Programming Hooks 101: Introduction

by Rick Marshall

Much of VA FileMan's success as a database management system grows out of its embracing the M design philosophy of opening doors rather than closing them. Just as M does not protect you from erasing your database, so it also permits you to program structured or not, functionally or logically, "spaghetti" code or Object Oriented Programming System (OOPS). Similarly, rather than adopt a single database model and enforce it, the FileMan developers have tried to permit the database designer to pick and choose elements from any database model as seems appropriate to solve the problem at hand. The key to maintaining this level of flexibility has always been the use of FileMan's programming hooks.

This is the first in a series of articles that will examine the programming hooks in FileMan. This article presents a list of these programming hooks. Future articles will drill into specific areas to examine how to use each hook and what variables can be relied upon within them. Broad and detailed understanding of when and how to use these hooks will distinguish novices from FileMan experts, and will increase your programming productivity.

What Is a Programming Hook?

Although FileMan has a vast number of features to build and use databases, most of these features are not hooks. For this series of articles, a programming hook is defined as a significant point in the sequence of events that make up a standard database activity. At each of these points, application developers or database designers can insert M code of their own.

Programming hooks let users weave into the fabric of database activities threads of their own to alter the flow of FileMan's basic activities. Unlike other features with strictly defined capabilities, hooks have undefined capabilities. They throw the doors wide open to every capability of the M programming language. Some hooks should have a wide range of possible features, while others must be extremely restricted. These articles will spell out the qualities of each hook.

The Locations of Hooks

Here is a complete list of the programming hooks in VA FileMan. In the next article, we'll begin to explore the effective uses of these hooks.

Reader

Reader Input Transform:

```
$P(DIR(0),"^",3,999)
```

Executable ?-Help:

```
$P(DIR("?"),"^",2,999)
```

Executable ??-Help:

```
$P(DIR("??"),"^",2,999)
```

Entry Execute Statement For Help Frame

Exit Execute Statement For Help Frame

Selection

Special Look-up:

```
^DD(filenumber,0,"DIC")
```

Pre-Lookup Transform:

```
^DD(filenumber,.01,7.5)
```

Selection Screen:

```
DIC("S") Parameter To DIC
```

File Screen:

```
^DD(filenumber,0,"SCR")
```

MUMPS Identifier:

```
^DD(filenumber,0,"ID","WRITE")
```

Selection Identifier:

```
DIC("W") Parameter To DIC
```

Selection Input Script:

```
DIC("DR") Parameter To DIC
```

Post-Action:

```
^DD(filenumber,0,"ACT")
```

Input

Adding a Record

LAYGO Input Script:

```
DIC("DR") Parameter To
FILE^DICN
```

LAYGO Screen:

```
^DD(filenumber,.01,"LAYGO",
#,0)
```

Editing a Field

Pointer/Set of Codes Screen:

```
^DD(filenumber,fieldnumber,
12.1)
```

Input Transform:

```
$P(^DD(filenumber,fieldnumber,
0),"^",5,999)
```

- Input Transform Prompt
- (Optional) Pattern Match (in X) Prompt

Executable Help Prompt
M Cross-references
Trigger Cross-references
Bulletin Cross-references & Servers
Variable Pointer Screens:

- `DIC("V")`

Editing a Record

Edit Field Prompt
Input Script:
```
DR Parameter To DIE
```
Input Template

Editing Every Word-Processing Field

Alternate Editor Activation Code
OK To Run Test
Return To Calling Editor

Deleting a Field

Deletion Screen:
```
^DD(filenumber,fieldnumber,
"DEL",#,0)
```

Deleting a Record

Deletion Screen:
```
^DD(filenumber,.01,"DEL",#,0)
```

ScreenMan Input

Form Pre Action

Form Post Action

Page Pre Action

Page Post Action

Form Data Validation

Block Pre Action

Block Post Action

Field Pre Action

Field Post Action

Field Branching Logic

Field Executable Caption

Field Executable Default

Field Post Action On Change

Output

Displaying a Field

Output Transform Prompt

Displaying a List

```
DIC("S") Parameter To DQ
^DICQ
```

Printing a Field

Word-Processing Field Windows

Searching

Search For Field Prompt
Search Template

Sorting

Sort By Prompt
BY parameter To DIP
Sort Template

Before Printing

Header Prompt
DHD Parameter To DIP
DIOBEG Parameter To DIP

Printing Each Record

Print Field Prompt
DIS Array Parameter To DIP
FLDS Parameter To DIP
Print Template

After Printing Each Record

DHIT Parameter To DIP

After Printing

Trailer (Header Prompt)
DHD Parameter To DIP
DIOEND Parameter To DIP

Data Exporter

File Header
Date Format
File Trailer

Security

Field Audit Condition:
```
^DD(filenumber,fieldnumber,"AX")
```

Installation

Environment Check Routine
Pre-Init After User Commit
Screen To Determine DD Update
Post-Initialization Routine

Version 20 Verified

VA FileMan version 20 was verified July 9, 1993, and released this fall. The key features include: file extract, data export, cancel print jobs, faster %RCR, and $NEXT elimination.

FileMan will create extract files based on existing files. We expect this feature to be useful for creating online archives. It will output data from FileMan files to files in the underlying operating system. Formats can be defined and used, such as SASS, Microsoft Word, and Lotus 1-2-3. See *Software $VIEW* on page 50 for additional information.

Users of Kernel will be able to pick the TaskMan User option to ask their queued jobs to stop running, but it is the responsibility of the application to determine whether they have been asked to stop. FileMan's queued print jobs now check, and will stop if the user asks them to. %XY^%RCR uses $QUERY to vastly increase the speed of array copying. Finally, $NEXT has been removed from all VA FileMan code. New data structures will be created free of $NEXT. Applications should convert their own executable code in the database to use $ORDER instead of $NEXT.

Forward your FileMan questions to the mail group FILEMAN DEVELOPMENT TEAM on the VA's FORUM system, or write to: VAISC6-San Francisco, Suite 600, 301 Howard Street, San Francisco, CA 94105.

Rick Marshall works at the Seattle Development Satellite office of the VA's San Francisco Information Systems Center. He is a member of the FileMan development team, teaches a VA Kernel class at the MTA annual meetings, and is active within the MDC.
