PROJECT REPORT

An Early Aramaic (Syriac) Word Processor under DOS.

SUNIL SIVANAND AND DANIEL BENJAMIN

The Vision

The project to develop an Aramaic Word Processor (AWP) started in Kuwait, sometime in 1985, when Daniel Benjamin, an expatriate native Aramaic speaker, was inspired by the introduction of Arabic word processors on personal computers. They started appearing in the corporate market as an alternative to typewriters, shortly after IBM launched the PC-AT model. Aramaic is the ancient Semitic language of Aram, a region in central Syria. It was the language spoken by Jesus, and the one used to write many of the original books of the Bible.

Daniel Benjamin had, a few years earlier, started work on an Aramaic typewriter, working with Hermes, the well-known typewriter manufacturer. He was hopeful of getting some of the Aramaic-speaking people scattered around the world to buy the new typewriters. However, the project had to be abandoned, as there were not enough purchase commitments to satisfy the minimum order quantity imposed by Hermes. But all was not lost. During his planning and working with Hermes, Daniel had done a lot of groundwork, preparing precision drawings of the intricate Aramaic fonts using high-resolution images, and had preserved almost all his work.

Before Daniel moved to Kuwait, he lived in Iraq, where he'd gained knowledge of the conventional die-cast Aramaic types from his father. The die-cast types were made when Daniel's elderly uncle, who had spent time in India as a deacon, was exposed to the techniques of preparing such types there. He worked with professional die-casters in India to prepare a type-set for Aramaic. These were used in what was possibly the first attempt to print Aramaic for the Church of the East (Nestorian Church). A set of these types was later obtained from the Nestorian Church in India by Daniel's father who used them to print Aramaic liturgical books.

Armed with all this experience, the type drawings prepared for the Hermes typewriter project, and a serious desire to preserve Aramaic, Daniel approached Sunil Sivanand, his friend and colleague at Kuwait National Petroleum Company (KNPC). Sunil, an Indian with no knowledge of Aramaic, worked in the company's Information Technology Division, where the technology infrastructure was largely built around main-frame and mini-computers. Despite the launch of personal computers several years ago, KNPC at the time had not seen a need to invest in them. PCs were still expensive and rare in Kuwait and considered far from affordable for home use. But, in 1983, foreseeing the IBM PC as a major trendsetter in technology, Sunil had purchased an 8080-based IBM PC from the UK and was using it at home, in his spare time. Most of his work on the PC was directed to becoming familiar with the technology and building small software applications. Soon, affordable IBM PC clones were becoming available in Kuwait, but few could put them to use except for word processing. Sunil had by then used his knowledge of the programming environment on the PC platform to build custom application software to enhance productivity for small and mid-sized businesses. In 1985, with the earnings from the sale of his custom-developed application software, Sunil invested in an IBM PC-AT clone. By the time Daniel talked to Sunil about his Aramaic word processor project, the first personal computers had just been purchased at KNPC, to be used for Arabic word processing. After Daniel and Sunil discussed the matter in detail, they agreed to work together on the AWP after office hours.

The Technology

When IBM PCs were first launched in 1981, they were equipped with either a monochrome or colour adapter to drive the appropriate type of monitor. These adapters came with fixed factory-loaded font and character sets burned into their read-only memory. The next generation of offerings for graphic adapters came in 1984, in the form of an Extended Graphic Adapter (EGA), with the launch of the IBM PC-AT. These had advanced features, particularly because they had a rewriteable memory that allowed for bit-maps of custom fonts to be loaded by a program running on the PC, and used to display text. EGAs could also

functionally serve the same purpose of the old display adapters, and were hence backward compatible and useable with the same monitors. The EGA, however, had a relatively short life, because within a few years, it was superseded by the Vector Graphic Array (VGA). The VGA survives even today, largely because of its design that allows screen resolution to be increased by adding more memory and processing power to the video adapter card.

The ability to download and use custom fonts on a printer had indeed been available long before the EGA appeared, and it solved the problem of having custom fonts on the computer's display. IBM's PC-DOS, and variants like MS-DOS, were the predominant (and possibly the only) operating systems available for IBM and other "compatible" PCs.

The software development environment was identified as Turbo C, as it offered the flexibility to address all the needs, including the ability to address low-level DOS calls to interact with the EGA display adapter.

The Awp Modules and Design

Once work on the project started, it was clear to Sunil that the following software components would be required to have the AWP working:

- Screen Font Editor
- Display Font Loader
- Printer Font Editor
- Printer Font Loader
- Keystroke Mapping
- Aramaic Context Handler and Text Direction Handling
- The Word Processor
- Screen and Printer Font Bit Maps

The AWP was designed to be modular, to allow for ease of software and font development. It would be ready for use only when all these components, explained below, were created and made available:

Screen Font Editor, which allowed the creation and refinement of bit map images for the computer's display, and to assign ASCII codes for invoking and displaying the bit-map image for every corresponding character.

Display Font Loader: A low-level program that could use the data saved by the screen font editor, to prepare a sequence of font upload instructions to load the bit-maps for the custom fonts into the EGA's memory

Printer Font Editor, which allowed the creation and refinement of bit map images which matched the font resolution for the printer, and to assign ASCII codes for invoking and displaying the bit-map image for every corresponding Aramaic character. This was to be developed separately, as printer bit-map resolutions were quite different from those available for the EGA. Also, it had to be flexible in order to support different kinds of printers and to allow for higher output quality.

Printer Font Loader: A low-level program that could use the data saved by the printer font editor, to prepare a sequence of font-upload instructions to load the bit-maps for the custom fonts into a printer's memory.

Keystroke Mapping: A module that maps a keystroke (or combination) to the associated ASCII character.

Aramaic Context Handler and Text Direction: Aramaic, unlike English, French and other languages that use the Latin script, is "context sensitive", meaning that the shape of letters change depending on its position in the word. This is determined by a set of predefined "rules". Moreover, Aramaic is written and read from right to left (unlike Latin and most other scripts) and this required the text to be displayed with cursor movements reversed.

The Word Processor: Essentially a text editor that allows for simple blocks of text to be typed, printed, edited. This module also carries out various file-management operations like Save, Open, etc. In this first version, there was no ability to format fonts, or paragraphs, as would be typically seen in word processors.

Screen and Printer Fonts Bit Maps: This comprises a database holding the exact shapes of each character, the alphabet and various shape variants in the Aramaic language. The shapes are held as bit-maps for each coded item. There is one set for representation on the screen, that has been prepared to correspond with the screen resolution and another set that has been prepared to correspond with the resolution required for a laser printer.

ܠܒܝܬܘܡ ܢܝܫܢܐ

ܦܝܛܝܟ ܟܘܡܝ ܕܝܝܗ ܝܥܩܘܒ ܡܥܠܡܢܐ: 58 1958

ܓܒ ܡܬܠܝܟ ܗܘܝܐ ܒܐܘܪܫܐ ܒܥܒܝܬܐ ܕܓܡܦܪ ܢܗܪܐ
ܕܒܝܡ ܘܕܝܐ ܒܪܗܒܓܘܬܐ ܒܚܦܘܠ ܛܘܪܐ
ܫܕܝ ܦܣܘܝ ܢܟܠ ܢܕ ܐܘܪܫܐ ܢܩܒ ܓܒܕܐ
ܠܒܘܒܘ ܦܢܐ ܘܢܣܝܗ ܒܣܒܩܐ ܦܠܠܒܡ ܒܕܕܐ

ܠܕܘܡܝܐ ܒܠܒܬܐ ܕܐܒܝ ܢܟܠ ܠܘܚܡܝܗ ܟܠ ܓܕܒܝ ܠܒ
ܘܟܠܐ ܠܗܐܘܬܐ ܒܒܝܬܐ ܕܕܝܠܗ ܝܗܒܝܬܝ ܠܒ
ܝܟܠܐ ܠܐܩܬܗ ܒܥܒܕܐ ܒܥܒܕܐ ܢܩܒ ܢܟܦܝ ܠܒ
ܒܢܬܐ ܘܓܠܬܐ ܦܡܓܢܡܒ ܠܗ ܝܗܦܥܕܝ ܠܒ

ܒܒܓܒܘܗ ܟܠܐ ܡܗܢܘܬܐ ܒܠ ܩܕܫܬܐ
ܘܟܠܐ ܡܗܣܐ ܠܗ ܢܗܪܐ ܕܗܒܓܐ ܐܘܟܠܐ ܒܓܒܬܐ
ܦܡܝܬܐ ܕܦܓܕܐ ܒܦܪܒܓܘܗ ܒܥܒ ܡܒ ܣܘܬܐ
ܘܦܠܦܐ ܒܝܡܫܐ ܕܝܡܫܐ ܕܢܦܕܐ ܟܠ ܣܘܐ ܐܬܐ

ܡܢ ܓܕܡܘܬܐ ܘܓܠܬܐ ܕܠܝܓܗ ܒܐܡܒܘܬܐ
ܢܒܬܗ ܡܕܡܒܬܐ ܘܡܣܟܬܐ ܕܡܬܐ ܒܓܒܓܘܬܐ
ܐܘܟܠܐ ܒܟܠܐ ܐܒܐ ܠܗ ܒܬܫܐ ܦܓܗܒܠܘܬܐ
ܦܓܓܠ ܕܩܩܐ ܡܦܓܐ ܠܦܘܬܐ ܒܗܘܒܣܘܬܐ

ܒܓܠܗܘ ܫܒܬܗ ܟܠܐ ܓܒܝܡ ܦܘܡܗ ܐܘܒܐܘܬܐ
ܟܠܐ ܒܐܒܦܦܐ ܐܘܟܠܐ ܒܝܠܩܐ ܕܗܕܒܡܘܬܐ
ܐܒܕܗ ܦܥܒܗܐ ܠܟܓܒܒ ܐܘܪܫܐ ܒܓܗܒܠܘܬܐ
ܒܐܟܠܐ ܘܕܦܐ ܕܗܘܒܦܢ ܫܢܬܗ ܕܟܠ ܒܠܒܘܬܐ

The Chronology

Since resources were limited, the approach taken was to build these modular items separately. Also, a perfected set of **Screen and Printer Fonts Bit Maps** would take time to develop as the process was inherently iterative and time-consuming. For convenience, the font and software development were done in separate locations as Daniel had the ability and knowledge to work on the fonts, while Sunil had the software-development skills. Hence it was decided to first develop the screen and printer font editors. Once these font editors were ready, Daniel set out to prepare the fonts, first for the screen and then for a low-resolution 24-pin matrix printer. The matrix printer quality did not appear to be very appealing, so it was decided to invest in a Canon laser printer that supported downloadable fonts.

The font editors for the screen and the printer were more or less similar in their operation and working. These were essential to commence the project and hence were the first to be built. They provided the user with an enlarged matrix, in the required bit-map resolution, on the computer's screen and allowed for each bit in the matrix to be turned on and off, thus producing the required shape for each character. The prepared bit-maps could be saved and retrieved for editing to refine the shape and quality. These font editors were used by Daniel to prepare the shape for each character, as it should visually appear on the screen and printed document.

Once the first versions of these font editors were in use, they were enhanced to add the ability to prepare a bit-map for download to the respective device and print or display a sample character to check out its shape as finally presented on the printer and the screen. To start with, a 24-pin dot matrix printer was used. As the project progressed, it was clear that to have professional printing quality, the resolution provided by the 24-pin matrix printer was not adequate. Hence a similar font editor for the Canon laser printer was developed. Preparation of fonts for the 24-pin matrix printer was quickly abandoned and efforts were directed at preparing the full set of fonts in high resolution. No attempt was made to use scalable fonts, as these were unknown at the time, so the AWP had only a single-sized set of fonts for display and printing.

As the fonts were being prepared, the programming and development work on the other components continued. The fonts were all stored as 8-bit ASCII characters, since Unicode was unknown at the time. There were challenges with staying within the restriction of 255 characters, as some of these had use as screen and printer control characters and the full 255 character set could not be used. This meant having to compromise, and drop some of Aramaic's context-sensitive shapes and ligatures.

The context analyzer was largely driven by a set of rules defined in tables holding data, with some of the more complex exceptions to the rules being hard coded into the routine. When the fonts were ready, and the first cut of the program modules also ready, Daniel and Sunil had to spend more time working together to refine the code and give final shapes to the context analyzer and the text editor. When their AWP's first version was ready, Daniel started testing it by preparing and typing larger blocks of text. At this stage, there were still many special situations that needed to be addressed, like text that appeared at the end of a line, or words that required breaks between lines or pages. These were corrected in the final version of the project. Daniel then left Kuwait to relocate to Chicago, where he continued using the AWP to produce a book, an anthology of his own Aramaic poems. It was also used to prepare other Aramaic content for circulation. Since Daniel had all the tools to make refinements to the fonts, he could continuously develop and refine the AWP's printed quality. Daniel Benjamin later adopted his experiences with the first Aramaic word processor to use more generic multi-lingual word processors that started appearing with the advancement of technology.