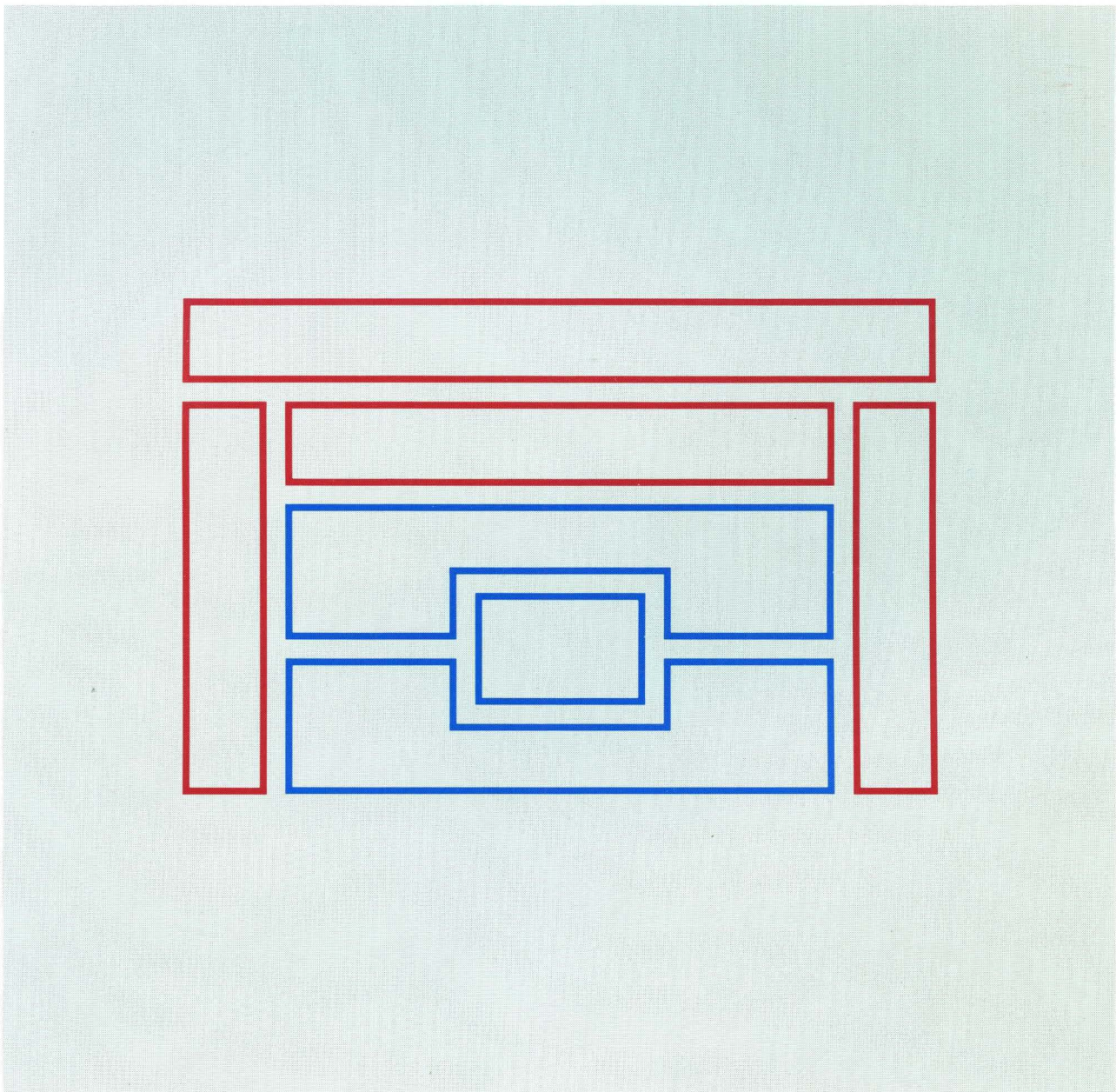
IBM

Systems Application Architecture

An Overview

GC26-4341-0



First Edition (May 1987)

This edition applies to the initial announcement of IBM's Systems Application Architecture.

Changes are made periodically to this publication; before using this publication in connection with the operation of IBM systems, consult the latest *IBM System/370, 30xx, and 4300 Processors Bibliography*, GC20-0001, *IBM System/36 Guide to Publications*, GC21-9015, or *IBM System/38 Guide to Publications*, GC21-7726, for the editions that are applicable and current.

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates.

Requests for IBM publications should be made to your IBM representative or to the IBM branch office serving your locality.

About This Manual

This manual is intended to help managers and technical personnel evaluate IBM's Systems Application Architecture, and do some preliminary, high-level planning for its implementation. More detailed information will be available in individual reference manuals.

Contents

Part One: Introduction

Part One: Introduction

Systems Application Architecture

Background

IBM offers systems based on several different hardware architectures and control programs. By pursuing a multiple-architecture strategy, IBM has been able to provide products with outstanding price/performance ratios across an ever-broadening spectrum of customer requirements. Today, IBM's systems span a nearly thousand-fold capacity range, and support the information processing needs of people in very different environments.

To make movement between these systems easier, to facilitate multi-system use, and to bring the breadth of IBM's product line to bear on customers' needs in all environments, IBM has introduced Systems Application Architecture. The results of Systems Application Architecture are intended to be:

- Programming skills that have broader applicability
- Applications that can be ported with less effort, or that can span systems
- User access to these applications that is simpler and more uniform.

What It Covers

Systems Application Architecture is a collection of selected software interfaces, conventions, and protocols that will be published in 1987. Systems Application Architecture will be the framework for development of consistent applications across the future offerings of the three major IBM computing environments:

- System/370 (TSO/E under MVS/XA, and CMS under VM)
- System/3X
- Personal Computer (Operating System/2 ™).[1]

These interfaces, conventions, and protocols are designed to provide an enhanced level of consistency in the following areas:

- Programming interface — the languages and services that application developers use in building their software
- User access — the use of screen panels, keyboard layouts, display options, and other interaction tools and techniques
- Communications support — the connectivity of systems and programs
- Applications — software built and supplied by IBM and other vendors.

[1] Operating System/2 is a trademark of the IBM Corporation.

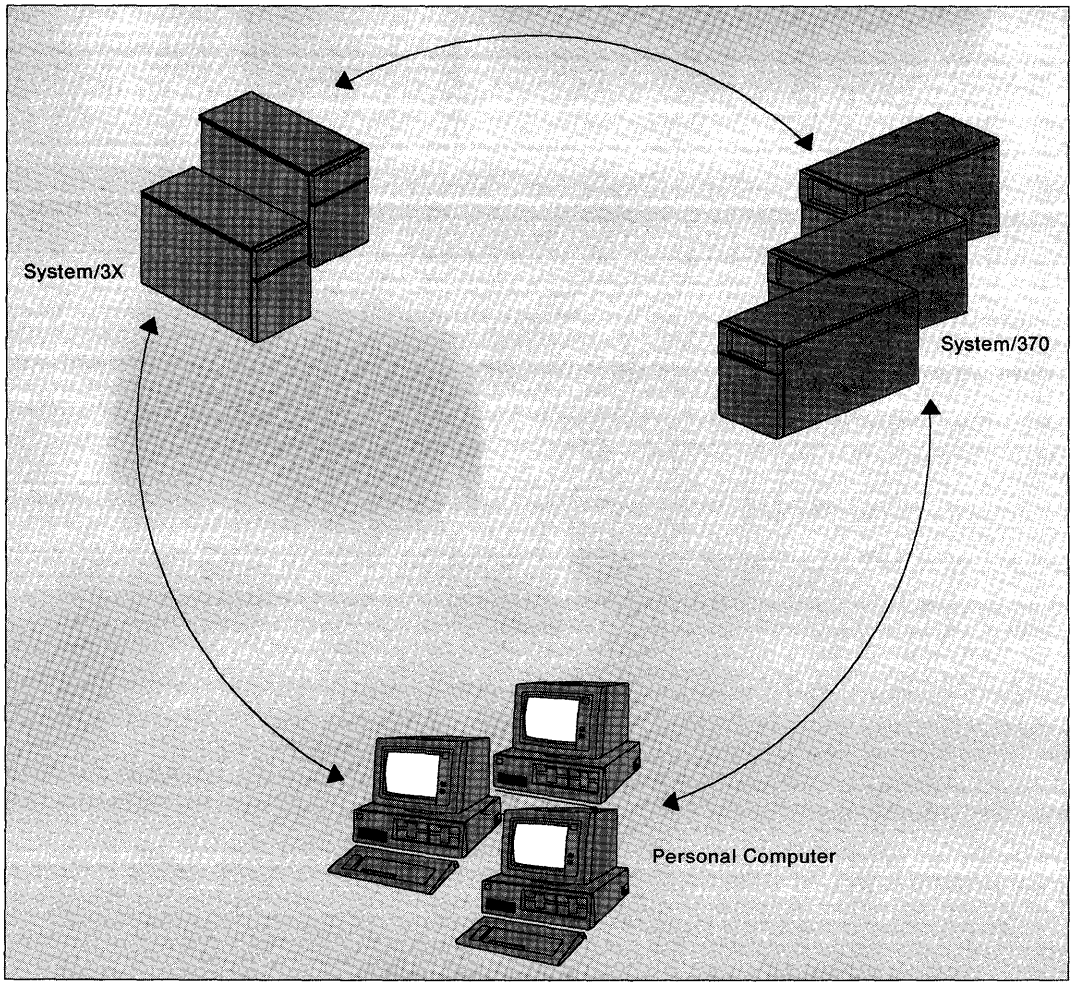


Figure 1. Large Enterprises Seek Multi-System Solutions

How It Gives Value

Systems Application Architecture will facilitate an increased level of consistency across the participating systems. As a result, the development and use of applications should be:

- Less expensive
- More timely.

Those users who need access to data on one system today and another system tomorrow can benefit. The programs they run are similar, and the actions they perform while running those programs are more uniform. Screens, keyboards, procedures — their appearance and behavior will often be the same. With less relearning, users have faster and easier access to data, and business efficiency increases.

Programmers, too, benefit from this increased consistency. Skills learned in one environment are transferrable to others, and there is less need for retraining or for system specialists. A programmer familiar with one environment can readily move to another and soon be productive.

Consistency means portability, and building applications for multi-system solutions becomes faster and easier. The source for a program built on one system can be taken to another system and implemented more smoothly. The effort expended in creating a general data processing solution is therefore lessened, along with development time.

Thus, there is much value in the growing consistency of the interface to the System/370, System/3X, and Personal Computer. The people who build the applications and the people who use the applications will all benefit.

In addition to these new benefits, in each of the Systems Application Architecture environments IBM plans to continue to support the execution of existing applications, thus conserving customers' current investments.

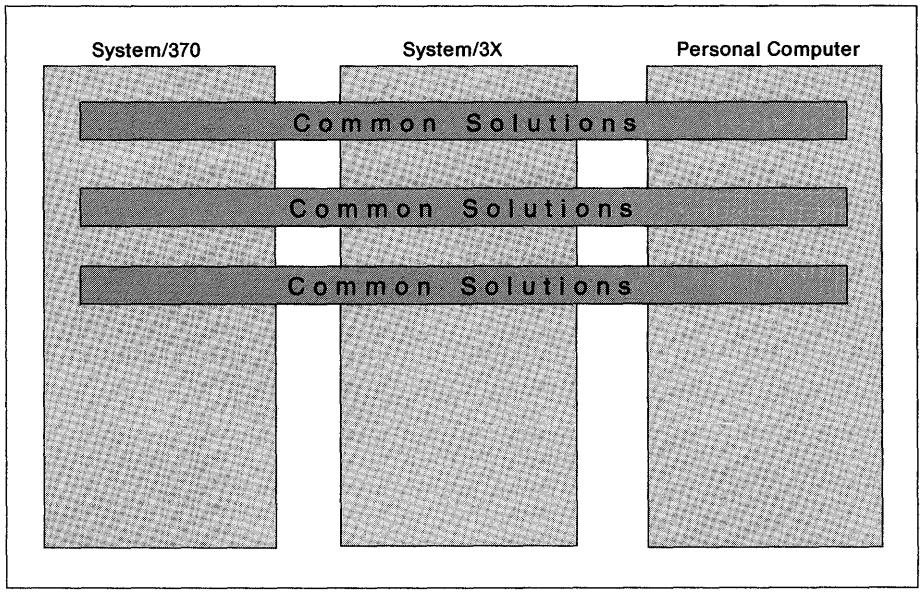


Figure 2. Consistency across Systems

Common Programming Interface

As has been noted, one important part of Systems Application Architecture is a common programming interface — the languages, commands, and calls that programmers employ. The components of the interface fall into two general categories, as follows:

- Languages

  Application Generator
  C
  COBOL
  FORTRAN
  Procedures Language

- Services

  Database Interface
  Dialog Interface
  Presentation Interface
  Query Interface

Specifications for Consistency

For each component of this programming interface, IBM is establishing a definition, or specification. A number of existing IBM products are already aligned with these specifications. More importantly, each specification will be used within IBM to promote and control software development and facilitate the consistency of all future products within the Systems Application Architecture framework. Those portions of the products controlled by the Common Programming Interface specifications are intended to have consistent implementation on the participating systems.

These IBM specifications have generally been developed with regard for established industry standards. Part Two of this manual (page 20) gives more detail on the content of each specification.

Advantages for Independent Software Vendors

Just as these specifications will be advantageous for the customers who use IBM software to build their own applications, they will benefit the software companies who build applications to sell. Using the common programming interfaces to IBM products, these companies can more easily design applications suitable for all the supported systems.

Customers seeking to reduce their data processing backlogs with prebuilt applications will also benefit, because the vendors will be able to produce applications more quickly and cost effectively.

Full Implementation, Plus

Over time, it is IBM's intent to supply products within Systems Application Architecture that will fully implement the interface specifications. In addition, some products may go beyond this to offer extensions that exploit features of the operating system or hardware on which they run.

When an application is being created for more than one system, its designers and coders can stay within the boundaries of the IBM specifications and obtain easier portability. If they want to create a program that takes advantage of one particular system, they can make use of that product's unique features. The choice is available.

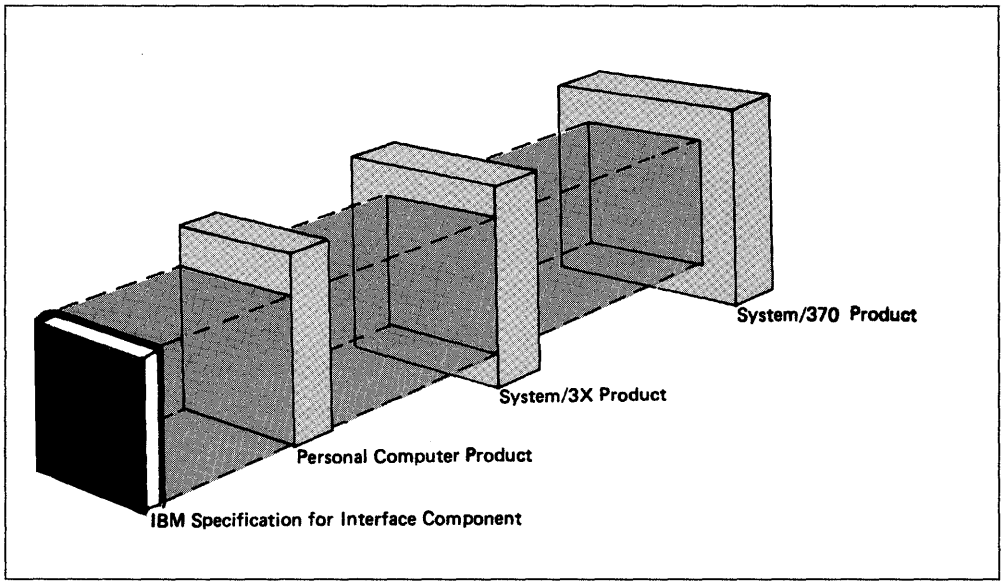


Figure 3. Products Will Meet or Exceed the Specification

Scope

The interface specifications cover many aspects of a programmer's interaction with selected IBM software. Other aspects will not immediately participate in this commonality. Things such as installation procedures, tuning considerations, and compile-time and run-time options are particular to individual products. Furthermore, other facets of the programming task, such as the use of job control language, will continue to be specific to operating systems and hardware, and remain outside the scope of the initial specifications.

Using the Programming Components Together

The components for which IBM specifications have been established offer a broad span of function that can meet most data processing needs.

Each component of the interface, as provided by applicable IBM products, is a valuable and efficient data processing aid in itself. Each is a tool that can be used on its own to get work done. In addition, many of these interface components can also be used together to create larger, more complex applications that accomplish sophisticated ends.

For example, an integrated application could be built from a combination of application generator and COBOL programs. Commands or queries could be incorporated into a FORTRAN program to obtain information from a data base. A program written in C could process data and then utilize the presentation interface to show results in a graphic format. Procedures language and the dialog interface could be used to tie together a multi-step application built of various program parts. The possibilities are many.

Because these components span the spectrum of common application needs, they offer the assistance customers require — whether to do a programming job using a single component, or to build a more complex data processing solution using multiple components of the commmon programming interface. And on top of this, of course, they offer all the value of cross-system consistency.

Common User Access

The Common User Access defines the specifications for the dialog between the human and the computer. It establishes how information appears on a display screen, and how people respond to that information. It includes definitions of interface elements and rules for interaction techniques such as panels, procedures for moving from one panel to another, choice selection, color and emphasis, messages, help, and terminology.

The Common User Access consolidates the latest technology in interface elements and techniques. Based on a foundation of fundamental concepts, and designed from the top down, its principles will apply across the range of Systems Application Architecture systems.

The Common User Access provides several advantages. Because it incorporates superior technology, it is by definition easier to learn and use. And because it is consistent across the participating systems, it has the benefits of familiarity. Users moving from application to application or from system to system need less time to adapt. And the programmers who create and maintain these applications should also be able to do so more quickly and easily — both in building the user interface portions of the applications, and in using the computer themselves.

Elements of the Interface

An interface between user and computer has three main components:

- The way the machine communicates with the user
- The way the user communicates with the machine
- What the user understands about the interface.

The first aspect is what the user perceives, what face the program and the hardware show to the person at the workstation, how the instructions and data are presented. The user has to recognize this information, understand it, and come up with an appropriate response. This response, consisting of established actions such as key selection or mouse movement, is the second aspect of the interface.

The third aspect, how users understand this entire process, is really part of the first two. Users' perceptions and expectations — their "conceptual models" — are influenced by their previous experience. To the extent that the interface is aligned with these expectations, it will be easy to use. To the extent that the interface is integrated and has a good overall design, it will be easy to learn.

The specifications being established for the Common User Access are intended to accomplish both these ends. They set forth a consistent set of concepts that are readily acquired and that provide continuity from application to application, system to system.

An interface can be consistent with respect to three broad categories or dimensions: physical, syntactical, and semantical.

- **Physical consistency** refers to the hardware: the keyboard layout, the location of keys, and the use of the mouse. For example, it would be physically consistent for the function keys to be in the same place on the keyboard regardless of the system being used. Likewise, it would be physically consistent for button 1 on a mouse to always select an item.

- **Syntactical consistency** refers to the sequence and order of appearance of elements on the display screen (the presentation language) and the sequence of keystrokes to request actions (the actions language). For example, it would be syntactically consistent to always center the panel title at the top of the panel.
- **Semantic consistency** refers to the meaning of the elements that make up the interface. For example, it would be semantically consistent for the command SAVE to have the same meaning (what is saved and what happens next) on all systems.

An interface for user access is dependent on device capabilities. An interface designed for an intelligent workstation will not have the same dynamic features as one for a less-sophisticated mainframe terminal. Thus, the design of a common user interface requires a balance between the desire for consistency and the desire for full exploitation of technological capabilities.

Flexibility

Although the Common User Access lays down rules for interface elements and interactions, it gives application designers a fair degree of flexibility. The Common User Access defines a number of types of panels and acknowledges that designers might require other, application-specific panel types not defined by the Common User Access. The Common User Access recommends that programmers try to use the defined panels but, if they can't, to use the common components of defined panels.

Evolution

The Common User Access is still evolving. It is being created for intelligent workstations and will grow through the midrange systems to the mainframe systems. The workstation was chosen as the starting point because it provides the greatest dynamic capabilities for the interface.

When the initial level of the Common User Access has been completely defined, those IBM specifications will be published and made generally available, and writing applications that follow these specifications will be encouraged.

In the future, the Common User Access will define added elements and interaction techniques as technologies evolve.

Summary of Benefits

A defined, common user interface benefits the user and the programmer. For both, the benefits are the same: saving time and money.

Users should benefit because they will need less time to learn how to use an application and, when using the application, take less time doing their work. A consistent interface also reduces users' frustration levels and makes them feel more comfortable.

A consistent user interface also benefits application designers and programmers. The Common User Access defines building blocks for an interface through specified interface elements and interaction techniques. These building blocks allow programmers to create and change applications more quickly and easily.

Common Communications Support

Common Communications Support is used to connect applications, systems, networks, and devices. This will be achieved by the consistent implementation of designated communication architectures in each of the Systems Application Architecture environments. These communication architectures are the building blocks for distributed function to be detailed in future announcements of the Common Programming Interface and IBM-provided Systems Application Architecture applications.

The architectures selected to date have been chosen from Systems Network Architecture (SNA) and international standards. Each was also included in the Open Communication Architectures announcement of September 16, 1986 (Announcement Letter 286-410), thus reaffirming IBM's commitment to openness.

Types of Support

Included in Common Communications Support at this time are data streams, application services, sessions services, network, and data link controls. The aspects of each of these are described below.

Data Streams

The **3270 Data Stream** consists of user-provided data and commands, as well as control information that governs the way data is handled and formatted by IBM displays and printers. The Systems Application Architecture computing environments will all support the 3270 Data Stream. In addition, the System/3X family will continue to support the 5250 Data Stream. The 3270 Data Stream is documented in *IBM 3270 Information Display System Data Stream Programmer's Reference*, GA23-0059.

Document Content Architecture (DCA) defines the rules for specifying the form and meaning of a text document. It provides for uniform interchange of textual information in the office environment and consists of format elements optimized for document revision. This is documented in *Document Content Architecture: Revisable-Form-Text Reference*, SC23-0758.

Intelligent Printer Data Stream (IPDS) is the high-function data stream intended for use with all-points-addressable page printers. Documentation of this data stream will be available shortly.

Application Services

SNA Distribution Services (SNADS) provides an asynchronous distribution capability in an SNA network, thereby avoiding the need for active sessions between the end points. SNADS is documented in *SNA Architecture Format and Protocol Reference Manual: Distribution Services*, SC30-3098.

Document Interchange Architecture (DIA) provides a set of protocols that define several common office functions performed cooperatively by IBM products. This is documented in *Document Interchange Architecture: Technical Reference*, SC23-0781.

SNA Network Management Architecture describes IBM's approach to managing communication networks. The protocols of problem management offer a vehicle

for monitoring network operations from a central location. This is documented in *SNA Format and Protocol Reference Manual Management Services*, SC30-3346.

Session Services

LU Type 6.2 is a program-to-program communication protocol. It defines a rich set of interprogram communication services, including a base and optional supplementary services. Support of the base is included in all IBM LU 6.2 products that expose an LU 6.2 application programming interface. This facilitates compatibility of communication functions across systems. LU 6.2 is documented in *SNA Format and Protocol Reference Manual: Architecture Logic for LU Type 6.2*, SC30-3269.

Network

Low-Entry Networking Nodes (Type 2.1 Nodes) support peer-to-peer communication. Type 2.1 nodes can be either programmable or fixed function systems. SNA Low-Entry Networking allows, through a common set of protocols, multiple and parallel SNA sessions to be established between Type 2.1 nodes that are directly attached to each other. Low-Entry Networking is documented in *SNA Format and Protocol Reference Manual: Architecture Logic for Type 2.1 Nodes*, SC30-3422.

Data Link Controls

Synchronous Data Link Control (SDLC) is a discipline for managing synchronous, code-transparent, serial-by-bit information transfer between nodes that are joined by telecommunication links. This is documented in *IBM Synchronous Data Link Control Concepts*, GA27-3093.

Token-Ring Network consists of a wiring system, a set of communication adapters (stations), and an access protocol that controls the sharing of the physical medium by the stations attached to the LAN. The IBM Token-Ring Architecture is based on the IEEE 802.2 and 802.5 standards. This is documented in *Token-Ring Network Architecture Reference* (part number 6165877).

X.25 defines a packet-mode interface for attaching data terminal equipment (DTE) such as host computers, communication controllers, and terminals to packet-switched data networks. An IBM-defined external specification, *The X.25 Interface for Attaching SNA Nodes to Packet-Switched Data Networks: General Information Manual* (GA27-3345), and the 1984 version of this interface (GA27-3761), describe the elements of CCITT X.25 that are applicable to IBM SNA products that can attach to X.25 networks. X.25 is shown here as a data link control to reflect its implementation within SNA. In the OSI Reference Model, X.25 is one of the options available for the physical, data link, and network layers.

Current Product Base

Most of the Common Communications Support architectures are already supported in the Systems Application Architecture environments. In MVS and VM, support is available through the following product sets:

Print Services Facility (PSF)
Graphical Data Display Manager (GDDM)
Netview
Virtual Telecommunications Access Method (ACF/VTAM)
Network Control Program (ACF/NCP)

In IBM Operating System/2, the architectures are supported primarily in the Communications Manager portion of the Enhanced Edition 1.1, and in the applications that use the Communications Manager services.

Future offerings of the System/3X will support the Common Communications Support protocols and interfaces. In the current System/36 and System/38 offerings, support is available throught the following product sets:

DisplayWrite/36
Personal Services/36
System/36 Office Management System
System/36 System Support Program and Communications Features
Personal Services/38
System/38 Control Program Facility and Communications Management Functions

Through communications networking and, in particular, through the Common Communications Support elements, existing applications developed for the current versions of the major IBM computing environments will be able to interact with applications, functions, and data in the Systems Application Architecture environments. This conserves current application investment in existing systems.

Expansion

Extensions to the above products and new Systems Application Architecture announcements for Common Applications will complete the product support for these Common Communications Support architectures in these environments.

As IBM expands the Systems Application Architecture, additional communications architectures will be evaluated for inclusion in Common Communications Support.

Common Applications

It is IBM's intent to develop applications that conform to Systems Application Architecture, making use of the common user access, the common programming interface, and the common communications support. These will be key applications that satisfy a significant customer need for use across the three Systems Application Architecture environments. Availability of these applications will help customers solve today's business problems in the most efficient and effective way.

Benefits

These IBM-written applications will obtain the same benefits from Systems Application Architecture as those applications written by customers and independent software vendors. They will offer consistency in how functions are implemented, how panels are laid out, and how the user navigates within the application in all the supported environments. Thus, an application initially developed for one environment and subsequently ported to another will appear consistent to the application user. This consistency will also apply to integrated "families" of applications which will be offered.

Users of MFI (Mainframe Interactive) workstations and IWS (Intelligent Workstations) could have a consistent set of defined functions, and where applicable, the IWS user may have additional defined functions available for the application or family of applications. Hosts controlling the MFI station will provide the user access functions. These functions will, within the limits of technology, have a reasonable consistency with the IWS's user access.

Initial Focus

Initially, the IBM application development effort will focus on integrated office and decision support. Later, it will expand into industry-specific applications.

In the initial focus on office applications, the elements being defined include:

- Document processing
- Document library
- Personal services, mail
- Decision support.

Current Product Base

Many of IBM's current office and decision support products already use Document Content Architecture (DCA), Document Interchange Architecture (DIA), and SNA Distribution Services (SNADS). These key architectures are part of Systems Application Architecture, and will conserve the investments that customers have made and offer a transition path to new Systems Application Architecture product offerings.

Continued Support in Other Areas

While this new emphasis will bring additional value to those interested in generalized business solutions, the familiar types of support will continue to be provided and enhanced.

- Continuity will be maintained with existing products and systems. This means that current investments in applications and knowledge will retain their value.
- Many other IBM systems and products will be offered outside the Systems Application Architecture, ensuring that customers with environment-specific business requirements will continue to have those needs satisfied.

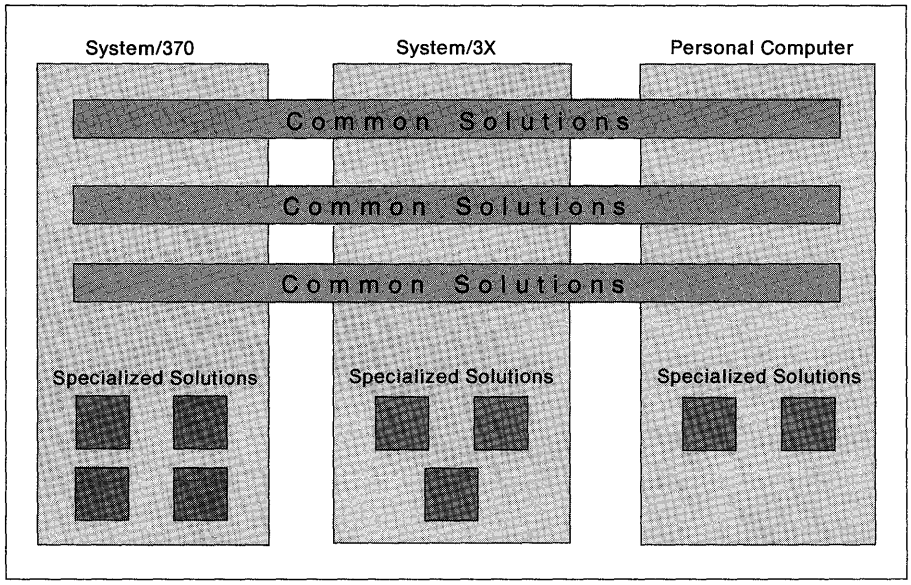


Figure 4. Both Common and Specialized Solutions Are Offered

Future Growth

In its initial form, Systems Application Architecture and its current IBM product implementations offer a starting point. It is IBM's intent that both definitions and implementations increase over time in an evolutionary process.

General Areas

Growth is expected to occur in three general ways:

1. The current specification levels will receive full implementation in all the Systems Application Architecture environments.
2. Individual interfaces will grow and become more powerful, incorporating new features into the original specifications.
3. Additional software interfaces and applications will be defined and included under the Systems Application Architecture umbrella.

Thus, consistency will grow, and with it — the value to customers.

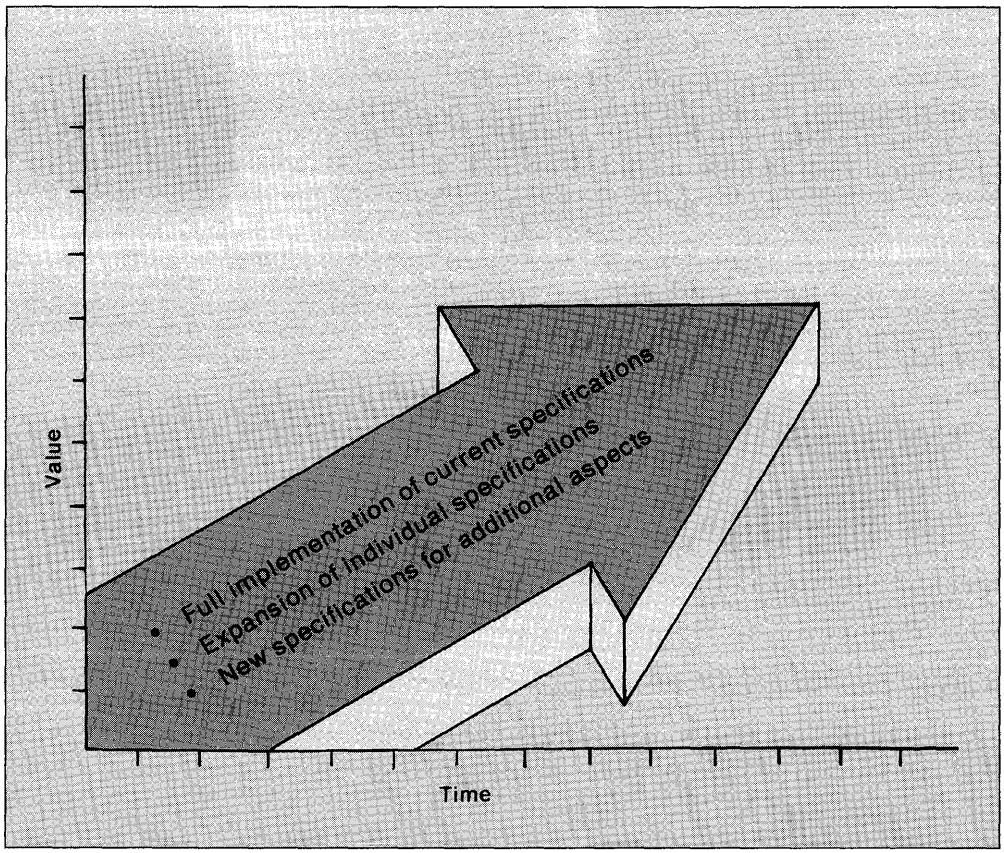


Figure 5. Future Growth

National Language Support

Systems Application Architecture provides IBM with the foundation to enhance the availability and consistency of national language implementation in software products. It is IBM's intent to develop the Systems Application Architecture specifications with support for the implementation of a broad set of national language representations.

Summary

Systems Application Architecture is a definition aimed at helping IBM's customers obtain business solutions. With it, IBM is initiating an evolution in cross-system consistency, while continuing to provide its established products and systems.

Systems Application Architecture is designed to afford growing commonality across the System/370 (TSO/E under MVS/XA, and CMS under VM), System/3X, and Personal Computer (Operating System/2) in the areas of programming interfaces, user access, communications support, and prebuilt applications, as well as national language support.

For each of these areas, IBM is establishing and will publish specifications — definitions of the elements that will be common across the three systems. Future IBM Systems Application Architecture products will then implement those specifications.

Benefits

IBM's Systems Application Architecture will offer customers several advantages:

- Programming skills that have broader use
- Applications that can be ported with less effort, or that can span systems
- User access to these applications that is simpler and more uniform.

Both development costs and training costs should be reduced. Customers seeking broad solutions for their personal, departmental, and enterprise-wide data processing needs will profit. And independent software vendors who choose to build on IBM products will benefit as well.

Future Growth

The initial IBM definitions and product implementation levels provide a starting point for cross-system consistency. Over time, both the definition and implementation of Systems Application Architecture will expand, and the level of consistency will increase.

Thus, customers will see an evolutionary process — a continual broadening and reinforcement of consistency, with an attendant increase in benefits.

Part Two: Contents of the Programming Interface

Part Two: Contents of the Programming Interface

On the following pages are descriptions of the components of the Common Programming Interface, with accompanying tables. Each table provides:

- A high-level list of the contents of the component — those elements in the IBM specification
- An indication of where each element is currently supported — that is, what Systems Application Architecture environments already have a released or announced licensed program which implements that interface element.

System/3X Considerations: Although the System/3X computing environment is not referenced in the tables, it is IBM's intent to implement the Common Programming Interface in future offerings of the System/3X. It is also IBM's intent that current System/36 and System/38 programming interfaces will continue to be supported on future offerings of the System/3X, conserving current application investment.

Reference Manuals

The tables here offer a general overview of the contents of the components of the interface. Comprehensive reference manuals are being prepared as well. These manuals will provide the full contents of each component, and constitute the IBM specifications for them. As the interface expands, the manuals will be updated accordingly.

Languages

Application Generator

An application generator is a generalized application development tool. It allows one to build some types of programs easily without using a traditional high-level language or being involved with system details. Application generators can be categorized as fourth-generation tools for the professional programmer.

Seated at the terminal, a programmer can create an application interactively. An application generator offers:

- A dialog-oriented, fill-in-the-blanks approach
- Immediate interactive syntax checking
- Prompting, tutorials, and a help facility.

Use of an application generator eliminates many of the steps required when creating applications using conventional methods. Each phase of certain projects — defining and validating screens, files, and logic; testing and debugging programs; and running trial executions of the application — can be completed under its guidance. Through each of these steps, it buffers the user from the complexities of system and data management.

IBM's application generator interface is tailor-made for supporting portability across systems. Its two-part structure — an application development portion (AD) and an application execution portion (AE) — allows an application to be built on one system and easily run on another (where the AE functions are installed). The AD function generates an application that is relatively independent of operating system and hardware considerations. When the application is actually used, AE handles the implementation and automatically adapts the code to the specific system it is running on.

The Interface for Systems Application Architecture

The interface specification is based on the existing System/370 Cross System Product/Application Execution. Thus, current product users will find the contents of the new interface to be familiar. For more information, see the table on the following pages.

The table below lists the elements currently in the application generator interface for Systems Application Architecture.

The table indicates (with an X) which systems already have an IBM licensed program announced or available that implements a particular language element.

On MVS and VM, the implementing product is Cross System Product/Application Execution, Version 3 (5668-814). An asterisk (*) indicates interface elements available in EZ-RUN Cross System Product/Application Execution (6317011 feature number 9375) running in PC DOS compatibility mode of Operating System/2.

Interface Element	MVS	VM	OS/2
Processing Statements			
Arithmetic operators	X	X	*
Conditional operators	X	X	*
MOVE data item content	X	X	*
MOVE corresponding at structure level	X	X	*
Logic controls (IF, ELSE, AND, OR, WHILE)	X	X	*
FIND in a table	X	X	*
RETRIEVE from table	X	X	*
SET record status	X	X	*
SET map status	X	X	*
SET map item attribute	X	X	*
TEST map, map or data item, record, or entry key	X	X	*
CALL application	X	X	*
TRANSFER to an application	X	X	*
PERFORM a process	X	X	*
Go to a process	X	X	*
Statement group subroutines	X	X	*
Process Options			
EXECUTE statements	X	X	*
DISPLAY a map	X	X	*
CONVERSE a map	X	X	*
ADD a record into a file	X	X	*
DELETE a record from a file	X	X	*
Read a record (INQUIRY)	X	X	*
Read record for update (UPDATE)	X	X	*
REPLACE a record	X	X	*
SCAN for next record	X	X	*
CLOSE a file or printer	X	X	*

Figure 6 (Part 1 of 3). Major Elements of the Application Generator Interface

Interface Element	MVS	VM	OS/2
Special Functions			
Transfer application name variable	X	X	*
Gregorian date	X	X	*
Julian date	X	X	*
Time	X	X	*
Exit from an application	X	X	*
Exit from statement group	X	X	*
Go To application flow logic	X	X	*
Entry key ID (AID)	X	X	*
MSG ID to be displayed	X	X	*
Map message field	X	X	*
User ID	X	X	*
Control hard I/O errors	X	X	*
Control arithmetic overflows	X	X	*
Test overflow results	X	X	*
Data base commitment control	X	X	*
Table FIND result (row number or array index)	X	X	*
File and Data Base			
Indexed access	X	X	*
Relative access	X	X	*
Serial access	X	X	*
Working storage definition	X	X	*
Redefined records	X	X	*
Key item name	X	X	*
Alternate record specification	X	X	*
Record Data Items			
Item name	X	X	*
Structures	X	X	*
Arrays	X	X	*
Binary data	X	X	*
Character data	X	X	*
Hexadecimal data	X	X	
Unsigned numeric data	X	X	*
Signed numeric data	X	X	*
Packed numeric data	X	X	*
Unsigned packed numeric data	X	X	*
Decimal positions	X	X	*

Figure 6 (Part 2 of 3). Major Elements of the Application Generator Interface

Interface Element	MVS	VM	OS/2
Reference and Edit Tables			
Reference table	X	X	*
Match valid edit table (include)	X	X	*
Match invalid edit table (exclude)	X	X	*
Range match valid edit table	X	X	*
Column definition	X	X	*
Contents definition	X	X	*
Screen and Printer Maps			
Map group name	X	X	*
Map name (within a group)	X	X	*
Map size for partial maps	X	X	*
Partial map position	X	X	*
Initial cursor position	X	X	*
Fold character input option	X	X	*
Help map name	X	X	*
Help PF key	X	X	*
Map formats	X	X	*
Map variable field name	X	X	*
Map field arrays	X	X	*
Field attribute definition	X	X	*
Display device selection	X	X	*
Map variable editing sequence	X	X	*
Map variable output editing	X	X	*
Map variable input editing and validation	X	X	*
Application			
Name	X	X	*
Type (main or called, transaction or batch)	X	X	*
Default help PF key	X	X	*
Process I/O option	X	X	*
Process object name	X	X	*
Process I/O error routine	X	X	*
Logic flow	X	X	*
Table and additional records list	X	X	*
Called parameter list	X	X	*

Figure 6 (Part 3 of 3). Major Elements of the Application Generator Interface

C

C is a programming language designed for a wide variety of programming tasks. It has been used for system-level code, text processing, graphics, and for development of engineering, scientific, and commercial applications.

The C language itself is compact, with function added through its library. This division has resulted in C being regarded as both flexible and efficient. An additional benefit is that the language is highly consistent across different systems.

C's flexibility permits its users to deal easily with machine-level entities at a low level, while at the same time having the high-level control and data structures found in other modern, structured programming languages.

Included is an extensive library of functions to provide input and output, mathematics, exception handling, string and character manipulation, dynamic memory management, as well as date and time manipulation. Use of this library helps to maintain program portability, because the underlying implementation details for the various operations need not be of concern to the programmer.

C supports numerous data types, including characters, integers, floating-point numbers and pointers — each in a variety of forms. In addition, C also supports data aggregates such as arrays, structures (records), unions, and enumerations.

The Interface for Systems Application Architecture

The interface specification has been developed in accord with the draft proposed American National Standard Programming Language - C (X3J11). For more information on the IBM specification, see the table on the following pages.

The table below lists the language elements currently in the C interface for Systems Application Architecture.

The table indicates (with an X) which systems already have an IBM licensed program announced or available that implements a particular language element. On Operating System/2 (Personal Computer), the implementing product will be IBM C/2.

An asterisk (*) indicates language elements that are consistent with the C Program Offerings for MVS and VM (5713-AAG and 5713-AAH). Although not part of Systems Application Architecture, these program offerings can be used to gain early experience with the C interface.

Language Element	MVS	VM	OS/2
Data Types			
`signed` keyword	*	*	X
`volatile` keyword	*	*	X
`const` keyword	*	*	X
`void *` pointers	*	*	X
enumerated datatype	*	*	X
`long double` datatype	*	*	X
Language Features			
Adjacent strings concatenated	*	*	X
Full function prototypes	*	*	X
Ref-Def model for externs	*	*	X
Preprocessor Directives			
if/ifdef/ifndef	*	*	X
else/elif/endif	*	*	X
define	*	*	X
line	*	*	X
include	*	*	X
pragma	*	*	X
undef	*	*	X
Escape sequences			
\a \b \f \n \r \t \v	*	*	X
ooo — octal	*	*	X
\x*hhh* — hexadecimal	*	*	X
Predefined Macros			
__LINE__ macro	*	*	X
__FILE__ macro	*	*	X

Figure 7 (Part 1 of 4). Major Elements of the C Interface

Language Element	MVS	VM	OS/2
Standard I/O:			
Standard streams			
stdin/stdout/stderr	*	*	X
File operations			
remove	*	*	X
rename	*	*	X
tmpfile/tmpnam	*	*	X
File access			
fclose	*	*	X
fflush	*	*	X
fopen	*	*	X
freopen	*	*	X
setbuf/setvbuf	*	*	X
Formatted I/O			
printf/fprintf/sprintf	*	*	X
vprintf/vfprintf/vsprintf	*	*	X
scanf/fscanf/sscanf	*	*	X
Character I/O			
fgetc/getc/getchar	*	*	X
fputc/putc/putchar	*	*	X
fgets/gets	*	*	X
fputs/puts	*	*	X
ungetc	*	*	X
Direct I/O			
fread/fwrite	*	*	X
ftell/fseek	*	*	X
rewind	*	*	X
Error-handling			
clearerr	*	*	X
feof	*	*	X
ferror	*	*	X

Figure 7 (Part 2 of 4). Major Elements of the C Interface

Language Element	MVS	VM	OS/2
Signal Handling:			
signal function	*	*	X
Signals Supported:			
SIGABRT	*	*	X
SIGFPE	*	*	X
SIGILL	*	*	X
SIGINT	*	*	X
SIGSEGV	*	*	X
SIGTERM	*	*	X
SIGUSR1	*	*	X
SIGUSR2	*	*	X
Non-Local Jumps:			
setjmp/longjmp	*	*	X
Mathematical:			
cos/sin/tan	*	*	X
acos/asin/atan/atan2	*	*	X
exp/log/log10	*	*	X
frexp/ldexp	*	*	X
modf/fmod	*	*	X
pow/sqrt	*	*	X
ceil/floor/fabs	*	*	X
Bessel functions	*	*	X
General Utilities:			
String conversion			
atof/atoi/atol	*	*	X
strtod/strtol	*	*	X
Pseudo-random numbers			
rand/srand	*	*	X
Memory management			
calloc/malloc/realloc	*	*	X
free	*	*	X
Environment interactions			
abort	*	*	X
exit	*	*	X
getenv	*	*	X
system	*	*	X
Searching and sorting			
bsearch/qsort			X
Integer arithmetic			
abs/labs	*	*	X

Figure 7 (Part 3 of 4). Major Elements of the C Interface

Language Element	MVS	VM	OS/2
Variable Arguments:			
va_start/va_arg/va_end	*	*	X
vprintf/vfprintf/vsprintf	*	*	X
String Operations:			
strlen	*	*	X
strstr			X
strtok	*	*	X
strpbrk	*	*	X
strcat/strncat	*	*	X
strcmp/strncmp	*	*	X
strcpy/strncpy	*	*	X
strchr/strrchr	*	*	X
strspn/strcspn	*	*	X
Memory Block Operations:			
memcpy, memcmp	*	*	X
memchr, memset	*	*	X
Date and Time:			
Time manipulation			
difftime	*	*	X
time	*	*	X
Time conversion			
asctime	*	*	X
ctime	*	*	X
gmtime	*	*	X
localtime	*	*	X
Character Handling:			
Character testing			
isalnum/isalpha/iscntrl	*	*	X
isdigit/isgraph/islower	*	*	X
isprint/ispunct/isspace	*	*	X
isupper/isxdigit	*	*	X
Character case mapping			
tolower/toupper	*	*	X

Figure 7 (Part 4 of 4). Major Elements of the C Interface

COBOL

COBOL is a widely-used application programming language. Its success is due largely to:

- Its power in handling the data processing needs of business
- The natural English-like appearance of the language, which makes it easy to write and maintain applications.

Programs written in COBOL are geared to handle large volumes of data from sources as different as magnetic tapes and display terminals. After manipulating this data, the programs can produce a variety of outputs, such as printed reports or files on disk storage devices. COBOL is also flexible enough to handle a wide range of data processing needs — from the overnight processing of a company payroll to the "I-need-an-answer-now" demands of insurance claim inquiries.

The Interface for Systems Application Architecture

In general, the language elements in the IBM specification fall into three categories:

- American National Standard Programming Language - COBOL, ANSI X3.23-1985, ISO standard 1989-1985, Intermediate Level
- Some elements from ANSI X3.23-1985, High Level
- IBM enhancements to this standard — such as COMP-3 and COMP-4 data items.

Because the IBM specification is based largely on these well-known and accepted industry standards, programmers will find several benefits:

- Familiarity
- Ability to use existing standard programs currently running on IBM systems
- Ability to take standard programs from non-IBM systems and run them on the supported architectures.

For enhanced programmer productivity and ease of use, IBM's specification will contain other language features beyond the industry standards. For more information, see the table on the following page.

The table below lists the language elements currently in the COBOL interface for Systems Application Architecture. This specification is based largely on the 1985 ANS COBOL standard, at the Intermediate Level. (Language elements that are new in the 1985 standard are printed in italics.) To this standard, IBM also adds certain extensions.

The table indicates (with an X) which systems already have an IBM licensed program announced or available that implements a particular language element.

On MVS and VM, the VS COBOL II product (5668-958) provides the COBOL interface, with the exception of certain 1985 ANS COBOL Intermediate Level items. On Operating System/2 (Personal Computer), the implementing product will be IBM COBOL/2.

Language Element	MVS	VM	OS/2
ANS LANGUAGE ELEMENTS			
NUCLEUS MODULE			
Language Concepts:			
COBOL words	X	X	X
Characters a-z in COBOL words			X
Literals	X	X	X
PICTURE character-strings	X	X	X
Comment-entries	X	X	X
Qualification	X	X	X
50 levels of qualifiers	X	X	X
Subscripting (data-name/literal)	X	X	X
Mixed indexes and subscripts			X
Relative subscripting			X
Subscripting (index-name)	X	X	X
Reference format:			
TITLE	X	X	X
Sequence number	X	X	X
Comment lines	X	X	X
Debugging lines	X	X	X
IDENTIFICATION DIVISION:			
PROGRAM-ID paragraph	X	X	X
AUTHOR paragraph	X	X	X

Figure 8 (Part 1 of 6). Major Elements of the COBOL Interface

Language Element	MVS	VM	OS/2
INSTALLATION paragraph	X	X	X
DATE-WRITTEN paragraph	X	X	X
DATE-COMPILED paragraph	X	X	X
SECURITY paragraph	X	X	X
ENVIRONMENT DIVISION:			
Configuration Section:			
SOURCE-COMPUTER paragraph	X	X	X
OBJECT-COMPUTER paragraph	X	X	X
SPECIAL-NAMES paragraph	X	X	X
Alphabet-name clause:			
word ALPHABET			X
STANDARD-2 option (ISO 7-Bit Code)			X
CLASS clause			X
Input-Output Section:			
File control entry:	X	X	X
ASSIGN TO literal			X
DATA DIVISION:			
Working-Storage Section:			
BLANK WHEN ZERO clause	X	X	X
Data-name or FILLER clause	X	X	X
JUSTIFIED clause	X	X	X
Level-number	X	X	X
OCCURS clause	X	X	X
VALUE clause allowed with OCCURS			X
PICTURE clause	X	X	X
REDEFINES clause	X	X	X
RENAMES clause	X	X	X
SIGN clause	X	X	X
SYNCHRONIZED clause	X	X	X
USAGE clause	X	X	X
BINARY			X
PACKED-DECIMAL			X
VALUE clause	X	X	X
PROCEDURE DIVISION:			
Arithmetic expressions	X	X	X
Conditional expressions	X	X	X
Relation condition:	X	X	X
Relational operators:	X	X	X
GREATER THAN OR EQUAL TO			X
> =			X

Figure 8 (Part 2 of 6). Major Elements of the COBOL Interface

Language Element	MVS	VM	OS/2
LESS THAN OR EQUAL TO			X
<=			X
Class condition:	X	X	X
ALPHABETIC (uppercase and lowercase alphabetic characters)			X
ALPHABETIC-LOWER			X
ALPHABETIC-UPPER			X
class-name			X
ACCEPT statement	X	X	X
ADD statement	X	X	X
TO identifier/literal GIVING identifier			X
NOT ON SIZE ERROR phrase			X
END-ADD phrase	X	X	X
ALTER statement	X	X	X
COMPUTE statement	X	X	X
NOT ON SIZE ERROR phrase			X
END-COMPUTE phrase	X	X	X
CONTINUE statement	X	X	X
DISPLAY statement	X	X	X
DIVIDE statement	X	X	X
NOT ON SIZE ERROR phrase			X
END-DIVIDE phrase	X	X	X
ENTER statement	X	X	X
EXIT statement	X	X	X
EVALUATE statement	X	X	X
GO TO statement	X	X	X
IF statement	X	X	X
END-IF phrase	X	X	X
INITIALIZE statement	X	X	X
INSPECT statement	X	X	X
CONVERTING phrase			X
MOVE statement	X	X	X
MULTIPLY statement	X	X	X
NOT ON SIZE ERROR phrase			X
END-MULTIPLY phrase	X	X	X
PERFORM statement	X	X	X
In-line Perform	X	X	X
END-PERFORM phrase	X	X	X
UNTIL phrase	X	X	X
WITH TEST BEFORE/AFTER phrase	X	X	X
VARYING phrase	X	X	X
WITH TEST BEFORE/AFTER phrase	X	X	X
SEARCH statement	X	X	X
END-SEARCH phrase	X	X	X

Figure 8 (Part 3 of 6). Major Elements of the COBOL Interface

Language Element	MVS	VM	OS/2
SEARCH ALL statement	X	X	X
END-SEARCH phrase	X	X	X
SET statement	X	X	X
Mnemonic-name TO ON/OFF			X
Condition-name TO TRUE	X	X	X
STOP statement	X	X	X
STRING statement	X	X	X
NOT ON OVERFLOW phrase			X
END-STRING phrase	X	X	X
SUBTRACT statement	X	X	X
NOT ON SIZE ERROR phrase			X
END-SUBTRACT phrase	X	X	X
UNSTRING statement	X	X	X
NOT ON OVERFLOW phrase			X
END-UNSTRING phrase	X	X	X
I-O MODULES			
ENVIRONMENT DIVISION:			
Input-Output Section:			
File control entry:			
SELECT clause	X	X	X
ASSIGN clause	X	X	X
ASSIGN TO literal			X
ORGANIZATION clause	X	X	X
ACCESS MODE clause	X	X	X
RECORD KEY clause	X	X	X
ALTERNATE RECORD KEY clause	X	X	X
FILE STATUS clause	X	X	X
RESERVE AREA clause	X	X	X
I-O-CONTROL paragraph:			
RERUN clause	X		X
SAME clause	X	X	X
MULTIPLE FILE TAPE clause	X	X	X
DATA DIVISION:			
File Section:			
FD file name	X	X	X
BLOCK CONTAINS clause	X	X	X
CODE-SET clause	X	X	X
DATA RECORDS clause	X	X	X
LABEL RECORDS clause	X	X	X
LINAGE clause	X	X	X

Figure 8 (Part 4 of 6). Major Elements of the COBOL Interface

Language Element	MVS	VM	OS/2
RECORD clause	X	X	X
VALUE OF clause	X	X	X
Record description entry	X	X	X
PROCEDURE DIVISION:			
CLOSE statement	X	X	X
DELETE statement	X	X	X
NOT INVALID KEY phrase			X
END-DELETE phrase	X	X	X
OPEN statement	X	X	X
READ statement	X	X	X
NOT AT END phrase			X
NOT INVALID KEY phrase			X
END-READ phrase	X	X	X
REWRITE statement	X	X	X
NOT INVALID KEY phrase			X
END-REWRITE phrase	X	X	X
START statement	X	X	X
NOT INVALID KEY phrase			X
END-START phrase	X	X	X
USE statement	X	X	X
WRITE statement	X	X	X
NOT AT END-OF-PAGE (EOP) phrase			X
NOT INVALID KEY phrase			X
END-WRITE phrase	X	X	X
File Status Codes	X	X	X
New File Status Codes			X
INTER-PROGRAM COMMUNICATION MODULE			
LINKAGE SECTION	X	X	X
PROCEDURE DIVISION:			
PROCEDURE DIVISION USING	X	X	X
CALL statement	X	X	X
ON EXCEPTION phrase			X
NOT ON EXCEPTION phrase			X
END-CALL phrase	X	X	X
CANCEL statement	X	X	X
EXIT PROGRAM statement	X	X	X

Figure 8 (Part 5 of 6). Major Elements of the COBOL Interface

Language Element	MVS	VM	OS/2
SORT-MERGE MODULE			
ENVIRONMENT DIVISION:			
Input-Output Section:			
FILE-CONTROL paragraph	X		X
File control entry	X		X
SELECT clause	X		X
ASSIGN clause	X		X
ASSIGN TO literal			X
SAME clause	X		X
DATA DIVISION:			
File Section:			
SD Sort-Merge File	X		X
Record description entry	X		X
DATA RECORDS clause	X		X
RECORD clause	X		X
PROCEDURE DIVISION:			
MERGE statement	X		X
RELEASE statement	X		X
RETURN statement	X		X
NOT AT END phrase			X
END-RETURN phrase	X		X
SORT statement	X		X
DUPLICATES phrase			X
SOURCE TEXT MANIPULATION MODULE			
COPY statement	X	X	X
IBM EXTENSIONS			
Special Register WHEN-COMPILED	X	X	X
ID DIVISION abbreviation	X	X	X
Optional IDENTIFICATION DIVISION paragraphs in any order	X	X	X
Second file status	X	X	X
USAGE COMPUTATIONAL-3 (COMP-3)	X	X	X
USAGE COMPUTATIONAL-4 (COMP-4)	X	X	X
ACCEPT FROM CONSOLE/SYSIN	X	X	X
DISPLAY UPON CONSOLE/SYSOUT	X	X	X
GOBACK statement	X	X	X
EJECT and SKIP1/2/3 statements	X	X	X
TITLE statement	X	X	X

Figure 8 (Part 6 of 6). Major Elements of the COBOL Interface

FORTRAN

FORTRAN is a programming language designed for jobs involving mathematical computations and other manipulation of numeric data. This makes it especially well-suited to scientific and engineering applications.

Because it is simple and easy to learn, and because it produces efficient code, FORTRAN is widely used. It forms a convenient and familiar tool for anyone involved in mathematical computation. Scientists, engineers, and students are only a few of the many people who use it.

The original FORTRAN was developed by IBM. Over the years, IBM has continued to enhance the language and to offer more powerful and sophisticated FORTRAN products with a variety of features on all its systems.

The Interface for Systems Application Architecture

The interface specification provides a language that has the familiar simplicity of its predecessors, but also incorporates new features. In general, the language elements fall into two categories:

- American National Standard Programming Language - FORTRAN, ANSI X3.9-1978 (FORTRAN 77), ISO standard, 1539-1980.
- Enhancements to this standard — such as the ability to use names that are up to 31 characters long.

Because the IBM specification is based on the ANS standard, users will obtain several benefits:

- Familiarity
- Ability to use existing programs currently running on IBM systems (many of which are based on the ANS 77 standard)
- Ability to take standard programs from non-IBM systems and run them on the supported IBM architectures.

For enhanced programmer productivity and ease of use, IBM's specification will contain language features beyond the industry standards. For more details, see the table on the following page.

The table below lists the language elements currently in the FORTRAN interface for Systems Application Architecture.

The table indicates (with an X) which systems already have an IBM licensed program announced or available that implements a particular language element.

On MVS and VM, the VS FORTRAN Version 2 product (5668-806) provides the FORTRAN interface, with the exception of the INCLUDE statement. On Operating System/2 (Personal Computer), the implementing product will be IBM FORTRAN/2.

Language Element	MVS	VM	OS/2
1977 ANS FORTRAN — all elements	X	X	X
INTEGER*2	X	X	X
LOGICAL*1	X	X	X
COMPLEX*16	X	X	X
COMPLEX*16 intrinsic functions	X	X	X
ISA S61.1 bit routines	X	X	X
IMPLICIT NONE	X	X	X
INCLUDE			X
31-character names	X	X	X
Mixed case names	X	X	X
IBM minimum compiler limits	X	X	X

Figure 9. Major Elements of the FORTRAN Interface

Procedures Language

IBM's procedures language allows one to write programs in a clear, structured way. In its REXX implementation, it has proved to be easy to learn and teach. The clarity it offers makes it useful to professional programmers and "casual" users alike. Powerful character and arithmetic abilities make it suited for programs large or small. Its conventional syntax provides a flexible language suited for use not only as a command and macro language, but as a full-function development language as well.

Ease of use is a distinctive feature of the procedures language. Furthermore, its design allows dynamic interpretation of statements, if desired.

The procedures language's other features include:

- Presentation of host commands to the system
- External and internal calling mechanisms
- Structured programming constructs, such as IF-THEN-ELSE and SELECT
- Expressions, including arithmetic, concatenation, comparative, and logical operators — all of which act upon variables that can contain either character strings or numbers
- Extensive string parsing by pattern matching; a string (possibly the input from an external source) can be split into parts and assigned to variables as needed
- Exception handling and tracing mechanisms.

The Interface for Systems Application Architecture

The interface specification is based largely on the REXX language currently available as part of the VM operating system. For more information on the contents of this specification, see the table on the following pages.

The table below lists the language elements currently in the procedures interface for Systems Application Architecture.

The table indicates (with an "X") which systems already have an IBM component announced or available that implements a particular language element.

The current implementation in the CMS component of VM/SP is the System Product Interpreter (also known as REXX).

Language Element	MVS	VM	OS/2
General Concepts			
Assignment statement		X	
Null clause		X	
Labels		X	
Commands		X	
Simple variables		X	
Compound variables		X	
Constant symbols		X	
Literal strings		X	
Hexadecimal literal strings		X	
Internal function calls		X	
External function calls		X	
Integer numbers		X	
Decimal numbers		X	
Exponential notation		X	
Instructions			
ADDRESS instruction		X	
ARG instruction		X	
CALL instruction		X	
simple DO		X	
repetitive DO		X	
DO FOREVER		X	
DO WHILE		X	
DO UNTIL		X	
controlled repetitive DO		X	
DROP		X	
EXIT		X	
IF-THEN-ELSE		X	
INTERPRET		X	
ITERATE		X	
LEAVE		X	
NOP		X	
NUMERIC DIGITS		X	
NUMERIC FORM		X	
NUMERIC FUZZ		X	
OPTIONS		X	
PARSE ARG		X	

Figure 10 (Part 1 of 4). Major Elements of the Procedures Language

Language Element	MVS	VM	OS/2
PARSE PULL		X	
PARSE SOURCE		X	
PARSE VALUE		X	
PARSE VAR		X	
PARSE VERSION		X	
PROCEDURE		X	
PROCEDURE EXPOSE		X	
PULL		X	
PUSH		X	
QUEUE		X	
RETURN		X	
SAY		X	
SELECT		X	
SIGNAL		X	
SIGNAL ON		X	
TRACE		X	
Functions			
ABBREV built-in function		X	
ABS built-in function		X	
ADDRESS built-in function		X	
ARG built-in function		X	
BITAND built-in function		X	
BITOR built-in function		X	
BITXOR built-in function		X	
CENTRE built-in function		X	
CENTER built-in function		X	
COMPARE built-in function		X	
COPIES built-in function		X	
C2D built-in function		X	
C2X built-in function		X	
DATATYPE built-in function		X	
DATE built-in function		X	
DELSTR built-in function		X	
DELWORD built-in function		X	
D2C built-in function		X	
D2X built-in function		X	
ERRORTEXT built-in function		X	
FORMAT built-in function		X	
INSERT built-in function		X	
LASTPOS built-in function		X	
LEFT built-in function		X	
LENGTH built-in function		X	
MAX built-in function		X	
MIN built-in function		X	
OVERLAY built-in function		X	
POS built-in function		X	

Figure 10 (Part 2 of 4). Major Elements of the Procedures Language

Language Element	MVS	VM	OS/2
QUEUED built-in function		X	
RANDOM built-in function		X	
REVERSE built-in function		X	
RIGHT built-in function		X	
SIGN built-in function		X	
SOURCELINE built-in function		X	
SPACE built-in function		X	
STRIP built-in function		X	
SUBSTR built-in function		X	
SUBWORD built-in function		X	
SYMBOL built-in function		X	
TIME built-in function		X	
TRACE built-in function		X	
TRANSLATE built-in function		X	
TRUNC built-in function		X	
VALUE built-in function		X	
VERIFY built-in function		X	
WORD built-in function		X	
WORDINDEX built-in function		X	
WORDLENGTH built-in function		X	
WORDS built-in function		X	
XRANGE built-in function		X	
X2C built-in function		X	
X2D built-in function		X	
Operators			
= comparative operator		X	
¬ = comparative operator		X	
> comparative operator		X	
< comparative operator		X	
< > comparative operator		X	
> < comparative operator		X	
> = comparative operator		X	
¬ < comparative operator		X	
< = comparative operator		X	
¬ > comparative operator		X	
= = comparative operator		X	
¬ = = comparative operator		X	
& boolean operator		X	
| boolean operator		X	
&& boolean operator		X	
Prefix ¬ boolean operator		X	
(blank) concatenation		X	
|| concatenation		X	
(abuttal) concatenation		X	
+ arithmetic operator		X	
- arithmetic operator		X	

Figure 10 (Part 3 of 4). Major Elements of the Procedures Language

Language Element	MVS	VM	OS/2
* arithmetic operator		X	
/ arithmetic operator		X	
% arithmetic operator		X	
// arithmetic operator		X	
** arithmetic operator		X	
Prefix - arithmetic operator		X	
Prefix + arithmetic operator		X	
Parentheses and operator precedence		X	
Parsing			
Word parsing		X	
Literal parsing triggers		X	
Variable parsing triggers		X	
Absolute column parsing		X	
Relative column parsing		X	
Special Features			
RC special variable		X	
RESULT special variable		X	
SIGL special variable		X	
Variable pool interface (EXECCOMM in CMS)		X	

Figure 10 (Part 4 of 4). Major Elements of the Procedures Language

Services

Database Interface

Access to data bases is provided through SQL (Structured Query Language). It allows users to define, retrieve, and manipulate information in a relational data base.

SQL is nonprocedural. Users specify what they want to do, and don't need to be concerned with how it's done. Syntax is straightforward and easy for even occasional users to learn and remember. Simple yet powerful, single statements can perform the same function as many lines of conventional code. SQL makes complex operations possible and typical tasks easy.

SQL also provides built-in functions and arithmetic operations, so programmers can perform immediate mathematical operations on data, often without having to write traditional programs. On the other hand, if noninteractive processing is desired, SQL statements can be imbedded in traditional application programs, written in such languages as COBOL or FORTRAN, and then precompiled.

SQL is based on the relational data model, an advanced technology that makes data easier to access and use. Various pieces of information can be viewed in relationship to each other without predefining that relationship in the data base structure. This easier, more flexible method allows users and applications to share and access data in an ad hoc fashion to support changing requirements for information.

The description of how data is stored or managed is not a part of the application. Therefore, storage and management of data can be optimized to each environment without impacting the portability of the application or the application development process.

The Interface for Systems Application Architecture

The specification for the SQL language provides a framework for similar data access in the supported architectures. This means that an application can be moved from environment to environment with minimal change.

IBM's specification has been developed with consideration for the American National Standard Database Language - SQL, ANSI X3.135-1986. The IBM specification is almost identical to the language in DB2 and SQL/DS, so current users of those System/370 licensed programs will have the benefits of familiarity. For more specifics, see the table on the following page.

The table below lists the language elements currently in the database interface for Systems Application Architecture.

The table indicates (with an X) which systems already have an IBM licensed program announced or available that implements a particular language element.

On MVS, the implementing product will be DB2 Release 3 (5740-XYR). On VM, it will be SQL/DS Version 2 (5688-004). On the Personal Computer, the implementation will be provided by the IBM Operating System/2 Database Manager.

Language Elements	MVS	VM	OS/2
SELECT Expressions			
SELECT list	X	X	X
FROM clause	X	X	X
WHERE clause	X	X	X
GROUP BY clause	X	X	X
HAVING clause	X	X	X
Data Definition			
CREATE TABLE	X	X	X
CREATE VIEW	X	X	X
ALTER TABLE	X	X	X
CREATE INDEX	X	X	X
DROP INDEX	X	X	X
DROP TABLE	X	X	X
DROP VIEW	X	X	X
COMMENT ON	X	X	X
Authorization			
GRANT	X	X	
REVOKE	X	X	
Basic Statements			
INSERT	X	X	X
SELECT	X	X	X
UPDATE ... WHERE	X	X	X
DELETE ... WHERE	X	X	X

Figure 11 (Part 1 of 3). Major Elements of the Database Interface

Language Elements	MVS	VM	OS/2
Cursor-Oriented Operations			
DECLARE CURSOR	X	X	X
ORDER BY	X	X	X
SELECT ... FOR UPDATE	X	X	X
UNION	X	X	
OPEN	X	X	X
FETCH	X	X	X
DELETE	X	X	X
UPDATE	X	X	X
CLOSE	X	X	X
Dynamic Facilities			
PREPARE	X	X	X
DESCRIBE	X	X	X
EXECUTE	X	X	X
EXECUTE IMMEDIATE	X	X	X
SQLDA	X	X	X
Miscellaneous Statements			
INCLUDE	X	X	X
LOCK TABLE	X	X	X
WHENEVER	X	X	X
Data Types			
CHARACTER	X	X	X
VARCHAR	X	X	X
LONG VARCHAR	X	X	X
INTEGER	X	X	X
SMALLINT	X	X	X
DECIMAL	X	X	X
FLOAT	X	X	X
DATE	X	X	X
TIME	X	X	X
TIMESTAMP	X	X	X

Figure 11 (Part 2 of 3). Major Elements of the Database Interface

Language Elements	MVS	VM	OS/2
Other Language Elements			
Null Values	X	X	X
Host Variable References	X	X	X
Indicator Variables	X	X	X
Column References	X	X	X
Ordinary Identifiers	X	X	X
Delimited Identifiers	X	X	X
Functions: AVG, MIN, MAX, SUM, COUNT	X	X	X
Arithmetic Operators	X	X	X
Comparison Operators	X	X	X
BETWEEN Predicate	X	X	X
IN Predicate	X	X	X
LIKE Predicate	X	X	X
IS NULL Predicate	X	X	X
ALL/ANY Predicate	X	X	X
EXISTS Predicate	X	X	X
Search Condition	X	X	X
SQLCA	X	X	X
Pre-Compilers			
C			X
COBOL	X	X	X
FORTRAN	X	X	

Figure 11 (Part 3 of 3). Major Elements of the Database Interface

Dialog Interface

Dialog services help programmers develop interactive applications. These services fall into two broad categories:

- The control of the display and interaction of panels containing constant and variable information on a screen, including:
  - menu selections
  - help information
  - data requests
  - messages
- The passing to the application of data and function requests from the user.

Thus, assistance is provided when a user is choosing general functions available in the whole system, or when data (input and responses) is being moved between a user and a specific application.

From the application developer's point of view, this management function can be thought of as a convenient, easy-to-use extension of the operating system's services. It helps the application developer code functions such as input field validation, message services, and help facilities.

Along with presentation, consistent dialog management buffers the programmer, the application, and the user from device-dependent considerations. This enhances application portability and simplifies application development. The programmer will use similar development tools across operating environments, gaining in productivity. And the users of these applications will have a consistent view across applications and systems.

The Interface for Systems Application Architecture

The interface specification for dialog is based on the existing Personal Computer EZ-VU Run Time Facility. Thus, current product users will find the new interface familiar. For more information, see the table on the following page.

The table below lists the elements currently in the dialog interface for Systems Application Architecture.

The table indicates (with an X) which systems already have an IBM licensed program announced or available that implements a particular element.

On the Personal Computer, the implementation will be provided by the IBM Operating System/2 Dialog Manager.

Dialog Service	MVS	VM	OS/2
Panel Services			
CONTROL (changes panel characteristics)			X
DMSELECT (selects another menu panel)			X
PANDEL (deletes a panel)			X
PANDISP (displays a panel)			X
PANREAD (gets input from a panel)			X
PANWRITE (writes a panel)			X
SETCUR (positions cursor)			X
SETMSG (writes a message)			X
Variable Services			
VDEFARR (defines an array)			X
VDEFINE (defines a scalar)			X
VDELETE (deletes a variable)			X
VGET (gets a variable)			X
VOBTAIN (gets a variable)			X
VPUT (puts a variable)			X
VRESET (deletes a pool)			X
VUPDATE (puts a variable)			X
Supported Languages			
C support			X
COBOL support			X
FORTRAN support			X

Figure 12. Major Elements of the Dialog Interface

Presentation Interface

Presentation services provide programmers and users with a comprehensive set of functions that allow information to be displayed or printed in the most effective manner.

The major functions provided are:

- A windowing system that the user can tailor to display selected information from one or several applications
- Support for presentation and interaction via the keyboard and mouse to enable applications to conform to the Common User Access
- Comprehensive graphics support
- Limited image support
- Saving and restoring graphics pictures
- Support for many types of display terminals, printers, and plotters.

By their very nature, presentation services buffer the programmer, the application, and the user from device-dependent considerations, thus enhancing application portability and simplifying application development. A cross-system specification augments these inherent benefits.

The Interface for Systems Application Architecture

The interface specification is based largely on the existing System/370 GDDM (Graphical Data Display Manager). Thus, the new specification will be familiar to current GDDM users.

For more information on the contents of the specification, see the table on the following pages.

The table below lists the calls currently in the presentation interface for Systems Application Architecture.

The table indicates (with an X) which systems already have an IBM licensed program announced or available that implements a particular call.

On MVS and VM, the GDDM/MVS and GDDM/VM products (5665-356 and 5664-200, respectively) provide the presentation interface, with the exception of certain elements (primarily in the area of windows). On the Personal Computer, the implementation will be provided by the IBM Operating System/2 Presentation Manager.

Language Element	MVS	VM	OS/2
Graphics:			
Floating-point: GSCP, GpsSetCurrentPosition, ...	X	X	
Fixed-point: GICP, GpiSetCurrentPosition, ...			X
Line functions:			
GS/GICP Gps/GpiSetCurrentPosition	X	X	X
GS/GIFLW Gps/GpiSetFracLineWidth	X	X	X
GS/GILINE Gps/GpiLine	X	X	X
GSLT GpiSetLineType	X	X	X
GSLW GpiSetLineWidth	X	X	X
GS/GIMOVE Gps/GpiMove	X	X	X
GS/GIPLNE Gps/GpiPolyLine	X	X	X
GS/GIQCP Gps/GpiQueryCurrentPosition	X	X	X
GSQFLW GpiQueryFracLineWidth	X	X	X
GSQLT GpiQueryLineType	X	X	X
GSQLW GpiQueryLineWidth	X	X	X
Arc functions:			
GSARC GpiCircArc	X	X	
GSELPS GpiEllipArc	X	X	
GS/GIPFLT Gps/GpiPolyFillet	X	X	X
Area functions:			
GSAREA GpiBeginArea	X	X	X
GSENDA GpiEndArea	X	X	X
GSPAT GpiSetPattern	X	X	X
GSQPAT GpiQueryPattern	X	X	X
Color and mix functions:			
GSBMIX GpiSetBackMix	X	X	X
GSCOL GpiSetColor	X	X	X
GSMIX GpiSetMix	X	X	X
GSQBMX GpiQueryBackMix	X	X	X

Figure 13 (Part 1 of 5). Major Elements of the Presentation Interface

Language Element	MVS	VM	OS/2
GSQCOL GpiQueryColor	X	X	X
GSQMIX GpiQueryMix	X	X	X
Character functions:			
GS/GICA Gps/GpiSetCharAngle	X	X	X
GS/GICB Gps/GpiSetCharBox	X	X	X
GS/GICBS Gps/GpiSetCharSpacing	X	X	X
GSCD GpiSetCharDirection	X	X	X
GS/GICH Gps/GpiSetCharShear	X	X	X
GSCHAP GpiCharString (at current position)	X	X	X
GS/GICHAR Gps/GpiCharStringAt (specified position)	X	X	X
GSCM GpiSetCharMode	X	X	X
GSCS GpiSetCharSet	X	X	X
GSTA GpiSetTextAlignment	X	X	X
GS/GIQCA Gps/GpiQueryCharAngle	X	X	X
GS/GIQCB Gps/GpiQueryCharBox	X	X	X
GS/GIQCBS Gps/GpiQueryCharSpacing	X	X	X
GSQCD GpiQueryCharDirection	X	X	X
GS/GIQCH Gps/GpiQueryCharShear	X	X	X
GSQCM GpiQueryCharMode	X	X	X
GSQCS GpiQueryCharSet	X	X	X
GSQTA GpiQueryTextAlignment	X	X	X
GS/GIQTB Gps/GpiQueryTextBox	X	X	X
Symbol set functions:			
GSDSS GpiLoadSymbolSet	X	X	X
GSRSS GpiDeleteSymbolSet	X	X	X
GSQNSS GpiQueryNumberSymbolSets	X	X	X
GSQSS GpiQuerySymbolSets	X	X	X
GSQSSD GpiQuerySymbolSetData	X	X	X
Marker functions:			
GS/GIMARK Gps/GpiMarker	X	X	X
GS/GIMB Gps/GpiSetMarkerBox	X	X	X
GS/GIMRKS Gps/GpiPolyMarker	X	X	X
GSMS GpiSetMarker	X	X	X
GS/GIQMB Gps/GpiQueryMarkerBox	X	X	X
GSQMS GpiQueryMarkerSymbol	X	X	X
Image function:			
GSIMG GpiImage	X	X	X
Transform functions:			
GS/GICALL Gps/GpiCallSegment	X	X	X

Figure 13 (Part 2 of 5). Major Elements of the Presentation Interface

Language Element	MVS	VM	OS/2
GS/GISCT Gps/GpiSetModelTransform	X	X	X
GS/GISORG Gps/GpiSetSegmentOrigin	X	X	X
GS/GISTFM Gps/GpiSetSegment TransformMatrix	X	X	
GS/GISAGA Gps/GpiSetSegmentTransform	X	X	X
GS/GISVL Gps/GpiSetViewingLimits	X	X	X
GS/GIUWIN Gps/GpiSetUniformWindow	X	X	X
GS/GIWIN Gps/GpiSetWindow	X	X	X
GS/GIQAGA Gps/GpiQuerySegment Transform	X	X	
GS/GIQORG Gps/GpiQuerySegmentOrigin	X	X	X
GS/GIQSVL Gps/GpiQuerySegmentViewing Limits	X	X	X
GS/GIQTFM Gps/GpiQuerySegmentTransform Matrix	X	X	
GS/GIQWIN Gps/GpiQueryWindow	X	X	X
General attribute and control functions:			
GSAM GpiSetAttrMode	X	X	X
GSCLR GpiClearGraphicsField	X	X	
GSDEFE GpiEndDefaults	X	X	X
GSDEFS GpiBeginDefaults	X	X	X
GSFLD GpiDefGraphicsField	X	X	X
GSPOP GpiPop (restore attributes)	X	X	X
GSQAM GpiQueryAttrMode	X	X	X
GSQFLD GpiQueryGraphicsField	X	X	X
Correlation and boundary functions:			
GS/GICORS Gps/GpiCorrelateStructure	X	X	X
GSTAG GpiSetTag	X	X	X
GSQTAG GpiQueryTag	X	X	X
Segment manipulation functions:			
GSSATI GpiSetInitialSegmentAttrs	X	X	X
GSSATS GpiSetSegmentAttrs	X	X	X
GSSCLS GpiCloseSegment	X	X	X
GSSDEL GpiDeleteSegment	X	X	X
GSSEG GpiOpenSegment	X	X	X
GSSPRI GpiSetSegmentPriority	X	X	X
GSQATI GpiQueryInitialSegmentAttrs	X	X	X
GSQATS GpiQuerySegmentAttrs	X	X	X
GSQPRI GpiQuerySegmentPriority	X	X	X
Metafile support:			
GSGET GpiGetGraphicsData	X	X	

Figure 13 (Part 3 of 5). Major Elements of the Presentation Interface

Language Element	MVS	VM	OS/2
GSGETE GpiEndGetGraphicsData	X	X	
GSGETS GpiBeginGetGraphicsData	X	X	
GSPUT GpiPutGraphicsData	X	X	
Picture exchange:			
Picture exchange	X	X	X
GSLOAD GpiLoadSegments	X	X	
GSSAVE GpiSaveSegments	X	X	
Windows (specialized windows and controls for Common User Access, including text and entry fields):			
Environment management:			
WMBEEP WinBeep			X
WMDISP WinDispatchMsg			X
WMDWP WinDefWindowProc			X
WMGET WinGetMsg			X
WMINIT WinInitialize			X
WMMINT WinSetMsgInterest			X
WMPEEK WinPeekMsg			X
WMPOST WinPostMsg			X
WMRCL WinRegisterClass			X
WMSCUP WinSetCursorPos			X
WMSCUR WinSetCursor			X
WMSEND WinSendMsg			X
WMSHCU WinShowCursor			X
WMSLE WinSetLastError			X
WMSSYS WinSetSysValue			X
WMTERM WinTerminate			X
WMWAIT WinWaitMsg			X
WMQCLI WinQueryClassInfo			X
WMQCLN WinQueryClassName			X
WMQCUP WinQueryCursorPos			X
WMQLE WinQueryLastError			X
WMQSYS WinQuerySysValue			X
WMQVER WinQueryVersion			X
Device management:			
WDCLS DevCloseDC (device context)			X
WDCRDC WinCreateWindowDC			X
WDOPEN DevOpenDC			X
WDQCAP DevQueryCaps			X

Figure 13 (Part 4 of 5). Major Elements of the Presentation Interface

Language Element	MVS	VM	OS/2
Window management:			
WICRT WinCreateWindow			X
WIDEL WinDestroyWindow			X
WIEWU WinEnableWindowUpdate			X
WISCRL WinScrollWindow			X
WISHOW WinShowWindow			X
WISMPS WinSetMultipleWindowPos			X
WISPOS WinSetWindowPos			X
WISTXT WinSetWindowText			X
WIQVIS WinIsWindowVisible			X
WIQRCT WinQueryWindowRect			X
WIQTXT WinQueryWindowText			X
Presentation space management:			
WPCRT GpiCreatePS (presentation space)			X
WPDEL GpiDestroyPS			X
WPQRY GpiQueryPS			X
Dialog and menu support:			
WCCRD WinCreateDlg			X
WCCRDB WinDlgBox			X
WCCRM WinCreateMenu			X
WCDELD WinDismissDlg			X
Utility functions:			
WUCOMP WinCompareStrings			X
WULOW WinLower (lowercase)			X
WUPPER WinUpper (uppercase)			X
WUQALP WinIsAlpha			X
Programming languages:			
COBOL (GSCP, ...)	X	X	X
FORTRAN (GSCP, ...)	X	X	X
C (GpiSetCurrentPosition, ...)			X

Figure 13 (Part 5 of 5). Major Elements of the Presentation Interface

Query Interface

Query and report writing services allow one to compose queries of a relational data base and to create reports based on the answers.

Query/report writing is an interactive service. Using an easy, menu-interaction approach, users can access and summarize information, and then format the results. This means that rather than having to write a program to produce answers, a person can merely create and run a query. Results can be obtained much more quickly.

The functions are easy to learn and use. The command set is simple, so one need know only a few commands to start becoming productive. Furthermore, online assistance guides users so they can make their requests more easily.

Query and reporting facilities are also available to applications through a program-to-program call interface. Applications can thus build and manipulate queries, procedures, and report specifications. Results can also be stored and then accessed by other applications.

Query can benefit a variety of users — from data processing experts to those who know nothing about data processing techniques. Query lets users concentrate on what they are trying to do, rather than how to do it.

The Interface for Systems Application Architecture

The interface specification for query is based on an extension of the interfaces found in today's QMF — the Query Management Facility familiar to System/370 users.

For more information on the contents of the query commands, see the table on the following pages.

The table below lists the interface elements currently in the query interface for Systems Application Architecture.

The table indicates (with an X) which systems already have an IBM licensed program announced or available that implements a particular element.

On MVS and VM, the Query Management Facility products (5668-AAA and 5668-721, respectively) provide the query interface, with some exceptions. An asterisk (*) indicates interface elements that will change in or be added to the current QMF. QMF intends to provide this implementation by December, 1988. The program call interface element (indicated by **) is planned to be described in the Systems Application Architecture query interface reference manual (to be available third quarter 1987), and is based on the command interface available in QMF today.

On the Personal Computer, the implementation will be provided by the IBM Operating System/2 Query Manager.

Interface Elements	MVS	VM	OS/2
Program Call			
Dialog Interface Select Service	**	**	
Commands			
Erase:			
Query	*	*	X
Form (report definition)	*	*	X
Procedure	*	*	X
Table	*	*	X
Export:			
Query	*	*	
Form (report definition)	*	*	
Procedure	*	*	
Table	*	*	X
Import:			
Query	*	*	
Form (report definition)	*	*	
Procedure	*	*	
Table	*	*	X
Print:			
Query	*	*	X
Form (report definition)	*	*	X
Procedure	*	*	X
Report	*	*	X

Figure 14 (Part 1 of 3). Major Elements of the Query Interface

Interface Elements	MVS	VM	OS/2
Run:			
Query	*	*	X
Procedure	*	*	X
Save:			
Table data	*	*	X
Exported Objects			
Query	X	X	
Procedure	X	X	
Table	X	X	X
Form (report definition)	X	X	
Query Capabilities			
SQL Selection	X	X	X
Prompted Query			
Table selection			X
Column selection			X
Row selection			X
Row ordering			X
Duplicates			X
Join columns			X
Reporting Capabilities			
Page:			
Heading text	X	X	X
Footing text	X	X	X
Blank lines before/after text	X	X	X
Text alignment	X	X	X
Numbering of pages	X	X	X
Column:			
Spacing	X	X	X
Width	X	X	X
Heading editing	X	X	X
Heading separators	X	X	X
Data editing	X	X	X
Aggregation:			
Average	X	X	X
Count	X	X	X
Maximum	X	X	X
Minimum	X	X	X
First	X	X	X
Last	X	X	X

Figure 14 (Part 2 of 3). Major Elements of the Query Interface

Interface Elements	MVS	VM	OS/2
Sum	X	X	X
Breaks on Columns:			
Six levels	X	X	X
Heading text	X	X	X
Footing text	X	X	X
Text alignment	X	X	X
Blank lines before/after text	X	X	X
Summary separators	X	X	X
Final summary:			
Final text	X	X	X
Text alignment	X	X	X
Blank lines before text	X	X	X
Summary separators	X	X	X
Detail Lines:			
Line spacing	X	X	X
Variables:			
Page	X	X	X
Date	X	X	X
Time	X	X	X
Print destination	X	X	X

Figure 14 (Part 3 of 3). Major Elements of the Query Interface

File Number
S370-20

GC26-4341-0

Printed in U.S.A.