Barcodes in a Printer-Independent Environment

by Ed de Moel and Cheryl Dougherty

This article is adapted from the **Proceedings** *of the 1993 MUMPS Users' Group-Europe meeting in Dublin, Ireland.*

Introduction

In any large computer system, software will have to deal with a mix of hardware: some of the users will be using the equipment that was acquired when the system was initially installed, some users will be using more modern equipment, and some departments can afford to buy more luxurious versions of the various devices that are used around computers, whereas the majority of the users will have to make do with the standard, general purpose, equipment. This is especially true in the case of the Composite Health Care System (CHCS), where some sites have over 1,000 terminals connected to a cluster of computers.

This paper will discuss how, in CHCS, it is becoming possible to print barcoded labels on any printer on the system, without losing the ability to produce labels that contain a fairly complex mix of data.

Possible Solutions

Since the CHCS system is based on the FileMan database management system, at first it was considered that all we would need was a print-template with some moderately complex output transformations. While the FileMan print-templates certainly are able to solve a large number of problems, there are some special issues to the printing of graphical information in general that make this approach less desirable. The special demands related to the sensitivity of certain barcode reading devices only make this more evident.

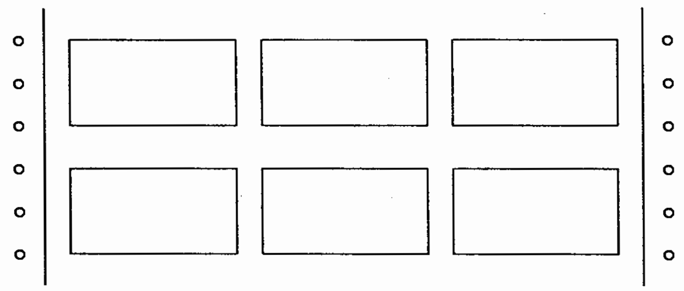
Furthermore, a solution was sought that would allow the end user to specify the layout of the labels to be printed. Any approach that would imply that end users would have to specify MUMPS code, even if only a "simple" function call to perform a data conversion, would create an insurmountable threshold for the people who would be using the system.

So, a new project was started that would allow the end user to define the layout of labels, and the appearance of the various data items on those labels. Key players in this project were the Lab Software Development Team, who created the data structures, and the Tools Team, who made the software to link these data structures and created the device drivers to make things work in a device-independent way.

Complications

The first complication to be solved was how to specify which data element from the CHCS database is to appear where on a label. At this moment, the CHCS database contains roughly four hundred FileMan files, with about sixteen thousand data fields. No single user can be expected to quickly identify a specific field in a data dictionary, and even when the user knows which field is intended, it often is a pointed-to-field (e.g., Name of Health Care Provider), and the end user would have to specify which pointer to the Health Care Provider file is intended (doctor who prescribes a drug, pharmacy assistant who fills the order, nurse who administers the drug, etcetera). This problem was overcome by building a set of tables, that allows for the definition of nicknames for fields in the database. For each functionality group in CHCS, a separate nickname table can be built. These tables have to be built in close cooperation between the customer and the various application teams, in order to ensure that the table links the correct data field to the nickname that is provided.

The next complication has to do with the nature of adhesive labels. Depending on the labels used, the forms that are fed through the printer can look quite different. One common format is the form with rows of three labels, eight rows per page:

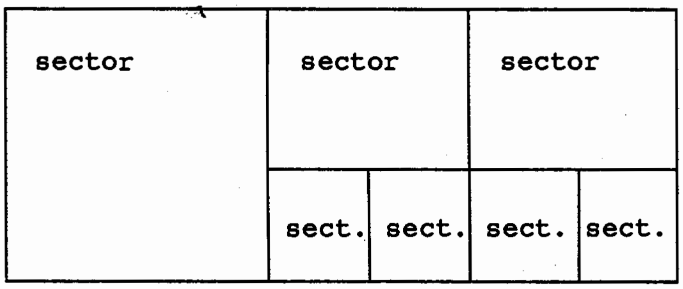


If all printers were capable of printing one label, and then moving the print-head upward and to the right to start printing the next, this wouldn't be anything special, but some printers can and some printers can not move "upward," so some special software would be required to make labels print, regardless of the peculiarities of the forms on which the labels are provided.

In order to make this work, a number of data structures were created:

- A description of the physical forms;
- A description of the placement of the labels on a form; and
- A description of the layout of a single label.

In order to make it easier for the end user to align groups of data on the label, a label can be organized in *sectors*, e.g., the most popular laboratory label would look like:



Within each sector, information can be placed:

- Fixed text; or
- Values taken from the database.

And, independent of its source, information can be represented in a number of ways. If the representation is shown as text, it can appear:

- Normal;
- **Boldface**;
- Underlined;
- *Italics*;
- 10 characters per inch;
- 12 characters per inch;
- 16 characters per inch (actually: 16.5 characters per inch, but let's say 16 for the purpose of this discussion); or
- Double-wide characters.

If the representation is as barcodes, it can appear:

- In any of the known barcode syntaxes (currently "2 of 5," "3 of 9" or "codabar");
- In "picket" orientation;
- In "ladder" orientation;
- Relative bar width can be varied based on ad hoc specifications;
- Horizontal and vertical dimensions can be varied based on ad hoc specifications.

Without going into too much detail about the barcodes themselves, the various syntaxes are different definitions of narrow and wide lines that form the barcodes. The "2 of 5" syntax uses only five lines per character, but can encode only digits; the "3 of 9" syntax, on the other hand, needs nine lines for each character, but is capable of encoding the complete ASCII set. A code with less lines is easier scanned, and creates less of a hazard for misinterpretation; a code with more lines allows for more detailed information to be encoded. Each application will make its own decision as to which syntax is more appropriate for its needs.

Solution

The CHCS barcode system ended up as the combination of three sets of utility routines:

- One set of routines converts an arbitrary text into a description of a set of patterns, depending on the syntax of the desired barcode. For each supported barcode syntax one such routine is created.
- One set of routines converts a barcode pattern description in the sequence of characters and printer control codes that will produce the desired barcode. For each supported printer (family) one such routine is created.
- One set of routines manipulates the information in the various tables that describe the layout of a label. Application-specific procedures define which fields appear where on the labels. Procedures prepared by the Tools Team interpret that information and call the procedures in the other two sets of utility routines to produce the labels on any printer.

The general sequence of function calls is:

- The desired output device is selected (in the standard way).
- A function is called that checks whether the desired label can be printed on the selected printer. If the current printer does not support all the required features, the application (not the tool) decides whether or not to proceed.

- For each label to be produced in the current run, set up local variables and call the procedure that enters one label into the print-buffer.
- When one horizontal "stroke" of labels is completed, the Tools utility routine will automatically cause these to be printed and the print-buffer will be cleared.
- When all labels are processed, a final call to clear the print-buffer is required.
- And finally, the output device needs to be closed.

While a label is processed, texts and graphics are inserted into a print-buffer, which resides in a MUMPS global variable: `^UTILITY("XBARUTL2",Y,X,. . . .)`. Because all information to be printed is stored here, organized by the vertical coordinate, the Tools software can organize the multiple labels for one horizontal "stroke," before printing any of these.

Because the control sequences that need to be sent to produce the graphics are also entered into this buffer, at the moment that the printing actually occurs, there are no problems left that result from having to "go back upwards" to print more text after a barcode has been printed (or vice versa).

Conclusion

The approach to link a small number of user-definable tables with a small number of device oriented utility routines provides the end user with the capability to define layout for labels on an ad-hoc basis. The available routines are modular to the level that:

- For each new printer family that is added to the system, one new routine will need to be written;
- For each new barcode-syntax that is added to the system, one new routine will need to be written;
- For each new data item that is needed on a label, a nickname and link to the actual data in the database will need to be set up.

This does leave the end user somewhat dependent on the software developers. This compromise, however, relieves the end user from the need for detailed knowledge of the construction of the database, and relieves the application programmers from the need to create call-back protocols to their software when the printer-specific software needs application specific procedures to obtain information to be barcoded. **M**

The paper upon which this article is based was given at the MUMPS Users' Group - Europe meeting in Dublin, 1993. The original was included in the *Proceedings of the Eighteenth Annual Meeting of the MUMPS Users' Group - Europe*. Permission to use this material was granted by that organization, which has since changed its name to M Technology Association-Europe.

Ed de Moel has been a member of MDCC-Europe since 1980 and a member of the MUMPS Development Committee (MDC) since 1987. Currently, Ed is vice-chair of the MDC. He has taught and worked with MUMPS at universities in the Netherlands since 1976. At Science Applications International Corporation (SAIC), he has worked on a system for archiving medical information on optical storage and various other software tools. His current activities involve software tools for performance measuring within the context of CHCS (Composite Health Care System), SAIC's hospital information system.

Cheryl Dougherty has worked with SAIC as a database administrator and as an M programmer. She was the Tools Team Leader from 1992 to 1993 when the barcode tools were produced.

M/MIIS Programmer/Analyst, Washington, DC

Consider full-time employment as an M/MIIS Programmer/Analyst at Sibley Memorial Hospital.
We offer:

- Quiet, residential location
- Small shop
- Free on-site parking
- Excellent benefits
- Relocation
- No travel.

For consideration, send resume to Karen Robertson Keck or call her collect at 202-537-4750 weekdays between 7:30 A.M. and 4:00 P.M.

Sibley Memorial Hospital
5255 Loughboro Road, N.W.
Washington, DC 20016

Equal Opportunity Employer