow guidelines developed and published by NIST.

APPENDIX A

FORMAL DESCRIPTION OF CGM

```
%-------------------------------------------------------------
"constraints":= {

   "ENCODING":= { "binary", "character", "clear text" }
   "MAX_STRING_LENGTH":= { * }     % ASSUMED TO BE UNIQUE FOR ALL STRINGS !
   "MAX_POINTS":= { * }           % ASSUMED TO BE UNIQUE FOR ALL POINT LISTS
!
   "MAX_NOPS":= { * }
   "VALID_CHARACTERS":= { * }
}

%-------------------------------------------------------------
"macro definitions":= {

   "device dependent background colour":={ <0,0,0> }

   "device dependent foreground colour":={ <1,1,1> }

   "standard string type":= {
       "MAX_STRING_LENGTH",
       "VALID_CHARACTERS"
   }

   "LINE_ATTRIBUTES":= {
      "ATTRIBUTES":= {
          "LINE BUNDLE INDEX",
          "LINE TYPE",
          "LINE WIDTH",
          "LINE COLOUR"
      }
   }

   "MARKER_ATTRIBUTES":= {
      "ATTRIBUTES":= {
          "MARKER BUNDLE INDEX",
          "MARKER TYPE",
          "MARKER SIZE",
          "MARKER COLOUR"
      }
   }
```

```
   "TEXT_ATTRIBUTES":= {
      "ATTRIBUTES":= {
          "TEXT BUNDLE INDEX",
          "TEXT FONT INDEX",
          "TEXT PRECISION",
          "TEXT COLOUR",
          "TEXT PATH",
          "TEXT ALIGNMENT",
          "CHARACTER EXPANSION FACTOR",
          "CHARACTER SPACING",
          "CHARACTER HEIGHT",
          "CHARACTER ORIENTATION",
          "CHARACTER SET INDEX",
          "ALTERNATE CHARACTER SET INDEX"
      }
   }

   "FILL_ATTRIBUTES":= {
      "ATTRIBUTES":= {
          "FILL BUNDLE INDEX",
          "FILL COLOUR",
          "FILL REFERENCE POINT",
          "INTERIOR STYLE",
          "HATCH INDEX",
          "PATTERN INDEX",
          "EDGE BUNDLE INDEX",
          "EDGE WIDTH",
          "EDGE TYPE",
          "EDGE COLOUR",
          "EDGE VISIBILITY",
          "PATTERN TABLE",
          "PATTERN SIZE",
          "COLOUR TABLE"
      }
   }
}

%-----------------------------------------------------------------
"delimiter elements":= {

   "NO-OPERATION":= { "MAX_NOPS" }

   "BEGIN METAFILE":= {
     "STRING":= { "USAGE":= [ "sometimes" ] { "always", "never" },
                  "standard string type"
     }
   }

   "END METAFILE":= { }
```

```
   "BEGIN PICTURE":= {
     "MAX_PICTURES":= { 32 }
     "STRING":= { "USAGE":= [ "sometimes" ] { "always", "never" },
                  "standard string type"
     }
   }

   "BEGIN PICTURE BODY":= { }

   "END PICTURE":= { }
}

%-----------------------------------------------------------------
"metafile descriptor elements":= {

   "METAFILE VERSION":= { * }

   "METAFILE DESCRIPTION":= {
     "STRING":= { "USAGE":= [ "sometimes" ] { "always", "never" },
                  "standard string type"
     }
   }

   "VDC TYPE":= [ "integer" ] { "real" }

   "INTEGER PRECISION":= [ 16 ] { 8, 24, 32 }

   "REAL  PRECISION":=          % before:  (0,9,23),  (0,12,52),  (1,16,16),
(1,32,32)
     { "32-bit floating point" } [ "32-bit fixed point" ]
     { "64-bit floating point" } { "64-bit fixed point" }

   "INDEX PRECISION":= [ 16 ] { 8, 24, 32 }

   "COLOUR PRECISION":= [ 8 ] { 16, 24, 32 }

   "COLOUR INDEX PRECISION":= [ 16 ] { 8, 24, 32 }

   "MAXIMUM COLOUR INDEX":= [ 63 ] { * }

   "COLOUR VALUE EXTENT":= [ <0,0,0>..<255,255,255> ]
                           { <*,*,*>..<*,*,*>       }

   "METAFILE ELEMENT LIST":= { "Drawing set", "Drawing-plus-control set" }


   "METAFILE DEFAULTS REPLACEMENT":= { "VDC TYPE",
                                       "INTEGER PRECISION",
                                       "REAL PRECISION",
```

```
    "INDEX PRECISION",
    "COLOUR INDEX PRECISION",
    "COLOUR VALUE EXTENT",
    "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 INTEGER PRECISION",
    "VDC REAL PRECISION",
    "AUXILIARY COLOUR",
    "TRANSPARENCY",
    "CLIP RECTANGLE",
    "CLIP INDICATOR",
    "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",
    "CHARACTER EXPANSION FACTOR",
    "CHARACTER SPACING",
    "TEXT COLOUR",
    "CHARACTER HEIGHT",
    "CHARACTER ORIENTATION",
    "TEXT PATH",
    "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",
    "COLOUR TABLE",
    "ASPECT SOURCE FLAGS"
}
```

```
   "FONT LIST":= { "MAX_NAMES":= { * },
                   "MAX_NAME_LENGTH":= { * },
                   "NAMES":= { "HERSHEY:*" }
   }

   "CHARACTER SET LIST":= {
          "MAX_CHARACTER_SETS":= { * },
          "MAX_CHARACTER_SET_LENGTH":= { * },
          "CHARACTER_SET_TYPE":= { "94-character G-set",
                                   "96-character G-set",
                                   "94-character multibyte G-set",
                                   "96-character multibyte G-set",
                                   "complete code" },
          "SEQUENCE_TAIL":= { "*" }
   }

   "CHARACTER CODING ANNOUNCER":= ( "*" ),
                                  [ "7-bit basic" ]
                                  { "8-bit basic",
                                    "7-bit extended",
                                    "8-bit extended" }
}

%-----------------------------------------------------------------
"picture descriptor elements":= {

   "SCALING MODE":= {
          "MODE":=  [ "abstract" ] { "metric" },
          "METRIC_SCALE_FACTOR":= { *.* },
          "PRECISION":= [ "32-bit fixed point" ]
   }

   "COLOUR SELECTION MODE":= [ "indexed" ] { "direct" }

   "LINE WIDTH SPECIFICATION MODE":= { "absolute" } [ "scaled" ]

   "MARKER SIZE SPECIFICATION MODE":= { "absolute" } [ "scaled" ]

   "EDGE WIDTH SPECIFICATION MODE":= { "absolute" } [ "scaled" ]

    "VDC EXTENT":= {
         "INT_AREA":=  [ <0,0>..<32767,32767> ] { <*,*>..<*,*> }
         "REAL_AREA":= [ <0.0,0.0>..<1.0,1.0> ] { <*.*,*.*>..<*.*,*.*> }
    }

   "BACKGROUND COLOUR":= [ "device dependent background colour" ]
                         { <*,*,*> }
}
```

```
%-------------------------------------------------------------
"control elements":= {

   "VDC INTEGER PRECISION":= [ 16 ] { 24, 32 }

   "VDC REAL PRECISION":=
     { "32-bit floating point" } [ "32-bit fixed point" ]
     { "64-bit floating point" } { "64-bit fixed point" }

   "AUXILIARY COLOUR":= {
       "INDEXED":= [ 0 ] { * },
       "DIRECT" := [ "device dependent background colour" ] { <*,*,*> }
   }

   "TRANSPARENCY":= [ "on" ] { "off" }

   "CLIP RECTANGLE":= [ "VDC EXTENT" ]
                      { "INT_AREA":=  { <*,*>..<*,*> }
                        "REAL_AREA":= { <*.*,*.*>..<*.*,*.*> }
                      }

   "CLIP INDICATOR":= [ "on" ] { "off" }
}

%-------------------------------------------------------------
"graphical primitive elements":= {

   "POLYLINE":= { "LINE_ATTRIBUTES",
                  "MAX_POINTS"
   }

   "DISJOINT POLYLINE":= { "LINE_ATTRIBUTES",
                           "MAX_POINTS"
   }

   "POLYMARKER":= { "LINE_ATTRIBUTES",
                    "MAX_POINTS"
   }

   "TEXT":= {
     "TEXT_ATTRIBUTES",
     "standard string type",
     "TEXT_FLAG":= { "final", "non-final" },
     "ALLOW_CHANGE":= { "TEXT FONT INDEX",
                        "CHARACTER EXPANSION FACTOR",
                        "CHARACTER SPACING",
                        "TEXT COLOUR",
                        "CHARACTER HEIGHT",
                        "CHARACTER SET INDEX"
     }
   }
```

```
"RESTRICTED TEXT":= {
  "TEXT_ATTRIBUTES",
  "standard string type",
  "TEXT_FLAG":= { "final", "non-final" },
  "ALLOW_CHANGE":= { "TEXT FONT INDEX",
                     "CHARACTER EXPANSION FACTOR",
                     "CHARACTER SPACING",
                     "TEXT COLOUR",
                     "CHARACTER HEIGHT",
                     "CHARACTER SET INDEX"
  }
}

"APPEND TEXT":= {
  "TEXT_ATTRIBUTES",
  "standard string type",
  "TEXT_FLAG":= { "final", "non-final" },
  "ALLOW_CHANGE":= { "TEXT FONT INDEX",
                     "CHARACTER EXPANSION FACTOR",
                     "CHARACTER SPACING",
                     "TEXT COLOUR",
                     "CHARACTER HEIGHT",
                     "CHARACTER SET INDEX"
  }
}

"POLYGON":= { "FILL_ATTRIBUTES",
              "MAX_POINTS"
}

"POLYGON SET":= {
  "FILL_ATTRIBUTES",
  "MAX_POINTS",
  "EDGE_OUT_FLAG":= { "invisible",       "visible",
                      "close/invisible", "close/visible"
  }
}

"CELL ARRAY":= {
  "MAX_COLUMNS":= { * },
  "MAX_ROWS":= { * },
  "LOCAL_COLOUR_PRECISION":= { 0, 1, 2, 4, 8, 16, 24, 32 },
  "LIST_MODE":= { "run-length", "packed" }
}

"GENERALIZED DRAWING PRIMITIVE":= {
  "ID":= { * },
  "MAX_DATA_RECORD_LENGTH":= { * }
}

"RECTANGLE":= { "FILL_ATTRIBUTES" }
```

```
    "CIRCLE":= { "FILL_ATTRIBUTES" }

    "CIRCULAR ARC 3 POINT":= { "LINE_ATTRIBUTES" }

    "CIRCULAR ARC 3 POINT CLOSE":= {
      "FILL_ATTRIBUTES",
      "CLOSURE":= { "pie", "chord" }
    }

    "CIRCULAR ARC CENTRE":= { "LINE_ATTRIBUTES" }

    "CIRCULAR ARC CENTRE CLOSE":= {
      "FILL_ATTRIBUTES",
      "CLOSURE":= { "pie", "chord" }
    }

    "ELLIPSE":= { "FILL_ATTRIBUTES" }

    "ELLIPTICAL ARC":= { "LINE_ATTRIBUTES" }

    "ELLIPTICAL ARC CLOSE":= {
      "FILL_ATTRIBUTES",
      "CLOSURE":= { "pie", "chord" }
    }
}

%-------------------------------------------------------------------
"attribute elements":= {

    "LINE BUNDLE INDEX":= [ 1 ] { 1..5 }

    "LINE TYPE":= ( *..-1 ) [ 1 ] { 1..5 }

    "LINE WIDTH":= {
        "ABSOLUTE":= [ *.* ] { *.* },
        "SCALED"  := { 1.0 } { *.* }
    }

    "LINE COLOUR":= {
        "INDEXED":= [ 1 ] { * },
        "DIRECT" := [ "device dependent foreground colour" ] { <*,*,*> }
    }

    "MARKER BUNDLE INDEX":= [ 1 ] { 1..* }

    "MARKER TYPE":= ( *..-1 ) [ 3 ] { 1..5 }

    "MARKER SIZE":= {
        "ABSOLUTE":= [ *.* ] { *.* },
        "SCALED"  := [ 1.0 ] { *.* }
    }
```

```
    "MARKER COLOUR":= {
        "INDEXED":= [ 1 ] { * },
        "DIRECT" := [ "device dependent foreground colour" ] { <*,*,*> }
    }

    "TEXT BUNDLE INDEX":= [ 1 ] { 1..* }

    "TEXT FONT INDEX":= [ 1 ] { 1..* }

    "TEXT PRECISION":= [ "string" ] { "character", "stroke" }

    "CHARACTER EXPANSION FACTOR":= [ 1.0 ] { 0.0..*.* }

    "CHARACTER SPACING":= [ 0.0 ] { 0.0..*.* }

    "TEXT COLOUR":= {
        "INDEXED":= [ 1 ] { * },
        "DIRECT" := [ "device dependent foreground colour" ] { <*,*,*> }
    }

    "CHARACTER HEIGHT":= { *.* }

    "CHARACTER ORIENTATION":= {
        "UP_VECTOR":= [ <0.0,1.0> ] { <*.*,*.*> },
        "BASE_VECTOR":= [ <1.0,0.0> ] { <*.*,*.*> }
    }

    "TEXT PATH":= [ "right" ] { "left", "up", "down" }

    "TEXT ALIGNMENT":= {
      "HORIZONTAL":= [ "normal" ]
                     { "left", "centre", "right", "continuous horizontal" }
      "VERTICAL"  := [ "normal" ]
                     { "top", "cap", "half", "base", "continuous vertical" }

    }

    "CHARACTER SET INDEX":= [ 1 ] { 1..* }          % ???, see CHARACTER SET
LIST

    "ALTERNATE CHARACTER SET INDEX":= [ 1 ] { 1..* }  % see CHARACTER SET LIST


    "FILL BUNDLE INDEX":= [ 1 ] { 1..* }

    "INTERIOR STYLE":= ( "*" )
                       [ "hollow" ]
                       { "solid", "pattern", "hatch", "empty" }
```

```
"FILL COLOUR":= {
    "INDEXED":= [ 1 ] { * },
    "DIRECT" := [ "device dependent foreground colour" ] { <*,*,*> }
}

"HATCH INDEX":= ( *..-1 ) [ 1 ] { 1..6 }

"PATTERN INDEX":= [ 1 ] { 1..* }

"EDGE BUNDLE INDEX":= [ 1 ] { 1..5 }

"EDGE TYPE":= ( *..-1 ) [ 1 ] { 1..5 }

"EDGE WIDTH":= {
    "ABSOLUTE":= [ *.* ] { *.* },
    "SCALED"  := [ 1.0 ] { *.* }
}

"EDGE COLOUR":= {
    "INDEXED":= [ 1 ] { * },
    "DIRECT" := [ "device dependent foreground colour" ] { <*,*,*> }
}

"EDGE VISIBILITY":= { "on" } [ "off" ]

"FILL REFERENCE POINT":= {
    "INT_POINT":=  [ <*,*> ] { <*,*> },
    "REAL_POINT":= [ <*.*, *.*> ] { <*.*, *.*> }
}

"PATTERN TABLE":= {
  "INDEX"  := [ 1 ] { 1..* },
  "COLUMNS":= [ 1 ] { 1..* },
  "ROWS"   := [ 1 ] { 1..* },
  "LOCAL_COLOUR_PRECISION":= { 0, 1, 2, 4, 8, 16, 24, 32 },
  }

"PATTERN SIZE":= {
    "HEIGHT_VECTOR":= [ <0.0,*.*> ] { <*.*,*.*> },
    "WIDTH_VECTOR" := [ <*.*,0.0> ] { <*.*,*.*> }
}

"COLOUR TABLE":= {
     "MAX_SIZE":= [ 255 ] { * },
     "INDEX":=    [ 0..255 ] { 0..* }
}
```

```
   "ASPECT SOURCE FLAGS":= {
     "LINE TYPE",
     "LINE WIDTH",
     "LINE COLOUR",
     "MARKER TYPE",
     "MARKER SIZE",
     "MARKER COLOUR",
     "TEXT FONT INDEX",
     "TEXT PRECISION",
     "CHARACTER EXPANSION FACTOR",
     "CHARACTER SPACING",
     "TEXT COLOUR",
     "INTERIOR STYLE",
     "FILL COLOUR",
     "HATCH INDEX",
     "PATTERN INDEX",
     "EDGE TYPE",
     "EDGE WIDTH",
     "EDGE COLOUR"
   }
}

%-----------------------------------------------------------------
"escape elements":= {

   "ESCAPE":= {
               "ID":= { * },
               "VALID_CHARACTERS":= { * },
               "MAX_DATA_RECORD_LENGTH":= { * }
   }
}

%-----------------------------------------------------------------
"external elements":= {

   "MESSAGE":= {
               "ACTION_FLAG":= { "no action", "action" }
               "MAX_STRING_LENGTH":= { * }
   }

   "APPLICATION DATA":= {
               "ID":= { * },
               "MAX_DATA_RECORD_LENGTH":= { * }
   }
}
```

APPENDIX B

CGM FORMAL DESCRIPTION GRAMMAR

B.1 Background

Some aspects of a profile can be specified very easily using a formalism whereas other aspects cannot. This can be illustrated by having a brief look at some of the aspects of a formal description:

* **Existence:** since it is not required to support all CGM elements defined by the standard, a formal description must contain information about which elements are supported. Note that the presence of a number of CGM elements is mandatory.

* **Characteristics:** an element usually contains some characteristics, like a mode in which an element can be used. Modes may be enumerated whereas other characteristics may be specified in a different way (see below). In a more general way characteristics tell something about the behavior of the associated elements.

* **Limits:** for some values associated with a particular element there may exist upper or lower bounds. For example in most applications there exists a maximum number of characters for strings. Note that this kind of information is more a characteristic of a characteristic than a characteristic of an element and, therefore, at another level. It becomes obvious that characteristics specified in a formal description are more hierarchical than linear.

* **Ranges:** if there exists an upper and a lower bound and every value in between these bounds is a possible value, a range can be specified. In practice, a limit can be expressed as a special case of a range, where, for example, the upper bound is set to the limit and the lower bound is allowed to be every value below the upper bound.

* **Enumerations:** if there is no order in a set of values, the set can be enumerated element by element. If there is an ordered set, and at least some of the possible values are consecutive, ranges can be used as an item of an enumeration.

* **Defaults:** some characteristics, like the line type, have a default value, when no assignment is made by the CGM generator. Default values are encoding specific and must be defined in the formal description.

* **Heuristics:** several features of an application cannot be specified in an easy way, because no assumptions about their characteristics can be made. An example is the setting of a color map: depending on the requirements of the application, different settings may be desired. The difficulty in finding a uniform specification scheme is illustrated by the following examples:

* a set of k colours should be arranged in a colour table with n entries (k <= n) in a way, that an arbitrary sequence of the k colours occurs periodically in the colour table.

* given the numbers k, l and m, with k+l+m=n, the color map should contain k, l and m entries of the colours C(k), C(l), and C(m) (varying in brigthness and saturation).

* an arbitrary color map should be forced to be ordered by a certain criteria, e.g., brightness.

* **Global information:** some information describes a feature of the application rather than a specific CGM element. An example is the encoding (character, binary or clear text) method, which is supported by the application.

The aspects mentioned above can be described independently from each other within a formal description. But since some CGM elements belong to more than one category, they must be described under several aspects. For example, if one first enumerates all the elements supported by the application and then defines the attributes of each of the elements in a separate step and so on, one will end up with an extremely redundant specification. Since redundant definitions are not only tedious, but also error-prone, the goal is to minimize the redundancy inherent in the grammar developed here. Therefore, each CGM element is described as a unit, using a notion that implies the definition of the element:

<ELEMENT> := <ELEMENT NAME> ':=' '{' <VALUES>* ¦ <ELEMENT>* '}'

Elements are:

* The 91 CGM elements as specified in ISO 8632, Part 3.

* Attributes, whose values are to be specified.

* Constraints, which are valid for the entire formal description, e.g., the maximum number of points in a point list "MAX_POINTS".

* Macros, whereupon each occurrence of the macro-name (except its definition) is replaced by the text given by its definition. Macros are allowed to be nested as long as recursion, either direct or indirect, is avoided.

Since an element has characteristics, which, in turn, can have different characteristics for itself, a recursive structure is appropriate. For example, the element METAFILE DESCRIPTION is associated with a string specifying the content of the metafile or a description of the product that generated the metafile. A string can have several characteristics, too; e.g., a maximum length or a required substring and so on. Therefore, the definition of <ELEMENT> builds a tree. Note that each <ELEMENT> is defined by <VALUES>* or <ELEMENT>* exclusively.

B.2 Principles

A formal description is a set of elements:

```
<FORMAL DESCRIPTION> := <ELEMENT>*
```

A basic value is called an atom. Since there are different modes in which an atom can be the value of an element, there are different modes of its specification. An atom can specify the standard values which are allowed for a specific element, default values, or values which are not defined by the CGM standard (private use of elements):

```
<VALUES> := { <PRIVATE> ¦ <DEFAULT> ¦ <STANDARD> }
```

The three modes are distinguished by the use of different brackets symbols to enclose a set of atoms:

```
<PRIVATE>  :=  '(' <ATOM>* ')'
<DEFAULT>  :=  '[' <ATOM>* ']'
<STANDARD> :=  '{' <ATOM>* '}'
```

Note that the default value does not have to be repeated in the list of standard values.

There is no specific order in which private, default and standard values must occur; that is, they can be mixed up.

Since different separators can be used, the appearance of an element definition can be designed in a fashion that is appropriate to the characteristic in question. Consequently, the following descriptions are equivalent:

```
{'a','b','c'} = {'a' 'b' 'c'} = {'a'},{'b'},{'c'} = {'a'}{'b'}{'c'}
```

For example, when enumerating some values (including a range), then:

```
{ -303..-301, 1..5 }
```

is more readable as an atom than

```
{ -303..-301 }, { 1..5 }
```

An atom is what it implies; namely, a character, a string, a number, an RGB value, a point with integer coordinates, or a point with real coordinates. Since RGB values and points are compound types, ranges of values are treated as atoms, too:

```
<ATOM> := <CHARACTER> ¦ <STRING> ¦ <NUMBER> ¦ <RGB> ¦ <POINT> ¦
          <RANGE>
```

The non-terminal symbols described by the preceding rules are:

```
<CHARACTER>     :=   '"'<LETTER>'"' ¦ <NUMBER>
<STRING>        :=   '"'<LETTER>*'"'
<NUMBER>        :=   <INTEGER> ¦ <REAL>
<RGB>           :=   '<' <INTEGER> <INTEGER> <INTEGER> '>'
<POINT>         :=   '<' <NUMBER> <NUMBER> '>'
<RANGE>         :=   <NUMBER>'..'<NUMBER> ¦
                     <LETTER>'..'<LETTER>
<INTEGER>       :=   ['+'¦'-']<DIGIT>+
<REAL>          :=   [['+'¦'-']<DIGIT>+]'.'<DIGIT>+
                     [['+'¦'-']'e'¦'E'¦<DIGIT>+]
```

The terminal symbols are:

```
<DIGIT>         :=   { '0'..'9' }
<LETTER>        :=   { 'a'..'z', 'A'..'Z', '0'..'9',
                       ' ', '+', '-', '#', ';', '/',
                       '(', ')', ',', '.', ''', '"',
                       '_', '$', '%' }
```

Note that all enumerations, such as element definition, atoms or even the components of a RGB value or the coordinates of a point can be separated either by a blank or by a comma.

Since not every character can be specified by an editor, characters are allowed to be specified by the corresponding ASCII code number.

B.3 An Example

Some of the mechanisms are illustrated by the following example:

```
"constraints" := {
   "MAX_STRING_LENGTH" := { 256 }
   "VALID_CHARACTERS"  := { 'a'..'z', 'A'..'Z', '0'..'9' }
}

"macro definitions" := {
   "standard string type" := { "MAX_STRING_LENGTH",
                               "VALID_CHARACTERS"   }
}

"delimiter elements" := {
   "BEGIN METAFILE" := {
      "STRING" := { "standard string type",
                    "USAGE" := [ "sometimes" ] { "always", "never"
},
                    "SUB_STRING" := { "example" }
      }
   }
}
```

It can be seen that some items are written in upper case and some in lower case. By convention the names of CGM elements (e.g., "BEGIN METAFILE"), constraints (e.g., "MAX_STRING_LENGTH") and characteristics (e.g., "USAGE") are written in upper case, whereas categories (e.g., "constraints"), macros (e.g., "standard string type") and enumerated string alternatives (e.g., "optional") are written in lower case. This convention was made to make a formal description more readable and is more a suggestion than an imperative. In any case, the expressions "example", "Example" and "EXAMPLE" are treated as being different.

A formal description is specified as a set of differences from the base CGM standard. For instance, in the CALS application profile, the maximum string length is restricted to 256 characters (see "MAX_STRING_LENGTH") and only letters and digits may appear in a string (see "VALID_CHARACTERS"). Characteristics like these are known as constraints. Each constraint is global unless they are specified in elements which define this constraint locally. For example, "MAX_STRING_LENGTH" restricts the maximum string length for all strings of the application. If a formal description contains the definition:

```
"BEGIN METAFILE" := { "MAX_STRING_LENGTH" := { 1024 },
                                    ...
                    }
```

the maximum length is set to 1024 characters for the element "BEGIN METAFILE" only, whereas the global definition holds for all other string elements.

An element name followed by ':=' denotes the definition of the designated element. If no ':=' appears, it is assumed that the element is defined as a constraint or a macro. In this case, the value defined there is substituted for the occurrence of the element name. The difference between constraint and a macro is that a constraint expresses some global properties of the formal description, whereas a macro doesn't have any meaning except that of textual substitution. It should be mentioned that macro-names are the only names that can be choosen arbitrarily. All other names must be chosen from a predefined set of keywords.

In the example, first "MAX_STRING_LENGTH" is set to 256 characters as a constraint. This definition is used to define the macro "standard string type". Later, the macro is used in a string specification. The definition of "STRING" in the example evaluates to:

```
"STRING" := { "MAX_STRING\_LENGTH" := { 256 },
              "VALID_CHARACTERS"   := { 'a'..'z', 'A'..'Z',
                                        '0'..'9' },
              "USAGE" := [ "sometimes" ] { "always", "never" },
              "SUB_STRING" := { "example" }
            }
```

The CGM standard allows the element "BEGIN METAFILE" to be followed by a string. The element "USAGE" specifies whether a string must appear or not. The possibilities are "always" (a string must appear), "never" (a string must not appear), and "optional" (a string may appear or not). These three values are enumerated in the example, where "optional" is set to be the default.

"SUB_STRING" specifies a string, which must occur somewhere in the string. In the example this substring is "example". As with numbers and letters, the wildcard '*' can be used to denote any number (for numbers), any letter (for characters), and any string (for strings). When specifying strings, the wildcard '?' (exactly one character) can also be used. This is illustrated by some examples:

```
*..10      :    a range up to 10 (no lower bound).
'a'..*     :    any character with an ascii code above 'a'.
'a'..'*'   :    any character between 'a' and '*' ('*' is a
                character).
"te*t"     :    any string with a leading 'te' and a trailing 't',
                for instance "test" or "text" (but also "tet").
"te?*t"    :    the same as the previous example, but "tet" doesn't
                match.
```

B.4 Notation

The syntax used to describe the grammar uses the following elements:

```
:=              :    separates the left and right side of a rule
{...}           :    grouping of symbols
<...>           :    enclosing of a non-terminal symbol
'...'           :    enclosing of a terminal symbol
<...>*          :    star closure (0 or more occurrences)
<...>+          :    plus closure (1 or more occurrences)
[...]           :    optional element (can be omitted)
¦               :    exclusive OR
'a'..'c'        :    shortcut for 'a','b','c'
{'a','b','c'}   :    one of 'a', 'b' or 'c'
```

Comments can be included in formal description to make it more readable. A comment is introduced by a '%', except when it occurs within a string embedded in "...". The characters of a line following a '%' (outside a string) are ignored.

APPENDIX C

GENERATOR QUESTIONNAIRE

delimiter elements

1 NO-OPERATION

1a) Does your implementation use this element? sometimes ☐ always ☐ never ☐

1b) What is the min. number of NO-OPS? ☐

1c) What is the max. number of NO-OPS? ☐

2 BEGIN METAFILE

*** Caution: this element is required ***

2a) Does your implementation use the string parameter? sometimes ☐ always ☐ never ☐

2b) What is the min. number of characters? ☐

2c) What is the max. number of characters? ☐

2d) In what range of ASCII characters does the value of a string–character–code always lie?

_____

_____

3 END METAFILE

*** Caution: this element is required ***

4 BEGIN PICTURE

*** Caution: this element is required ***

4a) What is the min. number of pictures in a CGM? ☐

4b) What is the max. number of pictures in a CGM? ☐

4c) Does your implementation use the string parameter? sometimes ☐ always ☐ never ☐

4d) What is the min. number of characters? ☐

4e) What is the max. number of characters? ☐

4f) In what range of ASCII characters does the value of a string–character–code always lie?

_____

_____

5 BEGIN PICTURE BODY

*** Caution: this element is required ***

6 END PICTURE

*** Caution: this element is required ***

metafile descriptor elements

7 METAFILE VERSION

*** Caution: this element is required ***

7a) Specify the version number which may appear in a CGM (enumerate)?

_____

_____

_____

_____

8 METAFILE DESCRIPTION

8a) Does your implementation use this element?

sometimes ☐
always ☐
never ☐

8b) Does your implementation use the string parameter?

sometimes ☐
always ☐
never ☐

8c) What is the min. number of characters? ☐

8d) What is the max. number of characters? ☐

8e) In what range of ASCII characters does the value of a string–character–code always lie?

_____

_____

9 VDC TYPE

9a) Does your implementation use this element?

sometimes ☐
always ☐
never ☐

9b) What VDC TYPEs does your implementation use?

integer ☐
real ☐

10 INTEGER PRECISION

10a) Does your implementation use this element?

sometimes ☐
always ☐
never ☐

10b) What INTEGER PRECISIONs does your implementation use?

16 ☐
8 ☐
24 ☐
32 ☐

11 REAL PRECISION

11a) Does your implementation use this element?

sometimes ☐
always ☐
never ☐

11b) What REAL PRECISIONs does your implementation use?

32-bit fixed point ☐
32-bit floating point ☐
64-bit floating point ☐
64-bit fixed point ☐

12 INDEX PRECISION

12a) Does your implementation use this element?

sometimes ☐
always ☐
never ☐

12b) What INDEX PRECISIONs does your implementation use?

16 ☐
8 ☐
24 ☐
32 ☐

13 COLOUR PRECISION

13a) Does your implementation use this element?

sometimes ☐
always ☐
never ☐

13b) What COLOUR PRECISIONs does your implementation use?

8 ☐
16 ☐
24 ☐
32 ☐

14 COLOUR INDEX PRECISION

14a) Does your implementation use this element?

sometimes ☐
always ☐
never ☐

14b) What COLOUR INDEX PRECISIONs does your implementation use?

16 ☐
8 ☐
24 ☐
32 ☐

15 MAXIMUM COLOUR INDEX

15a) Does your implementation use this element?

sometimes ☐
always ☐
never ☐

15b) What is the min. value for MAXIMUM COLOUR INDEX? ☐

15c) What is the max. value for MAXIMUM COLOUR INDEX? ☐

16 COLOUR VALUE EXTENT

16a) Does your implementation use this element?

sometimes ☐
always ☐
never ☐

16b) In what range of RGB values does the value of the COLOUR VALUE EXTENT always lie?

_____

_____

17 METAFILE ELEMENT LIST

*** Caution: this element is required ***

17a) Does your implementation use the shorthand pseudoelements?

sometimes ☐
always ☐
never ☐

17b) What shorthand pseudoelements does your implementation use?

Drawing set ☐
Drawing-plus-control set ☐

18 METAFILE DEFAULTS REPLACEMENT

18a) Does your implementation use this element?

sometimes ☐
always ☐
never ☐

18b) What METAFILE DEFAULTS REPLACEMENT does your implementation use?

VDC TYPE ☐
INTEGER PRECISION ☐
REAL PRECISION ☐
INDEX PRECISION ☐
COLOUR INDEX PRECISION ☐
COLOUR VALUE EXTENT ☐
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 INTEGER PRECISION ☐
VDC REAL PRECISION ☐
AUXILIARY COLOUR ☐
TRANSPARENCY ☐
CLIP RECTANGLE ☐
CLIP INDICATOR ☐
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 ☐
CHARACTER EXPANSION FACTOR ☐
CHARACTER SPACING ☐
TEXT COLOUR ☐
CHARACTER HEIGHT ☐
CHARACTER ORIENTATION ☐
TEXT PATH ☐
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 ☐
COLOUR TABLE ☐
ASPECT SOURCE FLAGS ☐

19 FONT LIST

19a) Does your implementation use this element?

sometimes ☐
always ☐
never ☐

19b) What is the min. number of font names? ☐

19c) What is the max. number of font names? ☐

19d) What is the min. length of a font name? ☐

19e) What is the max. length of a font name? ☐

19f) Specify the font names which may appear in the parameter list (list or describe)?

_____

_____

_____

_____

20 CHARACTER SET LIST

20a) Does your implementation use this element?

sometimes ☐
always ☐
never ☐

20b) What is the min. number of character sets? ☐

20c) What is the max. number of character sets? ☐

20d) What is the min. length of a sequence tail? ☐

20e) What is the max. length of a sequence tail? ☐

20f) What CHARACTER_SET_TYPE does your implementation use?

94-character G-set ☐
96-character G-set ☐
94-character multibyte G-set ☐
96-character multibyte G-set ☐
complete code ☐

21 CHARACTER CODING ANNOUNCER

21a) Does your implementation use this element?

sometimes ☐
always ☐
never ☐

21b) What CHARACTER CODING ANNOUNCER does your implementation use?

7-bit basic ☐
8-bit basic ☐
7-bit extended ☐
8-bit extended ☐
something else ☐

21c) Does your implementation use negative values for CHARACTER CODING ANNOUNCER to indicate private use?

yes ☐
no ☐

picture descriptor elements

22 SCALING MODE

22a) Does your implementation use this element?

sometimes ☐
always ☐
never ☐

22b) What SCALING MODEs does your implementation use?

abstract ☐
metric ☐

22c) In what range of real numbers does the value of the METRIC_SCALE_FACTOR always lie?

_____

_____

22d) What METRIC SCALE FACTOR precisions does your implementation use?

32-bit fixed point ☐

23 COLOUR SELECTION MODE

23a) Does your implementation use this element?

sometimes ☐
always ☐
never ☐

23b) What COLOUR SELECTION MODEs does your implementation use?

indexed ☐
direct ☐

24 LINE WIDTH SPECIFICATION MODE

24a) Does your implementation use this element?

sometimes ☐
always ☐
never ☐

24b) What LINE WIDTH SPECIFICATION MODEs does your implementation use?

scaled ☐
absolute ☐

25 MARKER SIZE SPECIFICATION MODE

25a) Does your implementation use this element?

sometimes ☐
always ☐
never ☐

25b) What MARKER SIZE SPECIFICATION MODEs does your implementation use?

scaled ☐
absolute ☐

26 EDGE WIDTH SPECIFICATION MODE

26a) Does your implementation use this element?

- sometimes ☐
- always ☐
- never ☐

26b) What EDGE WIDTH SPECIFICATION MODEs does your implementation use?

- scaled ☐
- absolute ☐

27 VDC EXTENT

27a) Does your implementation use this element?

- sometimes ☐
- always ☐
- never ☐

27b) Does your implementation use address spaces where the first point is not to the left and below the second point?

- sometimes ☐
- always ☐
- never ☐

28 BACKGROUND COLOUR

28a) Does your implementation use this element?

- sometimes ☐
- always ☐
- never ☐

control elements

29 VDC INTEGER PRECISION

29a) Does your implementation use this element?

- sometimes ☐
- always ☐
- never ☐

29b) What VDC INTEGER PRECISIONs does your implementation use?

- 16 ☐
- 24 ☐
- 32 ☐

30 VDC REAL PRECISION

30a) Does your implementation use this element?

- sometimes ☐
- always ☐
- never ☐

30b) What VDC REAL PRECISIONs does your implementation use?

- 32-bit fixed point ☐
- 32-bit floating point ☐
- 64-bit floating point ☐
- 64-bit fixed point ☐

31 AUXILIARY COLOUR

31a) Does your implementation use this element?

sometimes ☐
always ☐
never ☐

32 TRANSPARENCY

32a) Does your implementation use this element?

sometimes ☐
always ☐
never ☐

32b) What on/off flags does your implementation use?

on ☐
off ☐

33 CLIP RECTANGLE

33a) Does your implementation use this element?

sometimes ☐
always ☐
never ☐

34 CLIP INDICATOR

34a) Does your implementation use this element?

sometimes ☐
always ☐
never ☐

34b) What clip indicator values does your implementation use?

on ☐
off ☐

graphical primitive elements

35 POLYLINE

35a) Does your implementation use this element?

sometimes ☐
always ☐
never ☐

35b) What is the min. number of points (coordinate pairs) in the parameter list? ☐

35c) What is the max. number of points (coordinate pairs) in the parameter list? ☐

36 DISJOINT POLYLINE

36a) Does your implementation use this element?

sometimes ☐
always ☐
never ☐

36b) What is the min. number of points (coordinate pairs) in the parameter list? ☐

36c) What is the max. number of points (coordinate pairs) in the parameter list? ☐

37 POLYMARKER

37a) Does your implementation use this element?

sometimes ☐
always ☐
never ☐

37b) What is the min. number of points (coordinate pairs) in the parameter list? ☐

37c) What is the max. number of points (coordinate pairs) in the parameter list? ☐

38 TEXT

38a) Does your implementation use this element?

sometimes ☐
always ☐
never ☐

38b) What is the min. number of characters in a text string? ☐

38c) What is the max. number of characters in a text string? ☐

38d) In what range of ASCII characters does the value of a string–character–code always lie?

_____

_____

38e) What final/non-final flags does your implementation use?

final ☐
non-final ☐

38f) What text attribute modifications between non–final text does your implementation use?

TEXT FONT INDEX ☐
CHARACTER EXPANSION FACTOR ☐
CHARACTER SPACING ☐

TEXT COLOUR ☐
CHARACTER HEIGHT ☐
CHARACTER SET INDEX ☐

39 RESTRICTED TEXT

39a) Does your implementation use this element?

sometimes ☐
always ☐
never ☐

39b) What is the min. number of characters in a text string? ☐

39c) What is the max. number of characters in a text string? ☐

39d) In what range of ASCII characters does the value of a string–character–code always lie?

_____

_____

39e) What final/non-final flags does your implementation use?

final ☐
non-final ☐

40 APPEND TEXT

40a) Does your implementation use this element?

sometimes ☐
always ☐
never ☐

40b) What is the min. number of characters in a text string? ☐

40c) What is the max. number of characters in a text string? ☐

40d) In what range of ASCII characters does the value of a string–character–code always lie?

_____

_____

40e) What final/non-final flags does your implementation use?

final ☐
non-final ☐

41 POLYGON

41a) Does your implementation use this element?

sometimes ☐
always ☐
never ☐

41b) What is the min. number of points (coordinate pairs) in the parameter list? ☐

41c) What is the max. number of points (coordinate pairs) in the parameter list? ☐

42 POLYGON SET

42a) Does your implementation use this element?

sometimes ☐
always ☐
never ☐

42b) What is the min. number of points (coordinate pairs) in the parameter list? ☐

42c) What is the max. number of points (coordinate pairs) in the parameter list? ☐

42d) What edge-out flags does your implementation use?

- invisible ☐
- visible ☐
- close/invisible ☐
- close/visible ☐

43 CELL ARRAY

43a) Does your implementation use this element?

- sometimes ☐
- always ☐
- never ☐

43b) What is the min. number of columns? ☐

43c) What is the max. number of columns? ☐

43d) What is the min. number of rows? ☐

43e) What is the max. number of rows? ☐

43f) What LOCAL COLOUR PRECISIONs does your implementation use?

- 0 ☐
- 1 ☐
- 2 ☐
- 4 ☐
- 8 ☐
- 16 ☐
- 24 ☐
- 32 ☐

43g) What cell representation modes does your implementation use?

- run-length ☐
- packed ☐

44 GENERALIZED DRAWING PRIMITIVE

44a) Does your implementation use this element?

- sometimes ☐
- always ☐
- never ☐

44b) In what range of indices does the value of the GDP identifier always lie?

_____

_____

44c) What is the min. number of characters in a data record string? ☐

44d) What is the max. number of characters in a data record string? ☐

45 RECTANGLE

45a) Does your implementation use this element?

- sometimes ☐
- always ☐
- never ☐

46 CIRCLE

46a) Does your implementation use this element?

sometimes ☐
always ☐
never ☐

47 CIRCULAR ARC 3 POINT

47a) Does your implementation use this element?

sometimes ☐
always ☐
never ☐

48 CIRCULAR ARC 3 POINT CLOSE

48a) Does your implementation use this element?

sometimes ☐
always ☐
never ☐

49 CIRCULAR ARC CENTRE

49a) Does your implementation use this element?

sometimes ☐
always ☐
never ☐

50 CIRCULAR ARC CENTRE CLOSE

50a) Does your implementation use this element?

sometimes ☐
always ☐
never ☐

51 ELLIPSE

51a) Does your implementation use this element?

sometimes ☐
always ☐
never ☐

52 ELLIPTICAL ARC

52a) Does your implementation use this element?

sometimes ☐
always ☐
never ☐

53 ELLIPTICAL ARC CLOSE

53a) Does your implementation use this element?

sometimes ☐
always ☐
never ☐

attribute elements

54 LINE BUNDLE INDEX

54a) Does your implementation use this element?

sometimes ☐
always ☐
never ☐

54b) In what range of indices does the value of the LINE BUNDLE INDEX always lie?

_____

_____

55 LINE TYPE

55a) Does your implementation use this element?

sometimes ☐
always ☐
never ☐

55b) In what range of indices does the value of the LINE TYPE always lie?

_____

_____

55c) Does your implementation use negative values for LINE TYPE to indicate private use?

yes ☐
no ☐

56 LINE WIDTH

56a) Does your implementation use this element?

sometimes ☐
always ☐
never ☐

57 LINE COLOUR

57a) Does your implementation use this element?

sometimes ☐
always ☐
never ☐

58 MARKER BUNDLE INDEX

58a) Does your implementation use this element?

sometimes ☐
always ☐
never ☐

58b) In what range of indices does the value of the MARKER BUNDLE INDEX always lie?

_____

_____

59 MARKER TYPE

59a) Does your implementation use this element?

sometimes ☐
always ☐
never ☐

59b) Does your implementation use negative values for MARKER TYPE to indicate private use?

yes ☐
no ☐

60 MARKER SIZE

60a) Does your implementation use this element?

sometimes ☐
always ☐
never ☐

61 MARKER COLOUR

61a) Does your implementation use this element?

sometimes ☐
always ☐
never ☐

62 TEXT BUNDLE INDEX

62a) Does your implementation use this element?

sometimes ☐
always ☐
never ☐

62b) In what range of indices does the value of the TEXT BUNDLE INDEX always lie?

_____

_____

63 TEXT FONT INDEX

63a) Does your implementation use this element?

sometimes ☐
always ☐
never ☐

63b) In what range of indices does the value of the TEXT FONT INDEX always lie?

_____

_____

64 TEXT PRECISION

64a) Does your implementation use this element?

sometimes ☐
always ☐
never ☐

64b) What TEXT PRECISION does your implementation use?

string ☐
character ☐
stroke ☐

65 CHARACTER EXPANSION FACTOR

65a) Does your implementation use this element?

sometimes ☐
always ☐
never ☐

65b) In what range of real numbers does the value of a CHARACTER EXPANSION FACTOR always lie?

_____

_____

66 CHARACTER SPACING

66a) Does your implementation use this element?

sometimes ☐
always ☐
never ☐

66b) In what range of real numbers does the value of a CHARACTER SPACING always lie?

_____

_____

67 TEXT COLOUR

67a) Does your implementation use this element?

sometimes ☐
always ☐
never ☐

68 CHARACTER HEIGHT

68a) Does your implementation use this element?

sometimes ☐
always ☐
never ☐

69 CHARACTER ORIENTATION

69a) Does your implementation use this element?

sometimes ☐
always ☐
never ☐

70 TEXT PATH

70a) Does your implementation use this element?

sometimes ☐
always ☐
never ☐

70b) What TEXT PATH values does your implementation use?

right ☐
left ☐
up ☐
down ☐

71 TEXT ALIGNMENT

71a) Does your implementation use this element?

sometimes ☐
always ☐
never ☐

71b) What horizontal alignment values does your implementation use?

normal ☐
left ☐
centre ☐

right ☐
continuous horizontal ☐

71c) What vertical alignment values does your implementation use?

normal ☐
top ☐
cap ☐

half ☐
base ☐
continuous vertical ☐

72 CHARACTER SET INDEX

72a) Does your implementation use this element?

sometimes ☐
always ☐
never ☐

72b) In what range of indices does the value of the CHARACTER SET INDEX always lie?

_____

_____

73 ALTERNATE CHARACTER SET INDEX

73a) Does your implementation use this element?

sometimes ☐
always ☐
never ☐

73b) In what range of indices does the value of the ALTERNATE CHARACTER SET INDEX always lie?

_____

_____

74 FILL BUNDLE INDEX

74a) Does your implementation use this element?

sometimes ☐
always ☐
never ☐

74b) In what range of indices does the value of the FILL BUNDLE INDEX always lie?

_____

_____

75 INTERIOR STYLE

75a) Does your implementation use this element?

sometimes ☐
always ☐
never ☐

75b) What INTERIOR STYLE does your implementation use?

hollow ☐
solid ☐
pattern ☐
hatch ☐
empty ☐
something else ☐

75c) Does your implementation use negative values for INTERIOR STYLE to indicate private use?

yes ☐
no ☐

76 FILL COLOUR

76a) Does your implementation use this element?

sometimes ☐
always ☐
never ☐

77 HATCH INDEX

77a) Does your implementation use this element?

sometimes ☐
always ☐
never ☐

77b) In what range of indices does the value of the HATCH INDEX always lie?

_____

_____

77c) Does your implementation use negative values for HATCH INDEX to indicate private use?

yes ☐
no ☐

78 PATTERN INDEX

78a) Does your implementation use this element?

sometimes ☐
always ☐
never ☐

78b) In what range of indices does the value of the PATTERN INDEX always lie?

_____

_____

79 EDGE BUNDLE INDEX

79a) Does your implementation use this element? sometimes ☐ always ☐ never ☐

79b) In what range of indices does the value of the EDGE BUNDLE INDEX always lie?

_____

_____

80 EDGE TYPE

80a) Does your implementation use this element? sometimes ☐ always ☐ never ☐

80b) In what range of indices does the value of the EDGE TYPE always lie?

_____

_____

80c) Does your implementation use negative values for EDGE TYPE to indicate private use? yes ☐ no ☐

81 EDGE WIDTH

81a) Does your implementation use this element? sometimes ☐ always ☐ never ☐

82 EDGE COLOUR

82a) Does your implementation use this element? sometimes ☐ always ☐ never ☐

83 EDGE VISIBILITY

83a) Does your implementation use this element? sometimes ☐ always ☐ never ☐

83b) What on/off flags does your implementation use?

off ☐
on ☐

84 FILL REFERENCE POINT

84a) Does your implementation use this element? sometimes ☐ always ☐ never ☐

85 PATTERN TABLE

85a) Does your implementation use this element?

sometimes ☐
always ☐
never ☐

85b) In what range of indices does the value of the INDEX always lie?

_____

_____

85c) In what range of integers does the value of the number of columns always lie?

_____

_____

85d) In what range of integers does the value of the number of rows always lie?

_____

_____

85e) What LOCAL COLOUR PRECISIONs does your implementation use?

0	☐	8	☐
1	☐	16	☐
2	☐	24	☐
4	☐	32	☐

86 PATTERN SIZE

86a) Does your implementation use this element?

sometimes ☐
always ☐
never ☐

87 COLOUR TABLE

87a) Does your implementation use this element?

sometimes ☐
always ☐
never ☐

87b) What is the min. number of COLOUR TABLE entries? ☐

87c) What is the max. number of COLOUR TABLE entries? ☐

87d) In what range of indices does the value of the INDEX always lie?

_____

_____

88 ASPECT SOURCE FLAGS

88a) Does your implementation use this element?

sometimes ☐
always ☐
never ☐

88b) What ASPECT SOURCE FLAGS does your implementation use?

	individual	bundled
LINE TYPE	☐	☐
LINE WIDTH	☐	☐
LINE COLOUR	☐	☐
MARKER TYPE	☐	☐
MARKER SIZE	☐	☐
MARKER COLOUR	☐	☐
TEXT FONT INDEX	☐	☐
TEXT PRECISION	☐	☐
CHARACTER EXPANSION FACTOR	☐	☐
CHARACTER SPACING	☐	☐
TEXT COLOUR	☐	☐
INTERIOR STYLE	☐	☐
FILL COLOUR	☐	☐
HATCH INDEX	☐	☐
PATTERN INDEX	☐	☐
EDGE TYPE	☐	☐
EDGE WIDTH	☐	☐
EDGE COLOUR	☐	☐

escape elements

89 ESCAPE

89a) Does your implementation use this element?

sometimes ☐
always ☐
never ☐

89b) In what range of indices does the value of the ESCAPE identifier always lie?

_____

_____

89c) What is the min. number of characters in a data record string? ☐

89d) What is the max. number of characters in a data record string? ☐

external elements

90 MESSAGE

90a) Does your implementation use this element?

sometimes ☐
always ☐
never ☐

90b) What action-required flags does your implementation use?

no action ☐
action ☐

90c) What is the min. number of characters in a message string? ☐

90d) What is the max. number of characters in a message string? ☐

91 APPLICATION DATA

91a) Does your implementation use this element?

sometimes ☐
always ☐
never ☐

91b) In what range of indices does the value of the APPLICATION DATA identifier always lie?

_____

_____

91c) What is the min. number of characters in a data record string? ☐

91d) What is the max. number of characters in a data record string? ☐

92 Primitives and attributes

Mark the attributes available for each of the listed elements

	POLYLINE	DISJOINT POLYLINE	CIRCULAR ARC 3 POINT	CIRCULAR ARC CENTRE	ELLIPTICAL ARC
LINE BUNDLE INDEX					
LINE TYPE					
LINE WIDTH					
LINE COLOUR					

	POLYMARKER
MARKER BUNDLE INDEX	
MARKER TYPE	
MARKER SIZE	
MARKER COLOUR	

	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			

	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								
COLOUR TABLE								

APPENDIX D

TEST SUITE DESCRIPTION DATABASE

ELEMENT Question Answer		*Implications for Testing* and Requirements on CGM Suite-under-Test
NO-OPERATION		
1a	sometimes used	At least 5 of the files in the CGM suite-under-test shall include this element.
1b	min. number of NO-OPS	At least one instance of a NO-OPERATION element which contains the minimum number of NO-OPs allowed shall appear in the files that comprise the CGM suite-under-test.
1c	max. number of NO-OPS	At least one instance of a NO-OPERATION element which contains the maximum number of NO-OPs allowed shall appear in the files that comprise the CGM suite-under-test. If there is no maximum, then use the value 500.
BEGIN METAFILE		*Element must always be present.*
2a	string sometimes used	At least 50% of the instances, but not more than 75% of the instances, of this element shall use the string parameter.
2b	minimum string length	At least one of the files in the CGM suite-under-test shall contain a string with the minimum number of characters.
2c	maximum string length	At least one of the files in the CGM suite-under-test shall contain a string with the maximum number of characters. If there is no maximum, then use the value 300.
2d	characters used	Every ASCII character code that can appear in the string parameter shall appear in at least one instance of this element.
END METAFILE		*Element must always be present.*
BEGIN PICTURE		*Element must always be present, if there is at least one picture in the metafile.*
4a	min. number of pictures	At least one instance of the files in the CGM suite-under-test shall contain the minimum number of pictures.

ELEMENT Question	Answer	*Implications for Testing* and Requirements on CGM Suite-under-Test
4b	max. number of pictures	At least one instance of the files in the CGM suite-under-test shall contain the maximum number of pictures. If there is no maximum, then use the value 5.
4c	string sometimes used	At least 50% of the instances, but not more than 75% of the instances, of this element shall use the string parameter.
4d	minimum string length	At least one of the files in the CGM suite-under-test shall contain a string with the minimum number of characters.
4e	maximum string length	At least one of the files in the CGM suite-under-test shall contain a string with the maximum number of characters. If there is no maximum, then use the value 300.
4f	characters used	Every ASCII character that can appear in the string parameter shall appear in at least one instance of this element.
BEGIN PICTURE BODY		*Element must always be present, if there is at least one picture in the metafile.*
END PICTURE		*Element must always be present, if there is at least one picture in the metafile.*
METAFILE VERSION		*Element must always be present.*
7a	version = 1	*This is the only version we can test.*
	version = 2 or 3	*Cannot be tested.*
	version > 3	*Error; cannot test until fixed.*
METAFILE DESCRIPTION		
8a	sometimes used	At least 50% of the files, but not more than 90% of the files, in the CGM suite-under-test shall include this element.
8b	string sometimes used	At least 50% of the instances, but not more than 75% of the instances, of this element shall use the string parameter.
8c	minimum string length	At least one of the files in the CGM suite-under-test shall contain a string with the minimum number of characters.

ELEMENT Question	Answer	*Implications for Testing* and Requirements on CGM Suite-under-Test
8d	maximum string length	At least one of the files in the CGM suite-under-test shall contain a string with the maximum number of characters. If there is no maximum, then use the value 300.
8e	characters used	Every ASCII character that can appear in the string parameter shall appear in at least one instance of this element.
VDC TYPE		
9a	when element not used	*All appropriate CGM elements should use integer VDC.*
	sometimes used	At least 50% of the files, but not more than 90% of the files, in the CGM suite-under-test shall include this element.
9b	both types supported	At least one third of the files in the CGM suite-under-test shall use the less frequently used VDC type.
INTEGER PRECISION		
10a	when element not used	*All appropriate CGM elements should use 16-bit integer precision.*
	sometimes used	At least 50% of the files, but not more than 90% of the files, in the CGM suite-under-test shall include this element.
10b	only one precision used	No requirement.
	exactly two precisions used	At least one third of the files in the CGM suite-under-test shall use the less frequently used integer precision.
	three or four precisions used	Each precision used shall be represented in at least 5 of the files in the CGM suite-under-test.
REAL PRECISION		
11a	when element not used	*All appropriate CGM elements should use 32-bit fixed point real precision.*
	sometimes used	At least 50% of the files, but not more than 90% of the files, in the CGM suite-under-test shall include this element.
11b	only one precision used	No requirement.
	exactly two precisions used	At least one third of the files in the CGM suite-under-test shall use the less frequently used real precision.

ELEMENT Question Answer	*Implications for Testing* and Requirements on CGM Suite-under-Test
three or four precisions used	Each precision used shall be represented in at least 5 of the files in the CGM suite-under-test.
INDEX PRECISION	
12a when element not used	*All appropriate CGM elements should use 16-bit index precision.*
sometimes used	At least 50% of the files, but not more than 90% of the files, in the CGM suite-under-test shall include this element.
12b only one precision used	No requirement.
exactly two precisions used	At least one third of the files in the CGM suite-under-test shall use the less frequently used index precision.
three or four precisions used	Each precision used shall be represented in at least 5 of the files in the CGM suite-under-test.
COLOUR PRECISION	
13a when element not used	*All appropriate CGM elements should use 8-bit colour precision.*
sometimes used	At least 50% of the files, but not more than 90% of the files, in the CGM suite-under-test shall include this element.
13b only one precision used	No requirement.
exactly two precisions used	At least one third of the files in the CGM suite-under-test shall use the less frequently used colour precision.
three or four precisions used	Each precision used shall be represented in at least 5 of the files in the CGM suite-under-test.
COLOUR INDEX PRECISION	
14a when element not used	*All appropriate CGM elements should use 16-bit colour index precision.*
sometimes used	At least 50% of the files, but not more than 90% of the files, in the CGM suite-under-test shall include this element.
14b only one precision used	No requirement.
exactly two precisions used	At least one third of the files in the CGM suite-under-test shall use the less frequently used colour index precision.
three or four precisions used	Each precision used shall be represented in at least 5 of the files in the CGM suite-under-test.

ELEMENT Question Answer	*Implications for Testing* and Requirements on CGM Suite-under-Test
MAXIMUM COLOUR INDEX	
15a when element not used	*All appropriate CGM elements should contain colour indices that do not exceed the value of 63.*
sometimes used	At least 50% of the files, but not more than 90% of the files, in the CGM suite-under-test shall include this element.
15b min. value of max. index	Use of the minimum value possible for the maximum colour index element shall be represented in at least 2 of the files in the CGM suite-under-test.
15c max. value of max. index	Use of the maximum value possible for the maximum colour index element shall be represented in at least 2 of the files in the CGM suite-under-test. If there is no maximum, then use the value 300.
COLOUR VALUE EXTENT	
16a when element not used	*All appropriate CGM elements should contain RGB component values that do not exceed the value of 255.*
sometimes used	At least 50% of the files, but not more than 90% of the files, in the CGM suite-under-test shall include this element.
16b colour value extent range	Use of the maximum values indicated for the RGB components of direct color specifications shall be represented in at least one of the files in the CGM suite-under-test.
METAFILE ELEMENT LIST	*Element must always be present.*
17a use no pseudoelements	No special requirement.
always use pseudoelements	No special requirement.
sometimes use pseudoelem.	At least one third of the files in the CGM suite-under-test shall use one of the pseudoelements.
17b use both pseudoelements	At least one third of the files in the CGM suite-under-test shall use the less frequently used of the pseudoelements.

ELEMENT Question Answer		*Implications for Testing* and Requirements on CGM Suite-under-Test
METAFILE DEFAULTS REPLACEMENT		
18a	sometimes used	At least 50%, but not more than 90%, of the files in the CGM suite-under-test shall include this element.
18b	list of elements	At least one example of each of the elements that can appear in a METAFILE DEFAULTS REPLACEMENT element shall occur within the set of files that comprise the CGM suite-under-test.
FONT LIST		
19a	sometimes used	At least 50%, but not more than 90%, of the files in the CGM suite-under-test shall include this element.
19b	min. number of font names	At least one of the files in the CGM suite-under-test shall contain a FONT LIST element that contains the minimum number of font names.
19c	max. number of font names	At least one of the files in the CGM suite-under-test shall contain a FONT LIST element that contains the maximum number of font names. If there is no maximum, then use the value 10.
19d	min. length of font name	At least one of the files in the CGM suite-under-test shall contain a FONT LIST element that contains a font name whose string length equals the minimum string length allowed for a font name.
19e	max. length of font name	At least one of the files in the CGM suite-under-test shall contain a FONT LIST element that contains a font name whose string length equals the maximum string length allowed for a font name. If there is no maximum, then use the value 300.
19f	allowed font names	At least one instance of each allowed font name shall appear in the files that comprise the CGM suite-under-test.

ELEMENT Question Answer	*Implications for Testing* and Requirements on CGM Suite-under-Test
CHARACTER SET LIST	
20a sometimes used	At least 25%, but not more than 90%, of the files in the CGM suite-under-test shall include this element.
20b min. number of char. sets	At least one of the files in the CGM suite-under-test shall contain a CHARACTER SET LIST element that contains the minimum number of character sets.
20c max. number of char. sets	At least one of the files in the CGM suite-under-test shall contain a CHARACTER SET LIST element that contains the maximum number of character sets. If there is no maximum, then use the value 5.
20d min. length of seq. tail	At least one of the files in the CGM suite-under-test shall contain a CHARACTER SET LIST element that contains a character set designation sequence tail whose string length equals the minimum string length allowed for a designation sequence tail.
20e max. length of seq. tail	At least one of the files in the CGM suite-under-test shall contain a CHARACTER SET LIST element that contains a character set designation sequence tail whose string length equals the maximum string length allowed for a designation sequence tail. If there is no maximum, then use the value 4.
20f only one char. set type	No requirement.
exactly two char. set types	At least one third of the files in the CGM suite-under-test shall use the less frequently used character set type.
three, four, or five types	Each character set type used shall be represented in at least one of the files in the CGM suite-under-test.
CHARACTER CODING ANNOUNCER	
21a sometimes used	At least 25%, but not more than 90%, of the files in the CGM suite-under-test shall include this element.

ELEMENT Question	Answer	*Implications for Testing* and Requirements on CGM Suite-under-Test
21b	only one coding announcer	No requirement.
	exactly two announcers	At least 5 of the files in the CGM suite-under-test shall use the less frequently used character coding announcer.
	three, four, or five types	Each character coding announcer used shall be represented in at least one of the files in the CGM suite-under-test.
21c	negative coding announcer	At least one file in the CGM suite-under-test shall contain an instance of private use.
SCALING MODE		
22a	when element not used	*All appropriate CGM elements should use abstract scaling mode.*
	sometimes used	At least 50% of the files, but not more than 90% of the files, in the CGM suite-under-test shall include this element.
22b	both modes supported	At least one-third of the files in the CGM suite-under-test shall use the less frequently used scaling mode.
22c	minimum metric scale factor	At least one of the files in the CGM suite-under-test with scaling mode *metric* shall contain a metric scale factor equal to the minimum value allowed for this parameter.
	maximum metric scale factor	At least one of the files in the CGM suite-under-test with scaling mode *metric* shall contain a metric scale factor equal to the maximum value allowed for this parameter. If there is no maximum, then use the value 10.0.
22d	metric s.f. precision	*Any answer other than 32-bit fixed point violates the CGM standard.*
COLOUR SELECTION MODE		
23a	when element not used	*All appropriate CGM elements should use indexed colour selection mode.*
	sometimes used	At least 50% of the files, but not more than 90% of the files, in the CGM suite-under-test shall include this element.

ELEMENT Question Answer	*Implications for Testing* and Requirements on CGM Suite-under-Test
23b both modes supported	At least one third of the files in the CGM suite-under-test shall use the less frequently used colour selection mode.
LINE WIDTH SPECIFICATION MODE	
24a when element not used	*All appropriate CGM elements should use scaled line width specification mode.*
sometimes used	At least 50% of the files, but not more than 90% of the files, in the CGM suite-under-test shall include this element.
24b both modes supported	At least one third of the files in the CGM suite-under-test shall use the less frequently used line width specification mode.
MARKER SIZE SPECIFICATION MODE	
25a when element not used	*All appropriate CGM elements should use scaled marker size specification mode.*
sometimes used	At least 50% of the files, but not more than 90% of the files, in the CGM suite-under-test shall include this element.
25b both modes supported	At least one third of the files in the CGM suite-under-test shall use the less frequently used marker size specification mode.
EDGE WIDTH SPECIFICATION MODE	
26a when element not used	*All appropriate CGM elements should use scaled edge width specification mode.*
sometimes used	At least 50% of the files, but not more than 90% of the files, in the CGM suite-under-test shall include this element.
26b both modes supported	At least one third of the files in the CGM suite-under-test shall use the less frequently used edge width specification mode.
VDC EXTENT	
27a when element not used	*If VDC points are expected to be visible in the corresponding CGM hardcopy, all VDC points should lie between (0,0) and (32767,32767) if VDC TYPE is integer and between (0.0,0.0) and (1.0, 1.0) if VDC TYPE is real.*

ELEMENT Question Answer	*Implications for Testing* and Requirements on CGM Suite-under-Test
sometimes used	At least 50% of the files, but not more than 90% of the files, in the CGM suite-under-test shall include this element.
27b sometimes or always	At least one of the files in the CGM suite-under-test shall contain an instance of each of the possible four relationships (below/left, below/right, above/left, above/right) between the first and second VDC points.
BACKGROUND COLOUR	
28a when element not used	*All CGM hardcopies will have a device-dependent background colour (that is, the default background colour will depend on the specific CGM interpret/output device combination). This could easily have the effect of "hiding" some primitives that are expected to be visible in the intended picture.*
sometimes used	At least 50% of the files, but not more than 90% of the files, in the CGM suite-under-test shall include this element.
VDC INTEGER PRECISION	
29a when element not used	*If VDC TYPE is integer, all appropriate CGM elements should use 16-bit integer VDC precision.*
sometimes used	At least 50% of the files, but not more than 90% of the files, in the CGM suite-under-test shall include this element.
29b only one precision used	No requirement.
two or three precisions used	Each precision used shall be represented in at least 5 of the files in the CGM suite-under-test.
VDC REAL PRECISION	
30a when element not used	*IF VDC TYPE is real, all appropriate CGM elements should use 32-bit fixed point real precision.*
sometimes used	At least 50% of the files, but not more than 90% of the files, in the CGM suite-under-test shall include this element.

ELEMENT Question	Answer	*Implications for Testing* and Requirements on CGM Suite-under-Test
30b	only one precision used	No requirement.
	exactly two precisions used	At least one third of the files in the CGM suite-under-test shall use the less frequently used real precision.
	three or four precisions used	Each precision used shall be represented in at least 5 of the files in the CGM suite-under-test.
AUXILIARY COLOUR		
31a	sometimes used	At least 5 of the files in the CGM suite-under-test shall include this element.
TRANSPARENCY		
32a	sometimes used	At least 5 of the files in the CGM suite-under-test shall include this element.
32b	both flags used	At least 2 of the files in the CGM suite-under-test shall use the less frequently used on/off flag value.
CLIP RECTANGLE		
33a	sometimes used	At least 25%, but not more than 90%, of the files in the CGM suite-under-test shall include this element.
CLIP INDICATOR		
34a	sometimes used	At least 25%, but not more than 90%, of the files in the CGM suite-under-test shall include this element.
34b	both values used	At least 5 of the files in the CGM suite-under-test shall use the less frequently used clip indicator value.
POLYLINE		
35a	sometimes used	At least 10, but not more than 90% of the files, in the CGM suite-under-test shall include this element.
35b	min. number of points	At least one of the files in the CGM suite-under-test shall contain a POLYLINE element with the number of coordinate points equal to the minimum number allowed.

ELEMENT Question Answer		*Implications for Testing* and Requirements on CGM Suite-under-Test
35c	max. number of points	At least one of the files in the CGM suite-under-test shall contain a POLYLINE element with the number of coordinate points equal to the maximum number allowed. If there is no maximum, then use 1200.
DISJOINT POLYLINE		
36a	sometimes used	At least 5 of the files, but all of the files, in the CGM suite-under-test shall include this element.
36b	min. number of points	At least one of the files in the CGM suite-under-test shall contain a DISJOINT POLYLINE element with the number of coordinate points equal to the minimum number allowed.
36c	max. number of points	At least one of the files in the CGM suite-under-test shall contain a DISJOINT POLYLINE element with the number of coordinate points equal to the maximum number allowed. If there is no maximum, then use 1200.
POLYMARKER		
37a	sometimes used	At least 10 of the files, but not more than 90% of the files, in the CGM suite-under-test shall include this element.
37b	min. number of points	At least one of the files in the CGM suite-under-test shall contain a POLYMARKER element with the number of coordinate points equal to the minimum number allowed.
37c	max. number of points	At least one of the files in the CGM suite-under-test shall contain a POLYMARKER element with the number of coordinate points equal to the maximum number allowed. If there is no maximum, then use 1200.
TEXT		
38a	sometimes used	At least 10 of the files in the CGM suite-under-test, but not more than 90% of the files, shall contain this element.

ELEMENT Question	Answer	*Implications for Testing* and Requirements on CGM Suite-under-Test
38b	minimum string length	At least one of the files in the CGM suite-under-test shall contain a text string with the minimum number of characters.
38c	maximum string length	At least one of the files in the CGM suite-under-test shall contain a text string with the maximum number of characters. If there is no maximum, then use the value 300.
38d	characters used	Every ASCII character code that can appear in the string parameter shall appear in at least one instance of this element.
38e	both types supported	At least 10 of the files in the CGM suite-under-test shall contain TEXT elements which use the less frequently used final/non-final flag.
38f	some attribute modifications	There shall be at least one example of a TEXT (non-final)/APPEND TEXT (final) pair with an intervening text attribute modification for each text attribute modification used.
RESTRICTED TEXT		
39a	sometimes used	At least 10 of the files in the CGM suite-under-test, but not more than 90% of the files, shall contain this element.
39b	minimum string length	At least one of the files in the CGM suite-under-test shall contain a text string with the minimum number of characters.
39c	maximum string length	At least one of the files in the CGM suite-under-test shall contain a text string with the maximum number of characters. If there is no maximum, then use the value 300.
39d	characters used	Every ASCII character code that can appear in the string parameter shall appear in at least one instance of this element.
39e	both types supported	At least 10 of the files in the CGM suite-under-test shall contain RESTRICTED TEXT elements which use the less frequently used final/non-final flag.

ELEMENT Question Answer	*Implications for Testing* and Requirements on CGM Suite-under-Test
39f some attribute modifications	There shall be at least one example of a RESTRICTED TEXT (non-final)/APPEND TEXT (final) pair with an intervening text attribute modification for each text attribute modification used.
APPEND TEXT	
40a sometimes used	At least 10 of the files in the CGM suite-under-test, but not more than 90% of the files, shall contain this element.
40b minimum string length	At least one of the files in the CGM suite-under-test shall contain a text string with the minimum number of characters.
40c maximum string length	At least one of the files in the CGM suite-under-test shall contain a text string with the maximum number of characters. If there is no maximum, then use the value 300.
40d characters used	Every ASCII character code that can appear in the string parameter shall appear in at least one instance of this element.
40e both types supported	At least 10 of the files in the CGM suite-under-test shall contain APPEND TEXT elements which use the less frequently used final/non-final flag.
40f some attribute modifications	There shall be at least one example of a APPEND TEXT (non-final)/APPEND TEXT (final) pair with an intervening text attribute modification for each text attribute modification used.
POLYGON	
41a sometimes used	At least 10, but not more than 90% of the files, in the CGM suite-under-test shall include this element.
41b min. number of points	At least one of the files in the CGM suite-under-test shall contain a POLYGON element with the number of coordinate points equal to the minimum number allowed.

ELEMENT Question	Answer	*Implications for Testing* and Requirements on CGM Suite-under-Test
41c	max. number of points	At least one of the files in the CGM suite-under-test shall contain a POLYGON element with the number of coordinate points equal to the maximum number allowed. If there is no maximum, then use 1200.
POLYGON SET		
42a	sometimes used	At least 10, but not more than 90% of the files, in the CGM suite-under-test shall include this element.
42b	min. number of points	At least one of the files in the CGM suite-under-test shall contain a POLYGON SET element with the number of coordinate points equal to the minimum number allowed.
42c	max. number of points	At least one of the files in the CGM suite-under-test shall contain a POLYGON SET element with the number of coordinate points equal to the maximum number allowed. If there is no maximum, then use 1200.
42d	only one edge-out flag used	No requirement.
	exactly two flag values used	At least 5 of the files in the CGM suite-under-test shall use the less frequently used edge-out flag value.
	three or four flag values used	Each edge out flag value used shall be represented in at least 3 of the files in the CGM suite-under-test.
CELL ARRAY		
43a	sometimes used	At least 10 of the files in the CGM suite-under-test shall include this element.
43b	min. number of columns	At least one of the instances of this element shall have the number of columns equal to the minimum value used.
43c	max. number of columns	At least one of the instances of this element shall have the number of columns equal to the maximum value used. If there is no maximum, use the value 1280.

ELEMENT Question	Answer	*Implications for Testing* and Requirements on CGM Suite-under-Test
43d	min. number of rows	At least one of the instances of this element shall have the number of rows equal to the minimum value used.
43e	max. number of rows	At least one of the instances of this element shall have the number of rows equal to the maximum value used. If there is no maximum, use the value 1024.
43f	local colour precisions	There shall be at least one example of a CELL ARRAY element for each possible local colour precision value used.
43g	both modes used	At least 5 of the instances of CELL ARRAY in the files in the CGM suite-under-test shall use the less frequently used cell representation mode value.
GENERALIZED DRAWING PRIMITIVE		
44a	sometimes used	At least 5 of the files in the CGM suite-under-test shall include this element.
44b	range of values for identifier	At least one instance of each allowed escape identifier shall appear in the files that comprise the CGM suite-under-test.
44c	min. length of data record	At least one instance of a GDP element which uses a data record of minimum length shall appear in the files that comprise the CGM suite-under-test.
44d	max. length of data record	At least one instance of a GDP element which uses a data record of maximum length shall appear in the files that comprise the CGM suite-under-test. If there is no maximum, then use the value 300.
RECTANGLE		
45a	sometimes used	At least 10, but not more than 90% of the files, in the CGM suite-under-test shall include this element.
CIRCLE		
46a	sometimes used	At least 10, but not more than 90% of the files, in the CGM suite-under-test shall include this element.

ELEMENT Question Answer	*Implications for Testing* and Requirements on CGM Suite-under-Test
CIRCULAR ARC 3 POINT	
47a sometimes used	At least 5, but not more than 90% of the files, in the CGM suite-under-test shall include this element.
CIRCULAR ARC 3 POINT CLOSE	
48a sometimes used	At least 5, but not more than 90% of the files, in the CGM suite-under-test shall include this element.
CIRCULAR ARC CENTRE	
49a sometimes used	At least 5, but not more than 90% of the files, in the CGM suite-under-test shall include this element.
CIRCULAR ARC CENTRE CLOSE	
50a sometimes used	At least 5, but not more than 90% of the files, in the CGM suite-under-test shall include this element.
ELLIPSE	
51a sometimes used	At least 10, but not more than 90% of the files, in the CGM suite-under-test shall include this element.
ELLIPTICAL ARC	
52a sometimes used	At least 5, but not more than 90% of the files, in the CGM suite-under-test shall include this element.
ELLIPTICAL ARC CLOSE	
53a sometimes used	At least 5, but not more than 90% of the files, in the CGM suite-under-test shall include this element.
LINE BUNDLE INDEX	
54a sometimes used	At least 5, but not more than 90% of the files, in the CGM suite-under-test shall include this element.

ELEMENT Question	Answer	*Implications for Testing* and Requirements on CGM Suite-under-Test
54b	range of indices	Minimum and maximum indices as well as at least one intermediate value shall be represented in the CGM suite-under-test.
LINE TYPE		
55a	sometimes used	At least 10, but not more than 90% of the files, in the CGM suite-under-test shall include this element.
55b	range of indices	Minimum and maximum indices as well as at least one intermediate value shall be represented in the CGM suite-under-test, including all standard values.
55c	private values	At least one example of each private value shall be represented in the CGM suite-under-test.
LINE WIDTH		
56a	sometimes used	At least 10, but not more than 90% of the files, in the CGM suite-under-test shall include this element.
LINE COLOUR		
57a	sometimes used	At least 10, but not more than 90% of the files, in the CGM suite-under-test shall include this element.
MARKER BUNDLE INDEX		
58a	sometimes used	At least 5, but not more than 90% of the files, in the CGM suite-under-test shall include this element.
58b	range of indices	Minimum and maximum indices as well as at least one intermediate value shall be represented in the CGM suite-under-test, including all standard values.
MARKER TYPE		
59a	sometimes used	At least 10, but not more than 90% of the files, in the CGM suite-under-test shall include this element.

ELEMENT Question Answer	*Implications for Testing* and Requirements on CGM Suite-under-Test
59b private values	At least one example of each private value shall be represented in the CGM suite-under-test.
MARKER SIZE	
60a sometimes used	At least 10, but not more than 90% of the files, in the CGM suite-under-test shall include this element.
MARKER COLOUR	
61a sometimes used	At least 10, but not more than 90% of the files, in the CGM suite-under-test shall include this element.
TEXT BUNDLE INDEX	
62a sometimes used	At least 5, but not more than 90% of the files, in the CGM suite-under-test shall include this element.
62b range of indices	Minimum and maximum indices as well as at least one intermediate value shall be represented in the CGM suite-under-test.
TEXT FONT INDEX	
63a sometimes used	At least 10, but not more than 90% of the files, in the CGM suite-under-test shall include this element.
63b range of indices	Minimum and maximum indices as well as at least one intermediate value shall be represented in the CGM suite-under-test.
TEXT PRECISION	
64a sometimes used	At least 5, but not more than 90% of the files, in the CGM suite-under-test shall include this element.
64b more than one precision	At least 3 of the files in the CGM suite-under-test shall contain TEXT elements which use the less frequently used text precisions.

ELEMENT Question Answer	*Implications for Testing* and Requirements on CGM Suite-under-Test
CHARACTER EXPANSION FACTOR	
65a sometimes used	At least 5, but not more than 90% of the files, in the CGM suite-under-test shall include this element.
65b range of reals	Minimum and maximum values as well as at least one intermediate value shall be represented in the CGM suite-under-test.
CHARACTER SPACING	
66a sometimes used	At least 5, but not more than 90% of the files, in the CGM suite-under-test shall include this element.
66b range of reals	Minimum and maximum values as well as at least one intermediate value shall be represented in the CGM suite-under-test.
TEXT COLOUR	
67a sometimes used	At least 10, but not more than 90% of the files, in the CGM suite-under-test shall include this element.
CHARACTER HEIGHT	
68a sometimes used	At least 10, but not more than 90% of the files, in the CGM suite-under-test shall include this element.
CHARACTER ORIENTATION	
69a sometimes used	At least 10, but not more than 90% of the files, in the CGM suite-under-test shall include this element.
TEXT PATH	
70a sometimes used	At least 5, but not more than 90% of the files, in the CGM suite-under-test shall include this element.
70b more than one value	At least one instance of each supported text path value shall be included in the CGM suite-under-test.

ELEMENT Question Answer	*Implications for Testing* and Requirements on CGM Suite-under-Test
TEXT ALIGNMENT	
71a sometimes used	At least 10, but not more than 90% of the files, in the CGM suite-under-test shall include this element.
71b more than one value	At least one instance of each supported horizontal alignment value shall be included in the CGM suite-under-test.
71c more than one value	At least one instance of each supported vertical alignment value shall be included in the CGM suite-under-test.
CHARACTER SET INDEX	
72a sometimes used	At least 2, but not more than 90% of the files, in the CGM suite-under-test shall include this element.
72b range of indices	Minimum and maximum indices as well as at least one intermediate value shall be represented in the CGM suite-under-test.
ALTERNATE CHARACTER SET INDEX	
73a sometimes used	At least 2, but not more than 90% of the files, in the CGM suite-under-test shall include this element.
73b range of indices	Minimum and maximum indices as well as at least one intermediate value shall be represented in the CGM suite-under-test.
FILL BUNDLE INDEX	
74a sometimes used	At least 5, but not more than 90% of the files, in the CGM suite-under-test shall include this element.
74b range of indices	Minimum and maximum indices as well as at least one intermediate value shall be represented in the CGM suite-under-test.

ELEMENT Question Answer	*Implications for Testing* and Requirements on CGM Suite-under-Test
INTERIOR STYLE	
75a sometimes used	At least 10, but not more than 90% of the files, in the CGM suite-under-test shall include this element.
75b more than one value	At least one instance of each supported interior style value shall be included in the CGM suite-under-test.
75c private values	At least one example of each private value shall be represented in the CGM suite-under-test.
FILL COLOUR	
76a sometimes used	At least 10, but not more than 90% of the files, in the CGM suite-under-test shall include this element.
HATCH INDEX	
77a sometimes used	At least 10, but not more than 90% of the files, in the CGM suite-under-test shall include this element.
77b range of indices	Minimum and maximum indices as well as at least one intermediate value shall be represented in the CGM suite-under-test.
77c private values	At least one example of each private value shall be represented in the CGM suite-under-test.
PATTERN INDEX	
78a sometimes used	At least 5, but not more than 90% of the files, in the CGM suite-under-test shall include this element.
78b range of indices	Minimum and maximum indices as well as at least one intermediate value shall be represented in the CGM suite-under-test.
EDGE BUNDLE INDEX	
79a sometimes used	At least 5, but not more than 90% of the files, in the CGM suite-under-test shall include this element.

ELEMENT Question Answer		*Implications for Testing* and Requirements on CGM Suite-under-Test
79b	range of indices	Minimum and maximum indices as well as at least one intermediate value shall be represented in the CGM suite-under-test.
EDGE TYPE		
80a	sometimes used	At least 5, but not more than 90% of the files, in the CGM suite-under-test shall include this element.
80b	range of indices	Minimum and maximum indices as well as at least one intermediate value shall be represented in the CGM suite-under-test, including all standard values.
80c	private values	At least one example of each private value shall be represented in the CGM suite-under-test.
EDGE WIDTH		
81a	sometimes used	At least 5, but not more than 90% of the files, in the CGM suite-under-test shall include this element.
EDGE COLOUR		
82a	sometimes used	At least 5, but not more than 90% of the files, in the CGM suite-under-test shall include this element.
EDGE VISIBILITY		
83a	sometimes used	At least 5, but not more than 90% of the files, in the CGM suite-under-test shall include this element.
83b	both values	At least 2 of the files in the CGM suite-under-test shall contain EDGE VISIBILITY elements which use the less frequently used on/off flag value.
FILL REFERENCE POINT		
84a	sometimes used	At least 2, but not more than 90% of the files, in the CGM suite-under-test shall include this element.

ELEMENT Question Answer	*Implications for Testing* and Requirements on CGM Suite-under-Test
PATTERN TABLE	
85a sometimes used	At least 2, but not more than 90% of the files, in the CGM suite-under-test shall include this element.
85b index value	At least three instances of this element shall be included in the CGM suite-under-test illustrating use of the minimum and maximum indices and one value in between.
85c number of columns	At least three instances of this element shall be included in the CGM suite-under-test illustrating use of the minimum and maximum number of columns and one value in between.
85d number of rows	At least three instances of this element shall be included in the CGM suite-under-test illustrating use of the minimum and maximum number of rows and one value in between.
85e local colour precisions	At least one instance of each value of local colour precision used shall be included in the CGM suite-under-test.
PATTERN SIZE	
86a sometimes used	At least 2, but not more than 90% of the files, in the CGM suite-under-test shall include this element.
COLOUR TABLE	
87a sometimes used	At least 10, but not more than 90% of the files, in the CGM suite-under-test shall include this element.
87b min. no. of entries	At least one instance of this element shall be included in the CGM suite-under-test illustrating the use of the minimum number of entries.
87c max. no. of entries	At least one instance of this element shall be included in the CGM suite-under-test illustrating the use of the maximum number of entries. If the maximum is unlimited, use the value 300.

ELEMENT Question	Answer	*Implications for Testing* and Requirements on CGM Suite-under-Test
87d	index value	At least three instances of this element shall be included in the CGM suite-under-test illustrating use of the minimum and maximum indices and one value in between.
ASPECT SOURCE FLAGS		
88a	sometimes used	At least 5, but not more than 90% of the files, in the CGM suite-under-test shall include this element.
88b	aspect source flags	At least one instance of each value of each aspect source flag used shall be included in the CGM suite-under-test.
ESCAPE		
89a	sometimes used	At least 5 of the files in the CGM suite-under-test shall include this element.
89b	range of values for identifier	At least one instance of each allowed escape identifier shall appear in the files that comprise the CGM suite-under-test.
89c	min. length of data record	At least one instance of an ESCAPE element which uses a data record of minimum length shall appear in the files that comprise the CGM suite-under-test.
89d	max. length of data record	At least one instance of an ESCAPE element which uses a data record of maximum length shall appear in the files that comprise the CGM suite-under-test. If there is no maximum, then use the value 300.
MESSAGE		
90a	sometimes used	At least 5 of the files in the CGM suite-under-test shall include this element.
90b	both values used	At least 2 of the files in the CGM suite-under-test containing the MESSAGE element shall use the less frequently used action-required flag value.
90c	min. length of string	At least one instance of a MESSAGE element which uses a message string of minimum length shall appear in the files that comprise the CGM suite-under-test.

ELEMENT Question	Answer	*Implications for Testing* and Requirements on CGM Suite-under-Test
90d	max. length of string	At least one instance of a MESSAGE element which uses a message string of maximum length shall appear in the files that comprise the CGM suite-under-test. If there is no maximum, then use the value 300.
APPLICATION DATA		
91a	sometimes used	At least 5 of the files in the CGM suite-under-test shall include this element.
91b	range of values for identifier	At least one instance of each allowed application data identifier shall appear in the files that comprise the CGM suite-under-test.
91c	min. length of data record	At least one instance of an APPLICATION DATA element which uses a data record of minimum length shall appear in the files that comprise the CGM suite-under-test.
91d	max. length of data record	At least one instance of an APPLICATION DATA element which uses a data record of maximum length shall appear in the files that comprise the CGM suite-under-test. If there is no maximum, then use the value 300.
PRIMITIVES AND ATTRIBUTES		
92	combinations	The CGM suite-under-test shall contain all frames of all Test Images (see Appendix E) that correspond to those combinations of primitives and attributes that are marked on the questionnaire.

APPENDIX E

ANNOTATED TEST IMAGES

E.1 Overview and Purpose

In this section are 18 figures that are intended to be used to produce a set of **test images** that have been designed to thoroughly check out many of the most likely combinations of graphical primitive CGM elements and graphical attribute CGM elements. Each figure in fact represents one or more **test images**. The exact content of each **test image** will be governed by two main factors:

(a) The amount of information that can fit on a single screen.

(b) The actual primitive-attribute combinations used by the CGM generator-under-test.

Each test image specified shall appear in the native suite and the corresponding CGM suite-under-test.

E.2 Terminology

Metafile: A metafile is a file that conforms to the requirements of FIPS 128. A metafile contains one or more pictures.

Picture: A picture is the visual representation of all the CGM elements contained between a BEGIN PICTURE BODY element and an END PICTURE element in a metafile. Pictures may be mapped one-to-one or many-to-one to metafiles.

Frame: A frame is all the visual information presented on a single screen or hardcopy. Each frame shall be mapped to a CGM picture within a metafile. There might be only one frame stored in each metafile.

Subframe: A subframe is the information contained within each box of the figures within this section. One or more subframes will comprise a frame.

Test Image: A **test image** consists of a tiling of one or more subframes that together exercise many of the legitimate primitive-attribute element combinations. The client has the freedom to decide how **test images** map to frames and how frames map to metafiles.

E.3 Creating the Test Images

In the following pages, instructions for creating **test images** are given. If any specific CGM primitive-attribute combination is not used by the **generator-under-test**, the client may omit the related subframe(s) from the **test image**. If any specific CGM element is not used at all by the **generator-under-test**, the whole **test image** may be omitted from the **native suite** and the counterpart **CGM suite-under-test**.

All subframes should use the CGM TEXT element to label the subframes showing exactly the attribute parameter values used for each object in the subframe.

The **test images** stipulated in this appendix can be used to satisfy some or all of the requirements specified in the test suite description.

E.4 The Annotated Subframes

E.4.1 POLYLINE

Figure E-1 shows the basic POLYLINE subframe. It is associated with the CGM POLYLINE element and the five standard LINE TYPE attributes (1 through 5).

To generate the test image, replicate the subframe by varying LINE WIDTH and LINE COLOUR. The line width and line colour parameters should take on their extreme values and at least one intermediate value.

E.4.2 DISJOINT POLYLINE

Figure E-2 shows the basic DISJOINT POLYLINE subframe. It is associated with the CGM DISJOINT POLYLINE element and the five standard LINE TYPE attributes (1 through 5).

To generate the test image, replicate the subframe by varying LINE WIDTH and LINE COLOUR. The line width and line colour parameters should take on their extreme values and at least one intermediate value.

E.4.3 CIRCULAR ARC 3 POINT

Figure E-3 shows the basic CIRCULAR ARC 3 POINT subframe. It is associated with the CGM CIRCULAR ARC 3 POINT element and the five standard LINE TYPE attributes (1 through 5).

To generate the test image, replicate the subframe by varying LINE WIDTH and LINE COLOUR. The line width and line colour parameters should take on their extreme values and at least one intermediate value.

E.4.4 CIRCULAR ARC CENTRE

Figure E-4 shows the basic CIRCULAR ARC CENTRE subframe. It is associated with the CGM CIRCULAR ARC CENTRE element and the five standard LINE TYPE attributes (1 through 5).

To generate the test image, replicate the subframe by varying LINE WIDTH and LINE COLOUR. The line width and line colour parameters should take on their extreme values and at least one intermediate value.

E.4.5 ELLIPTICAL ARC

Figure E-5 shows the basic ELLIPTICAL ARC subframe. It is associated with the CGM ELLIPTICAL ARC element and the five standard LINE TYPE attributes (1 through 5).

To generate the test image, replicate the subframe by varying LINE WIDTH and LINE COLOUR. The line width and line colour parameters should take on their extreme values and at least one intermediate value.

E.4.6 POLYMARKER

Figure E-6 shows the basic POLYMARKER subframe. It is associated with the CGM POLYMARKER element and the five standard MARKER TYPE attributes (1 through 5).

To generate the test image, replicate the subframe by varying MARKER SIZE and MARKER COLOUR. The marker size and marker colour parameters should take on their extreme values and at least one intermediate value.

Figure E-1. POLYLINE Subframe.

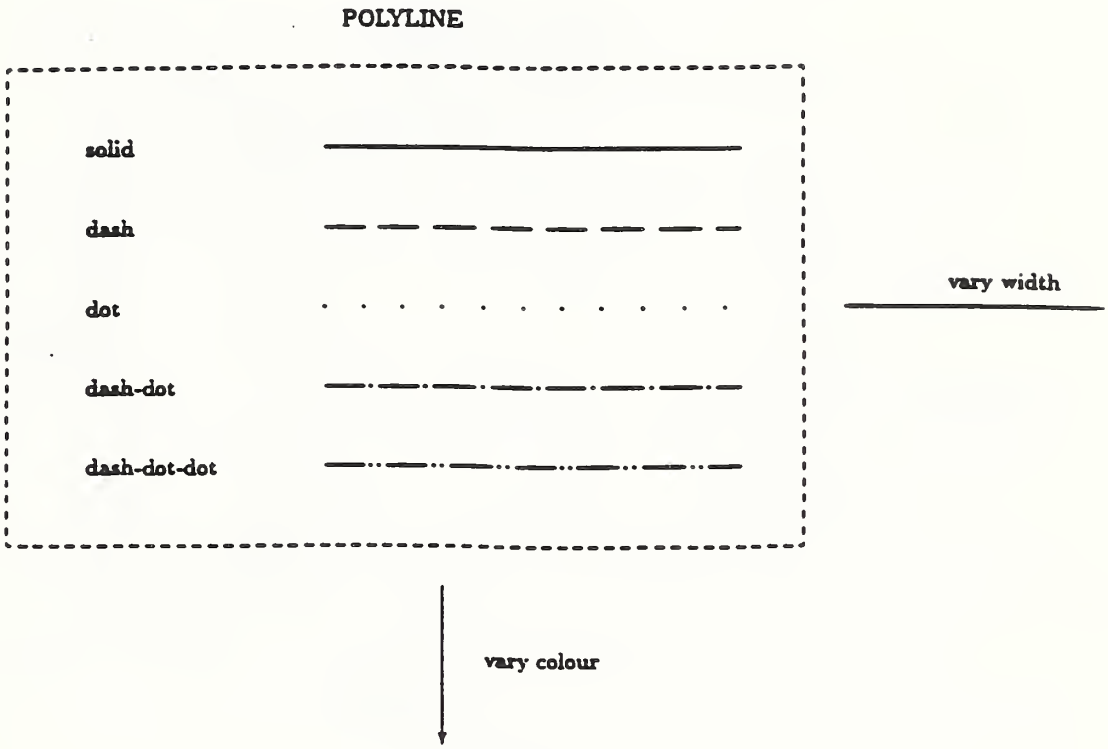


Figure E-2. DISJOINT POLYLINE Subframe.

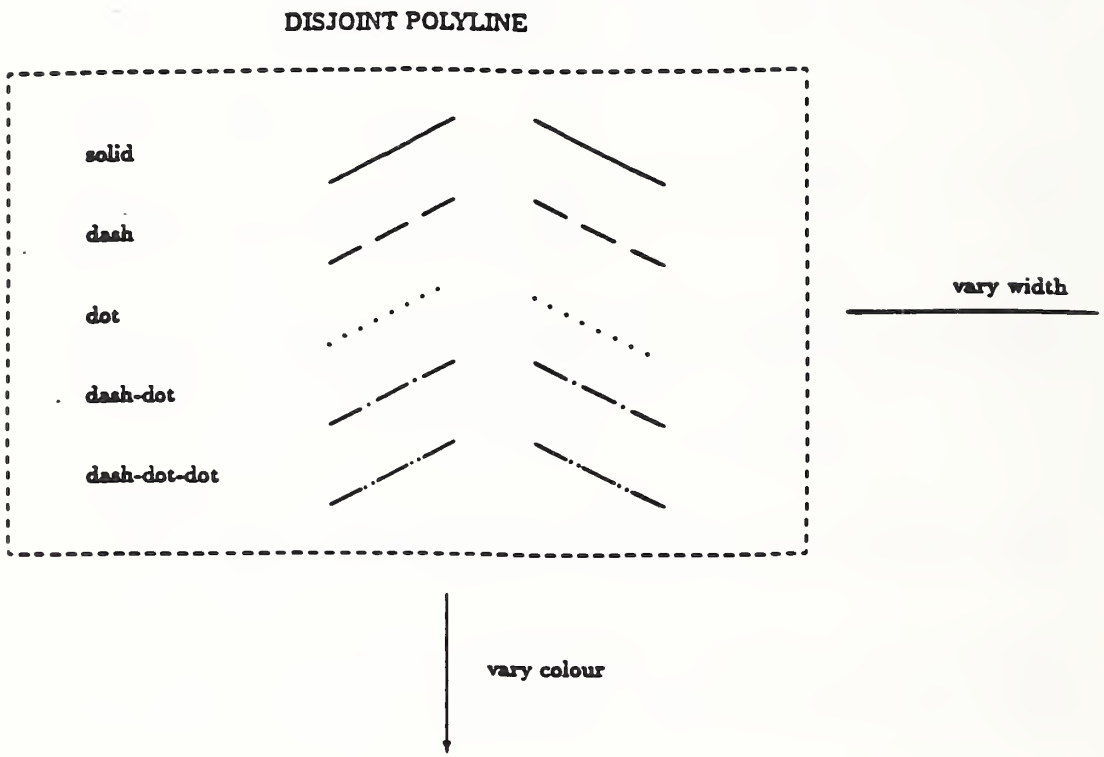


Figure E-3. CIRCULAR ARC 3 POINT Subframe.

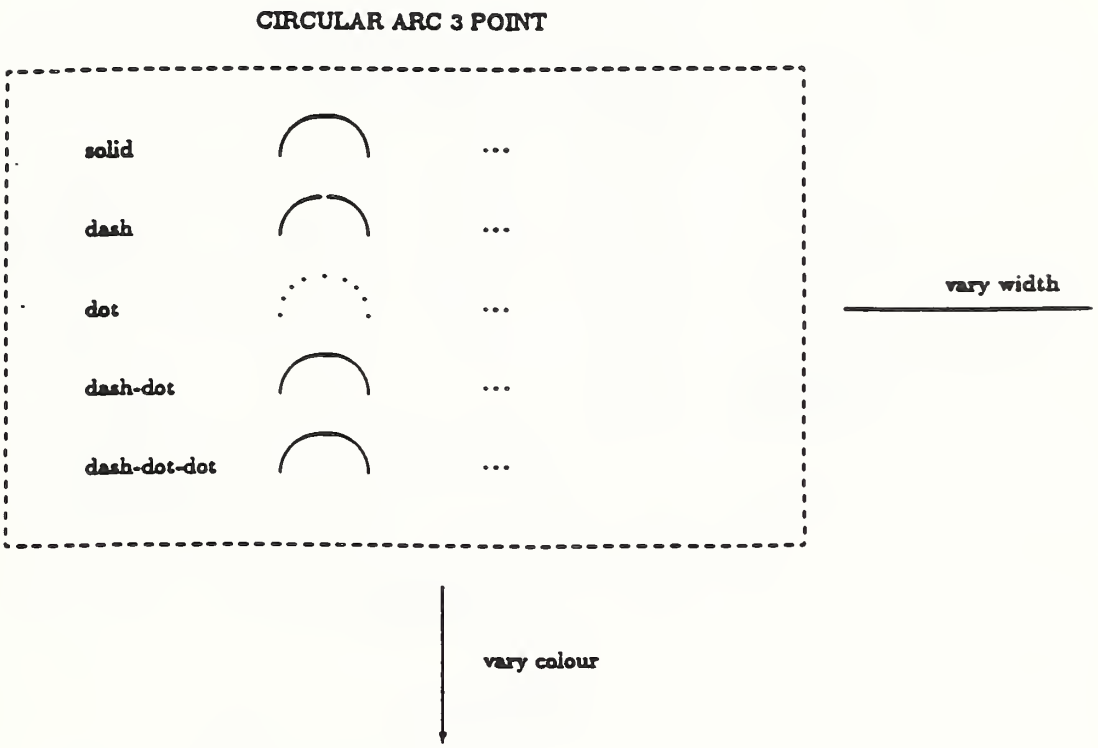


Figure E-4. CIRCULAR ARC CENTRE Subframe.

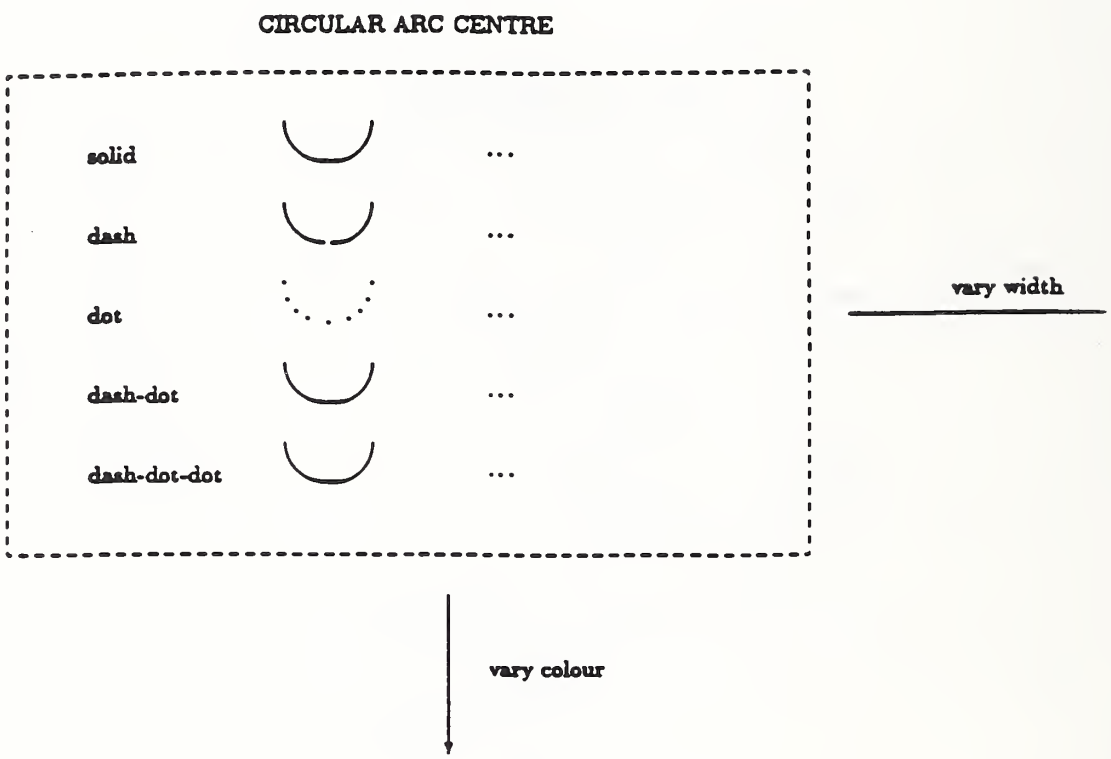


Figure E-5. ELLIPTICAL ARC Subframe.

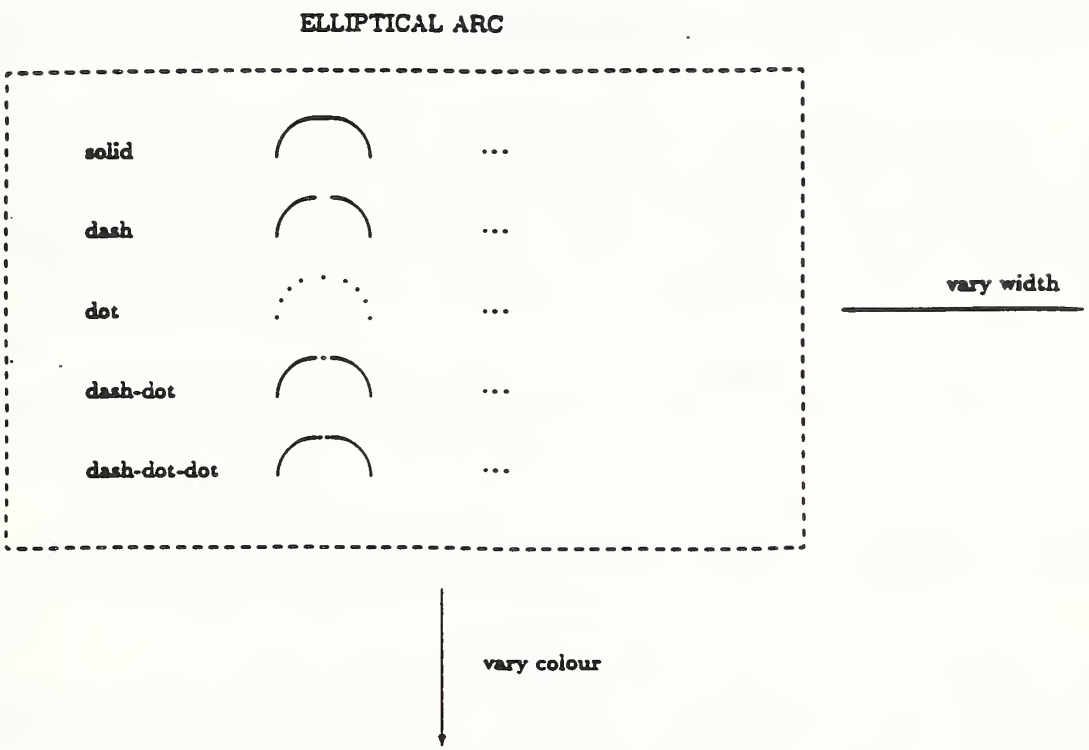
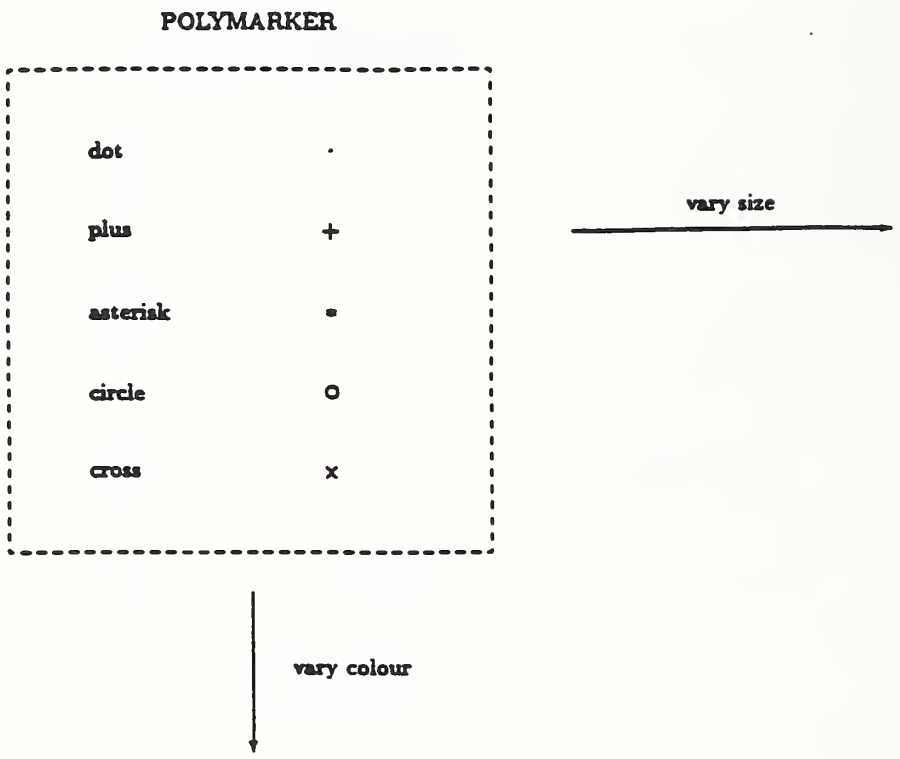


Figure E-6. DISJOINT POLYLINE Subframe.



E.4.7 TEXT (part 1)

Figure E-7 shows the first of four basic TEXT subframes. It is associated with the CGM TEXT element and TEXT PRECISION character.

To generate the first of the TEXT test images, replicate the subframe by varying CHARACTER ORIENTATION, TEXT PATH, and CHARACTER EXPANSION FACTOR. The values for path, orientation, and character expansion factor should take on at least the values illustrated in the subframe. Different TEXT COLOURs may be used with each different subframe.

A similar test image for RESTRICTED TEXT shall be generated in the same way as for this TEXT test image.

E.4.8 TEXT (part 2)

Figure E-8 shows the second of four basic TEXT subframes. It is associated with the CGM TEXT element and the text attributes TEXT FONT INDEX, TEXT HEIGHT, CHARACTER EXPANSION FACTOR, and CHARACTER SPACING.

To generate the second of the TEXT test images, replicate the subframe by varying TEXT HEIGHT, CHARACTER EXPANSION FACTOR, and CHARACTER SPACING. One frame should be generated for each font index used. The values for height, expansion factor, and spacing should take on their extreme values and at least one intermediate value. Different TEXT COLOURs should be used with each different subframe.

A similar test image for RESTRICTED TEXT shall be generated in the same way as for this TEXT test image.

E.4.9 TEXT (part 3)

Figure E-9 shows the third of four basic TEXT subframes. It is associated with the CGM TEXT element and various TEXT ALIGNMENT settings.

To generate the third of the TEXT test images, replicate the subframe by varying CHARACTER ORIENTATION, TEXT PATH, and CHARACTER EXPANSION FACTOR. The values for path, orientation, and character expansion factor should take on at least the values illustrated in the subframe. Different TEXT COLOURs may be used with each different subframe.

A similar test image for RESTRICTED TEXT shall be generated in the same way as for this TEXT test image.

E.4.10 TEXT (part 4)

Figure E-10 shows the fourth of four basic TEXT subframes. It is associated with the CGM TEXT (non-final) and CGM APPEND TEXT (final) element and various attribute settings that are changed between the TEXT and the APPEND TEXT element. Subframes are then varied in a pattern similar to that for figure E-7.

E.4.11 RECTANGLE

Figure E-11 shows the basic RECTANGLE subframe. It is associated with the CGM RECTANGLE element and the five standard INTERIOR STYLE values (hollow, solid, pattern, hatch, and empty).

To generate the test image, replicate the subframe by varying PATTERN INDEX, HATCH INDEX, EDGE TYPE and EDGE WIDTH. The hatch index and edge type parameters should take on all their standard values. The pattern index and edge width parameters should take on their extreme values and at least one intermediate value. Randomly vary FILL COLOUR and EDGE COLOUR to show the effects of these CGM attribute elements.

E.4.12 POLYGON

Figure E-12 shows the basic POLYGON subframe. It is associated with the CGM POLYGON element and the five standard INTERIOR STYLE values (hollow, solid, pattern, hatch, and empty).

To generate the test image, replicate the subframe by varying PATTERN INDEX, HATCH INDEX, EDGE TYPE and EDGE WIDTH. The hatch index and edge type parameters should take on all their standard values. The pattern index and edge width parameters should take on their extreme values and at least one intermediate value. Randomly vary FILL COLOUR, EDGE COLOUR, and EDGE VISIBILITY to show the effects of these CGM attribute elements.

E.4.13 POLYGON SET

Figure E-13 shows the basic POLYGON SET subframe. It is associated with the CGM POLYGON SET element and the five standard INTERIOR STYLE values (hollow, solid, pattern, hatch, and empty).
To generate the test image, replicate the subframe by varying PATTERN INDEX, HATCH INDEX, EDGE TYPE and EDGE WIDTH. The hatch index and edge type parameters should take on all their standard values. The pattern index and edge width parameters should take on their extreme values and at least one intermediate value. Randomly vary FILL COLOUR, EDGE COLOUR, and EDGE VISIBILITY and vary the edge-out flags to show the effects of these parameter settings and attribute elements.

Figure E-7. TEXT Subframe 1.

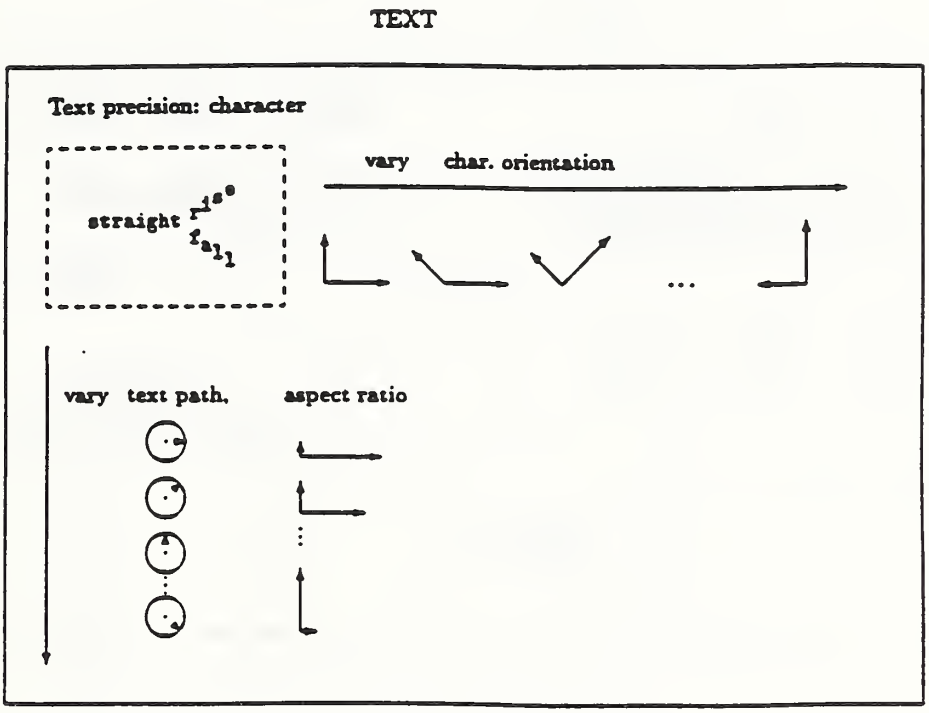


Figure E-8. TEXT Subframe 2.

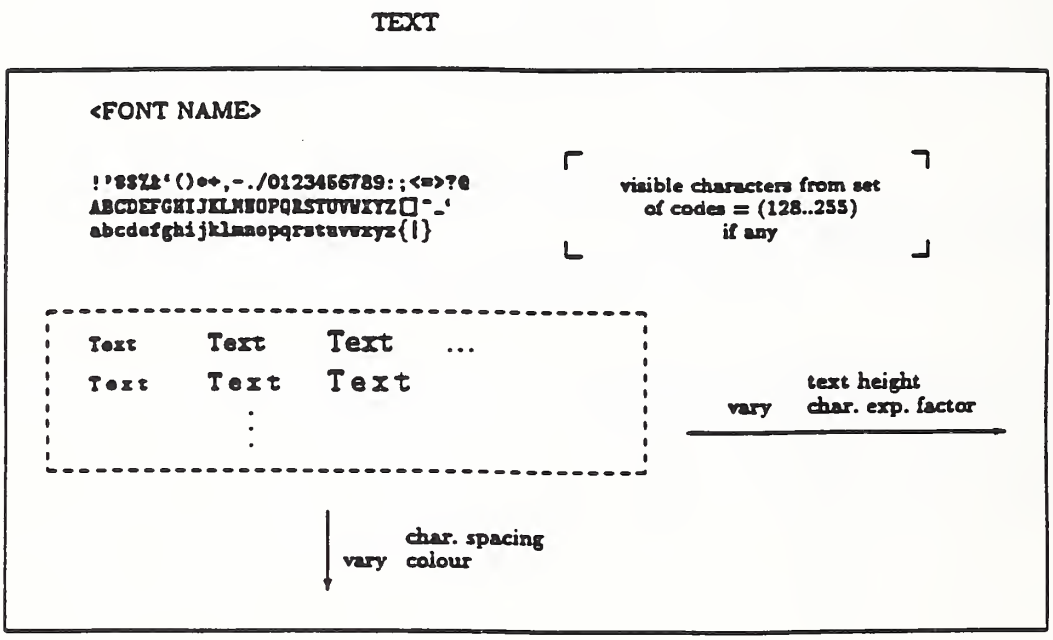


Figure E-9. TEXT Subframe 3.

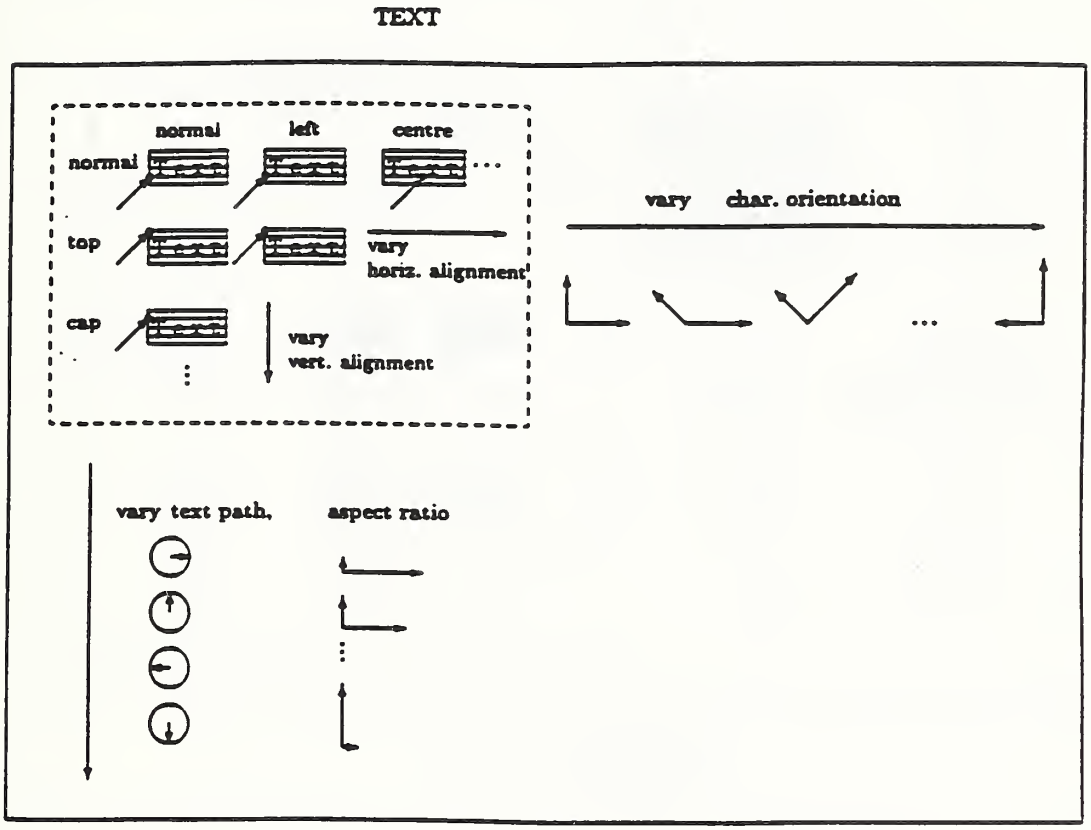


Figure E-10. TEXT Subframe 4.

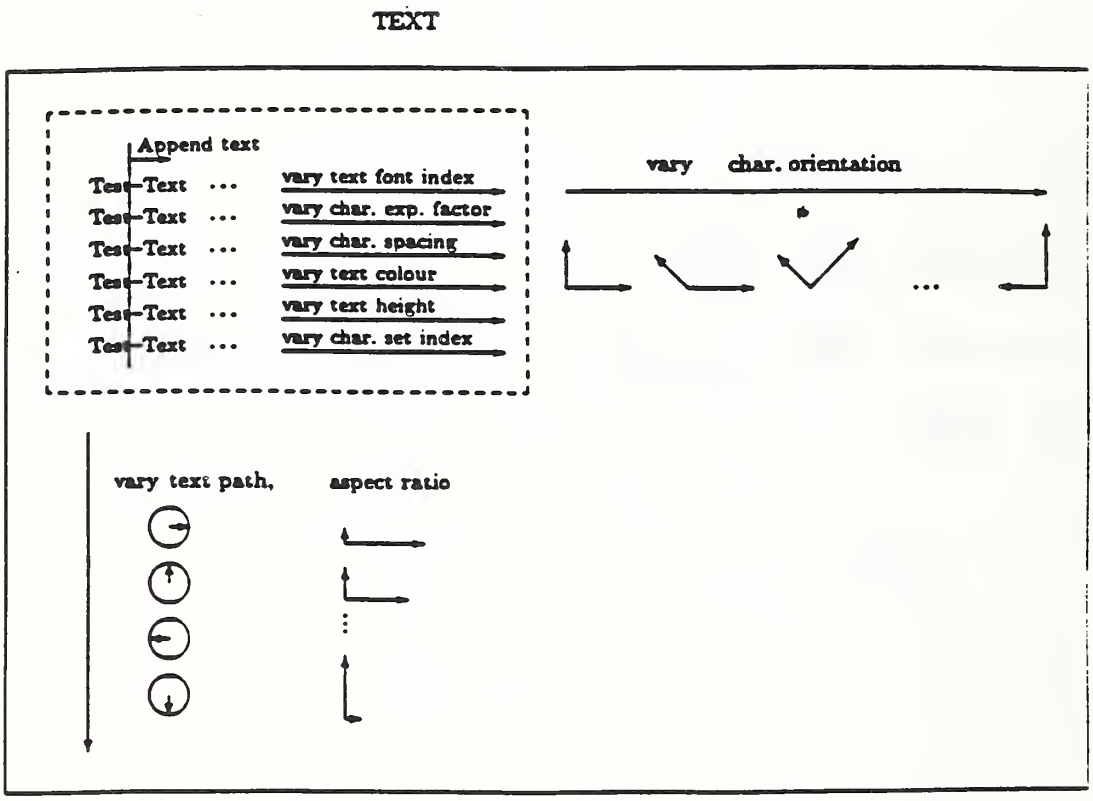


Figure E-11. RECTANGLE Subframe.

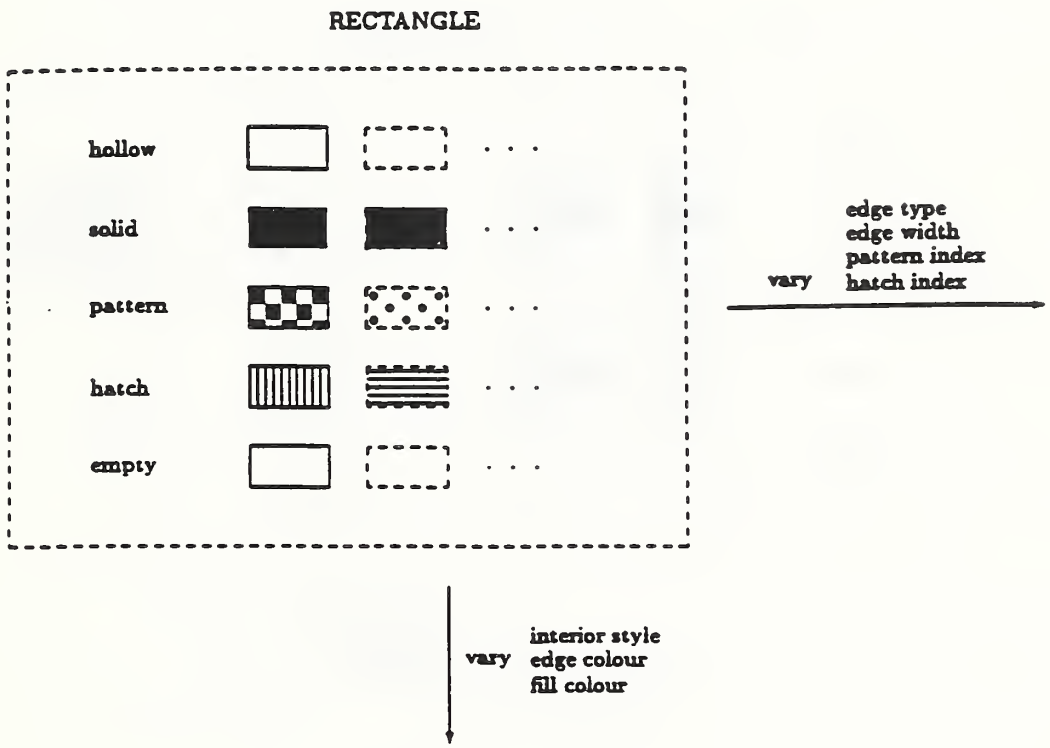


Figure E-12. POLYGON Subframe.

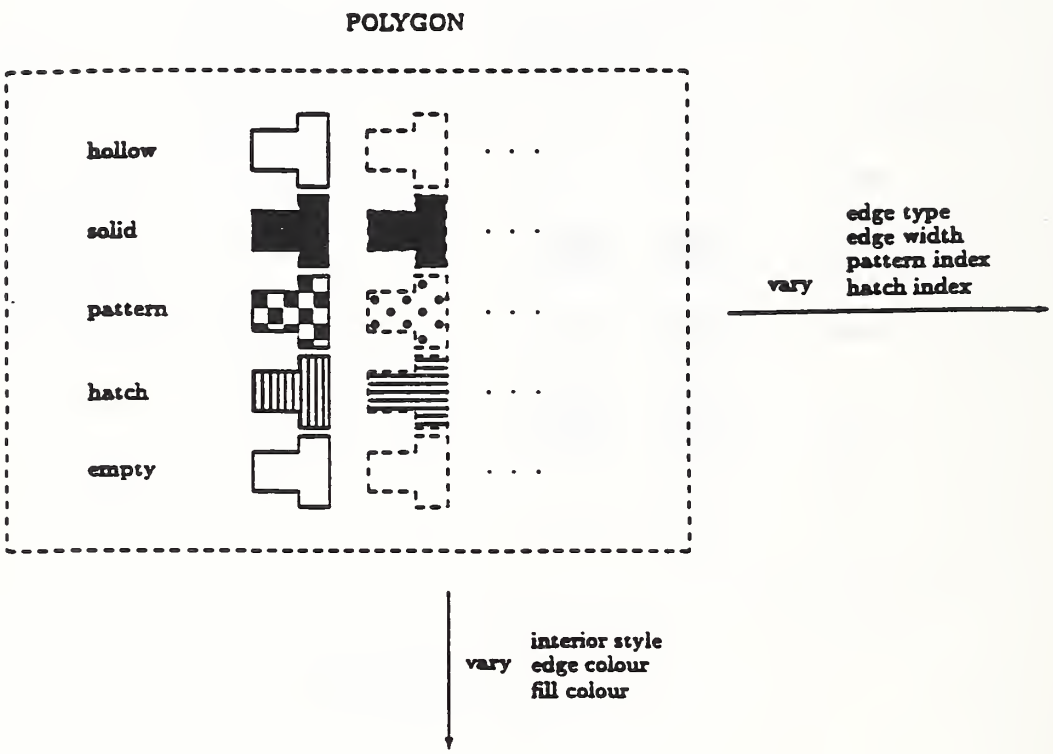
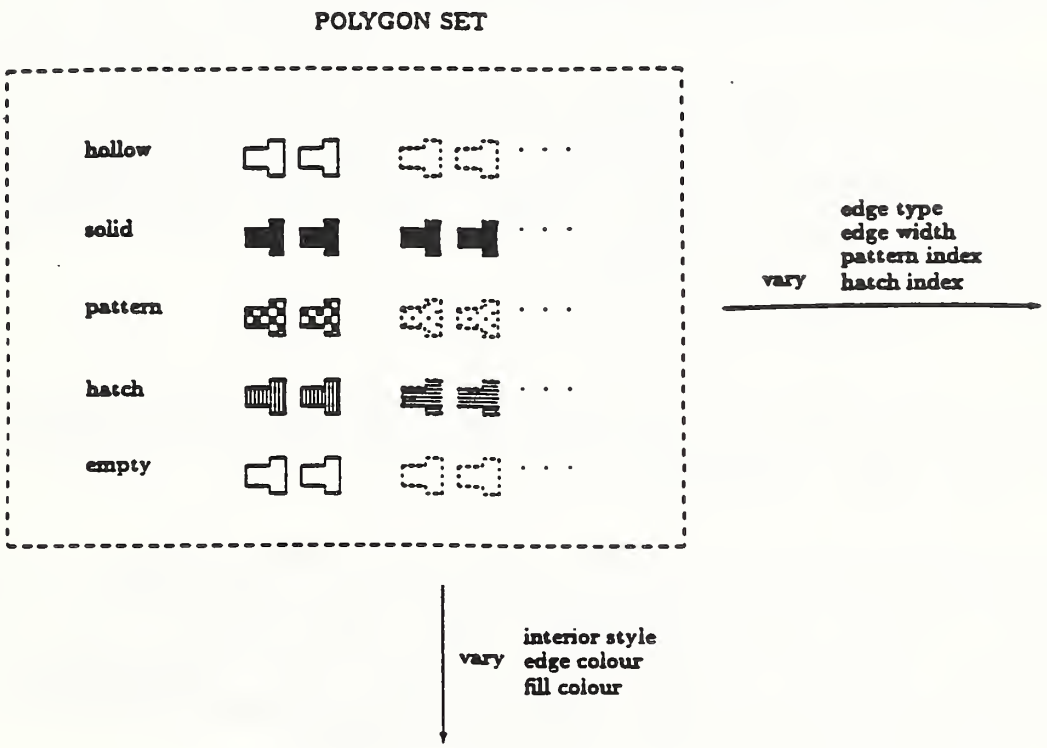


Figure E-13. POLYGON SET Subframe.



E.4.14 CIRCLE

Figure E-14 shows the basic CIRCLE subframe. It is associated with the CGM CIRCLE element and the five standard INTERIOR STYLE values (hollow, solid, pattern, hatch, and empty).

To generate the test image, replicate the subframe by varying PATTERN INDEX, HATCH INDEX, EDGE TYPE and EDGE WIDTH. The hatch index and edge type parameters should take on all their standard values. The pattern index and edge width parameters should take on their extreme values and at least one intermediate value. Randomly vary FILL COLOUR and EDGE COLOUR to show the effects of these CGM attribute elements.

E.4.15 CIRCULAR ARC CENTRE CLOSE

Figure E-15 shows the basic CIRCULAR ARC CENTRE CLOSE subframe. It is associated with the CGM CIRCULAR ARC CENTRE CLOSE element and the five standard INTERIOR STYLE values (hollow, solid, pattern, hatch, and empty).

To generate the test image, replicate the subframe by varying PATTERN INDEX, HATCH INDEX, EDGE TYPE and EDGE WIDTH. The hatch index and edge type parameters should take on all their standard values. The pattern index and edge width parameters should take on their extreme values and at least one intermediate value. Randomly vary FILL COLOUR and EDGE COLOUR to show the effects of these CGM attribute elements.

E.4.16 CIRCULAR ARC 3 POINT CLOSE

Figure E-16 shows the basic CIRCULAR ARC 3 POINT CLOSE subframe. It is associated with the CGM CIRCULAR ARC 3 POINT CLOSE element and the five standard INTERIOR STYLE values (hollow, solid, pattern, hatch, and empty).

To generate the test image, replicate the subframe by varying PATTERN INDEX, HATCH INDEX, EDGE TYPE and EDGE WIDTH. The hatch index and edge type parameters should take on all their standard values. The pattern index and edge width parameters should take on their extreme values and at least one intermediate value. Randomly vary FILL COLOUR, EDGE COLOUR, and EDGE VISIBILITY to show the effects of these CGM attribute elements.

E.4.17 ELLIPTICAL ARC CLOSE

Figure E-17 shows the basic ELLIPTICAL ARC CLOSE subframe. It is associated with the CGM ELLIPTICAL ARC CLOSE element and the five standard INTERIOR STYLE values (hollow, solid, pattern, hatch, and empty).

To generate the test image, replicate the subframe by varying PATTERN INDEX, HATCH INDEX, EDGE TYPE and EDGE WIDTH. The hatch index and edge type parameters should take on all their standard values. The pattern index and edge width parameters should take on their extreme values and at least one intermediate value. Randomly vary FILL COLOUR and EDGE COLOUR to show the effects of these CGM attribute elements.

E.4.18 ELLIPSE

Figure E-18 shows the basic ELLIPSE subframe. It is associated with the CGM ELLIPSE element and the five standard INTERIOR STYLE values (hollow, solid, pattern, hatch, and empty).

To generate the test image, replicate the subframe by varying PATTERN INDEX, HATCH INDEX, EDGE TYPE and EDGE WIDTH. The hatch index and edge type parameters should take on all their standard values. The pattern index and edge width parameters should take on their extreme values and at least one intermediate value. Randomly vary FILL COLOUR and EDGE COLOUR to show the effects of these CGM attribute elements.

Figure E-14. CIRCLE Subframe.

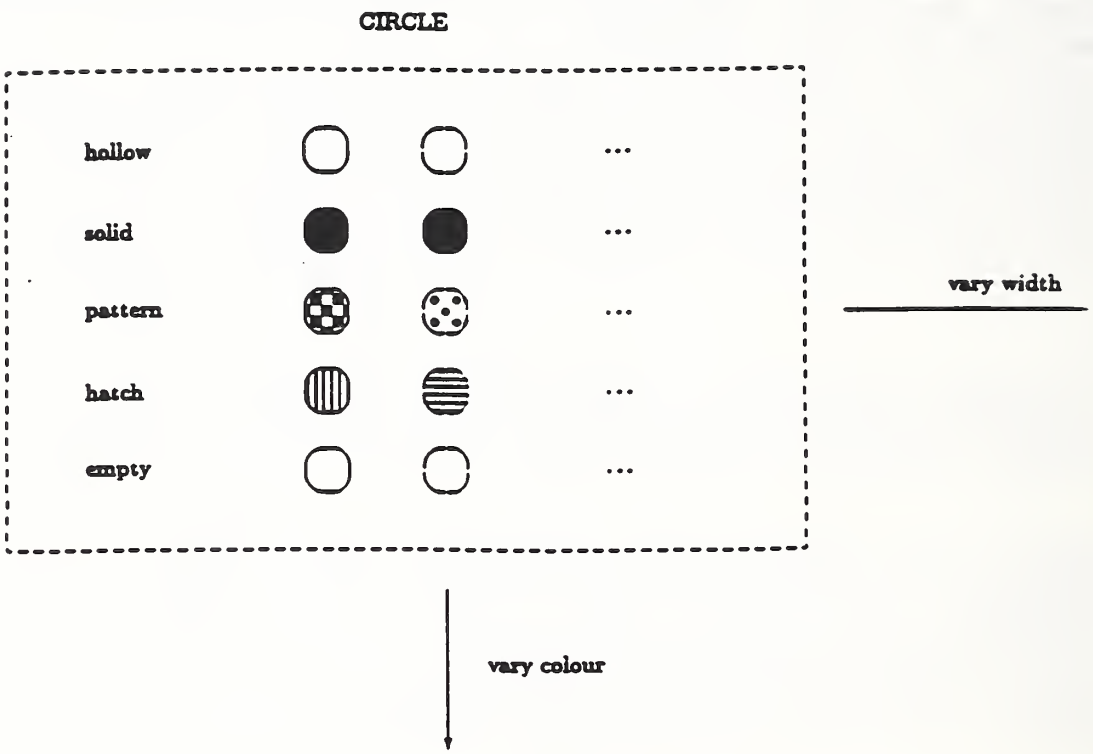


Figure E-15. CIRCULAR ARC CENTRE CLOSE Subframe.

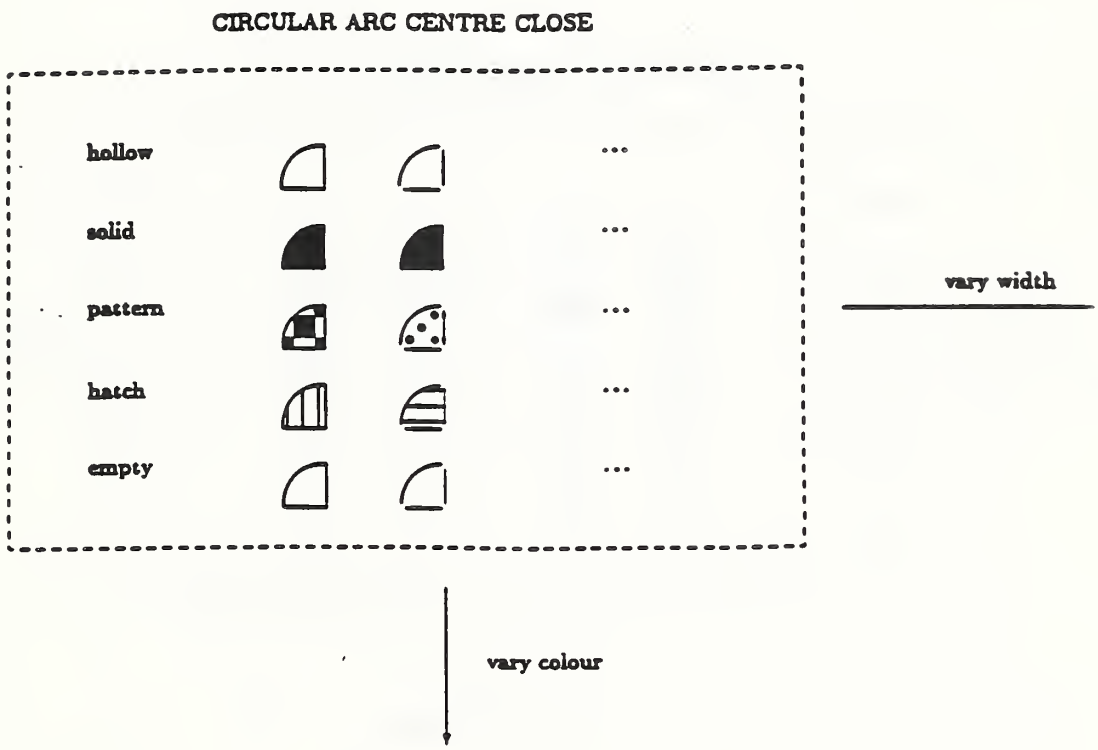


Figure E-16. CIRCULAR ARC 3 POINT CLOSE Subframe.

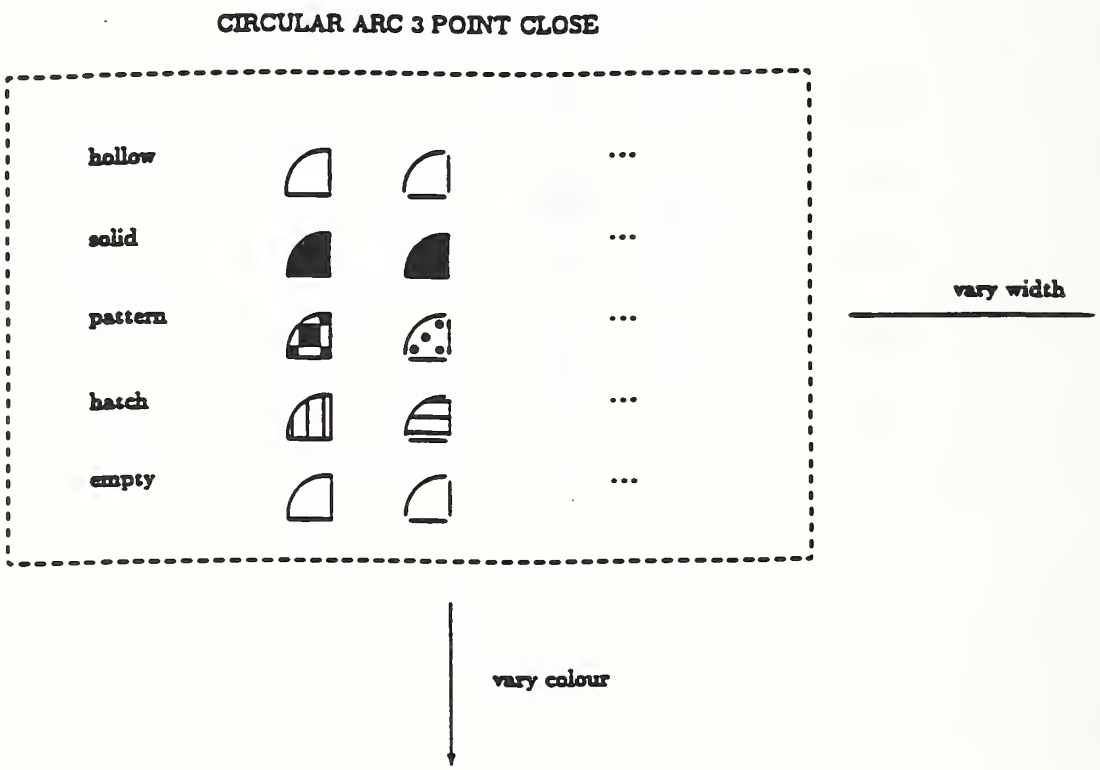


Figure E-17. ELLIPTICAL ARC CLOSE Subframe.

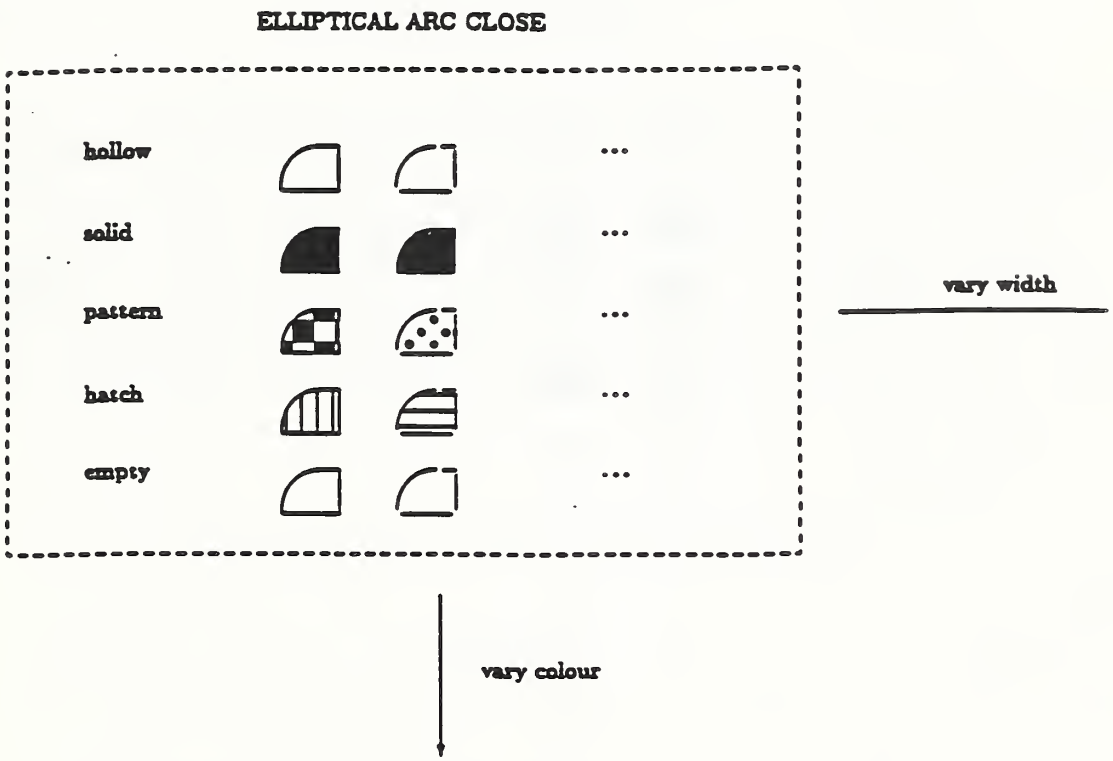
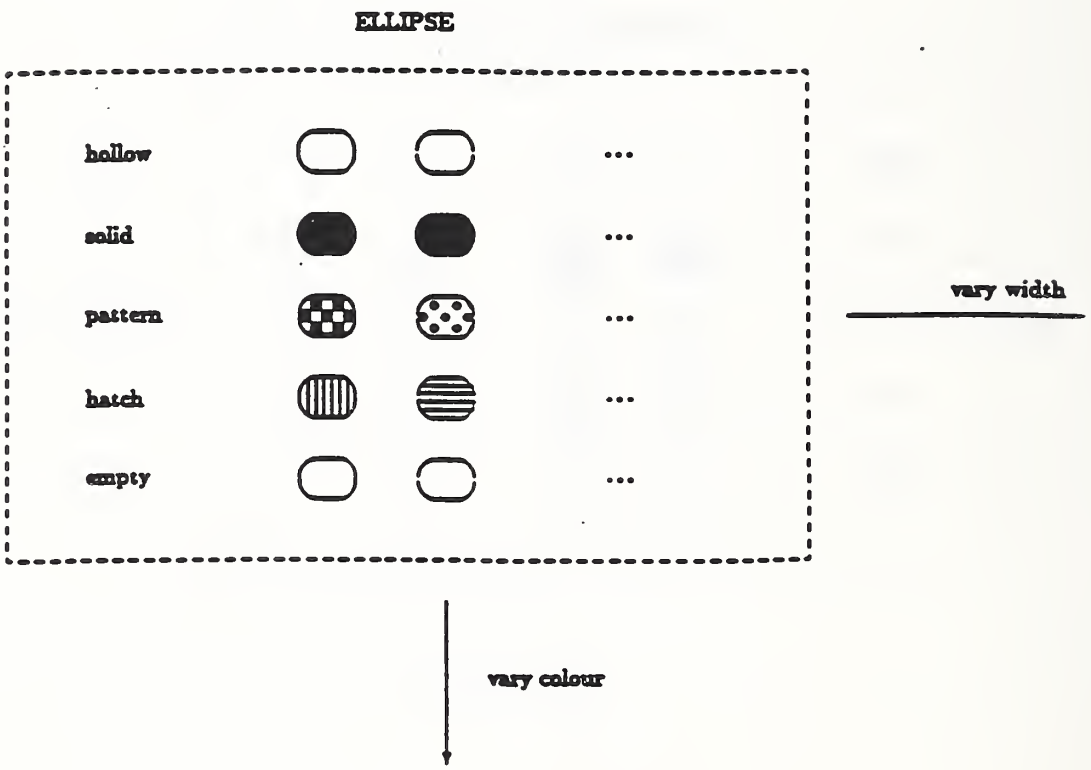


Figure E-18. ELLIPSE Subframe.



NIST-114A
(REV. 3-90)

U.S. DEPARTMENT OF COMMERCE
NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY

BIBLIOGRAPHIC DATA SHEET

1. PUBLICATION OR REPORT NUMBER
NISTIR 4806

2. PERFORMING ORGANIZATION REPORT NUMBER

3. PUBLICATION DATE
MARCH 1992

4. TITLE AND SUBTITLE
Procedures Manual for Testing CGM Generator Products That Claim Conformance to FIPS 128 and MIL-D-28003

5. AUTHOR(S)
Daniel R. Benigni, Editor

6. PERFORMING ORGANIZATION (IF JOINT OR OTHER THAN NIST, SEE INSTRUCTIONS)
U.S. DEPARTMENT OF COMMERCE
NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY
GAITHERSBURG, MD 20899

7. CONTRACT/GRANT NUMBER

8. TYPE OF REPORT AND PERIOD COVERED
1/90 through 12/92

9. SPONSORING ORGANIZATION NAME AND COMPLETE ADDRESS (STREET, CITY, STATE, ZIP)
Defense CALS Executive Office
Pentagon, Room 3D833
Washington, DC 20301-8000

10. SUPPLEMENTARY NOTES

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.)

A total Computer Graphics Metafile (CGM) conformance test suite that tests both the CGM standard (FIPS 128) and the CGM Application Profile for CALS (MIL-D-28003) must test three things: metafiles, generators, and interpreters. NIST has developed a test tool for testing metafiles, and a conformance testing service has begun. This report provides a procedures manual specifying the methodology and details for testing conformance of CGM generator products. The procedures enable a tester to verify that a CGM generator produces conforming metafiles which accurately and correctly define the intended picture.

12. KEY WORDS (6 TO 12 ENTRIES; ALPHABETICAL ORDER; CAPITALIZE ONLY PROPER NAMES; AND SEPARATE KEY WORDS BY SEMICOLONS)
application profile; CALS; CGM; CGM generator testing; Computer Graphics Metafile; conformance testing; Federal Information Processing Standard 128, Military Specification MIL-D-28003; metafiles; procedures manual.

13. AVAILABILITY

[X] UNLIMITED

[ ] FOR OFFICIAL DISTRIBUTION. DO NOT RELEASE TO NATIONAL TECHNICAL INFORMATION SERVICE (NTIS).

[ ] ORDER FROM SUPERINTENDENT OF DOCUMENTS, U.S. GOVERNMENT PRINTING OFFICE, WASHINGTON, DC 20402.

[X] ORDER FROM NATIONAL TECHNICAL INFORMATION SERVICE (NTIS), SPRINGFIELD, VA 22161.

14. NUMBER OF PRINTED PAGES
136

15. PRICE
A07

ELECTRONIC FORM