Technical
Report

Santa Teresa
Laboratory
San Jose, CA

IBM

TR

INTERACTIVE PROGRAMMING IN A MULTIFACETED ENVIRONMENT:
The APL2 Connection to IBM Program Products

by Harlan Crowder
Dick Dunbar

May 1986

TR 03.288

May 1986
TR 03.288

Interactive Programming in a Multifaceted Environment

The APL2 Connection to IBM Program Products

Harlan Crowder
IBM Santa Teresa Laboratory

Dick Dunbar
IBM Almaden Research Center

General Products Division
Santa Teresa Laboratory
San Jose, California

The illustrations in this report were created using Interactive Presentation Graphics, Version 2, and Graphical Data Display Manager, Interactive Chart Utility running with APL2. Text and graphics were integrated using the Document Composition Facility. The report was set in Helvetica type and produced on the IBM 4250 printer.

Interactive Programming in a Multifaceted Environment

The APL2 Connection to IBM Program Products

Application programs written in APL2 have access to a variety of IBM program products. These products provide services to APL2 applications, including user dialog management, graphics, relational database control and management, and high-performance processing.

This monograph briefly describes the IBM program products that complement APL2, and explains how they are used in the APL interactive programming environment.

Reprints of this report may be obtained by writing Harlan Crowder at the following address:

IBM Corporation
Department M30
555 Bailey Avenue
San Jose, CA 95150
USA

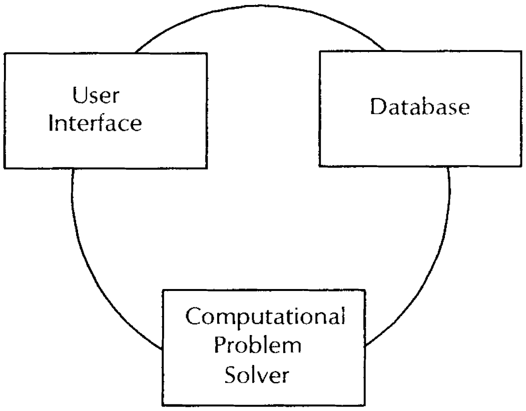
Introduction

The APL2 program product offers a variety of fully supported interfaces to other IBM products in the VM/CMS, MVS/TSO, and MVS/XA operating system environments. These include the Graphical Data Display Manager (GDDM) for user dialogs, device control, and graphics; the Interactive System Productivity Facility (ISPF) for user interface management; IBM DATABASE 2 (DB2) and Structured Query Language/Data System (SQL/DS) for relational database access and management; VS FORTRAN and IBM S/370 assembler language for high-performance processing and access to existing compiled subroutine libraries; and, under VM/CMS, the VM/SP System Product Interpreter (REXX) for host-system support, string processing, and general procedural language programming.

This monograph gives a brief introduction to these various IBM products, and describes how they are used in the APL2 programming environment to complement and extend the power and versatility of the APL2 language and system.

The Application System Triad

Most computer application systems have three main components: the user interface, the database, and the computational problem solver components.



The ***User Interface*** component is that part of the application system that manages the system's interactions with the user; it can involve the use of command processors, menu drivers, and graphics. Various facilities in the APL2 language and system are specifically intended for constructing user interface components; these include the APL2 Session Manager, the FORMAT primitive, and the various input/output modes. Some applications, however, require a more versatile and robust user interface than can be conveniently constructed in APL. The section below on the User Interface describes the use of GDDM and ISPF in APL applications.

The ***Database*** component manages the storage and access of problem data associated with the application. In APL, the traditional database component is the workspace, a reservoir for both data and programs. In APL2, relational data is handled as general arrays in a natural and easy way. The Database section below describes the use of the IBM relational database management systems, DB2 and SQL/DS, in the APL2 environment.

The ***Computational Problem Solver*** component is that part of an application program that performs the computationally intensive or sophisticated portion of solution processing. Until recently, APL applications were limited because the problem solver component could be coded in APL only. The Names Association facility of APL2 allows APL programs to access external routines coded in S/370 assembler language, FORTRAN, and, under VM/CMS, REXX. The use of external routines will be described in the Computational Problem Solver section.

The User Interface

This section describes the use of GDDM and ISPF for building the user interface component of APL application systems.

APL2 and GDDM

The GDDM program product is a format manager that processes both graphics and alphanumerics on display devices, printers, and plotters. The major component of GDDM is a set of functions for drawing pictures and controlling text. GDDM operates in the VM/CMS, MVS/TSO, and MVS/XA environments. For more general information, see *GDDM General Information*, IBM form number GC33-0100. For more technical information, see *GDDM Application Programming Guide*, IBM form number SC33-0148.

In addition to the GDDM Base product, GDDM supports a variety of separately orderable features. For example, the Presentation Graphics Feature (PGF) provides a set of functions for producing business and engineering charts and graphs; included in the PGF is the Interactive Chart Utility (ICU) that allows the easy construction of basic plots and charts in an interactive mode. The Interactive Map Definition (IMD) provides a way to create screen images that define the format of data to be displayed and processed; these images can be saved in files for later access and use by the application program.

Using GDDM with APL2

The APL2 shared variable interface to GDDM allows data, in the form of APL arrays, to be sent directly to GDDM for processing. The GDMX workspace, distributed as part of the APL2 program product, is a simple, easy-to-use functional interface to GDDM that takes advantage of APL2 general arrays to formulate and manage data for graphics and user dialog applications. For example, the GDDM function GSCOL is used to set the color of subsequent graphic operations on display devices. Using GDMX, the APL statement to change the GDDM color to red is

```
        'GSCOL' GDMX 2
```

The left argument to GDMX is the name of the GDDM function; the right argument is the operand for the GDDM function (in this example, 2 is the GDDM color code for red).

Using the Interactive Chart Utility (ICU) with APL2

The ICU is an optional component of GDDM that allows quick and easy production of charts and business graphics. Operated from a series of menus and help screens, it is particularly useful for people with limited computer experience. The following chart types can be constructed using the ICU:

bar charts
histograms
line graphs
pie charts
polar charts
scatter plots
surface charts
tower charts
Venn diagrams

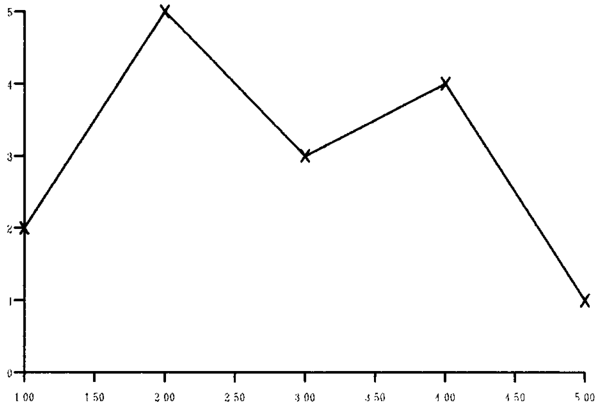
For more information, see *GDDM Interactive Chart Utility User's Guide*, IBM form number SC33-0111.

In APL2, ICU is invoked using the special CHART call of the GDDM interface. This general facility allows a versatile application programming interface to the ICU; the user can easily tailor the ICU invocation to fit specific needs.

As an example, we consider a very simple interface that takes a vector or matrix argument and produces a simple plot for a vector, and multiple plots for rows of a matrix. Using this interface in an APL function called *CHART*, the APL expression

```
      CHART 2 5 3 4 1
```

produces the following ICU picture:

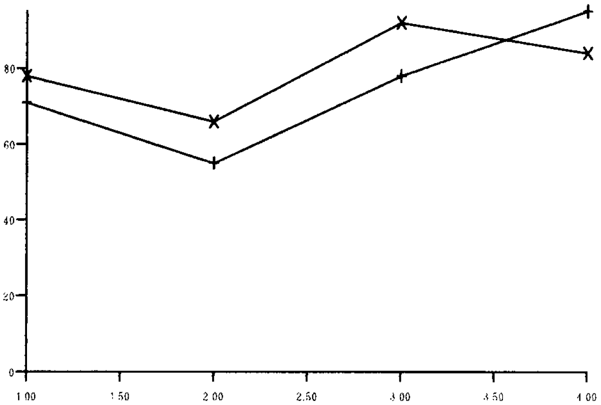


As a more interesting example, suppose we have sales data for two products, for four quarters of the year, expressed as a two-by-four matrix *SALES*. The sequence

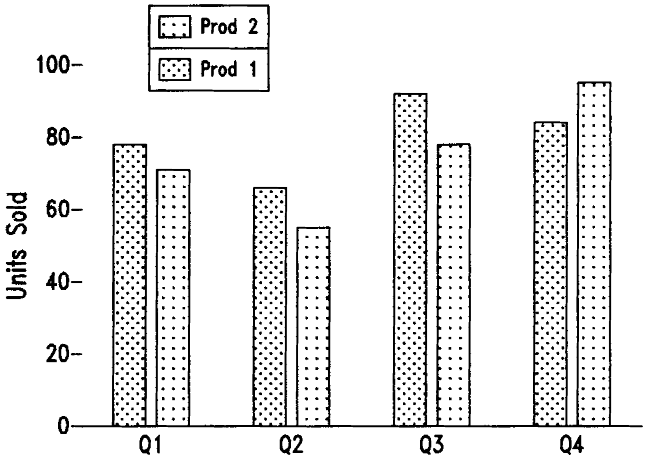
```
      SALES
78 66 92 84
71 55 78 95

      CHART SALES
```

produces the following ICU plot:



Manipulating this plot using the interactive facilities provided by ICU produces this chart:



The ICU, in conjunction with APL2, is a powerful tool for quickly producing graphic charts for data comparison and analysis, and provides the ability to manipulate the pictures to produce good business and presentation graphics.

Using the Interactive Map Definition (IMD) with APL2

The IMD allows the design and management of terminal screens and panels for APL applications; once designed, these images are stored in files for use by the application. Because these images are processed and formatted at the time they are produced, the application is relieved of the burden of processing screen formats at execution time; the result is simpler program code for screen processing and management, and significantly faster execution time. The GDMX function interface to GDDM can be used in APL2 applications to manage IMD screens and associated data.

For more information about the use of the IMD, see *GDDM Interactive Map Definition User's Guide*, IBM form number SC33-0154.

APL2 and ISPF

The ISPF program product is a set of executable routines that may be invoked from application systems ISPF provides the ability to execute special interactive programs called *dialogs* that provide interactive control for applications. ISPF operates in conjunction with APL2 in the VM/CMS, MVS/TSO, and MVS/XA environments.

The available services that use ISPF dialogs include

- Identifying various choices of processing routines available in an application.
- Invoking a requested routine, based on the user's choice from within a dialog.
- Prompting the user for data entry.
- Processing user data into application work areas.
- Checking data to verify that it is appropriate for the application.
- If the data is not appropriate for the application, identifying the error to the user and prompting for reentry.
- Providing online documentation, consisting of messages and tutorial displays, to help the user in processing an application.

For more information about ISPF, see *Interactive System Productivity Facility General Information*, IBM form number GC34-2078.

Using ISPF with APL2

APL2 and ISPF communicate through a shared variable interface, Auxiliary Processor 317 (AP317). This interface allows full access to ISPF services from within an APL2 application. A distributed workspace named ISPEXEC, provided with the ISPF product, contains functions for aiding in the use of ISPF with APL2.

A unique feature of ISPF running with APL2 is the ISPF ability to use APL2 as a subsystem. In this mode of operation, ISPF dialogs can pass APL statements to the APL2 environment for execution and obtain results back from APL2 for display or additional processing.

For more information about using ISPF in conjunction with APL2, see *ISPF Dialog Management Services*, IBM form number SC34-2173.

The Database

This Section describes the use of DB2 and SQL/DS for building the database component of APL2 application systems.

APL2 and SQL

DB2 and SQL/DS are the IBM relational database management systems. APL2 uses the SQL interface to communicate with SQL/DS running in the VM/SP environment and with DB2 running in the MVS/TSO and MVS/XA environments.

DB2 and SQL/DS both use and manage data stored as relational tables. Because tables are simple and familiar (telephone books, airline schedules, and bank statements are all table structures), most people can understand and use them easily. For example, a collection of employee data could be stored as the following table:

EMPLOYEE
========

ID	NAME	INT	DEPT	YEARS	SALARY
113	ADAMS	SA	D01	12	36000
104	BANKS	JA	D04	15	35000
107	CROW	PJ	D02	6	24000
106	DEAN	RA	D02	12	38000
108	EATON	FA	D03	18	40000
109	FARR	JJ	D01	25	50000
103	GALVIN	JE	D04	5	27000
110	HARVEY	HP	D04	23	45000
101	INGRAM	MD	D01	2	18000
114	JACKSON	MA	D02	1	16000
102	KAHAN	BA	D03	6	32000
111	LAMAR	WJ	D02	21	45000
105	MULVEY	JS	D04	3	21000
112	NELSON	AB	D04	7	32000

Application programs communicate with DB2 and SQL/DS using the Structured Query Language (SQL). SQL is a nonprocedural language; users specify what they want to do, not how to do it. The same language is used for all operations on relational tables, including definition, retrieval, and manipulation.

As an example, the following SQL statement could be used to process the above EMPLOYEE table to obtain the names, department, and salary of all employees with salaries less than 25,000.

```
SELECT NAME, DEPT, SALARY
  FROM EMPLOYEE
  WHERE SALARY < 25000
```

Using DB2 and SQL/DS with APL2

In APL2, communication with SQL/DS and DB2 are handled by the SQL Auxiliary Processor, AP127. This shared variable interface allows APL2 applications to pass SQL statements, in the form of APL character vectors or matrices, to the database systems, and receive tables from these systems in the form of APL2 arrays containing both character and numeric data. For example, the SQL statement above, applied to the EMPLOYEE table, would produce an APL2 matrix of the form:

```
CROW     D02  24000
INGRAM   D01  18000
JACKSON  D02  16000
MULVEY   D04  21000
```

The SQL distributed workspace, supplied with the APL2 program product, contains a set of programs for aiding the use of DB2 and SQL/DS from APL application programs. The workspace contains a variety of functions for communicating with the database systems via AP127:

- Data access functions, which pass SQL requests to AP127.
- User support functions, which create common sequences of requests and pass them to AP127.
- Task control functions, which allow management of the APL2/SQL interface environment.
- A defined operator *UNTIL*, which creates a derived function that processes a stack of requests to AP127.

For more information on the database management systems, see *SQL/Data System General Information*, IBM form number GH24-5012, and *IBM DATABASE 2 General Information*, IBM form number GC26-4073. For more information about the APL2/SQL interface, see *APL2 Programming: Using Structured Query Language (SQL)*, IBM form number SH20-9217.

APL2 Relational Applications

The APL2/SQL interface, combined with the ability to easily represent relational data as APL2 arrays, makes APL2 a natural adjunct to DB2 and SQL/DS. The following are potential application areas of APL2 in a relational environment.

Interactive relational application systems -- The ease with which APL2 can create, access, and manipulate relational data makes it a good candidate for implementing interactive relational application systems. The SQL interface, combined with the APL2 interfaces to GDDM and ISPF, provides a powerful and versatile tool for the interactive analysis and management of relational data.

Relational data model design -- A major obstacle to the use of relational database systems is the design and implementation of a data model for a particular application. The data model design phase usually requires experimentation and analysis to arrive at the correct set of relational tables for the problem at hand. Because prototype systems are easy to design and implement in APL, it is the ideal tool for this job.

Small multi-user applications -- APL2 provides a unique set of facilities for building multi-user application systems. The user-to-user shared variable interface facility of APL2 allows applications to communicate directly and asynchronously among different virtual machines (in VM/CMS) or TSO address spaces (in MVS/TSO). This facility allows the easy construction of multi-user relational applications, using a single-user SQL interface server, accessible from multiple users.

The Computational Problem Solver

This section introduces the facility in APL2 that allows APL applications to access external programs written in S/370 assembler language, FORTRAN, and, under VM/CMS, REXX.

The Name Association Facility

The APL2 Name Association facility allows portions of APL2 application systems to be written in FORTRAN and S/370 assembler language. These compiled language functions are used in the application the same as if they had been written in the APL2 programming language. This powerful facility offers a variety of advantages for production APL2 applications:

Use of existing programs and subroutine libraries -- Existing procedures written in FORTRAN or assembler can now be used directly, without modification, in APL2 applications; they do not need to be translated into APL. This means that programs from compiled subroutine libraries can now be used like APL functions in APL2 applications.

Improved execution performance -- Computationally intensive portions of applications may become the bottleneck that limits either the capacity or performance of the application. The Name Association facility allows these bottleneck portions to be written in compiled code. Because external objects are syntactically equivalent to APL objects that they replace, changing the reference to APL objects in the workspace to the external name reference is sufficient; the remainder of the application is not effected.

Easier maintenance of shared programs -- Shared programs are crucial when applications require that everyone use the same code. For example, some applications require functions that control access to critical resources, such as files or communication facilities. By their nature, these functions are subject to periodic modification. They are good candidates for external objects because an external function that is centrally accessible is easier to modify than the equivalent function distributed in a number of separate workspaces.

Access to the IBM 3090 Vector Facility -- APL2 applications can use the IBM 3090 Vector Facility by calling FORTRAN subroutines compiled with VS FORTRAN Version 2. This allows APL2 to exploit the latest in high-performance processing technology for a variety of applications areas.

The Name Association facility also allows APL2 under VM/CMS to call functions written in the REXX programming language. REXX has a variety of facilities for accessing information about the host system and other application subsystems; this information is now easily available to APL2 applications. REXX also has language features that aid in string handling and parsing; APL2 applications that require string processing can take advantage of these REXX functions.

For more information about the APL2 Name Association facility, see *APL2 Programming: System Services Reference* (Version 1, Release 2), IBM form number SH20-9218. For more information about FORTRAN, see *VS FORTRAN Version 2 Programming Guide*, IBM form number SC26-4222. For more information about REXX, see *VM/SP System Product Interpreter Reference*, IBM form number SC24-5238.

APL2 and VS FORTRAN

We demonstrate the use of FORTRAN subroutines in APL2 with a simple example. The following FORTRAN program, named SDF, is used to calculate the standard deviation of a list of real numbers:

```
      SUBROUTINE SDF(S,N,X)
C   COMPUTE STANDARD DEVIATION
C   OF N NUMBERS X
      REAL*8 X(N),S,A
      INTEGER*4 N
      A=0.
      DO 10 I=1,N
   10 A=A+X(I)
      A=A/N
      S=0.
      DO 20 I=1,N
   20 S=S+(X(I)-A)**2
      S=DSQRT(S/N)
      RETURN
      END
```

The following sequence shows how SDF is used in APL2:

```
⍝ MAKE SDF KNOWN TO APL2...

      3 11 ⎕NA 'SDF'
1

⍝ A TYPICAL LIST OF NUMBERS...

      Q1
5 11 3 24 8

⍝ APPLICATION OF SDF...

      SDF (0 (⍴Q1) Q1)
7.414
```

The first APL2 statement uses the Name Association system function *⎕NA* to make the FORTRAN subroutine SDF known to APL2. When SDF is applied to an input list of numbers, its argument is a 3-item array corresponding to the three parameters of SDF: a place-holder for the result, the length of the input list, and the input list itself. We can use SDF in a more natural APL2 style by embedding it in an APL2 function. The following function *SD*, given a list of numbers, constructs the required argument and applies SDF to compute the standard deviation:

```
     ∇
[0]    Z←SD X
[1]   ⍝ APL COVER FUNCTION FOR
[2]   ⍝ FORTRAN ROUTINE SDF
[3]    Z←SDF (0 (⍴X) X)
     ∇
```

SD can be used in various ways consistent with APL2 syntax; the actual standard deviation calculation is performed in FORTRAN:

```
      Q1
5 11 3 24 8

      SD Q1
7.414

      Q2
17 3 8 11
      Q3
3 4 3 6
      Q4
8 10 9 2.

⍝ APPLY SD TO EACH ITEM
⍝     OF A NESTED ARRAY...

      SD¨ Q1 Q2 Q3 Q4
7.414  5.068  1.225  3.112
```

This combination of APL-FORTRAN hybrid functions leads naturally to a programming style that takes advantage of the strong points of both languages: APL for versatility and ease-of-use, and FORTRAN for processing power.

APL2 and REXX

APL2 can execute REXX functions in three forms:

- Primitive REXX functions that are part of the REXX interpreter.
- REXX programs contained in files on disk.
- REXX functions that are represented as character arrays in the APL workspace.

The function USERID is a primitive REXX function; it returns the user's VM/CMS user identification. The following sequence shows its use in APL2:

```
      3 10 ⎕NA 'USERID'    ⍝ Make USERID known to APL2
1

      USERID ⍳0
CROWDER
```

The REXX function USERID requires an empty argument; in REXX, the call would be `USERID()`. In APL2, USERID is called with an empty array argument.

The following example shows the use of dynamic REXX execution where the REXX function is derived from a character array in the workspace. It uses the special REXX interpreter built-in function ∆EXEC that is supplied with the APL2 program product. In this example, the derived REXX function returns the first character string token that follows a left parenthesis in its argument string.

```
      3 10 ⎕NA '∆EXEC'    ⍝ MAKE ∆EXEC KNOWN TO APL2
1

      R←'ARG PARMS ''('' FIRST REST' 'RETURN FIRST'

      R ∆EXEC 'PARM1 PARM2 ( OPT1 OPT2 OPT3'
OPT1
```

The use of REXX in conjunction with APL2 significantly extends the power and versatility of APL for a variety of applications.

Conclusion

The APL2 program product ability to access and use the services and capabilities of other IBM program products makes APL2 a powerful tool for system integration. In this monograph, we have briefly illustrated how APL2 interacts with various IBM products for constructing the user interface, database, and computational problem solver components of application systems.

Acknowledgment

Special thanks to Betty Faith and Sheryl Winton for their critical reading of this paper; their suggestions improved both the style and content.