گزارش کامپیوتر

ماهنامهٔ انجمن انفورماتیک ایران

انجمن انفورماتیک ایران

INFORMATICS SOCIETY OF IRAN

ویژه‌نامهٔ زمستان ۱۳۷۱

Farsi Computing Review

April 1993

Editor: Amin Mohadjer

Published by: Informatics Society of Iran &
BRAIN Computer Systems Group

Farsi Computing Review

Editor: Amin Mohadjer

Farsi Computing Review (FCR) is a quarterly journal published in English by Informatics Society of Iran in co-operation with BRAIN Computer Systems Group.

Subscription enquiries inside Iran and Middle East should be directly addressed to: Amin Mohadjer, P.O. Box 14455-161, Tehran, Iran. Tel/Fax: +98 21 980-102.

Distributor of Farsi Computing Review outside of Iran (except Middle East) is EleTek Systems. For further information, contact: c/o Dr. Nasser Modiri, 1107 Second Ave., Suite 613, Redwood City, Ca 94063, USA. Tel/Fax: (415) 369-8967. e-mail: `nasser@unet.net.com`

Anyone interested in acting as editor for a special issue of Farsi Computing Review devoted to a particular topic is invited to contact the editor. Technical papers appearing in this issue and opinions expressed in contributions are to be construed as those of the individual author rather than the position of FCR.

Advertising Sales: Inquiries concerning advertising should be forwarded to Advertising Sales, BRAIN Computer Systems Group, P.O. Box 14455-161, (21) 980-102.

This issue is partly prepared using Ventura Publisher and printed on a HP Laser III printer. The full contents of Farsi Computing Review is also available in electronic format.

Editorial and all other correspondence and enquiries should be addressed to Amin Mohadjer, Editor, P.O. Box 14455-161, Tehran, Iran.

Farsi Computing Review

A Quarterly Journal on Farsi Computing
Volume 1 ∗ Number 1 ∗ April 1993

Editor: Amin Mohadjer

In this issue:

Published by Informatics Society of Iran in Co-operation with
BRAIN Computer Systems Group
P.O. Box 14455-161, Tehran

Editorial

The idea of coming out with a journal on the subject of Farsi Computing was shaped in 1991. At then, we published BRAIN newsletter and were determined to establish some special interest groups with newsletters of their own to focus on more specific and detailed topics. The topic of Farsi Computing was the most sounding for two reasons; it was an original idea which could also attract the attention of researchers in Arabic countries, and also many problems were around in adopting computer systems to our native language 'Farsi'. Unfortunately, soon we found out that publishing magazines and running special interest groups in Iran is not as easy as we thought.

BRAIN newsletter was ceased to publish after four issues and our ideas were turned to dust. However, we found an alternative. We could continue working under the cover of another magazine and the best possible choice was *Computer Report*, the magazine of Informatics Society of Iran.

The first edition of Farsi Computing Review which you have in hand features five original papers. In their paper "A Persian Codeset for Information Interchange", Saeid Kazemi and Mammad M. Zadeh propose a standard for Farsi code set. The other paper, "Farsi GUIs" written by Dr. Abdulah Al-Salamah from King Saud University was supposed to be printed in next issue but with a little bit of luck we could manage its inclusion in this issue. Behzad Torabi of Iran System wrote us "An Approach to UNIX Farsi System" and finally me and my colleague, Mahmood R. Ziaei wrote "Farsi Computing" and "Farsi Character Set: The Problem of Multiple Standards", respectively.

Not all available Farsi character sets and keyboard layouts are included in appendices to this edition. That's also true when said about catalog of Farsi and English publications on Farsi Computing. However, I hope that in next editions those which have left out will got their place.

I would like to thank several people who provided me with enthusiasm, encouragement, and suggestions that helped to shape Farsi Computing Review. I am especially indebted to Saeed Vahid, Mehdi Alipour, Nasser Modiri, Anoosh Hosseini, and people in ISI for their support.

I welcome your comments and opinions regarding materials printed in this edition and am looking forward to having your contribution in this journal.

Amin Mohadjer
Editor
February 27, 1993

A Persian Codeset for Information Interchange*

Mammad M. Zadeh محمّد محسن‌زاده
Saied Kazemi سعید کاظمی
Parsis, Inc.† شرکت پارسیس

August 5, 1992

Abstract

A standard character encoding method is essential for the effective exchange of information among computer systems and peripherals. The most widespread encoding for English characters is the 7-bit American Standard Code for Information Interchange (ASCII). Additional standards have been developed by various international committees to support other languages. But to date, there is no well accepted standard character encoding method for Persian. Although in recent years some progress has been made, no true standard has emerged. Traditionally, Persian has been considered as an extension to existing Arabic codesets, but this approach has several drawbacks. In this paper we will discuss the strengths and shortcomings of some existing codesets as well as outlining the requirements for Persian information interchange. In conclusion, we will propose a codeset for Persian information interchange, and an encoding scheme for Persian fonts that meet all of the outlined requirements.

*Copyright ©1992 by Parsis, Inc. Permission to use, copy, and distribute this paper is hereby granted, provided that this copyright notice appears in all copies.
†Parsis, Inc., 1736 Franklin St., Suite B, Santa Monica, CA 90404, USA. Tel:(310) 917-2650 Fax:(310) 917-2653

1 Introduction

For decades now, the 7-bit American Standard Code for Information Interchange (ASCII) has been the standard character encoding method for communication between computer hardware and software. As its name suggests, ASCII is an *American* standard for English; it does not meet the requirements of other languages.

In this paper we will explore limitations of existing Persian codesets, outline the requirements of a complete Persian character encoding scheme, and propose a *Farsi Standard Code for Information Interchange* (called FSCII) as well as a *Farsi Standard Font Encoding* (called FSFE).

1.1 Requirements

The criteria used in specifying our requirements were based on the ability to enter, store, process, display, and print any Persian document, no matter how complicated it may be. We use the term "Persian document" in its broadest sense: we mean such things as newspaper articles, scientific reports, educational text books, government documents, poetry and classic literature. Thus, a complete Persian codeset should be capable of encoding any published Persian text. The requirements were:

1. Ability to include all Persian characters.
2. Ability to coexist with the ASCII codeset in a document and be "properly" displayed without requiring any structure on the document in the form of embedded commands.
3. Ability to include all Arabic characters for representing syntactically correct Arabic text.
4. Ability to efficiently handle special Persian requirements such as non-spacing marks (حرکات), punctuation marks, numeric characters, mathematical symbols, and compound words.

It should be emphasized that, if these requirements were met, it would be possible to take advantage of many existing text-based applications in Persian.

1.2 Extensions to ASCII Codeset

The first 7-bit International Standards Organization (ISO) codeset supported certain Western European languages (e.g., French) by replacing several characters in the ASCII codeset. The problem with this scheme was that only a subset of all necessary characters could be included in the encoding space of 7 bits (128 characters). Moreover, since the 7-bit ISO standard modified the ASCII codeset, systems that employed the ASCII codeset were not compatible with this new ISO standard.

To overcome the shortcomings of a 7-bit encoding scheme, ISO extended the ASCII codeset to 8 bits (256 characters) and defined a complete set of characters needed to support most Western European languages. This standard is known as the ISO 8859 8-bit standard codeset that is now widely used in Europe. Other 8-bit standards are being developed by ISO to support the Greek, Cyrillic, Arabic and Hebrew languages [1].

1.3 Codesets for Multilingual Environments

For almost all computers, a character is an 8-bit quantity. This means that a character can have one of 256 different code values. As mentioned earlier, ASCII has standardized the meaning of code values 0 to 127. If the codeset of a given language can fit within the range 128 to 255, the codeset of that language can easily coexist with ASCII in a document.

In general, issues involving multilingual documents can be grouped into two categories:

1. Documents that contain 7-bit ASCII and another 7-bit codeset in the range 128 to 255.
2. Documents that contain 7-bit ASCII and two or more 7-bit or 8-bit codesets.

Documents in the first category are relatively easy to deal with. That is, there would be no ambiguity in identifying the correct codeset of any given character in the document (code values 0-127 are ASCII and 128-255 are the other codeset).

Documents in the second category are more difficult to deal with since code values 128-255 of different codesets overlap and cause ambiguity. To illustrate this point, consider the case when ISO 8859 and the 8-bit Arabic standard coexist in a document. In such a case, a code value of 155 would be impossible to identify. That is, one could not tell which codeset it belongs to.

Problems of multilingual support are now being addressed by ANSI X3L2, ISO Draft International Standard 10646, and the Unicode Consortium. The Unicode Consortium, for example, is developing a fixed-width 16-bit codeset that represents all languages of the world without any ambiguity [3]. But until a standard 16-bit codeset is adopted by the computer industry, an effective method for identifying the codeset of any given character in a multilingual document is needed. A common method has been the insertion of special *embedded commands* [2] within the document. For example, if a 2-byte character sequence such as 255,17 can be interpreted as a switch to codeset number 17, the following sequence of characters:

255	17	89	65	37	255	24	65	255	2	91	72

can be interpreted as a 6-character segment of a document of which the first three characters belong to codeset 17, the fourth character belongs to codeset 24, and the last two characters belong to codeset 2. Therefore, when the document is scanned serially, there is no ambiguity as to which codeset each character belongs.

Embedded commands, however, have two major drawbacks that make their usage undesirable:

1. For random accesses in a document, the insertion of embedded commands is of no value unless the "vicinity" of the accessed location is scanned to find and interpret the commands. For many applications, such as data base management systems, finding and interpreting the vicinity of an accessed location is not acceptable.
2. Insertion of embedded commands in a document forces any application operating on the document to know the "syntax" of the commands and be able to parse them. This means that none of the existing thousands of applications can work with such documents.

Therefore, since the usage of embedded commands severely limits the usability and portability of documents, a properly designed Persian codeset for encoding *bilingual English-Persian* documents should not require the assistance of such commands.

1.4 Existing Persian Codesets

Traditionally, international standards organizations have considered Persian to be an extension of their standard Arabic codesets [3]. Although the Persian character set is a superset of Arabic, there are some key differences that affect the design of a Persian codeset. For instance, Persian characters have a different collation order, and the construction of compound Persian words require two forms for some characters (see Section 2.3).

In recent years, several codesets for Persian characters have been developed. Most of these codesets, however, are ad hoc schemes that have not been published. An exception has been a codeset published by the Iranian Institute of Standards and Industrial Research. This institute has defined the ISIRI 2900 codeset as the standard Persian codeset [9]. The ISIRI 2900 codeset has addressed many issues of the Persian language and has been an important step in finalizing a standard codeset for Persian. However, in its current form, it falls short of being a final standard.

2 The Persian Language

In order to define a codeset for proper representation of Persian text, we have to address some of the more important aspects of the Persian language. These aspects include: flow of text from right to left, a character set composed of Persian and Arabic characters, multiple shapes for each character (depending on its position in a word), non-spacing marks, special punctuation marks, and compound words.

2.1 Direction of Text

Persian texts read from right to left. At first glance, the relevance of the text's direction to a character encoding scheme may not be obvious. However, to coexist with the ASCII codeset in a document (see Section 1.1), we shall show that direction should be an intrinsic property of each character in a bilingual document. This implies that even those characters that have the same shape in English and Persian (such as '.', '+', or even a space ' ') should be included in a Persian codeset.

There has been much discussion about direction as an intrinsic property of characters in several papers and technical reports [2, 4, 6, 7] and the reader is strongly encouraged to read them. However, we will briefly discuss the subject here because of its importance for the design of a Persian codeset.

Let's define text that reads from left to right as *L-text* and text that reads from right to left as *R-text*. The next two lines are examples of L-text and R-text respectively:

This text reads from left to right.

این متن از راست به چپ خوانده میشود.

The problem we are addressing is how to correctly display a line containing both L-text and R-text without inserting embedded commands. To illustrate this problem, we will give a simple example

Typing ﻣ
Typing ﻣﺘ
Typing ﻣﺘﻨ
Typing متن
Typing متن ﻓ
Typing متن فا
. . .

Figure 1: Example of a reflected segment.

in which we assume that the *current document direction* [4] of our display is *L-mode*, meaning that paragraphs are anchored from the left. Suppose that we want to enter and display the following line:

Typing متن فارسی in English text is fun.

We refer to the order in which characters are entered as the *logical order*. Therefore, the logical order for the above line is:

T	y	p	i	n	g		ﻣ	ﺗ	ﻧ		ﻓ	ﺍ	ﺭ	ﺳ	ﯾ		i	n …

Note that when displaying this line, we don't want to see the text in its logical order; instead, we want to see it in its *visual order*. To maintain correct visual order when encountering R-text, we have to start a *reflected* segment – a segment where characters are pushed in the opposite direction. Figure 1 shows how R-text is reflected in our example. As soon as we encounter the next L-text, the reflected segment ends and characters are displayed in the "normal" way.

Therefore, to produce the correct visual order, it is important to recognize L-text and R-text without ambiguity. Since we do not want to impose any structure on a document, the only solution is to recognize these segments by looking at the code for each character. We can thus define *L-characters* and *R-characters* for representing English and Persian characters respectively. Now, it should be evident that we need an R-character for space (' ') in addition to the L-character space of ASCII. If the space between the words "متن" and "فارسی" is changed to an L-character, the visual order of the line becomes:

Typing متن فارسی in English text is fun.

This happens because the L-character space ends the reflected segment "متن", and the next R-character starts the next reflected segment "فارسی".

The above example illustrates why direction must be an intrinsic property of each character and why we must have two different codes for those characters that are represented by similar glyphs in English and Persian.

2.2 The Persian Character Set

With the exception of four extra characters ('پ', 'چ', 'ژ', and 'گ'), the Persian character set is very similar to Arabic. Like Arabic, characters may assume up to four different shapes depending on their position in a word. For example, 'ﻫ', 'ﻬ', 'ﻪ', and 'ه' are different shapes of the same character. The proper shape of a character can be determined automatically during the display of text and need not be stored in a file. The process that determines the shape of each character is sometimes called *contextual analysis*. This process can be efficiently performed by various output devices such as terminals, terminal emulators, windowing systems, and printer preprocessors. Therefore, it is not necessary to include all four shapes of a given character in a Persian codeset.

Persian also employs the Arabic non-spacing marks which act as vowels. The usage of non-spacing marks in Persian is not as widespread as in Arabic. However, they cannot be omitted from a Persian codeset since their use is essential for both indicating the proper pronunciation, and in case Arabic text is included in a Persian document. Classical Persian literature uses a mixture of Persian and Arabic texts. For instance, the first verse (مصراع) in Hafez's book of poetry is entirely in Arabic, as shown below:

«اَلا يا اَيُّهَا السّاقى اَدِرْ كَأساً و ناوِلْها   كه عشق آسان نمود اوّل ولى افتاد مشكلها»

The punctuation marks used in Persian are similar to English, except that guillemets ('«' and '»') are often used as quotation marks.

Persian numerals are different from English but, like English, mathematical expressions are read from left to right. Thus Persian numerals are not R-characters. Numerals, the decimal separator, and other mathematical operators are considered special L-characters (refer to Section 3). The following example illustrates some of these points [6]:

...کلمه جدید از عبارت زیر تعیین میشود:

$$\frac{n-۱}{۲m}+\frac{۲(n-۱)(n-۲)}{۳m^{۲}}+\frac{۳(n-۱)(n-۲)}{۴m^{۳}}+\dots$$

که تقریباً مساوی $\frac{log_e(۱-\alpha)}{\alpha}+\frac{۱}{(۱-\alpha)}$ است که در آن $e=۲/۷۱۸۲۸$ میباشد. بنابراین اضافه کردن کلمات جدید با کمک الگوریتم F کار چندان...

2.3 Compound Words and Plurals

Persian differs from Arabic in two major ways: 1) it has numerous compound words, and 2) it uses a suffix to indicate plurals. Compound words are composed of two or more words with no spaces in between. Similarly, Persian words are pluralized by appending the suffix "ها" also with no space. Both these cases pose a particular problem for the contextual analysis of Persian texts.

In case of compound words, contextual analysis should not join adjacent words that compose a compound word. For example, the word "چشم‌انداز", is composed of the words "چشم" and "انداز". The contextual analysis algorithm has to be instructed to use the ending form of 'ـم' (not its middle form 'ـمـ') yielding "چشم‌انداز" (not "چشمانداز"). In case of pluralization, the contextual analysis

algorithm is faced with the same problem. For example, the plural form of "دهکده" is "دهکده‌ها" and not "دهکدهها".

The most elegant solution to support this feature of Persian is to assign two codes for each character that has a different glyph for its middle and ending form. This way, one can explicitly choose the ending form when entering compound words [5]. Codesets that do not distinguish between the middle and ending forms of a character usually employ a special invisible non-joining character to delimit the words in a compound word. Most Arabic codesets take advantage of this non-joining character for composing compound words [3]. This approach may be adequate for Arabic, but is completely impractical for Persian [5] due to the abundance of compound words in Persian.

3 Design Goals

Alphabet: Support for all Persian and Arabic characters is essential. The ISIRI 2900 codeset has left out three Arabic characters: 'إ', 'ة', and 'ي'. As mentioned in Section 2.2, many works of classical Persian and Islamic literatures use extensive Arabic text and any encoding scheme must be able to encode these texts.

Numerals: Numerals in Persian look different from numerals in English. Both numerals and mathematical symbols in Persian are considered special L-characters. They differ from other L-characters (such as English characters) in that they do not end the reflection if they occur inside a reflected R-segment; instead, this special L-segment becomes a nested reflection itself [7]. Therefore, the Persian codeset should also include mathematical symbols. Although mathematical symbols used in Persian resemble those used in English, their glyphs are usually specifically designed to match Persian text and numerals. The ISIRI 2900 codeset does not include the division symbol '÷', and relies on the '/' symbol, whic[illegible] used as the decimal separator in Persian, and not as a division symbol. For example:

۱۰/۲۳ ÷ ۳ = ۳/۴۱

Non-spacing Marks: Non-spacing (diacritical) marks are essential in representing Persian literature and Arabic texts (see Section 2.2). They are needed to show correct pronunciation of words and also to distinguish between different words (with different meanings) that are spelled the same way, but pronounced differently. As an example, consider the words: "گَرد", "گِرد", and "گُرد" which, without non-spacing marks, are all written the same way but have three different pronunciations and three different meanings. Sometimes it is possible to guess the correct pronunciation, and thus the correct meaning of a word from its context in a sentence, but not always. Consider the following example:

کی رفت؟
کِی رفت؟

in which, without the non-spacing mark ('ـَ'), the word "کی" can be pronounced in two different ways, yielding different valid meanings in each sentence. The ISIRI 2900 codeset does not incorporate any non-spacing marks.

Punctuations Marks: A complete set of punctuation marks is essential for representing modern Persian texts. Most punctuation marks in Persian resemble those used in English; except that the comma, the semicolon, and the question mark in Persian are represented by '،', '؛', and '؟' respectively. Although the other punctuation marks look the same as English punctuation marks, we still need a complete set for Persian since, when used in Persian, their direction is from right to left. The ISIRI 2900 codeset has left out most punctuation marks (e.g., the semicolon, the single quotation marks, the guillemets, square brackets, curly braces). The following example is taken from a poem by a famous contemporary poet:

آشیانی بود؛ مسکین در حصار عزلتش محصور؛
آشیان بود آن، که در هم ریخت، ویران کرد، با خود برد...
[آیا هیچ داند باد؟]
(مهدی اخوان‌ثالث، آخر شاهنامه، مرثیه، انتشارات مروارید، صفحه ۱۲۸)

Compound Words: The codeset should aid the contextual analysis algorithm to determine the proper shape of characters for compound Persian words. As described in Section 2.3, we have to assign two codes for each character that has a different middle and ending form.

Collation: The characters in a Persian codeset should be organized in the proper collating order, which is slightly different from Arabic [9].

Bilingual Text: The codeset should be able to support both ASCII and Persian texts in the same document without using embedded commands for switching between languages (see Section 1.3). Therefore, we need two 7-bit codesets arranged into a single 8-bit space such that characters 0-127 are reserved for ASCII and 128-255 are reserved for the Persian codeset. As a result, English and Persian characters can be easily distinguished by their eighth bit. The ISIRI 2900 codeset occupies the same space as ASCII characters, therefore requiring special embedded commands to switch between codesets.

Bidirectional Text: The codeset should define a complete character set for Persian in order for display devices to perform the proper reflection algorithm without using embedded commands (Section 2.1). Therefore, we have to include all numerals, mathematical symbols, punctuation marks, and even the space character in a Persian codeset [7]. The ISIRI 2900 codeset does not define a complete set of R-characters for Persian.

	0x8	0x9	0xA	0xB	0xC	0xD	0xE	0xF
0x0		~	SP	۰	أ	ﭼ	ص	ک
0x1	#	^	!	۱	ؤ	چ	ﺿ	ﮔ
0x2	*	@	"	۲	إ	ﺣ	ض	گ
0x3	_	&	=	۳	ﺋ	ح	ﻃ	ﻟ
0x4	<	َ	ریال	۴	ئ	ﺧ	ط	ل
0x5	>	ِ	،	۵	ﺑ	خ	ﻇ	ﻣ
0x6	{	ٰ	:	۶	ب	د	ظ	م
0x7	}	ً	؟	۷	ﭘ	ذ	ﻋ	ﻧ
0x8	[	ُ	(	۸	پ	ر	ع	ن
0x9	]		)	۹	ة	ز	ﻏ	و
0xA	’	ٍ	×	؛	ﺗ	ژ	غ	ﻫ
0xB	‘	ٌ	+	«	ت	ﺳ	ﻓ	ه
0xC	\		÷	»	ﺛ	س	ف	ﯾ
0xD	|	ّ	−	آ	ث	ﺷ	ﻗ	ی
0xE	%	ْ	.	ا	ﺟ	ش	ق	ي
0xF	$	ٔ	/	ء	ج	ﺻ	ﻛ	

Table 1: The FSCII Codeset.

4 The Proposed Codesets

We propose a new codeset that has two components: the FSCII codeset for information interchange, and the Farsi Standard Font Encoding (FSFE) for encoding character glyphs. Files stored on disk will contain FSCII characters in logical order. Display agents such as terminals, terminal emulators, windowing systems, and printer preprocessors will be responsible for performing contextual analysis on the FSCII input and displaying the proper shape of each character from a given Persian font. In order to standardize the font encoding, we defined FSFE which includes all possible glyphs for Persian characters.

4.1 The Farsi Standard Code for Information Interchange

Table 1 shows the proposed FSCII codeset. It is an 8-bit codeset that assigns codes `0x0` to `0x7F` to the ASCII characters and codes `0x80` (upper left) to `0xFF` (lower right) to Persian characters.

FSCII achieves all of the goals set out in Section 3. Character codes `0x80` to `0x9F` have been assigned to special symbols and non-spacing marks. Character codes `0xA0` to `0xBC` include the right-to-left space character, punctuation marks and mathematical symbols. Finally, `0xBD` to `0xFE` includes all other Persian and Arabic characters. Table 2 illustrates sample usages for some of the

Code	*Name*	*Glyph*	*Examples*
0x96	alef above	ٰ	اسمٰعیل، موسیٰ
0x97	fathatan	ً	مثلاً، عیناً
0x98	dammatan	ٌ	مضافٌالیه، مشبهٌبه
0x99	kasratan	ٍ	اَحَد
0x9F	ya maksureh	ٔ	همهٔ شهر، نامهٔ تو
0xBF	hamza	ء	انشاء
0xC0	hamza on alef	أ	تأمین، جرأت
0xC1	hamza on waw	ؤ	مؤمن، لؤلؤ
0xC2	hamza under alef	إ	إیواء (جا دادن)
0xC3	hamza on yeh	ئ	مسئله، شیئ
0xC9	teh marbuta	ة	صلوة، دائرةالمعارف

Table 2: Sample usage of some FSCII codes.

FSCII characters that may need clarification.

4.2 The Farsi Standard Font Encoding

As we described in Section 2.2, terminals, terminal emulators, windowing systems, and printer preprocessors (filters) must be able to perform contextual analysis on the text and display the final form of a text in the correct visual order. The encoding scheme for display fonts is outlined in Table 3. Since Persian displays accept FSCII characters as input, they can easily recognize L-text and R-text and switch their currently displayed font accordingly.

Notice that codes `0x0` to `0x1F` have not been defined and codes `0x20` to `0x7F` correspond to FSCII codes `0xA0` to `0xFF`. Codes `0x80` to `0x9F` correspond to FSCII codes `0x80` to `0x9F`. Codes `0xA0` to `0xE0` include the additional glyphs. Non-spacing marks assigned from `0xE6` to `0xEF` differ from those defined in `0x96` to `0x9F` in that they are the connected form used by terminals that cannot place non-spacing marks on top of characters [7]. Codes in the last column include ligatures that are commonly used in Persian.

FSCII input (usually from either a file or keyboard) is translated into FSFE for display devices. Remember that the range `0x0` to `0x7F` of FSCII is the same as ASCII. A code in this range is not a Persian character and is not translated into a code in FSFE. A FSCII code in the range `0x80` to `0xFE` is translated into FSFE according to the algorithm shown in Figure 2 (written in the style of the C programming language).

	0x2	0x3	0x4	0x5	0x6	0x7	0x8	0x9	0xA	0xB	0xC	0xD	0xE	0xF
0x0	SP	۰	أ	چـ	ص	ک		˜	ـآ	ـجـ	ـش	ـق	ـي	لا
0x1	!	۱	ؤ	چ	ضـ	گـ	#	^	ـا	ـج	ـصـ	ـکـ		ـلا
0x2	"	۲	إ	حـ	ض	گ	*	@	ـأ	ـچـ	ـص	ـک		لآ
0x3	=	۳	ئـ	ح	طـ	لـ	_	&	ـؤ	ـچ	ـضـ	ـگـ		ـلآ
0x4	ریال	۴	ئ	خـ	ط	ل	<	,	ـإ	ـحـ	ـض	ـگ		لأ
0x5	،	۵	بـ	خ	ظـ	مـ	>	-	ـئـ	ـح	ـطـ	ـلـ		ـلأ
0x6	:	۶	ب	د	ظ	م	{	'	ـئ	ـخـ	ـط	ـل	ـٰ	لإ
0x7	؟	۷	پـ	ذ	عـ	نـ	}	ً	ـبـ	ـخ	ـظـ	ـمـ	ـً	ـلإ
0x8	(	۸	پ	ر	ع	ن	[	ٌ	ـب	ـد	ـظ	ـم	ـٌ	الله
0x9	)	۹	ة	ز	غـ	و	]	ٍ	ـپـ	ـذ	ـعـ	ـنـ	ـٍ	في
0xA	×	؛	تـ	ژ	غ	هـ	’	َ	ـپ	ـر	ـع	ـن	ـَ	ًّ
0xB	+	«	ت	سـ	فـ	ه	‘	ُ	ـة	ـز	ـغـ	ـو	ـُ	ٔ
0xC	÷	»	ثـ	س	ف	یـ	\	ِ	ـتـ	ـژ	ـغ	ـهـ	ـِ	ٍّ
0xD	−	آ	ث	شـ	قـ	ی	|	ّ	ـت	ـسـ	ـفـ	ـه	ـّ	
0xE	.	ا	جـ	ش	ق	ي	%	ْ	ـثـ	ـس	ـف	ـیـ	ـْ	
0xF	/	ء	ج	صـ	کـ		$	ٔ	ـث	ـشـ	ـقـ	ـی	ـٔ	

Table 3: The Persian Standard Font Encoding.

To illustrate a sample translation performed by the algorithm of Figure 2, an example of FSCII and FSFE encodings of the phrase "همهٔ شهر" is shown below (the first row is FSCII and the second row is FSFE):

0xFA ه	0xF5 م	0xFA ه	0x9F ٔ	0xDD ش	0xFA ه	0xD8 ر
0x7A هـ	0xD7 ـمـ	0xDD ـه	0x9F ٔ	0x5D شـ	0xDC ـهـ	0xBA ـر

5 Experience with FSCII and FSFE Codesets

The proposed codesets, FSCII and FSFE, are the direct results of four years of designing and prototyping a comprehensive programming environment for developing Persian applications. This work has been performed on a network of Sun Microsystems SPARCstations running Solaris 1.0 and OpenWindows 2.0 (X11 Windowing System). This particular configuration was chosen because of Solaris's internationalization enhancements to UNIX, its powerful windowing system, and its open architecture.

```
if (current char == farsi) {
        if (previous char != attachable from left)
                current char = first FSFE form;
        else
                current char = middle FSFE form;
}

if (current char == farsi && previous char == farsi) {
        if (current char != attachable from right &&
            previous char != last FSFE form) {
                if (previous char == first FSFE form)
                        previous char = last-by-itself FSFE from;
                else if (previous char == middle FSFE form)
                        previous char = last-attached FSFE from;
        }
}
```

Figure 2: FSCII to FSFE conversion algorithm.

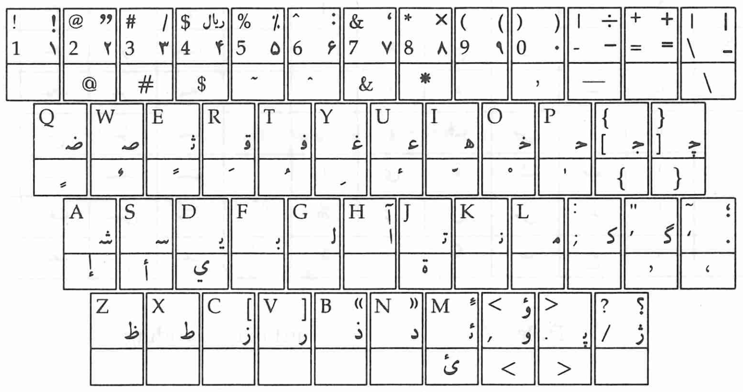
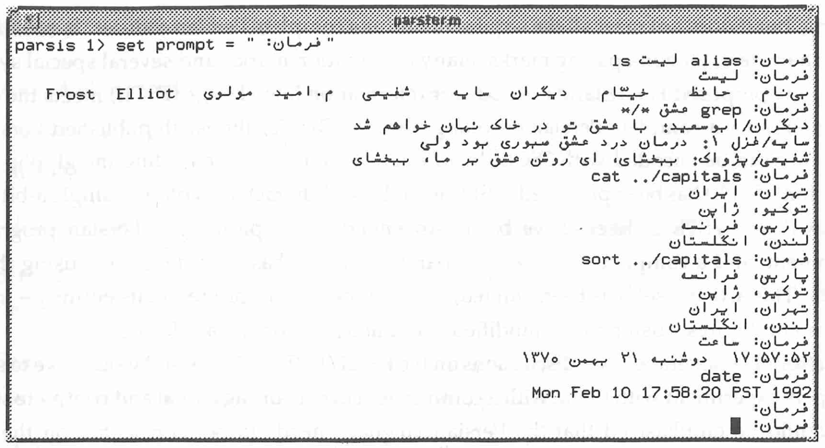


Figure 3: The Persian keyboard layout.

The first step in this effort was the development of *ParsTerm*, a Persian terminal emulator for the X11 Windowing System. Today, advanced windowing systems allow us to implement such terminal emulators entirely in software. ParsTerm runs as an X client and takes control of the keyboard and screen processing. The user can select the current document direction (*L-mode* or *R-mode*), as well as the current input characters (*L-characters* or *R-characters*). If R-character input is selected, all keyboard events are translated according to the Persian keyboard layout (Figure 3). This keyboard layout was based on the ISIRI 2901 standard layout [10] with modifications for new FSCII characters and the SPARCstation keyboard. For example, the 'J' key generates the FSCII code for 'ﺗ' (code 0xCA) when unshifted, 'ﺖ' (code 0xCB) when shifted, and 'ﺓ' (code 0xC9) when holding down a special modifier key (usually Alt).

The screen management code in ParsTerm performs the bidirectional laying out of text on the screen, as explained in Section 2.1. Also, ParsTerm performs contextual analysis on its input and



```
parsis 1> set prompt = " فرمان: "
                                        فرمان: alias لیست ls
                                                  فرمان: لیست
  بی‌نام: حافظ   خیام   دیگران   سایه   شفیعی   م.امید   مولوی   Elliot   Frost
                                            فرمان: grep عشق */*
              دیگران/ابوسعید: با عشق تو در خاک نهان خواهم شد
                        سایه/غزل ۱: درمان درد عشق صبوری بود ولی
              شفیعی/پژواک: ببخشای، ای روشن عشق بر ما، ببخشای
                                        فرمان: cat ../capitals
                                                  تهران، ایران
                                                    توکیو، ژاپن
                                                  پاریس، فرانسه
                                                 لندن، انگلستان
                                       فرمان: sort ../capitals
                                                  پاریس، فرانسه
                                                    توکیو، ژاپن
                                                  تهران، ایران
                                                 لندن، انگلستان
                                                    فرمان: ساعت
                               ۱۷:۵۷:۵۲   دوشنبه ۲۱ بهمن ۱۳۷۰
                                                    فرمان: date
                                  Mon Feb 10 17:58:20 PST 1992
                                                         فرمان:
                                                       فرمان: █
```

Figure 4: A sample ParsTerm session.

thereby converts FSCII to FSFE. As a result, many text-based programs written with no knowledge about right-to-left languages can be used with Persian texts. For example, this document was prepared using the unmodified executables of the standard *vi* and *Emacs* editors. Other applications, such as database management systems, could also be "Persianized" in this way. Figure 4 shows a screen dump of a sample session in ParsTerm.

We also used an enhanced version of the powerful TEX typesetting system [6] that is capable of mixing right-to-left with left-to-right texts for the final typesetting of this article. First, the TEX document was created and edited using *vi*, then the document was passed through a series of filters that performed the contextual analysis, as well as Persian ligature processing, and then it was processed by TEX.

6 Summary and Conclusion

Adoption of a standard encoding scheme for Persian is essential for development of Persian applications and advancement of computer industry in Iran. Such a codeset should include all possible Persian and Arabic characters, as well as numerals, non-spacing marks, and punctuation marks. We have shown that, in order to mix right-to-left text with left-to-right text without using embedded commands, direction has to be an intrinsic property of each character. Therefore, a Persian codeset should also include symbols that are common with the ASCII codeset. In addition, display terminals and windowing systems can perform the contextual analysis on the incoming input stream and choose the correct shape of characters. Moreover, compound words and rules for plurals in Persian make it necessary to include two forms for most characters: one that follows the normal contextual analysis rules, and another that ensures the ending form of that character (regardless of its surrounding characters) to be displayed.

The ISIRI 2900 codeset has taken an important step in defining a standard for Persian character encoding, but it does not meet all the requirements for a complete Persian codeset. It excludes some Arabic characters, all non-spacing marks, many punctuation marks, and several special symbols.

The new proposed Farsi Standard Code for Information Interchange (FSCII) meets the outlined requirements for a complete Persian encoding scheme. That is, almost all published Persian texts in existence can be encoded with FSCII. In addition, a standard for encoding the glyphs within a Persian font (FSFE) has been proposed. FSFE assembles all character glyphs in a single 8-bit codeset. Both FSCII and FSFE codesets have been implemented in a prototyped Persian programming environment, and a complete document preparation system has been developed using these two codesets. This article itself has been entered, stored, typeset, and printed in its entirety – including all tables and figures – using the unmodified executables of existing applications.

We strongly encourage open discussions on the FSCII/FSFE codesets and would like to see other developers experiment with them, with a commitment to expediting a final and complete standard.

It should be emphasized that the Persian language needs to be represented on the various international standards committees in order to address issues specific to Persian and not have it be considered an extension of Arabic.

7 Acknowledgements

We would like to acknowledge the efforts of the Iranian Institute for Standards and Industrial Research for their pioneering work in this area. Also, special thanks to Hafez Dadeh-Negar, Inc., (شرکت حافظ داده‌نگار), for providing the Persian fonts used in typesetting this paper.

References

[1] *Unicode: A New Standard for International Data Sets*. SunWorld, July 1991, Vol. 4(7), pp. 62-68.

[2] Joseph D. Becker. *Multilingual Word Processing*. Scientific American, July 1984, Vol. 251, No. 1, pp. 96-107.

[3] *The Unicode Standard: Worldwide Character Encoding*, Addison-Wesley, 1991.

[4] D. Berry and U. Habusha. Vi.iv, *a Bi-Directional Version of the* vi *Full-Screen Editor*. Electronic Publishing, Vol. 3(2), pp. 3-29.

[5] M. Sanati, M. Dadashzadeh, and M. Dadfar. *Iranian Standard Code for Information Interchange (ISCII)*. Computer Standards and Interfaces 6 (1987) pp. 427-432.

[6] D. E. Knuth and P. MacKay. *Mixing right-to-left texts with left-to-right texts*. TUGboat, Vol. 8(1987) pp. 14-25.

[7] M. Zadeh and S. Kazemi. *ParsTerm: A Persian Terminal Emulator for X Windows*. Parsis, Inc.

[8] M. Zadeh and S. Kazemi. *ParsTEX: A Persian Typesetting System*. Parsis, Inc.

[۹] مؤسسه استاندارد و تحقیقات صنعتی: کُد تبادل اطلاعات در فارسی، شمارهٔ استاندارد ۲۹۰۰، چاپ اوّل، شهریور ۱۳۶۷.

[۱۰] مؤسسه استاندارد و تحقیقات صنعتی: حروف و علائم زبان فارسی روی صفحه کلید کامپیوتر، شمارهٔ استاندارد ۲۹۰۱، چاپ اوّل، شهریور ۱۳۶۷.

This page is intentionally left blank.

Farsi Character Set: The Problem of Multiple Standards

Mahmood Reza Ziaei
BRAIN Computer Systems Group[†]

January 3, 1993

Abstract

Iran's computer community has not yet agreed on a unique Farsi character set for the purpose of information interchange. Different vendors have come with their own character sets which are in no way compatible with each other. This paper explains the reason behind the availability of these multiple standards. The author brings in focus the characteristics of commonly used Farsi code sets and concludes that introduction of a unique standard might not be the one and only solution ahead.

† BRAIN Computer Systems Group, P.O. Box 14455-161, Tehran, Iran.
Tel: (021) 634-279 Fax: (021) 980-102

1 Farsi Standards: The First Step

As yet, there is no widely accepted standard available for Farsi character set. Standard 2900 on Iranian Standard Code for Information Interchange and Standard 2901 on keyboard layout for Farsi language were released in 1989 by Institute of Standards and Industrial Research of Iran and have remained unchanged since then.

The 7-bit code of standard 2900 uses ISO code extension facilities to define a G1 set of 94 Farsi characters. Users can switch between G1 and G0 (English characters) sets with the aid of control characters. Excluding mainframes and larger systems, the time standard 2900 was introduced, most of computers in use were operating in an 8-bit environment, therefore no way to look back and practice standard 2900.

So far, no 8-bit standard has been offered for Farsi information interchange but work is on the progress toward preparing an early draft of this standard. For a more detailed discussion of Farsi character sets see article done by *M. Ziaei* entitled *"Farsi Character Set: The Problem of Multiple Standards"* printed in this issue.

As far as keyboard layout is concerned, major software vendors are agreed on using standard 2901 although *Rayaneh Saz*, *Dadeh Kavi*, and *Microsoft* are insisting on their own way of laying out Farsi keys on the keyboard.

1.1 Farsi Character Sets in Use

Standard 2900 was not well-received by software developers and went obsolete shortly after being released. *Iran System* and *SinaSoft* Farsi character sets were and still are the most widely-used code sets today. Rayaneh Saz, Dadeh Kavi, and *Alis* Farsi character sets are also used [see appendix A]. SinaSoft character set is supporting that of Iran System, but not vice versa. Neither are supporting standard 2900 or those of other companies.

Not bringing in light the disagreement over the order of characters in Farsi code sets, software vendors are not agreed on the number of characters that should be placed is such a table. Since a Farsi letter might have up to four cases (English has just two), many argue that all possible four shapes of letters such as ع should be included in the character set.

Another company, Microsoft, has entered the fray with its new character sets called *Microsoft Farsi code* and *Microsoft Farsi font*. Microsoft Farsi code is used for the purpose of storing, retrieving, and transmission of Farsi data while Microsoft Farsi font is used for the representation of Farsi data on the screens.

The availability of so many Farsi character sets has led to incompatibility between data files created by various applications. Users who are running a software of company *x*, are forced to have *x* character set in place. Surprisingly, software vendors do not seem much concerned in supporting file formats of their rivals through conversion tables or other utility programs.

1.2 The Order of Farsi Characters

Despite major differences, almost all Farsi standards are sharing a common point; they do not touch the English characters placed in the first half of the ASCII table. Since the order in which Farsi

characters are placed in the code set is a determining factor when it comes to sorting data, these standards have managed it somehow to keep the order of characters in the table just as the order of letters in Farsi alphabet.

Since codes 176 to 223 are reserved for tabulation characters, some developers prefer not to replace characters residing in these places with Farsi characters. This, in spite of promoting the compatibility and portability of code set, brings to attention the problem of space limits for the placement of Farsi characters. Both Iran System and SinaSoft character sets have placed Farsi characters out of this boundary.

1.3 The Forthcoming 8-bit Standard

A committee is currently investigating the technical sides of an 8-bit Farsi character set and is supposed to announce a new standard for the replacement of standard 2900. The new standard aims to solve the problem of incompatibility between existing standards and make it possible for users to easily transfers files from one application to another. Apparently, the new standard does not intend to add one more new character set to already crowded list of code sets. Instead, it sets a protocol for data transfer and interchange. The new standard will allow software developers to continue building applications based on their own character sets but encourages them to use its proposed character set for the purpose of data transfer and interchange.

The Farsi data can be displayed, stored, and retrieved as before but when it comes to transferring data between computers in a network or between users through communication lines, applications should use a filter to convert the data to the specified standard format prior to dispatching.

On the receiver side, another filter is used to convert data to the original format or any other format preferred by user. Although this would work fine as far as DOS and Windows environments are concerned, Unix users might face some difficulties since they can hook up to numerous system resources and transfer data without going through passage filters.

2 Generating Farsi Fonts for Text Mode

In order to display Farsi information on the screen, Farsi characters should be designed and placed in the character set. In Farsi systems, mathematical, Latin, and graphical characters residing in the second half of the ASCII character set (128..255) are being replaced with Farsi characters. Depending on the type of graphics adaptor in place, this could be done through software or hardware means.

2.1 Farsi Fonts

Each font is designed as a meaningful matrix pattern which is copied to an area of memory. Since on a typical IBM PC-compatible VGA adaptor, characters are constrained to a 9- by 16-pixel matrix cell, it is difficult to design a good-looking font for Farsi characters in text mode. The problem is that the vertical and horizontal ranges of Farsi and Arabic characters are beyond the range of this space. The existence of diacritical marks in Farsi is adding to this complexity.

To come up with a solution, some of Farsi software developers including SinaSoft have allotted two character spaces to displaying a single Farsi character such as ت.

The allowed matrix cell dimensions for CGA, Hercules, and EGA are 8x8, 8x14, and 8x16 respectively. To facilitate the process of designing fonts, some vendors have developed font editors which enable users to design new fonts or modify existing fonts easily. Font editors save designated fonts to a file in a special format. Later, this file is used to download fonts into computer.

2.2 Graphics Adaptors

Only EGA, VGA or higher graphics adaptors are capable of downloading user-defined fonts through use of device drivers or software routines.

To download Farsi characters into MDA, Hercules, and CGA graphics adaptors, one should have its ROM BIOS re-programmed with an EPROM programmer. Once this took place, users with these types of adaptors are not able to work with any character set except the one their ROM has been prepared for. It seems not to be a good deal in a confusing market of character sets. That's why EGA and VGA cards are the most widely used graphic adaptors in Iran today.

In VGA and EGA, BIOS load generator function 11h is used to download user-defined fonts [3]. This function consists of fifteen sub-functions for loading text fonts either from ROM or from a memory area in which user-defined characters are residing. After designing the proper font for a Farsi character in a font definition table, the address of table (array) is passed to this function along with its allotted ASCII code.

Farsi developers usually write device drivers to download user-defined fonts during system boot phase. In this case the downloaded Farsi characters are remained in memory while the system is power on. Since this might cause some conflict with other application programs system is running, developers are switching to another method which is mounting the Farsi module on the Farsi application itself.

3 Farsi Keyboards

Luckily, there is a general agreement on the way Farsi letters should be assigned to keys on the keyboard. With some little modifications, vendors have accepted Farsi keyboard layout expressed in standard 2901.

It is long expressed that English/US QWERTY keyboards are not as efficient as much as their counterpart Dvorak's keyboard since the latter has been designed having the results of several efficiency tests in mind. Apparently, the layout of Farsi keyboard is not based on any scientific research done in order to determine the most frequently used letters and put them on the most convenient keys for typing.

Like English keyboard in which upper-case letters are typed using the combination of Shift key and lower case letters, Farsi upper case letters are typed in a similar way. However, since a number of Farsi characters have more than just two cases, some vendors have also allotted keys to represent these forms.

Standard 2901 says that software algorithms should take the responsibility for recognizing which form of a character should be displayed according to context. However, since standard 2901 is itself

based on standard 2900 and that is being revised, it is not unlikely to have a new standard Farsi keyboard soon.

3.1 Intelligent and Dumb Keyboards

The Farsi alphabet consists of 32 basic characters (Arabic consists of 28 characters) and the stretch character used for beautifying Farsi scripts. A Farsi character might have up to four different shapes.

ﻊ   ﻉ   ﻌ   ﻋ

Figure 1. Each Farsi charactrer may have four different shapes.

Standard 2901 has tried to minimize the number of keys put on the Farsi keyboard to just one key for each letter and not to have keys assigned for all cases. This puts an extra burden on the shoulders of software to determine where and when to use which form of the character.

Keyboards with this feature are called intelligent keyboards since the user just types the letter and the proper shape of character would be determined according to the context.

Dumb Farsi keyboards are those that, like Farsi typewriters have defined keys for all cases of a character. Those in favour of this kind of keyboard argue that for professional typists, who have got used to Farsi typewriters, this style of keyboard is both more familiar and productive. Dadeh Kavi has his keyboard layout arranged in this way.

3.2 Diacritical Marks

The standard 2901 does not specify keys for representing diacritical marks. Although diacritical marks are not frequently used in modern Farsi (unlike Arabic), one might want to include such marks in his or her scripts.

Dealing with diacritical marks on the display is not possible. IBM PC compatibles and computer terminals have a fixed cell size for displaying characters. One or the other character can exclusively occupy the cell space on the screen, but not both at the same time. This makes it impossible to put diacritical marks on Farsi letters as far as working in text mode is concerned.

However, when it comes to printing, a 9- or 24-pin dot matrix printer can easily have diacritical marks printed in the same cell as of the letter.

The solution that some vendors have come up with it is to display the diacritical mark in character cell next to the letter that should be vowelized. The major drawback is that this causes inconsistency between Farsi letters and reduce the readability of Farsi script. Almost all text-based Farsi desktop publishing packages are using this method of representation to put vowels on letters.

3.3 Support for Farsi Keyboard

The majority of Farsi programs in market are not offering support for different types of keyboards. The most widely used keyboard layouts are those of standard 2901, Iran System, and SinaSoft. Layout conforming to IBM, Facit, and Olivetti typewriters are also used. Also, the Farsi applications are not generally supposed to offer users a way of defining his own customized keyboard layout. Standard 820 explains the Farsi typewriters keyboard layout.

4 Printing in Farsi

The early printers employed for printing Farsi scripts were daisy-wheel and chain printers. Although these printers were producing better quality Farsi texts than today's dot-matrix printers, they were slower. The other major problem was that these printers were not capable of printing bilingual Farsi/Latin scripts. To print in Farsi, users have to remove standard Latin wheel or chain belt, replacing it with one having Farsi letters embossed on it. Therefore, users were able to print pure Farsi or Latin scripts but not a mixture of two.

4.1 Downloading Farsi Fonts into Printers

9- and 24-pin dot-matrix printers are not subject to same restriction, since they let users to define fonts for a brand new alphabet while preserving the standard Latin/English fonts. It is all done through use of printer drivers which download Farsi characters into printer memory during system start-up or upon user's request. They can be embedded as a device statement in CONFIG.SYS file or as an stand-alone .EXE file.

Apart from this, some Farsi applications automatically do the job when being started by users. Printer manufacturers are dedicating escape sequence commands to this purpose which are used by software developers to define new character sets for printers. The process is similar to downloading Farsi fonts to display adaptors. Using the proper escape sequence users can replace the characters in the second half of the printer code table with Farsi fonts.

However, some dot-matrix printers do not allow downloading Farsi characters into certain places due to hardware incompatibility. With such printers it is no longer possible to keep a one to one relation between Farsi characters downloaded into display adaptor and those downloaded into printer.

Certain solutions have been proposed to address this problem. The technique frequently used is to build an intermediate layer or filter to do some pre-processing prior to printing Farsi scripts. Since it is quite possible to download all needed Farsi characters to printer (no matter the place), such a filter can be used to scan characters in data file which is to be sent to printer. When it encounters a Farsi character (e.g. character 149) which whose font is not residing on cell 149 of printer character table, but on cell 255, the filter will add the displacement value of x (here x = 255-149) to character code in order to find access to proper printable character on printer memory.

4.2 Printing Diacritical Marks

The problem of printing vowelized Farsi characters is largely solved with printers built-in capabilities. Majority of text printers are capable of moving print head a character back to put the vowels and sounds on already printed characters. Since, like displays, printed characters are also constrained to a certain size of print matrix cell, it is difficult to produce beautiful printed Farsi letters particularly on the 9-pin dot-matrix printers. However, 24-pin, ink-jet, and laser printers have no problem in

producing eye-catching Farsi scripts.

5 Editing in Bilingual

In Farsi, like Arabic, the direction of writing is right-to-left for text and left-to-right for numbers. Whenever a Farsi character is being typed, depending on the fact that it is a numeral or an alphabetical character, the cursor should show different behaviors. In case the incoming character is a numeral, the cursor stands motionless, shifting the numbers one place to the left as they appear on the screen. Meanwhile, typing a literal will cause the cursor to jump over the numerals and starts accepting literals (see following figure).

Character typed		Farsi text layout
none	(initial state)	سال_
٧	(7)	سال_٧
٢	(2)	سال_٧٢
م	(m)	سال٧٢م_

5.1 Handling Mixed Scripts

Due to the ever-increasing presence of Latin-based technical terms in modern Farsi, it is essential to offer support for typesetting of composite texts.

The technique developed to address this need is somewhat analogous to one used for handling Farsi numerals. Almost all Farsi applications, have a key devoted for switching between Farsi/English languages. When in English mode, the cursor remains motionless and incoming characters shift one place to the left as they appear.

Naturally, after any keystroke BIOS shifts the cursor one place to the right, so to make a program write in Farsi it is essential to bypass BIOS control over display. To achieve this, Farsi applications are controlling the direction of writing in the level of display.

5.2 Editing Features

The editing features used by Farsi developers are similar to those of their English counterparts. In a typical input line designed to accept Farsi input, backspace is used to delete the last entered character, although it moves in the opposite direction. The input line should support both insert and overwrite modes. Functions defined for other keys should comply with non-written standard for keys. Home and End keys are taking the cursor to the start or end of Farsi string currently being edited in input line. Control-right and Control-left are used to move cursor word by word in backward and forward positions. Delete key is used to erase the character residing on the cursor.

5.3 Contextual-Analysis Technique

The Farsi script is context sensitive. The shape of most of the characters depends on their position within a word and the characters adjacent to them. Each Farsi character may represent up to four possible cases of which only one would be correct in a particular situation (figure 1). It's up to the algorithm to find out from the context which form of the character should be chosen. Characters that are ending the words are written in their upper-case format. Like English, a blank space is used to separate the words. When a character stands alone or is at the end of a word, it is written in upper-case (bold-stroke). When it occurs in the middle of a word, it is usually joining to the subsequent letter. The algorithms used by different vendors to address this problem are much and less the same.

6 Searching and Sorting

Prior to performing sort and search operations on Farsi data, it is necessary to call string processing routines to reformat Farsi data into a more understandable pattern.

Some characters including stretch character (used to increase the clarity of Farsi texts) have no place in Farsi alphabet and should be removed. In addition, since a Farsi character may be represented by up to four different cases, each being coded as a separate character, provisions should be considered to replace the different occasions of these cases with just one out of four.

Next, comes inverting Farsi strings, since when sorting, strings are being compared character by character from left to right. Figure three shows a Farsi string before and after being processed by string manipulation routines. Once, these processes are accomplished, well-established sorting algorithms can be called to perform sort operation. Powerful software development kits (such as *Borland's database toolbox* and *Novell's Btrieve*) are available which provides developers with pre-written sorting, indexing and searching routines.

A معرفت B تفرعم

Figure 3. A Farsi string should be inverted prior to sort operation
A: In non-inverted state B: After being inverted

Although, the routines to perform needed operations on Farsi strings can operate on stored data, it is more convenient to invoke them at the time of data-entry. This will help to store data in ready-for-sorting format. The point is during data-entry and storage phase, due to operator typing speed and the I/O delays, these routines can use these dead-times to perform their tasks.

These routines should also be invoked when user makes a query to convert the query to proper format to be used to retrieve matching data. The technique usually built into Farsi applications is to store the data in a sort order using binary tree algorithm in indexed files. This makes quick and immediate search operations possible.

7 Localization of Systems and Applications

So far, the process of translating software application programs into Farsi has been largely based on making modifications in object code and not in the source level. Apart from translating the contents of menus, status lines, windows, dialogue boxes, warning messages, and help files into Farsi, it is necessary to find a way of feeding Farsi data into the application instead of Latin data.

To make these applications accept Farsi inputs from user, use of Terminate and Stay Resident (TSR) programs has proved to be helpful. Whenever the user is in input line to enter Farsi data, with a single keystroke he or she can invoke the TSR program already installed in the memory and start typing in Farsi. Since a TSR program can work independent of application environment in which it is called, users may also use them to type Farsi in DOS command line. Using TSRs, applications such as spreadsheets and database management systems can work fine in Farsi.

7.1 Farsi Applications

When it comes to Farsi word processing and desktop publishing, use of TSR programs is not much helpful. Many have found it much easier to develop such applications from scratch rather than making changes in the package originally developed to be used by English users. Editing and word processing features needed for manipulation of Farsi texts, a wide range of Farsi typefaces, and Farsi spell-checking capabilities can hardly all be embedded into a package throughout making changes to object code. So, there is no surprise that today there are several Farsi DTP and word processing packages with reasonable capabilities available.

References

[1] Tayli, Murat and I. Al-Salamah, Abdullah. Building Bilingual Microcomputer Systems.
In Communications of the ACM, Volume 33, Number 5, May 1990, pp. 495-504.
[2] Smith, Ben. Around the World in Text Displays.
In BYTE, Volume 15, Number 5, May 1990, pp. 262-268.
[3] Phoenix Technologies Ltd. System BIOS for IBM PC/XT/AT Computers and Compatibles.
Addison Wesley, 1989.
[4] Kane, Gerry. CRT Controller Handbook.
Osborne/McGraw-Hill, Berkeley, California, 1980.

This page is intentionally left blank.

Farsi GUIs

by
Dr. Abdullah Ibrahim Al Salamah
Dean, College of Computer and Information Sciences
King Saud University
P.O. Box 51178
Riyadh 11543
Kingdom of Saudi Arabia

Introduction

The introduction of English-based personal computers to the Middle East spurred the development of bilingual systems and user interfaces. Many countries have successfully implemented Arabic/English interfaces. These developments, however, were not coordinated nor centrally planned. This led to the development of several incompatible Arabic character sets. Nevertheless, the development of Arabic Windows™ by Microsoft® has by virtue of its elegance emerged as a de facto standard for Arabic/English graphical user interfaces.

Currently there are efforts to develop adequate Farsi character sets. One trend is certain - there will probably not be a centrally coordinated standardization process for Farsi. Undoubtedly, a de facto standard will emerge. Hopefully, it will be adopted because of its adherence to the principles outlined in this paper, and not because of its sheer volume in the market place.

Because of the close similarity between Arabic and Farsi, there need not be a long development time in the construction of a Farsi character set. This paper proposes that Arabic Windows is an excellent model to follow because of its support for the following theoretical principles.

Bilingual Systems - Design Guidelines

Ideally, a number of design guidelines are necessary for implementing bilingual systems and are presented below. Arabic and English are used as example languages, however, these principles are applicable to Farsi and English as well.

- The bilingual software process must mainly be a software solution without restriction.
- In order to address a large audience, the core of the bilingual system must run on widely accessible powerful workstations.
- Problems inherent to the bilingual nature of typical input/output functions must be solved whenever possible at the workstation level.
- Basic workstation capabilities must include advanced display management and resource multiplexing mechanisms.
- The workstation operating system must support the desired bilingual user interfaces in order to provide a complete and natural processing environment.
- The final system must include bilingual development tools and programming interfaces to existing programming languages at the application layer.[1]

Graphical User Interfaces
One goal of all software developers is the perfectly intuitive interface - one that does not need a manual. Such an interface makes the software accessible to more people. In the late 1970's, developers at Xerox PARC took the initial big step by developing a Graphical User Interface (GUI) for the SmallTalk system. The work of Xerox was later refined and popularized by Apple Computer with its line of Macintosh personal computers.

Besides replacing command lines with a full-screen interactive display, GUIs aim to standardize the interface for operations common to all applications such as opening, saving, and printing files. The GUI screen displays a visual representation of the system. The user interacts with the system through various symbols and objects. The graphical approach makes programs significantly easier because it prompts with visual cues instead of waiting for imperative commands. A graphical interface allows the user to see more information at a glance. And last, but not least, GUIs have the potential to retain the user's interest by being more aesthetically pleasing.[3]

Platforms
The IBM-PC/AT/80x86 family of computers is popular and widely available due to clone manufacturers. Many different models in vast quantities exist providing a wide spectrum of power/performance at an affordable price. Furthermore, its open architecture supports a wide range of software and can be used as building blocks to form more complex systems through various networking strategies. IBM-PC and AT clones are the most common personal computers in the Arab world. An Arabic GUI for this family of machines is necessary to provide its users with the power of a GUI as well as attracting potential new users who may otherwise be afraid of using computers.

Microsoft Windows
Microsoft introduced Windows 1 in the early 1980's, providing GUI capabilities for the IBM-PC family of computers. In successive versions of Windows, Microsoft has steadily added features that integrate the system's modularity around the concept of a document. Indeed, we currently can have a single document that employs the services of several applications simultaneously. Multimedia titles combine an extensive variety of data formats (text, picture, voice, ...) into a single presentation. The impact of this innovative environment has a chance to change the way we work with computers.

Windows is a DOS-based package that, in a sense, extends its power. It is also, in some sense, an operating system that takes control over the machine and all connected devices. Programs developed for Windows cannot run without it since Windows provides CPU and memory management in addition to graphics display management (see Figure 1).

Windows shares the screen by creating separate windows for each program. The window's border completely contains the output of the program. In order to protect the output in other

windows, Windows itself enforces this rule and will not draw any program's output outside of its own client area. If a drawing's coordinates fall outside of the windows, the drawing is truncated.

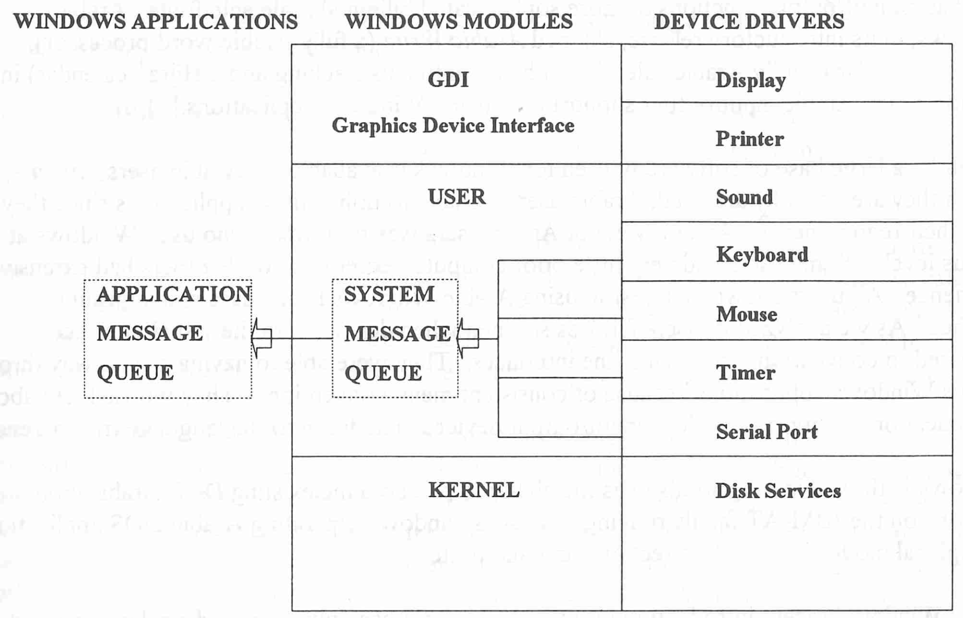


Figure 1. Windows Design Outline

Windows promises to free programmers from accounting for every possible variety of monitor, printer, and input device. Each device driver need be written only once. Instead of each software developer writing its own complete set, each hardware manufacturer writes one general driver for its products to run under Windows. Microsoft includes many drivers with Windows; others are available from the manufacturer. From the application's point of view, it sees generic devices supported by Windows rather than specific devices. By the same token, each device driver works with every Windows application that needs that type of generic device.

Multitasking, device independence, objects embedding and linking, and a standard graphical interface distinguish Windows from other programming environments. These powerful features challenge developers to rethink their program designs. Before the advent of Windows, applications were tending to grow larger and larger, acquiring ever more features and trying to be everything to everybody. Under DOS, size was the only answer to the diverse needs of a large user base. Windows allows another approach. It still supports large applications, but it also permits smaller applications to act in concert, even if each comes from a different developer. Windows applications exchange data, each acting as a module in a larger system. All the modules run concurrently. They may be added or changed individually, allowing the user to select exactly the desired mix of features. The combined power and flexibility of several small applications can equal or exceed that of a single large program.[3][7]

Microsoft Windows with Arabic Support (Arabic GUI)
Microsoft introduced Windows v3.0 with Arabic Support by the end of 1991. Arabic Windows supports Arabic at its internal level providing the most bilingual transparent environment ranging from basic input/output functions to more sophisticated bilingual scale able fonts. Arabic Windows, in its introductory release, shipped *Arabic Write* (a fully Arabic word processor), *Arabic Calendar* (a fully Arabic calendar with appointments tracking and a Hijra[1] calendar) in addition to the Arabic input/output support of existing Windows applications.[5][6]

Currently, a large base of software written for Windows is available for Arabic users. Even though they are not fully arabized, Arabic users are able to utilize these applications since they serve their requirement[2]. An interview of Arabic users was performed who used Windows at various levels. Some users had very little prior computer experience while others had extensive experience. All users showed interest in using Arabic Windows for its ease and appealing interface. As we analyzed it, users felt less strained when dealing with the visual interface compared to conventional command line interfaces. They were able to navigate their way through various Windows applications because of consistent menu conventions. They worried less about an application's support of various input/output devices in addition to the language transparency.

Windows with Arabic support also has the ability to preserve the existing DOS Arabization solutions on the IBM-AT family running as a DOS window. Operating Arabic DOS applications in graphical mode supports bi-directional cut and paste.

Arabic Windows' recent introduction into the Arab world has only permitted a relatively short time to judge its impact. Notwithstanding, it has received great acceptance among Arab computer users and it has begun to be a de facto standard for several organizations. A similar reaction for Farsi bilingual systems could occur if a Farsi Windows system could be developed.

Arabic Windows and Systems Integration
Systems integration is a necessary requirement in today's environment. Various heterogeneous systems are viewed as building blocks which must be integrated together to form a larger system. Such integrated environments are called *OPEN SYSTEMS*. The components are dissimilar - manufactured by different suppliers and running different operating systems. Yet they are connected in a network providing one working environment where information is transparently exchanged and computing power is incrementally increased.

The concept of openness is appealing to users in the computer industry because it dissociates them from monopolizing manufacturers. Given a specification, users can integrate several operating systems on different hardware platforms supplied by different vendors in addition to a variety of accessories.

1 Hijra calendar is a lunar calendar used by Moslems.
2 Microsoft already announced that it will ship a fully arabized version of Windows in addition to a line of its software: *Microsoft Word* word processor, *Microsoft Excel* spreadsheet, *Microsoft Access* database, and the Arabic Software Development Kit for Windows development tools (Visual Basic, C/C++).

Microsoft has put great effort in attracting software developers to join the Microsoft "family" by releasing development tools and system documentation at affordable costs. This has motivated developers to release Windows versions of their software packages and tools in addition to creating new packages for Windows. These tools cover a wide range of applications from word-processing, drafting, utilities, networking, etc.[3]

Windows' peculiar design, supported by its large base of software tools and packages, allows for smooth integration with other systems. A variety of networking protocols and network interface tools is available for Windows.

Arab computer professionals now have the chance to create integrated systems that comprise bilingual GUI interfaces. The IBM-AT family of machines (and clones) running Windows can serve as low cost bilingual workstations in addition to more powerful servers that are connected on the same network. Windows with Arabic support is ideal as a bilingual interface tool to create a bilingual open system architecture. The same could hold true for a Farsi Windows system.

One workstation can connect to more that one server at a time. The user can then process files transparently on many servers (concurrent with other users, and possibly on different operating systems). In an integrated environment, remote output devices are accessible in a transparent and bilingual fashion. The underlying network takes the responsibility of delivering the request to its destination which can be distributed on several nodes (points) on the network. Adding Windows network oriented tools (electronic mail, electronic scheduler) in the desired language (Arabic, Farsi, etc.) along with other bilingual desktop tools, provides integrated bilingual office automation at its best for end users.

Anticipated Problems in an Integrated Bilingual Environment
The inadequacy of existing standards for bilingual character code sets has resulted in many diverse code sets that have been adapted by several bilingual applications on various platforms. This current diversity is a barrier to transparent information exchange in the bilingual integrated environment.

Windows with Arabic support, has the ability to emulate any of the known Arabic character sets adapted by known applications. Furthermore, a file conversion utility is provided that translates files between several Arabic character codes.

Third party Arabic software houses have announced that they will develop software packages that address the problem of multiple character sets in an integrated environment. These packages will make use of the emulation feature in Windows.

Conclusion

3 Microsoft announced that it will release an Arabic development library for some Windows development tools.

Arabic Windows is a fine example of a de facto standard Arabic GUI that follows strong principles for bilingual design. The Farsi bilingual effort can learn from the development experiences of Arabic bilingual systems. Adherence to the principles given above should guide the design of Farsi bilingual systems. Practical and elegant implementation of those principles (as done for Arabic Windows) should produce a working Farsi model upon which countless Farsi applications can be developed.

References

1. Tayli, M., Al-Salamah, A., Building Bilingual Microcomputer Systems. *Communications of the ACM* 33,5 (May 1990), 495-504.
2. Belfakih M., Arabization: Analysis, Recommendation and Implementation Plan. *High Commission for the Development of ArRiyadh* (May 1991).
3. Myers, B. and Doner, C., Programmer's Introduction to Windows 3.1, *Sybex Tech Publication*, First Edition 1992.
4. Arab Standards and Meteorology Organization, 8-bit coded Arabic/Latin character set for information interchange, ASMO DS 708 Tech. Report, 1985.
5. Microsoft Corporation, *Microsoft Windows 3.1 User's Guide*, No. PC21669-0492, 1992.
6. Microsoft Corporation, *Microsoft Windows 3.1 with Arabic Support, User's Guide Supplement*, No. 0592-PN31317, 1992.
7. Petzold, C., Windows version 3.1, *Microsoft Systems Journal* 6, 5 (Sept/Oct 1991) 17-26.

An Approach to UNIX Farsi System

Behzad Torabi
Iran System[†]

December 14, 1992

Abstract

Although single-user DOS-based machines still are undisputed rulers of Iran's computer market and industry, those organizations in search of more powerful systems have started taking UNIX seriously. However, work should have been done to add Farsi capabilities to UNIX. Following a brief history, the author takes a look at efforts made in Tehran- based Iran System Company to develop Farsi tools and applications for UNIX. As a part of UNIX Farsi System, the development of a Farsi translation layer under which Farsi applications are running is explained. The article also focuses on Iran System's strategy in choosing UNIX as the platform of choice.

† Iran Systems, P.O. Box 15875-3884, Tehran, Iran.
Tel: (021) 628-055 Fax: (021) 622-141

1 History

When early personal computers arrived in Iran, there was little Farsi computing around. On the mainframe side, IBM with its family of 3270 Farsi terminals and Farsi chain-printers was the biggest player in the market. The way that IBM addressed the problem of working in Farsi was quite simple. Using a switch, data terminals could shift back and forth between Farsi and English text-entry modes. This switch was also inverting the direction in which characters were entered, therefore making it possible for users to enter Farsi texts (like Arabic, the direction of text in Farsi is from right to left).

On the printer side, the English chain should be replaced with a Farsi one and after that, it was possible to make Farsi hard copies. However, the major disadvantage of such a solution was that of machines' nature. They had been originally designed to work in English and the data could never be stored in Farsi. Another disadvantage was the lack of mixed-lingual text-entry capabilities. English is the language of choice in technical and scientific communities and beside supporting the native languages of users, a capable computer system should also offer support for bilingual text-entry.

With the arrival of first personal computers to the country and the lack of support from Western computer firms, Iran's computer community was faced with a challenge of great magnitude: making personal computers work in Farsi.

The first personal computers which were Iranized were NCR DMV series. After that, IBM PCs and compatible machines should have been modified to work in Farsi. On the printer side, Epson's FX-100 series were among first printers offering bilingual Farsi/Latin capabilities. Since there was no organization to supervise and conduct the process of Iranizing computer systems, some private companies had to take over. As the result, multiple standards for Farsi character sets emerged on IBM machines.

On the software side, some poor quality Farsi editors were all that were available for Iranian users. Not a single Farsi database management system, word processor or utility package was available and every thing should be started from scratch.

Faced with limitations in accessing technical documents and materials on one side, and lack of hard currency on the other, Iranian system developers and programmers had to go through the tough way, finding their own path throughout the system and finding out about everything with a trial and error method. Today, the situation is much different. A wide range of Farsi applications and tools are available in the market. Iranized database management systems and desktop publishers, accounting applications, Farsi DOS extenders, and almost every thing else that users may need is present.

DOS-based systems are around every where and, wide spread practice of software piracy have put bundles of free packages and applications from every category at the hand of users!

2 UNIX

Up to 1988, so much work had been done for DOS-based PCs. The prices were down enough for most government departments and private companies to purchase personal computers and peripherals, but medium and large size companies and organizations were still faced with problems. From one side, the amount of data in their offices was much more than what a DOS-based PC could handle and from the other side, they needed to share data among users in various ways.

One solution was buying a mainframe, but comparing with PCs, mainframes were and still are too expensive to be purchased and maintained. There remains two other choices, networking and switching to UNIX.

We, at Iran System, were faced with this problem, too. We saw a large potential in market for much stronger systems, but as a PC manufacturer, we had to stick with IBM compatibility. In the summer of 1988, we introduced our 386-based computers. These machines were powerful enough to handle tasks much stronger than a simple DOS session, so we decided to choose between DOS-based networks and UNIX.

UNIX won for many reasons, among them the power, built-in networking, multitasking along with multiuser capabilities, and expandability in much lower costs compared to DOS-based networks.

Modifying UNIX to work in Farsi was not an easy task at all. There were limitations, restricting our access to all aspects that could be considered Hi-Tech, and in a market that was crying for more and more computing power, we didn't have much time to spend around.

3 Hardware Side

Although the problem of displaying Farsi characters on PC display units had been solved long ago, the way of accessing and loading them under UNIX was very different. On the other hand, under UNIX the system console (i.e. the PC's display unit) may be used as 12 separate ANSI compatible terminals, each of them having its own way of handling Farsi characters. The diversity of terminals that UNIX could support was another problem. Choosing the simplest way, we decided to start with just two main series, WYSE and DEC's VT series.

Iran System's Farsi character set is based on bimodal appearance of Farsi alphabet, so we did not have to worry about processing each character before sending it to the terminal (the task is known as contextual analysis).

Thanks to UNIX's same view of all peripherals, handling Farsi on printers comes much the same way of the terminals. Some of the special purpose devices were not capable of accepting Farsi downloadable characters and therefore, minor hardware changes should have been done to build Farsi codes into them.

Parallel to the team working on hardware side of problem, another team focused on developing system and application programs. The most wanted tools for the new born Farsi UNIX was bilingual English/Farsi text editor, database management system, word processor, and Farsi shell. The latter was very important because UNIX is very odd looking at the first glance for DOS users and we had to provide ease of use and user friendliness that came with DOS-based systems for our users to make them feel at home.

With the goal of making UNIX friendly and acceptable for users familiar with DOS, we took our software models from the DOS world. Our bilingual editor, BE, is an UNIX menu driven equivalent of PE2 in DOS and our Farsi shell, ABZAR, is very much like an extended version of PCSHELL.

We also chose the UNIX version of FoxBase as our Farsi database management system for its widely acceptance in DOS world, the ability of creating similar environments for both DOS and UNIX users, and application and data portability between two operating systems. This made the task of putting UNIX and DOS machines on the same network easier by making same data transparent for both worlds.

For our word processor, KELK, we had no model to imitate. Because of the challenges that an UNIX word processor should meet, simply no other word processor was available to fulfill our requirements. A DOS version of KELK was also developed to keep the idea of having a DOS equivalent for all of our tools. A Farsi calendar and date completed our Farsi package.

To be an assist to the system developers writing applications under our Farsi UNIX environment, an advanced screen editor (ISE) with professional editing features was developed along with a Farsi programming language and database management system (ISPL) with built-in SQL and BASIC like syntax. We also developed a programmers' hexadecimal editor (MABNA).

When the job was finished and the Farsi UNIX came to the market, we were noticed that market's demand for distributed databases was much more than we thought. Rapid enhancements in country's telecommunications networks now make it possible to have reliable communication interchange between two or more computers in different locations of the country, and 9600 data compression modems are working well enough.

This made us to go for a real networking database and we figured out that Informix is a proper choice. Our Farsi solution for Informix was adding a Farsi phase to its compiler to alter the generated code. This brought comfort to Farsi application developers since they no longer were caring about how the program will work in Farsi mode.

4 A Farsi Layer for UNIX:

It is necessary to have a translation layer for application programs running under a Farsi multiuser and multitasking operating system. Of course, such a layer increases the system load since for every active task, at least one translation task should be activate and this puts an extra burden on the system. However, there is no other alternative when one is bringing an English application to an Eastern language like Farsi or Arabic (another choice which is coding Farsi applications from scratch do not seem much appealing!)

This translation layer, like many other software solutions, can be placed in system's terminal. Although using a Farsi data terminal removes the translation layer's load from central computer, from our point of view it has the following disadvantages:

- What should be done to the current installation base of terminals? Most of our customers who were using Iran System UNIX did not have the financial conditions to discard all of their terminals and purchase new ones.

- For cost efficiency purposes, many customers are using their outdated XTs and ATs with terminal emulator programs. The Farsi terminal was not an answer to this kind of users at all.

- The system's console does not take advantages of the Farsi terminal. So it should either remain untouched or a Farsi translation layer should be written to just handle jobs that had been activated from console.

Having above mentioned factors in mind we turned to Farsi translation layer alternative. Before going deep into the design, it's a good idea to discuss Farsi DOS extenders first. It would help us in getting a better view of what a Farsi translation layer looks like.

A DOS translation layer is a TSR (Terminate and Stay Resident program) that filters the keyboard input and screen output of desired program or whole operating environment and makes certain changes in them.

Input filter replaces the address of system's keyboard interrupt, so when user activates the language-switch key, each English character will be mapped into its Farsi equivalent. When a program tries to access the display unit, the cursor movement will be reversed and Farsi characters will be interpreted and displayed by reading and writing the PC's screen memory. Since DOS is a single-task operating system, a TSR program can easily take over all system resources. As long as TSR translation layer is in charge, DOS has no control over what happens to the system.

On UNIX side, the main concept remains the same. Any program that uses the translation layer should be surrounded between I/O filters. But UNIX is a multiuser and multitasking environment with full control over computer's hardware. The main differences between DOS and UNIX translation layer programs are:

- Because ordinary users never have a permission to access computer hardware under UNIX, direct hardware accesses through user-level programs is not possible.

- Almost all UNIX programs run on text-based terminals with no screen memory. So there is no ready-to-read screen output available.

- TSR concept simply does not exist under UNIX. Because the environment is a multitasking one, a program is either running or not running and the concept of Terminate and Stay Resident simply has no meaning. Of course, a program can run either in foreground or in background and a running program can be in sleep mode until something wakes it up.

Having these facts in hand, something for UNIX itself can be worked out. At first, because there is no access to system's hardware, the available system resources (i.e. standard input, standard output and standard error) should be used. In our Farsi layer, two programs are responsible for taking care of the job, one filters input and the other filters output of the program that we want to make it work in Farsi. These two programs keep track of their status by using IPC (interprocess communication) methods prepared by UNIX. In this way, they both will always know the exact situation of main program.

In main program's point of view, keyboard input and screen output come and go just the same way they did before. Here are some reasons that why this type of translation program suits best for our needs:

- Because all the translation job is handled by translation program, there is no need for making changes in existing terminal emulators and PCs can be used instead of terminals as before.

• All types of Farsi terminals are supported and there is no need to add new hardware for using translation layer program.

• Because of bimodal nature of our Farsi code set, data remain transparent between DOS and UNIX application software programs. In addition, there is no need to printer filtering programs since the same data will be sent to the printer.

• The price that user pays to get advantage of Farsi layer with his or her library of English software products, compared to other solutions, is very low and this persuades users which are in doubt regarding choosing a proper platform, to make up their minds and switch from DOS to UNIX. Users that are currently using UNIX will be sure that their investments are safe.

5 Conclusion

The UNIX Farsi System is made of following parts:

• Hardware that Farsi characters are directly built into it
• Drivers for hardware devices which are capable of downloading Farsi code set
• Original Farsi programs (shell, editor, word processor,...)
• Tools to be used by system developer
• Farsi translation layer

Farsi Computing: An Overview

Amin Mohadjer
BRAIN Computer Systems Group[†]

January 17, 1993

Abstract

In Iran, a wide range of computing tools, originally designed to process Latin-based scripts, have been in use for decades despite the linguistic barriers. The process of localizing Latin-based systems and applications, in order to make computing and the use of computers easier for Farsi users is well underway through foreign vendors and domestic developers. This article illustrates the problems around regarding computing in Farsi. Some of the techniques long used by developers of Farsi systems are brought in focus. Since, DOS is still the environment of choice for the majority of Iranian software developers, most of topics discussed here are exclusive to DOS. Some of the problems such as having multiple standards for Farsi information interchange are also discussed.

† BRAIN Computer Systems Group, P.O. Box 14455-161, Tehran, Iran.
Tel: (021) 980-102 Fax: (021) 980-102

1 Introduction

During the past several years, Iran's data processing community has suffered from the lack of a well-defined standard for Farsi character set. Despite the frustrating debate on setting a unique standard, there are still doubts that whether we will ever agree on a standard character set for our language "Farsi" in a near future.

It was just ten years ago, when PC computing found its way through country's data processing environments. Before that, the computing was mainly done on mainframes, most of them IBM 370 and compatible series. IBM mainframes were based on EBCDIC character set.

Since 7-bit EBCDIC does not provide enough space for having both English and Farsi characters simultaneously, users of such systems were forced to use either English or Farsi characters at one time by switching back and forth between these two sets. In practice, this caused some problems; it was not possible to have bilingual texts, and texts in each of languages should be printed on separate printers.

Then came ASCII and soon became the standard of choice for PCs. The 8-bit ASCII was capable of having 256 characters defined, enough to contain both English and Farsi characters. However, to have Farsi characters defined, the European and mathematical characters in the second half of the code set (codes 128 to 255) should be dismissed.

Due to the hardware limitation of then, it was not possible to download fonts for Farsi characters into video controllers. In order to make a PC work in Farsi, the characters should be built into EPROMs using EPROM programmer devices. Since not everybody had an EPROM programmer or the necessary technical know-how, only a few companies made their own character sets available, perhaps two or three.

With the introduction of downloadable printers and graphics boards, many software developers found it easier to design their own character sets and this eventually led to the numerous character sets available today.

Different motives could be seen for the companies to deliver their own character sets. One is a matter of taste. Different character shapes, which will be discussed shortly, were supposed to be not the same from the point of view of readability and visual effects.

Also, since our national standard organization was out of the game, software developers were thinking that if the program they were producing dominated the market, they would be recognized as the first companies who set the standard through the vast acceptance of their character sets.

However, till this moment, no organization has come up with a widely acceptable Farsi standard. The blame may go to software companies since they insist on their own character sets and are not ready to give them up in favour of having a single one.

But, one should also be aware of the cause that made these companies go their own way and that is of Farsi script. To be fair, we should say that the blame must at least be shared. At present, the question that comes to mind is whether we should continue our efforts (rather fights!) to eventually agree on a standard or we can find an easy, acceptable and inexpensive way of supporting all available character sets so it would no longer matters for users which set they are using and at the same time these numerous sets can live peacefully together.

Of course, there are technical problems and some other major obstacles that should be addressed. And ultimately, will the result be acceptable for the developers and end users?

2 Character Sets

As it was mentioned before, one of the factors that has greatly contributed to the emergence of multiple Farsi character sets is Farsi script, itself.

In Farsi, very much like Arabic, the shapes of letters depend on their position within a word, and their adjacent letters. Some of Farsi code sets provide distinct shapes for upper-case and lower-case letters. That is for a given letter such as ب two distinct shapes are considered, one for ﺑ (lower case) and another for ب (upper case). Therefore, in this class of code sets each upper case character gets its own code space in the table.

On the other hand, there is another class of code sets which do not allot code spaces to upper case letters but introduce a "tail" character, which concatenates to the lower case characters to form the upper case letters (figure 1). To name a few characters, ب , پ , ت , س , and ص are generated this way.

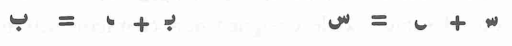


Figure 1. A "tail" character concatenates to the lower case to form upper case

Putting aside the difficulties of handling mentioned two-form characters, working with four-form characters is another problem. Depending on the type of Farsi standard in place, a four-form character such as ع (its different forms are ع , ﻊ , ﻋ , and ﻌ) can either be expressed using four different character codes or we can use just two codes for ﻋ and ﻌ and add a tail ـ when appropriate.

There is another group with quite different shapes which are also making troubles. Characters such as ق, م , and ل can not be expressed by appending a tail to their lower case formats, so we must consider two separate codes for their upper- and lower case formats.

Two letters need further briefing. ی and ه are the last two letters of Farsi alphabet. Although, ه is a four-form character (ه , ﻬ , ﻫ , and ﻪ), a great number of character sets have just two codes devoted to represent all four forms. ه is used in place of both ه and ﻫ while ﻬ is used for the other two. The upper case ی gets two distinct shapes in modern Farsi scripts; ی is used when the character comes stand-alone and ـی is used when it is to be concatenated to the previous character. Some character sets consider just one ی for both these purposes. There are also problems regarding handling لا , ا , ئ , and ة characters which are not discussed here due to space limits.

3 Monitor and Printer Support

As we mentioned earlier the first printers and graphics boards did not support font downloading and were loading their fonts from ROM chips. In order to add Farsi capabilities to their systems, users had to have their font generator ROMs replaced by EPROMs in which Latin characters residing in the second half of the code set had been replaced by Farsi characters.

With the introduction of EGA and VGA cards on IBM PCs, Mackintosh computers which generally work in graphics mode, and downloadable printers, creating new character sets and working with them became a relatively easy task. The above mentioned reason and the fact that software can be copied easily and freely in Iran, it became possible for almost every PC user to own a downloader.

Despite the easy task of downloading fonts, there were still a few hardware that cause problems. For instance, some STAR printers do not allow downloading characters for some ASCII codes in the second half of the character set. Some other printers did not allow to even have just half of Farsi characters defined at one time. These problems caused many companies, mainly those who were selling these stuff, to design new character sets that would work without problems on their products.

Some unpredictable things also happened with the introduction of VGA boards. A character in VGA is 8 pixel wide but it is shown in a 9 pixel wide block. In other words one blank pixel would always be left between two adjacent characters. This caused a very unimpressive visual effect.

On the other hand the tabulation characters which lie in the ASCII range 176 to 223 were showing a very interesting property. That is the CRT controller duplicates the 8th pixel column to 9th column. In this way you could make characters like ═ and ╡bind closely to each other. This feature raised this idea to use mentioned area of ASCII character sets for concatenating Farsi characters. In all previous character sets this area was left untouched since they were needed in many applications. But by the time VGA was introduced some people designed new character sets by sacrificing the tabulation characters for keeping the consistency between Farsi characters on screens.

Another feature built in VGA boards permits having 512 different characters displayed at one time. This is possible by omitting the intensity attribute. This feature brought the idea to some people to make use of it for solving all the existing problems. But in practice new problems were emerged. Storing a 9 bit character code was both a cumbersome task and also there was a question of portability. Printing characters of such a code set, if not impossible, could slow down the activity for a considerable amount of time.

It is clear that different hardware configurations have had a considerable impact on the evolution of so many new character sets.

4 Editing

In a standard Farsi typewriter, letters are typed individually and there is no smart process to determine the right shape of the letter according to context. It is up to typist to use shift or other control key to type the proper shapes. In computers, in contrast, one can write line editing programs. This provides the ability to make an analysis on the incoming keystroke and the string entered before to figure out the appropriate form of character, if necessary.

So far, several algorithms have been devised to do such analysis and conversion. Some of these algorithms rely on just the previous entered key. Some others look at the character before the cursor in the line buffer. In the latter solution if you move the cursor along the line you can be sure that the correct shape for any newly typed character would be determined according to the character preceding the cursor.

Another conversion that occurs when you enter a key is done on the character before the cursor. Generally there are two basic modes of editing in use. In one, when keys are entered the lower case shape will be considered as the default. If you type a concatenating character in front of it no change will occur but if you type a blank the previous character will be converted to its upper case form. In the other method, the upper case mode is selected when you enter the character. Then, if you enter a concatenating character following it, the program automatically converts it to the appropriate form.

A few problems occur with the automatic conversion of codes, no matter which method you choose. Suppose you are to enter the word; هفت حوض. With either of the methods discussed earlier the word will be entered as هفتحوض which is not quite what we intended to have. A solution to such problems may be a virtual space between هفت and حوض by using an unused key to enter it.

Another problem is that in Farsi script we use ﻻ instead of ﻟﺎ and it is expected that the editor program replace the two characters ﻟﺎ with the one character ﻻ .

Of course more can be said on the different approaches the software developers have taken in their editor programs, and the keyboard layout as one of the most affecting factor needs further considerations; but, here we just took a look at some aspects of editing modes in order to show their impact on the selection of a character set.

As you've seen the dynamic analysis and conversion of characters in Farsi editors make them distinguishable from their Latin languages counterparts (besides you have to handle letters from right to left while digits are entered using the same left to right scheme). Because of that, many developers decided to sacrifice some character codes in favor of having simpler and fewer code. Some letters that were omitted under this condition are the stand-alone ی; as well as ﻫ , ﻬ, ﻻ, and ﺋ.

In the same way processing of with-tail characters is relatively simpler than those characters which have distinct upper case and lower case codes, hence some favours the with-tail characters (no need to mention that when printed, with-tail characters have a more impressing appearance).

Some developers have tried to Iranize original packages, especially database management systems to give Farsi users full advantage of the complexity and power built into these products with a minimum of investment. Since such companies did not have access to source programs, they had to write interrupt service routines in form of resident programs or device drivers.

These interrupt routines which replace the keyboard and sometimes display interrupt service routines allow to make necessary analyses and conversions on the input key. Almost all such systems could only rely on the previous key typed and there was no way to analyze the user's input buffer according to cursor's position. This caused that those companies present character sets which need little analyses and simple conversions.

Now, with the ever increasing demand for original, complete, and reliable packages and services the original companies offer on their products, we see less such illegal modifications and reselling of packages. On the other hand since some of these original companies now have representatives in Iran, it has become possible for them to work on the source programs and hence they may present much cleaner customized packages. I hope that this will help to make companies agree on a standard set. In this way those who have presented character sets to get easy and unlawful profits from packages, no longer will be able to dominate the market.

5 Storing and Sorting Farsi Data

In this section, we will discuss the problem of storing and sorting Farsi texts and data and its impact on choosing a character set.

Two cases should be studied under this category. The first is when you write the sorting routines by yourself or have a toolbox or a file system package to do the job for you but you can determine your own comparison routine.

In this case there is no need to worry about how to store the data, since regardless of method you can perform enough modifications on the two strings you want to be compared. These conversions usually can not be performed prior to storing (though some do it), since some information may be lost in this way. To name a few of such conversions that are useful to make comparison more reliable, we may mention: removing blank spaces, stretch characters, and some punctuation characters, converting multiform characters to just one character (this conversion must preserve the alphabetic order), and inverting the string so that we may use the hardware comparison operation which usually performs from left to right.

Sometimes this last conversion is performed during storing so that the time for each comparison will be reduced and hence the time of occasional sorting that often takes lots of time.

The other case is when you use a package or toolbox that does not allow you to define a comparison routine. In other words it uses the conventional string comparison operation which translates to a single machine instructions.

Design of a character set for such system must be done carefully since extraneous characters and even the order of multiform characters can cause serious problems. Unfortunately, it seems to be almost impossible to design a perfect character set for this case so that it would eliminate every problem.

6 A Solution !

By the way, although numerous factors should be considered in designing a Farsi character set, I tried to enlist some of the most important ones. Now we go back to the question that whether we shall continue our debate on setting one single standard or there are ways to have a few sets (each of them being selected from a larger group of character sets) and have them work beside each other. The remaining part of the article will consider the possibility of developing software to provide different sets as options for users or even allow the exchange of information between existing software by adding conversion routines to them.

I propose that old software be modified in their new version so that they provide possibilities for the user to import or export files of different character sets (some companies have already tried to do that, and it seems that it has proven to be practical). New software products should provide options (e.g. in form of menus) for selecting the desired character set to be used in the environment of the package. This may satisfy the taste of a large majority of users. The technical aspects of such system is under investigation by the author and early studies reveal that this is quite possible.

7 Conclusion

The last word is that while we have not been able to come to an agreement on a standard character set after so many years, why don't we sit and select a few, that is three or four, sets as our standard, each representing the useful aspects we have already discussed? Obviously the major problem in this selection is that sets must be convertible to each other and this conversion must be done in an inexpensive and reliable way. I hope that future efforts for setting the standard will eventually lead to a happy ending that put a firm basis for future developments of Farsi software programs.

Catalog of Publications on Farsi and Arabic Computing

Compiled by: Amin Mohadjer

The following bibliography contains the list of published papers, articles, and conference proceedings which in one way or another are related to the topic of Farsi/Arabic Computing. Since the preparation of a more complete version of this list was not possible, a preliminary version is introduced here. This list is sorted according to the names of authors and will be regularly revised and updated to include new entries.
In case, you have authored a paper or article in above mentioned area and your name is not included here, please be kind enough to provide us with a copy of your paper to be appended to this list (the address is given below). Copies of some of the articles listed here are available upon paying a nominal fee (to cover postage expenses). Send your requests to the following address:

Amin Mohadjer
P.O. Box 14455-161
Tehran, Iran.

1. Appropriate Technology Ltd. *APTEC Arabic Utilities Programming Manual v2.* London, 1984.

2. Arab Standards and Metrology Organization. *8-bit Coded Arabic/Latin Character Set for Information Interchange.* ASMO DS 708 Tech. Rep., 1985.

3. Arab Standards and Metrology Organization. *Data Processing 7-bit Coded Arabic Character Set for Information Interchange.* ASMO 449 Tech. Rep., Amman Jordan, 1982.

4. *ASCII Encoding of the Persian Language.* NOOR Vol. 2, Issue 11, EleTek Systems, 1992.

5. Becker, J. D. *Arabic Word Processing.* Communications of the ACM, Vol. 30, No. 7, July 1987, pp. 600-610.

6. Dadashzadeh, M., Dadfar, M. B., and Sanati, M. *Writing A Farsi Printer Driver for MS-DOS (Abstract).* In Proc. of First International Computer Conf. in Science, Technology, and Medicine, Isfahan, Iran, Dec. 26-28, 1991.

7. Hosseini, Anoosh. *Development of A Persian Graphical User Interface (Abstract).* In Proc. of First Int'l Computer Conf. in Science, Technology, and Medicine, Isfahan, Iran, Dec. 26-28, 1991.

8. Hyder, S. S. *A System for Generating Urdu/Farsi/Arabic Script.* Information Processing 71 (Proc. of IFIP Congress), North-Holland, Amsterdam, 1972.

9. International Business Machines, Inc. *Personal Computer National Supplement Arabic.* No. 8223445. 1985.

10. International Organization for Standardization. *Information Processing - 8-bit Single-byte Coded Graphic Character Sets - Part 6: Latin/Arabic Alphabet.* ISO/TC 97, ISO 8859-6 0, Tech. Rep., 1987.

11. International Organization for Standardization. *Information Processing - Arabic 7-bit Coded Character Set for Information Interchange.* ISO/TC 97, ISO 9036 Tech. Rep. 1987.

12. Knuth, Donald E. and MacKay, Pierre. *Mixing Right-to-Left Texts with Left-to-Right Texts.* TUGboat, Vol. 8 (1987), No. 1, pp. 14-17.

13. Kostopoulos, George K. *Design of A Universal Text Processing System and Its Handling of Farsi (Abstract).* In Proc. of First Int'l Computer Conf. in Science, Technology, and Medicine, Isfahan, Iran, Dec. 26-28, 1991.

14. MacKay, P. *Typesetting Problem Scripts.* Byte, Vol. 11, No. 2 (Feb. 1986), 201-218.

15. Manzer, M., and Mahmoud, N. N. *A Special Purpose Computer for Arabic Text Processing.* In Proceedings of Tenth National Computer Conference (Jeddah, Saudi Arabia, Feb. 28-March 2). 1988, pp. 679-688.

16. *Microsoft MS-DOS User's Guide Arabic Supplement.* Microsoft, 1988.

17. Modiri, Nasser. *How to Connect Iran to a Wide Area Network.* NOOR Vol. 2, Issue 12, EleTek Systems, 1992, pp. 521-527.

18. Mohammad-Zadeh, M. *Persian Operating System and Editor.* M.S. Thesis, Sharif University of Technology, Tehran, Iran, 1986.

19. Mohsen Zadeh, Mohammad and Kazemi, Saied. *A Persian Codeset for Information Interchange.* Parsis, Inc. April 10, 1992

20. *O1 Systems ITHA User's Guide and Advanced Programming Reference Rel. 3.0.* Bahrain, 1988.

21. Parhami, Behrooz. *Decomposition of Characters into Line Segments.* Proc. of the Seminar on Educational Problems and Teaching Methods for a First Course of Computer Science, Tehran, Oct. 1975, pp. 41-65.

22. Parhami, Behrooz. *Impact of the Farsi Language on Computing in Iran.* Mideast Computer, Vol. 1, No. 1, pp. 6-7, Sep. 18, 1978.

23. Parhami, Behrooz. *Language-Dependent Considerations for Computer Applications in Farsi and Arabic Speaking Countries.* System Approach for Development (Proc. of IFAC Conf.), North-Holland, Amsterdam, 1981, pp. 507- 513.

24. Parhami, Behrooz. *On the Use of Farsi and Arabic Languages in Computer- Based Information Systems.* Proc. of the Symp. on Linguistic Implications of Computer-Based Information Systems, New Delhi, India, Nov. 1978.

25. Parhami, Behrooz. *Optically Weighted Dot-Matrix Farsi and Arabic Numerals.* Information Technology 78 (Proc. of the Third Jerusalem Conference.), North-Holland, Amsterdam, Aug. 1978, pp. 207-210.

26. Parhami, Behrooz. *Standard Farsi Information Interchange Code and Keyboard Layout: A Unified Proposal.* Journal of the Institution of Electrical and Telecommunications Engineers, Vol. 30, No. 6, pp. 179-183, 1984.

27. Parhami, Behrooz and Mavaddat, F. *Computers and the Farsi Language: A Survey of Problem Areas.* Information Processing 77 (Proc. of IFIP Congress), North-Holland, Amsterdam, 1977, pp. 673-676.

28. Parhami, Behrooz and Taraghi, M. *Automatic Recognition of Printed Farsi Texts (Summary).* Proc. of the Conf. on Pattern Recognition, Oxford, England, Jan. 1980.

29. Parhami, Behrooz and Taraghi, M. *Automatic Recognition of Printed Farsi Texts.* Pattern Recognition, Vol. 14, Nos. 1-6, pp. 395-403, 1981.

30. Saad, H., Adnan, N., and Mohammad S. *Recent Experience in the Development of a Standard Arabic Information Processing Glossary.* In Proceedings of Eleventh National Computer Conference (Dharan, Saudi Arabia, Mar. 4-7). 1989, pp. 373-381.

31. Sanati, Mohammad, Dadashzadeh, M., and Dadfar, M. *Iranian Standard Code for Information Interchange (ISCII).* In Computer Standards & Interfaces, Dec. 1987.

32. Saudi Soft. *AL MUSSAED AL ARABI.* Jeddah, Saudi Arabia, 1986.

33. Smith, Ben. *Around the World in Text Displays.* Byte, Vol. 15, No. 5, May 1990, pp. 262-268.

34. Social.Culture.Iranian NewsGroup. *Iran's Connectivity to World-Wide Networks.* NOOR Vol. 2, Issue 7, EleTek Systems, 1992, pp. 370-372.

35. El-sheikh, T. S., and Guindi, R. M. *Computer Recognition of Arabic Cursive Scripts.* Pattern Recognition, Vol. 21, No. 4, pp. 293-302, 1988.

36. Tayli, Murat. *Integrated Arabic System.* In Proceedings of the First KSU Symp. on Computer Arabization (Riyadh, Saudi Arabia, Apr. 6-9). 1987, pp. 135-143.

37. Tayli, Murat. *Integrated Arabic System Technical Information and Programming Manual.* College of Computer & Inf. Sci., King Saud University, Saudi Arabia, 1988.

38. Tayli, Murat and Al-Salamah, Abdullah I. *Building Bilingual Microcomputer Systems.* Communications of the ACM, Vol. 33, No. 5, May 1990, pp. 495-504.

39. Tayli, Murat and Nafisah, M. *Intelligent Arabic Workstation.* In Proceedings of the Ninth National Computer Conference (Riyadh, Saudi Arabia, Sept. 23-27). 1986, pp. 10-2-1 to 10-2-8.

مقاله‌نامهٔ «کامپیوتر و زبان فارسی»

تهیه کننده: امین مهاجر

فهرست زیر شامل مقالاتی می‌شود که تاکنون درزمینهٔ سیستمهای فارسی، کارکرد دو زبانهٔ سیستمها و سایر مسائلی که به‌نحوی با کارکرد کامپیوتر در فارسی ارتباط داشته‌اند، نوشته شده‌است. بیشتر این مقالات در مجلات فارسی زبان کامپیوتری به‌چاپ رسیده و سعی بر آن بوده‌است که حتی مجلاتی که موضوع اصلی آنان کامپیوتر هم نیست بازبینی شوند. مقالات بدون در نظر گرفتن کیفیت آنها و تنها بر اساس کلید «کامپیوتر و زبان فارسی» انتخاب شده‌اند، گرچه ممکن است که عنوان مقاله‌ای خاص به‌طور کامل محتوای آنرا نرساند. فهرست زیر بر اساس نام پدید آورند گان مرتب گردیده‌است.

در صورتی که شما مقاله‌ای در زمینهٔ فوق نوشته‌اید که در جایی به‌چاپ رسیده، ولی نام آن در فهرست پایین به‌چشم نمی‌خورد، خواهشمندیم یک کپی از مقاله را به‌همراه آدرس تماس خود برای ما به نشانی قید شده بفرستید تا به‌فهرست اضافه شود. متن کامل بیشتر مقالات فهرست پایین جمع آوری شده است. برای دریافت رونوشت مقالهٔ دلخواه لطفا تقاضایی به‌همراه شماره و عنوان مقاله و مقداری تمبر باطل نشده (برای هزینه‌های پستی و تکثیر) به آدرس زیر ارسال کنید:
صندوق پستی ۱۶۱-۱۴۴۵۵
تهران

(۱) احمد زاده، محمد رضا و کبیر، احسان‌اله. «شکستن کلمات تایپ شدهٔ فارسی به حروف». گزارش اولین کنفرانس بین‌المللی کامپیوتر در علوم، فنون و پزشکی در ایران، ۷-۵ دی‌ماه ۱۳۷۰، صفحهٔ ۶-۱.
(۲) افشار مهاجر، کامران. «زیبایی شناسی حروف». صنعت چاپ، شمارهٔ ۱۱۳، اردیبهشت ۱۳۷۱، صفحهٔ ۳۶-۳۳.
(۳) امامی، کریم. «لزوم بازنگری در شیوه نگارش خط فارسی». آدینه، شمارهٔ ۷۴ و ۷۳، شهریور ۱۳۷۱، صفحهٔ ۱۹-۱۸.
(۴) باطنی، محمد رضا. «نگاهی تازه به شیوه خط فارسی». آدینه، شماره ۷۵، آبان ۱۳۷۱، صفحهٔ ۴۵-۴۴.
(۵) پرهامی، بهروز. «استاندارد پیشنهادی برای کد تبادل اطلاعات در فارسی». گزارش کامپیوتر، شمارهٔ ۱۵، شهریور ۱۳۵۹، صفحهٔ ۴-۳.
(۶) پرهامی، بهروز. «دربارهٔ علامتی برای نمایش واحد پول ایران». گزارش کامپیوتر، شمارهٔ ۵۲، فروردین و اردیبهشت ۱۳۶۳، صفحهٔ ۲۲.
(۷) پرهامی، بهروز. «شناسائی متون چاپی فارسی توسط کامپیوتر». گزارش کامپیوتر، شمارهٔ ۲۷، شهریور ۱۳۶۰، صفحهٔ ۱۱-۷.
(۸) پرهامی، بهروز. «مسائل مربوط به زبان و خط در کاربردهای کامپیوتر». گزارش کامپیوتر، شمارهٔ ۲۹، آبان ۱۳۶۰، صفحهٔ ۱۲-۵.
(۹) جابری پور، قاسم. «پردازش کار آ: تعیین کنندهٔ مهم در تدوین استاندارد رمز تبادل اطلاعات در فارسی». گزارش کامپیوتر، شمارهٔ ۷۶، اردیبهشت ۱۳۶۵، صفحهٔ ۲۳-۱۹.
(۱۰) جابری پور، قاسم. «فارسی کردن زبانهای برنامه‌سازی سطح بالا». گزارش کامپیوتر، شمارهٔ ۶۸، شهریور ۱۳۶۴، صفحهٔ ۷-۵.
(۱۱) جابری پور، قاسم. «یک راه‌حل ساده برای مرتب‌سازی اطلاعات فارسی». گزارش کامپیوتر، شمارهٔ ۷۴، اسفند ۱۳۶۴، صفحهٔ ۸-۳.
(۱۲) جذبی، محمد هاشم. «راه‌حلی در رابطه با چاپ اعداد فارسی به کمک بسته نرم‌افزاری EDIAC». گزارش کامپیوتر، شمارهٔ ۷۷، خرداد ۱۳۶۵، صفحهٔ ۱۰.
(۱۳) جذبی، محمد هاشم. «استفاده از امکانات چاپگر Epson FX-100 برای چاپ اعداد فارسی». گزارش کامپیوتر، شمارهٔ ۷۷، خرداد ۱۳۶۵، صفحهٔ ۱۷.

(۱۴) جم زاد، منصور. «برنامه‌ای برای تعیین معادلهای حرفی اعداد». گزارش کامپیوتر، شمارهٔ ۴۱، بهمن ۱۳۶۱، صفحهٔ ۶-۵.
(۱۵) جم زاد، منصور. «پایانهٔ فارسی با حداقل تعداد کلید». گزارش کامپیوتر، شمارهٔ ۳۴، اردیبهشت ۱۳۶۱، صفحهٔ ۶-۵.
(۱۶) جوان، داریوش. «طرح پیشنهادی زبان فارسی برای کامپیوتر». گزارش کامپیوتر، شمارهٔ ۲۲، فروردین ۱۳۶۰، صفحهٔ ۱۲.
(۱۷) حسیبیان، محمد رضا. «تغییر خط فارسی ماشینهای باروز سری اِل ۹۰۰۰». گزارش کامپیوتر، شمارهٔ ۴۰، آذر و دی ۱۳۶۱، صفحهٔ ۶-۵.
(۱۸) خلخالی، رشید. «برنامه متن‌نگار فارسی - لاتین برای اسپکتروم». علم الکترونیک و کامپیوتر، شمارهٔ ۱۸۷، خرداد ۷۱، صفحهٔ ۱۲-۶.
(۱۹) دبیرخانهٔ شورای عالی انفورماتیک کشور. «خبرنامهٔ انفورماتیک ویژهٔ سمینار نشر کامپیوتری». ویژه‌نامهٔ خبرنامهٔ انفورماتیک، سازمان برنامه و بودجه، اردیبهشت ۱۳۷۰، ۱۸۰ صفحه.
(۲۰) دوستی، پرویز. «چاپ خط درشت فارسی توسط کامپیوتر». گزارش کامپیوتر، شمارهٔ ۳۲، بهمن و اسفند ۱۳۶۰، صفحهٔ ۱۸-۱۷.
(۲۱) رهبر، حمید رضا. «کارکرد فارسی و دو زبانه بر روی کامپیوتر (قسمت اول)». گزارش کامپیوتر، شمارهٔ ۷۶، اردیبهشت ۱۳۶۵، صفحهٔ ۹.
(۲۲) رهبر، حمید رضا. «کارکرد فارسی و دو زبانه بر روی کامپیوتر (قسمت دوم)». گزارش کامپیوتر، شمارهٔ ۷۷، خرداد ۱۳۶۵، صفحهٔ ۱۶.
(۲۳) شربیانی، احمد. «تاریخچهٔ استفاده از کامپیوتر در صنعت چاپ و نشر». خبرنامهٔ انفورماتیک (ویژهٔ سمینار نشر کامپیوتری)، اردیبهشت ۱۳۷۰، صفحهٔ ۲۶-۷.
(۲۴) شربیانی، احمد. «نشر کامپیوتری». صنعت چاپ، شمارهٔ ۱۱۳، اردیبهشت ۱۳۷۱، صفحهٔ ۴۱-۳۷.
(۲۵) «صفحه کلیدهای فارسی». صنعت روز، صفحهٔ ۲۷-۲۴.
(۲۶) صنعتی، جعفر. «کد استاندارد برای تبادل اطلاعات در فارسی». سمینار شرکت داده‌پردازی، آذرماه ۱۳۶۴.
(۲۷) صنعتی، محمد. «دشواری‌های زبان فارسی با کامپیوتر». آدینه، شمارهٔ ۷۲، مرداد ۱۳۷۱، صفحهٔ ۵۷-۵۶.
(۲۸) صنعتی، محمد. «کد استاندارد برای تبادل اطلاعات در فارسی». گزارش کامپیوتر، شمارهٔ ۷۶، اردیبهشت ۱۳۶۵، صفحهٔ ۱۸-۱۵.
(۲۹) صنعتی، محمد. «نرم‌افزارهای نشر کامپیوتری: ویژگی‌ها و امکانات». صنعت چاپ، شمارهٔ ۱۱۳، اردیبهشت ۱۳۷۱، صفحهٔ ۲۵-۲۳.
(۳۰) «طراحی قلم‌ها». صنعت چاپ، شمارهٔ ۱۱۳، اردیبهشت ۱۳۷۱، صفحهٔ ۲۹-۲۸.
(۳۱) عبدالله‌زاده حقیقی، حسین. «طراحی حروف فارسی». صنعت چاپ، شمارهٔ ۱۱۳، اردیبهشت ۱۳۷۱، صفحهٔ ۳۰.
(۳۲) «فارسی کردن اسپکتروم». علم الکترونیک و کامپیوتر، شمارهٔ ۱۱۱، شهریور ۱۳۶۸، صفحهٔ ۵۴.
(۳۳) فیروزبخت، حسین و انزانی، امیر اسعد. «طرح چند زبانه کردن چاپگرهای کامپیوتری اِل‌ای ۳۶ و اِل‌ای ۱۸۰». گزارش کامپیوتر، شمارهٔ ۳۵، خرداد ۱۳۶۱، صفحهٔ ۱۴-۱۳.
(۳۴) «فونت فارسی کمودور ۶۴». علم الکترونیک و کامپیوتر، سال ۱۶، شمارهٔ ۱۹۱، صفحه ۵ تا ۷، مرداد ۱۳۷۱.
(۳۵) کابلی، ایرج. «فراخوان به فارسی‌نویسان و پیش‌نهاد به تاجیکان». آدینه، شمارهٔ ۷۲، مرداد ۱۳۷۱، صفحهٔ ۵۵-۴۹.
(۳۶) کمیتهٔ مطالعاتی استاندارد تبادل اطلاعات در فارسی. «پیشنهاد نهائی کد استاندارد تبادل اطلاعات در فارسی». گزارش کامپیوتر، شمارهٔ ۷۶، اردیبهشت ۱۳۶۵، صفحهٔ ۷-۳.
(۳۷) کمیتهٔ مطالعاتی استاندارد تبادل اطلاعات در فارسی. «مختصری دربارهٔ روش استاندارد برای توسعه کد ISO». گزارش کامپیوتر، شمارهٔ ۷۶، اردیبهشت ۱۳۶۵، صفحهٔ ۹-۸.
(۳۸) کمیتهٔ مطالعاتی استاندارد تبادل اطلاعات در فارسی. «مختصری دربارهٔ کد تک‌نمادی برای الفبای فارسی». گزارش کامپیوتر، شمارهٔ ۷۶، اردیبهشت ۱۳۶۵، صفحهٔ ۱۰.
(۳۹) کمیتهٔ مطالعاتی استاندارد تبادل اطلاعات در فارسی. «نکاتی دربارهٔ طرح صفحه کلیدهای فارسی و دوزبانه». گزارش کامپیوتر، شمارهٔ ۷۶، اردیبهشت ۱۳۶۵، صفحهٔ ۱۴-۱۳.
(۴۰) کمیتهٔ مطالعاتی استاندارد تبادل اطلاعات در فارسی. «نکاتی دربارهٔ نمایش علائم فارسی در دستگاههای خروجی».

گزارش کامپیوتر، شمارهٔ ٧٦، اردیبهشت ١٣٦٥، صفحهٔ ١٢-١١.

(٤١) محمدی‌فر، محمد رضا. «زبان و اطلاعات». مجلهٔ زبانشناسی، سال ششم، شمارهٔ اول، پیاپی ١١، دی ١٣٦٨.

(٤٢) محمدی‌فر، محمد رضا. «کامپیوتر در زبانشناسی (١)». ریزپردازنده، شمارهٔ ١، مرداد ١٣٦٩، صفحهٔ ١١-٩.

(٤٣) موسسهٔ استاندارد و تحقیقات صنعتی ایران. «حروف فارسی در ماشینهای تحریر». استاندارد شمارهٔ ٨٢٠، آذرماه ١٣٥٥.

(٤٤) موسسهٔ استاندارد و تحقیقات صنعتی ایران. «طرز قرار گرفتن حروف و علائم زبان فارسی بر روی صفحه کلید کامپیوتر». استاندارد شمارهٔ ٢٩٠١، چاپ دوم، مرداد ماه ١٣٦٨.

(٤٥) موسسهٔ استاندارد و تحقیقات صنعتی ایران. «کد تبادل اطلاعات به‌زبان فارسی». شمارهٔ استاندارد ٢٩٠٠، چاپ دوم، مرداد ماه ١٣٦٨.

(٤٦) میرزا حسابی، محمد حسین. «زبانهای برنامه‌نویسی فارسی». گزارش کامپیوتر، شمارهٔ ٨١، آذر ١٣٦٥، صفحهٔ ٢٠.

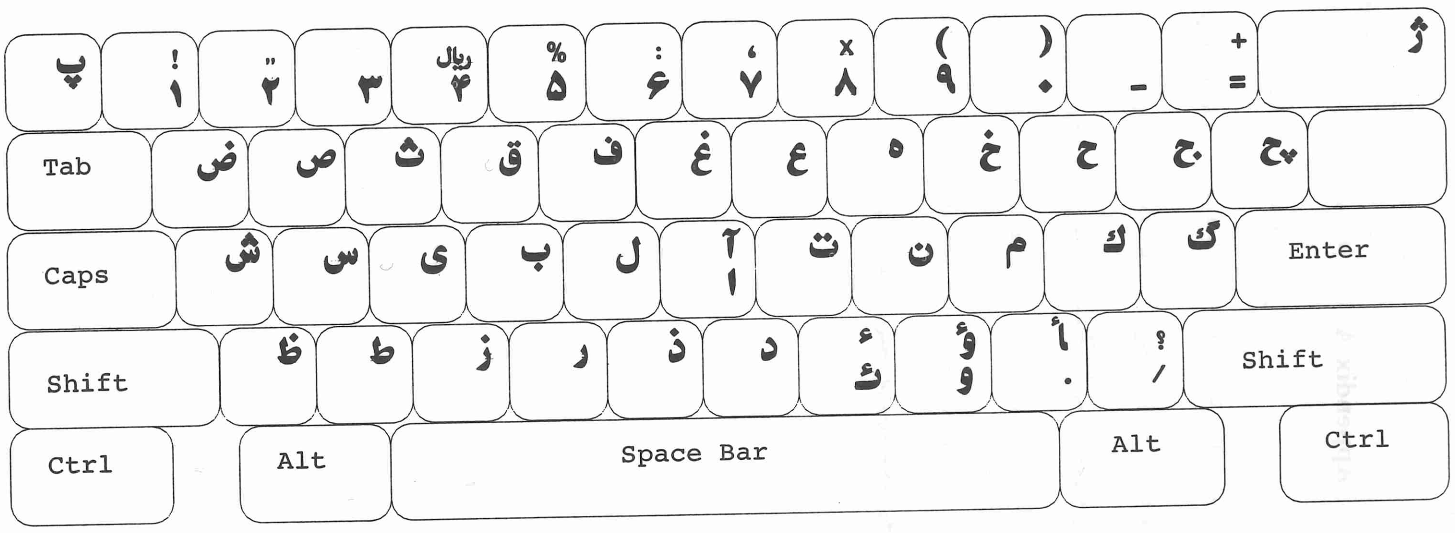
(٤٧) میرزا عبداللهی، محمد. «ترجمهٔ متن‌های سادهٔ علمی از انگلیسی به فارسی بوسیلهٔ کامپیوتر». گزارش کامپیوتر، شمارهٔ ٨، بهمن ١٣٥٨، صفحهٔ ١٠.

(٤٨) «نگارش جدید سیستم دوزبانهٔ فارسی - لاتین پانیذ (نگارش ٥)». علم الکترونیک و کامپیوتر، شمارهٔ ١٨٥، اردیبهشت ٧١، صفحهٔ ٣١-٣٠.

(٤٩) هال، پاتریک. ترجمهٔ پرهامی، بهروز. «کامپیوتر و کشورهای عرب‌زبان». گزارش کامپیوتر، شمارهٔ ٢٥، تیر ١٣٦٠، صفحهٔ ١٦-١٥.

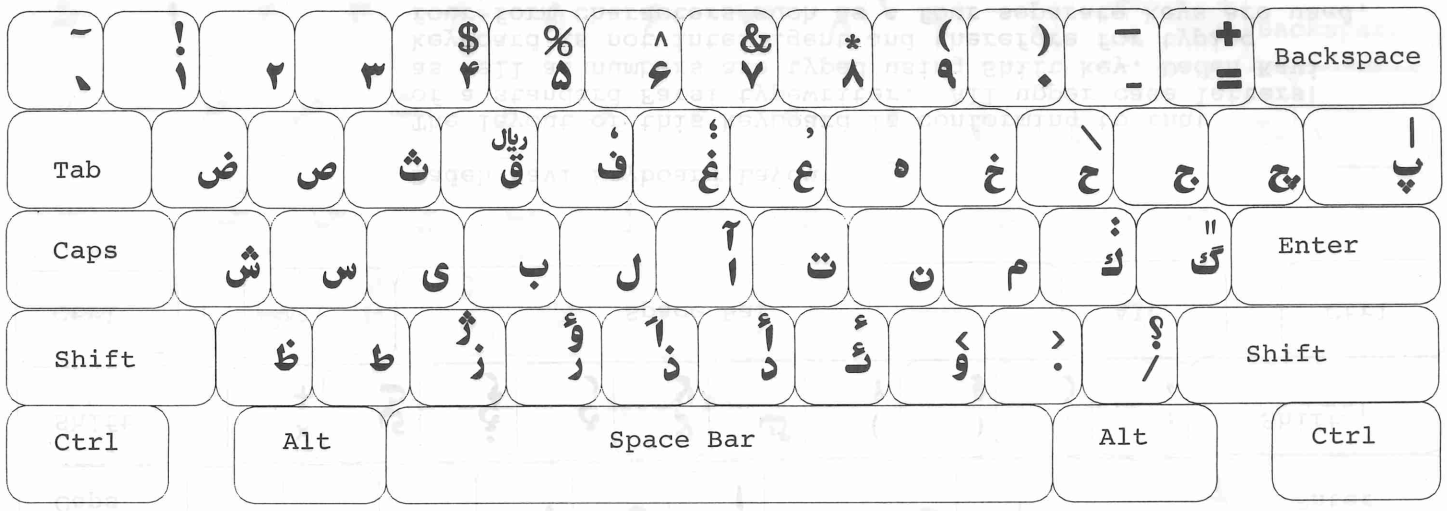
(٥٠) هروی، ایرج. «شیوه‌ها و ابزار طراحی فونت فارسی». خبرنامهٔ انفورماتیک (ویژهٔ سمینار نشر کامپیوتری)، اردیبهشت ١٣٧٠، صفحهٔ ١٢٤-١٢١.

Appendix A



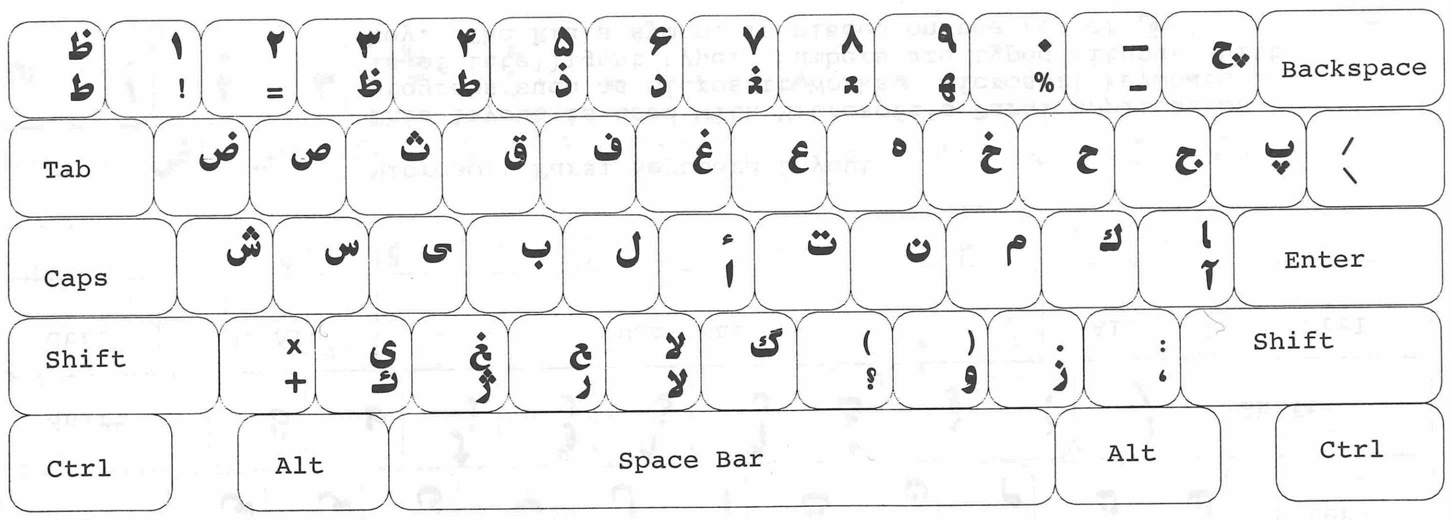
Standard 2900 Keyboard Layout

This layout is the most widely used keyboard layout today. Iran System is also using this layout. However, rumors are heard about possible changes to be made in this layout by Iranian national standard organization.



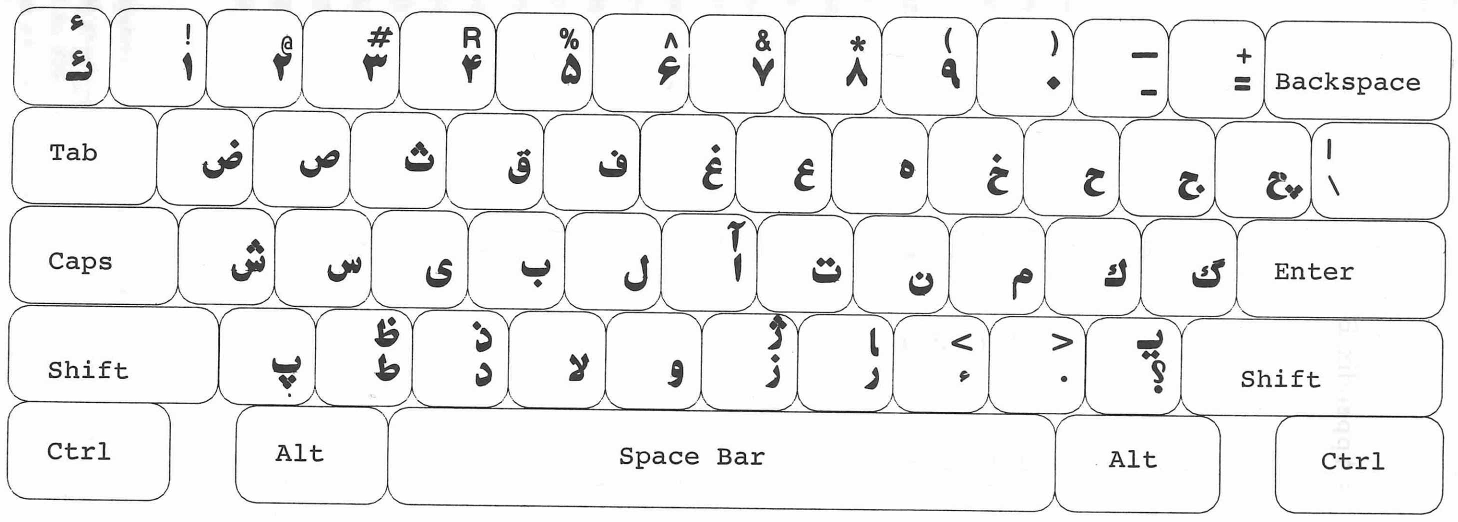
Microsoft Farsi Keyboard Layout

This layout is used with Microsoft's Farsi application programs such as Microsoft Works. Microsoft keyboard is of intelligent type. Numbers are typed without Shift key. The Rials symbol is placed on the top of ق .



Dadeh Kavi Keyboard Layout

The layout of this keyboard is conforming to that of a standard Farsi typewriter. All upper case letters as well as numbers are typed using Shift key. Dadeh Kavi keyboard is not intelligent and therefore for typing four-form characters such as ع four separate keys are used.



Rayaneh Saz Keyboard Layout

Rayaneh Saz keyboard is of intelligent type. Numbers are being typed without Shift key. This layout is not of a common use.

Appendix B

128	٠	160	خ	192		224	ظ
129	١	161	خـ	193		225	ع
130	٢	162	د	194		226	ـع
131	٣	163	ذ	195		227	ـعـ
132	٤	164	ر	196		228	عـ
133	٥	165	ز	197		229	غ
134	٦	166	ژ	198		230	ـغ
135	٧	167	س	199		231	ـغـ
136	٨	168	سـ	200		232	غـ
137	٩	169	ش	201		233	ف
138	،	170	شـ	202		234	فـ
139	ـ	171	ص	203		235	ق
140	؟	172	صـ	204		236	قـ
141	آ	173	ض	205		237	ک
142	ئـ	174	ضـ	206		238	کـ
143	ء	175	ط	207		239	گ
144	ا	176		208	Graphical Characters	240	گـ
145	ـا	177		209		241	ل
146	ب	178		210		242	لا
147	بـ	179		211		243	لـ
148	پ	180		212		244	م
149	پـ	181		213		245	مـ
150	ت	182		214		246	ن
151	تـ	183	Graphical Characters	215		247	نـ
152	ث	184		216		248	و
153	ثـ	185		217		249	ه
154	ج	186		218		250	ـهـ
155	جـ	187		219		251	هـ
156	چ	188		220		252	ـی
157	چـ	189		221		253	ی
158	ح	190		222		254	یـ
159	حـ	191		223		255	

Iran System Farsi Character Set

Original graphical characters are remained preserved in this character set but no code has been allotted to Rials symbol. It is believed that Iran System character set is the most widely-used character set today. All possible four cases of a Farsi letter has been built into this codeset.

128	!	160	ٮ	192		224	ط
129	"	161	خ	193		225	ظ
130	x	162	خـ	194		226	ع
131	—	163	د	195		227	ـع
132	%	164	ذ	196		228	ـعـ
133	:	165	ر	197		229	عـ
134	؟	166	ز	198		230	غ
135	،	167	ژ	199		231	ـغ
136	/	168	س	200		232	ـغـ
137	آ	169	سـ	201		233	غـ
138	أ	170	ش	202		234	فـ
139	ا	171	شـ	203		235	ق
140	ـا	172	ص	204		236	قـ
141	ء	173	صـ	205		237	کـ
142	ـأ	174	ض	206		238	گـ
143	ؤ	175	ضـ	207		239	ل
144	ٮ	176		208	Graphical Characters	240	لا
145	ئـ	177		209		241	لـ
146	ب	178		210		242	م
147	بـ	179		211		243	مـ
148	پ	180		212		244	ن
149	پـ	181		213		245	نـ
150	ت	182	Graphical Characters	214		246	و
151	تـ	183		215		247	ه
152	ث	184		216		248	ـهـ
153	ثـ	185		217		249	ـهـ
154	ج	186		218		250	هـ
155	جـ	187		219		251	ی
156	چ	188		220		252	ـی
157	چـ	189		221		253	یـ
158	ح	190		222		254	یـ
159	حـ	191		223		255	

SinaSoft Farsi Character Set

In this codeset, those upper-case Farsi letters with a "tail" at the end are being represented by a combination of two characters (e.g. character "ش" is displayed when character "ٮ" (code 160) and character "ش" (code 170) are joined together. SinaSoft character set has preserved graphical characters residing in the second half of the code but has no code allotted to Rials symbol.

128	٠	160	حـ	192		224	ط
129	١	161	خ	193		225	ظ
130	٢	162	خـ	194		226	ع
131	٣	163	د	195		227	ـع
132	٤	164	ذ	196		228	ـعـ
133	٥	165	ر	197		229	عـ
134	٦	166	ز	198		230	غ
135	٧	167	ژ	199		231	ـغ
136	٨	168	س	200		232	ـغـ
137	٩	169	سـ	201		233	غـ
138	x	170	ش	202		234	ف
139	؟	171	شـ	203		235	فـ
140	R	172	ص	204		236	ق
141	ء	173	صـ	205		237	قـ
142	ـٔ	174	ض	206		238	ک
143	ئـ	175	ضـ	207		239	کـ
144	آ	176		208	Graphical Characters	240	گ
145	ا	177		209		241	گـ
146	ـا	178		210		242	ل
147	ب	179		211		243	لـ
148	بـ	180		212		244	لا
149	پ	181		213		245	م
150	پـ	182	Graphical Characters	214		246	مـ
151	ت	183		215		247	ن
152	تـ	184		216		248	نـ
153	ث	185		217		249	و
154	ثـ	186		218		250	ه
155	ج	187		219		251	هـ
156	جـ	188		220		252	ی
157	چ	189		221		253	ـی
158	چـ	190		222		254	یـ
159	ح	191		223		255	

Rayaneh Saz Farsi Character Set

Developed by Rayaneh Saz Co., this codeset is not of a general use. Graphical characters are preserved and there is a symbol for representing Rials. With just one code difference, Rayaneh Saz codeset is conforming to Iran System Character set. All possible four cases of a Farsi character have been built into this codeset.

000	NUL	032	SP	064	بـ	096	ضـ
001	SOH	033	!	065	ب	097	ض
002	STX	034	"	066	پـ	098	ط
003	ETX	035	=	067	پ	099	ط
004	EOT	036	ریال	068	تـ	100	ظ
005	ENQ	037	%	069	ت	101	ظ
006	ACK	038	:	070	ثـ	102	عـ
007	BEL	039	؟	071	ث	103	ع
008	BS	040	(	072	جـ	104	غـ
009	HT	041	)	073	ج	105	غ
010	LF	042	x	074	چـ	106	فـ
011	VT	043	+	075	چ	107	ف
012	FF	044	،	076	حـ	108	قـ
013	CR	045	–	077	ح	109	ق
014	SO	046	.	078	خـ	110	کـ
015	SI	047	/	079	خ	111	ک
016	DLE	048	٠	080	د	112	گـ
017	DC1	049	١	081	د	113	گ
018	DC2	050	٢	082	ذ	114	لـ
019	DC3	051	٣	083	ذ	115	ل
020	DC4	052	٤	084	ر	116	مـ
021	NAK	053	٥	085	ر	117	م
022	SYN	054	٦	086	ز	118	نـ
023	ETB	055	٧	087	ز	119	ن
024	CAN	056	٨	088	ژ	120	و
025	EM	057	٩	089	ژ	121	و
026	SUB	058	آ	090	سـ	122	هـ
027	ESC	059	ا	091	س	123	ه
028	FS	060	ء	092	شـ	124	يـ
029	GS	061	ـأ	093	ش	125	ی
030	RS	062	ؤ	094	صـ	126	NA
031	US	063	ئـ	095	ص	127	DEL

Iranin Standard Code for Information Interchange (Standard 2900)

This is a 7-bit character set and is not in use today. Just two out of four possible cases that a Farsi letter might take are included in this codeset. The first 32 codes are control characters. The character set is complying with ISO extension techniques for 7-bit codesets. Using control characters SI and SO, codeset G1 (this codeset) or codeset G0 (origianl 7-bit ASCII codeset) can be activated.

000	NUL	032	SP	064	@	096	‘
001	SOH	033	!	065	ء	097	ف
002	STX	034	"	066	آ	098	ق
003	ETX	035	#	067	ا	099	ک
004	EOT	036	ریال	068	أ	100	گ
005	ENQ	037	%	069	أ	101	ل
006	ACK	038	&	070	ب	102	م
007	BEL	039	’	071	پ	103	ن
008	BS	040	)	072	ت	104	و
009	HT	041	(	073	ث	105	ؤ
010	LF	042	*	074	ج	106	هـ
011	VT	043	+	075	چ	107	ـۀ
012	FF	044	،	076	ح	108	ىٰ
013	CR	045	–	077	خ	109	ی
014	SO	046	.	078	د	110	ئ
015	SI	047	/	079	ذ	111	–
016	DLE	048	٠	080	ر	112	ـَ
017	DC1	049	١	081	ز	113	ـُ
018	DC2	050	٢	082	ژ	114	
019	DC3	051	٣	083	س	115	ـً
020	DC4	052	٤	084	ش	116	ـْ
021	NAK	053	٥	085	ص	117	
022	SYN	054	٦	086	ض	118	
023	ETB	055	٧	087	ط	119	
024	CAN	.056	٨	088	ظ	120	
025	EM	057	٩	089	ع	121	HSP
026	SUB	058	:	090	غ	122	NSP
027	ESC	059	؛	091	]	123	}
028	FS	060	>	092	\	124	|
029	GS	061	=	093	[	125	{
030	RS	062	<	094	^	126	~
031	US	063	؟	095	–	127	

alis Farsi Character Set

Like ISCII, alis codeset is a 7-bit character set. The first 32 codes are allotted to control characters. Each Farsi letter has been defined just once and has occupied just one code space (one out of four possible cases). Codes are also allotted to diacritical marks.

128		160	FSP	192	بـ	224	ع
129	@	161	/	193	پ	225	عـ
130	*	162	.	194	پـ	226	ـع
131	R	163	؟	195	ت	227	ـعـ
132	ـ	164	!	196	تـ	228	غ
133	ا	165	:	197	ث	229	غـ
134	~	166	؛	198	ثـ	230	ـغ
135	&	167	،	199	ج	231	ـغـ
136		168	‘	200	جـ	232	ف
137		169	’	201	چ	233	فـ
138		170		202	چـ	234	ق
139	{	171	ـً	203	ح	235	قـ
140	}	172	ـَ	204	حـ	236	ک
141		173	ـُ	205	خ	237	کـ
142	«	174	ˆ	206	خـ	238	گ
143	»	175	¦	207	د	239	گـ
144	(	176	٠	208	ذ	240	ل
145	)	177	١	209	ر	241	لا
146	[	178	٢	210	ز	242	لـ
147	]	179	٣	211	ژ	243	م
148	—	180	٤	212	س	244	مـ
149	#	181	٥	213	سـ	245	ن
150	%	182	٦	214	ش	246	نـ
151	,	183	٧	215	شـ	247	و
152	<	184	٨	216	ص	248	ه
153	=	185	٩	217	صـ	249	هـ
154	>	186	ـّ	218	ض	250	ـلا
155	\	187	آ	219	ضـ	251	ـهـ
156	x	188	ء	220	ط	252	ی
157	\	189	ا	221	طـ	253	یـ
158	+	190	ـا	222	ظ	254	ـی
159	–	191	ب	223	ظـ	255	ئـ

Dadeh Kavi Farsi Character Set

TEX–e–paarsi (Farsi version of TEX) developed by Dadeh Kavi of Iran is based on this character set. There are also minor changes in the first half of the codes. Graphical characters in the second half have been overwritten. Code 160 is allotted to Farsi space (FSP). Codes not allotted to Farsi characters are reserved for future extensions.

128	NSP	160		192		224	چ
129	LSP	161		193		225	ح
130		162		194		226	خ
131		163		195		227	د
132	FSP	164	ء	196		228	ذ
133		165	آ	197		229	ر
134	BDG	166	ا	198		230	ز
135		167	أ	199		231	ژ
136		168	أ	200		232	س
137	ريال	169	ب	201		233	ش
138		170	پ	202		234	ص
139		171	ت	203		235	ض
140		172	ث	204		236	ط
141	SLB	173	ج	205		237	ظ
142	SFB	174		206		238	ع
143		175		207	Unchanged	239	غ
144		176		208		240	
145		177		209		241	ف
146		178		210		242	ق
147		179		211		243	ک
148		180		212		244	گ
149		181		213		245	ل
150		182		214		246	م
151	Unchanged	183	Unchanged	215		247	ن
152		184		216		248	و
153		185		217		249	ؤ
154		186		218		250	.
155		187		219		251	ه
156		188		220		252	ی
157		189		221		253	ئـ
158		190		222		254	
159		191		223		255	

Microsoft Farsi Code

This character set is used by Microsoft for the purpose of storing and retrieving Farsi data but not for representation of Farsi information on the screen [see next pages]. Only one out of four possible cases that a Farsi letter might take are included in this code. It is up to algorithms to determine the proper shape of character on screen in accordance with neighbouring characters. Codes 143 to 163 and 174 to 223 are remained unchanged.

000	NUL	032		064		096	
001	٠	033		065		097	
002	١	034		066		098	
003	٢	035		067		099	
004	٣	036		068		100	
005	٤	037		069		101	
006	٥	038		070		102	
007	٦	039		071		103	
008	٧	040		072		104	
009	٨	041		073		105	
010	٩	042		074		106	
011	سر	043		075		107	
012	شر	044		076		108	
013	صر	045		077		109	
014	ضر	046		078		110	
015	ـ	047		079		111	
016	DLE	048		080		112	
017	DC1	049	Unchanged	081	Unchanged	113	Unchanged
018	DC2	050		082		114	
019	ریال	051		083		115	
020	DC4	052		084		116	
021	؟	053		085		117	
022	،	054		086		118	
023	؛	055		087		119	
024	CAN	056		088		120	
025	EM	057		089		121	
026	SUB	058		090		122	
027	ESC	059		091		123	
028	ى	060		092		124	
029	GS	061		093		125	
030	RS	062		094		126	
031	US	063		095		127	

Microsoft Farsi Font

This character set is a version of Microsoft Arabic "Naskh" font. This codeset is used for the sole purpose of displaying information on the screen. This is the only character set that has placed Farsi character both in the first and second half of code table. Graphical character are preserved, although some of control characters (codes 0 to 32) are overwritten.

128	ء	160	شـ	192		224	ـغـ
129	آ	161	ص	193		225	ف
130	ا	162	صـ	194		226	فـ
131	ـا	163	ض	195		227	ق
132	أ	164	ضـ	196		228	قـ
133	ـأ	165	ط	197		229	کـ
134	أ	166	ظ	198		230	ک
135	ـأ	167	ع	199		231	گـ
136	ب	168	ـع	200		232	گ
137	بـ	169	عـ	201		233	ل
138	پ	170	ـعـ	202		234	لـ
139	پـ	171	غ	203		235	م
140	ت	172	ـغ	204		236	مـ
141	تـ	173	غـ	205		237	ن
142	ث	174		206		238	نـ
143	ثـ	175		207		239	و
144	ج	176		208	Graphical Characters	240	
145	جـ	177		209		241	ؤ
146	چ	178		210		242	ه
147	چـ	179		211		243	هـ
148	ح	180		212		244	ـهـ
149	حـ	181		213		245	ی
150	خ	182	Graphical Characters	214		246	ی
151	خـ	183		215		247	يـ
152	د	184		216		248	ئـ
153	ذ	185		217		249	ﻻ
154	ر	186		218		250	.
155	ز	187		219		251	ﻷ
156	ژ	188		220		252	ﻵ
157	س	189		221		253	لـ
158	سـ	190		222		254	
159	ش	191		223		255	ـلـ

Microsoft Farsi Font (continued)

128	NUL	160	SP	192	آ	224	ه
129	SOH	161	PSP	193	ا	225	ی
130	STX	162	PCN	194	ء	226	[
131	ETX	163	!	195	ب	227	]
132	EOT	164	ریال	196	پ	228	{
133	ENQ	165	%	197	ت	229	}
134	ACK	166	.	198	ث	230	«
135	BEL	167	,	199	ج	231	»
136	BS	168	(	200	چ	232	*
137	HT	169	)	201	ح	233	_
138	LF	170	x	202	خ	234	|
139	VT	171	+	203	د	235	\
140	FF	172	،	204	ذ	236	
141	CR	173	-	205	ر	237	
142	SO	174	/	206	ز	238	
143	SI	175	/	207	ژ	239	
144	DLE	176	۰	208	س	240	ـَ
145	DC1	177	۱	209	ش	241	ـِ
146	DC2	178	۲	210	ص	242	ـُ
147	DC3	179	۳	211	ض	243	ـً
148	DC4	180	٤	212	ط	244	ـٌ
149	NAK	181	٥	213	ظ	245	ـٍ
150	SYN	182	٦	214	ع	246	ـّ
151	ETB	183	۷	215	غ	247	ـْ
152	CAN	184	۸	216	ف	248	
153	EM	185	۹	217	ق	249	
154	SUB	186	:	218	ک	250	
155	ESC	187	؛	219	گ	251	
156	FS	188	<	220	ل	252	
157	GS	189	=	221	م	253	
158	RS	190	>	222	ن	254	
159	US	191	؟	223	و	255	DEL

G1 Table of 8-bit Farsi Standard Character Set

This code set is to replace 7-bit ISCII for Farsi information interchange in an 8-bit environment. Just one out of four possible cases that a Farsi letter might take, are included in this code set since the sole purpose of this standard will be for Farsi data storage and retrieval. This standard can be seen as a protocol for Farsi data communication in a LAN or WAN and allows vendors to use the character set of their own for the representaion of Farsi data on the screens.

128	NUL	160	SP	192	آ	224	ه
129	SOH	161	PSP	193	ا	225	ی
130	STX	162	PCN	194	ء	226	[
131	ETX	163	!	195	ب	227	]
132	EOT	164	ریال	196	پ	228	{
133	ENQ	165	%	197	ت	229	}
134	ACK	166	.	198	ث	230	«
135	BEL	167	,	199	ج	231	»
136	BS	168	(	200	چ	232	*
137	HT	169	)	201	ح	233	_
138	LF	170	x	202	خ	234	|
139	VT	171	+	203	د	235	\
140	FF	172	،	204	ذ	236	
141	CR	173	-	205	ر	237	
142	SO	174	/	206	ز	238	
143	SI	175	/	207	ژ	239	
144	DLE	176	۰	208	س	240	ـَ
145	DC1	177	۱	209	ش	241	ـِ
146	DC2	178	۲	210	ص	242	ـُ
147	DC3	179	۳	211	ض	243	ـً
148	DC4	180	٤	212	ط	244	ـٌ
149	NAK	181	٥	213	ظ	245	ـٍ
150	SYN	182	٦	214	ع	246	ـّ
151	ETB	183	۷	215	غ	247	ـْ
152	CAN	184	۸	216	ف	248	إ
153	EM	185	۹	217	ق	249	ة
154	SUB	186	:	218	ک	250	ك
155	ESC	187	؛	219	گ	251	ى
156	FS	188	<	220	ل	252	
157	GS	189	=	221	م	253	
158	RS	190	>	222	ن	254	
159	US	191	؟	223	و	255	DEL

G2 Table of 8-bit Farsi Standard Character Set

The G2 table has four characters more than G1 [see previous page] to offer support for 8-bit Arabic Standard character set, ISO 8859/6.

NOOR

A Technical Newsletter for Iranian Universities

NOOR is a monthly technical publication designed to promote and assist the transfer of technical knowledge and scientific information to Iranian Universities. NOOR was founded in 1991 by Dr. Nasser Modiri for the sole purpose of communication of the latest trends and issues in technology and science to the Iranian technical and scientific community. NOOR in no way is associated with any religious, political or profit-seeking venture nor does it endorse any such activities.

Some of the regular Iranian universities which receive NOOR at this time are: Uo Isfahan, Uo Shiraz, Uo Orumiyeh, Uo Sharif, Uo Ferdowsi, Uo Bu-Ali Sina, and Uo Science and Technology.

Contributors to NOOR present a wide variety of backgrounds, for example, Computer Communications, Computers Graphics, Computer Languages, Architectures and Compilers for High Performance Systems, Software Testing and Automated Test Tools, Software Production Processes, Quality Engineering, Testware Engineering, System Theory and Feedback Analysis, Robot and Machine Vision, and Medical Image Processing.

To receive a single **Free** copy of the NOOR, please write to:

Dr. Nasser Modiri
1107 Second Ave., Ste. 613
Redwood City, CA 94063, USA.
(415) 369-8967

Farsi Computing Review
سردبیر: امین مهاجر (۹۸۰۱۰۲)

ویژه‌نامهٔ **گزارش کامپیوتر**
ماهنامهٔ انجمن انفورماتیک ایران
صاحب‌امتیاز : انجمن انفورماتیک ایران
مدیر مسئول : ابراهیم نقیب‌زاده مشایخ
سردبیر : علی پارسا

صفحه‌بندی و حروفچینی : امین مهاجر
لیتوگرافی : **کوروش**
چاپ : چاپخانهٔ مروی
دفتر انجمن: خیابان دکتر فاطمی، خیابان پروین اعتصامی، کوچه دوم، پلاک ۲۱، طبقهٔ سوم.
ساعات کار دفتر: هرروز بجز پنج‌شنبه‌ها از ۹ تا ۱۶.
شمارهٔ تلفن دفتر: ۶۵۶۶۸۷
شمارهٔ حساب بانکی انجمن: ۱۱۱۷ بانک صادرات شعبه ۱۶۵۲
نشانی پستی انجمن: انجمن انفورماتیک ایران، صندوق پستی ۱۱۹۶-۱۴۱۵۵ تهران.
برای اشتراک داخل کشور این ویژه‌نامه با نشانی پستی انجمن مکاتبه کنید.

CALL FOR PAPERS AND CONTRIBUTIONS

Farsi Computing Review

A Quarterly Journal on Farsi Computing

Scope

A large number of countries, notably in the Middle East, have been trying to adopt the computer systems initially designed for the English language and Roman characters to the needs of their native tongues. Although, with the support of western countries, the Arabization of computer systems was successfully achieved, such a support was not offered to Farsi users.

It took Farsi application developers years to come up with various techniques and algorithms to solve this problem, and of course, it was led to incompatibility and lack of standards among software programs. Part of this dilemma was due to lack of reliable and documented sources of information such as books, magazines, journals, and papers.

Farsi Computing Review provides opportunity for researchers, developers and users of Farsi systems to share findings, exchange ideas and experiences, and establish ties with each other.

It provides insiders of industry and academics with a reliable flow of information. The materials printed in journal are relevant for many domains. *Farsi Computing Review* will be of interest to a broad spectrum of professionals ranging from theoreticians to system and application developers, from researchers to authors and end-users.

Topics

We encourage innovative submissions in any area concerned with Farsi Computing research development and practice. The language of journal is in English. The areas of interest include but are not limited to the following

- *Modification of software programs for working in Farsi*
- *Standadization and compatibility of Farsi packages*
- *Farsi computing environments*
- *Design of algorithms for Farsi applications*
- *Iranizing hardware platforms and peripherals*
- *Farsi word processing and desktop publishing*
- *Farsi font generation (on screen and printout)*
- *Farsi character sequencing*
- *Conversion utilities for Farsi applications*
- *Graphical-based Farsi packages*
- *Farsi software development toolkits*

Additional Information

For more information or to receive future mailings and submission guidelines, please contact:

Amin Mohadjer
P.O. Box 14455-161
Tehran, Iran
Phone/Fax: +98 21 980102

Published by
BRAIN Computer Systems Group in co-operation with
Informatics Society of Iran