*[This document was presented as a set of slides. It is reproduced here in general using a smaller font.]*

# Japan's Proposal to FORTRAN8x
# -- Extension of CHARACTER Type for National Character Handling --

CONTENTS



FORTRAN WG of Japanese National Committee
for ISO/TC97/SC22

August , 1987

---

## Needs for various kinds of codes

- In the world, there are large number of people who wish to use FORTRAN and their native language is not English. Their needs for the handling of the national character code are strong.

- KANJI is now widely used on computers.
  - in Japan, recent innovations in word processing facilitated the KANJI input/output problem.
  - There are more than 10,000 characters in KANJI. Therefore KANJI cannot be mapped into any single byte code.

Reasons to be intrinsic, not derived

- Efficiency
  - We wish to process codes efficiently for our own language.

- Availability
  - We need the conformance of specifications between one byte code and multi byte codes.
  - We wish to manipulate not only KANJI hut also other kinds of characters(e.g. ASCII data) in the same program.

- Japan's experience of implementation
  - In Japan, most vendors have already implemented the data type for KANJI and this feature has been supplied to users, This implementation is successfully carried out as the intrinsic data type for FORTRAN77.
  - New character type and its constant has necessarily been adopted.

---

Language Specifications (1/3)

- New CHARACTER type and new CHARACTER constant
  - A processor may have a number of character sets, each is associated to a distinct character type. (For the present the number would be one or two.) The first of them (current one) shall include the FORTRAN character set, and others shall include the blank character.
  - The length of a character string is not the number of occupied bytes but the number of characters in the string.
  - User need not know and need not handle escape sequences.

- Operations and assignments are also applied to the new character type similarly to the current character type in general; Some of such features are processor dependent. (e.g. collating sequence)

Language Specifications (2/3)

- CHARACTER type-statement

R502 type-spec is ~
or CHARACTER [kind-length-selector]
or ~ *[editor's note: this looks like a typo]*

R508 kind-length-selector is(type-param-value)
or (LEN=type-param-value [,KIND=char-kind ] )
or (KIND=char-kind [,LEN=type-param-value ] )
or * char-length [, ]

R5xx char-kind is scalar-int~constant

The default char-kind is one.

The relationship between the value of a char-kind and the associated character set is processor-dependent. (The programmer would be allowed to change this correspondence through a compiler parameter, if the processor can handle multiple character sets.)

ex: CHARACTER (KIND=2) A, B

---

Language Specifications (3/3)

- Character Constant

R414 char-literal-constant is [char-kind ] ' [character ] ...'
or [char-kind ] " [character ] ..."

R5xx char-kind is scalar-int-constant

char-kind shall represent the kind of the contained characters.
char-kind may be omitted only when char-kind is one.

```
ex:  'lst kind'    ! lst kind character constant
    2'第２種'      ! 2nd kind character constant
    1'      '      ! lst kind black
```

- Character Constant Edit Descriptor

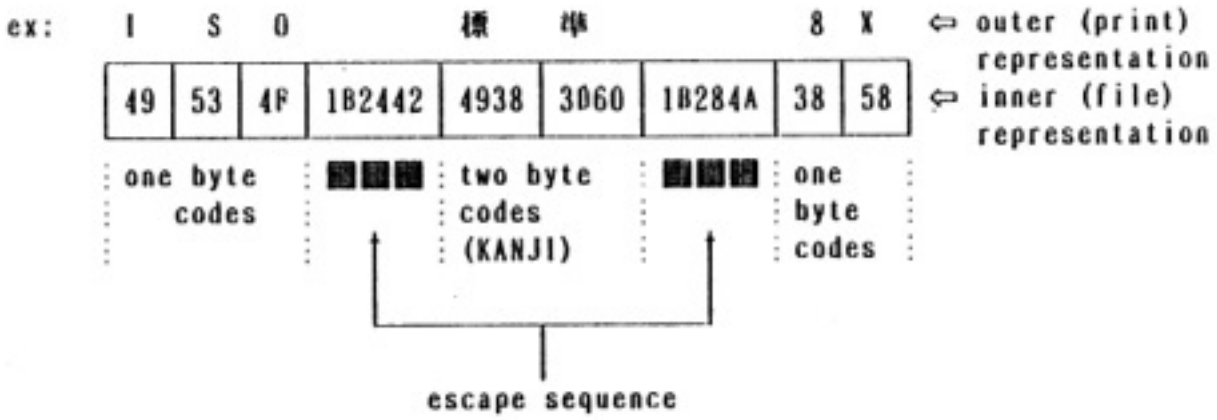
  Character constant edit descriptor is defined similarly to character constant.

- Character Editing

  A [w ] edit descriptor is used, where w is a character length.

Implementation - Japan's experience

- Escape sequences are necessary to distinguish the character kind in the same data stream.

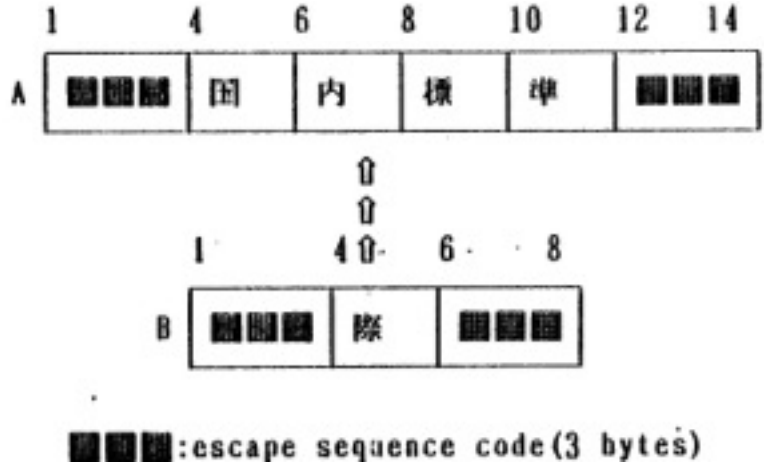



- Escape sequences are handled only by the processor,
  - The data in the program has no escape sequence,
  - Insertion and deletion of the escape sequences are done by the processor when input or output of the data is performed,

Example of KANJI handling feature

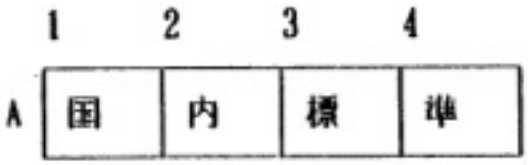
- When KANJI feature is not supported, user must take care of the escape sequences.

```
      PROGRAM   KANJI1
      CHARACTER A*14, B*8
      A= '国内標準'
      B= '際'
      WRITE(2, 100) A
      A(6:7)=B(4:5)
      WRITE(2, 100) A
100   FORMAT (1X , A14)
      STOP
      END
```




- When KANJI feature is supported, user can handle KANJI strings easily.

```
      PROGRAM   KANJI2
      CHARACTER (KIND=2) A*4
      A=2'国内標準'
      WRITE(2, 100) A
      A(2:2)=2 '際'
      WRITE(2, 100) A
100   FORMAT (1X , A4)
      STOP
      END
```



1987年　カレンダ

8月

| 日 | 月 | 火 | 水 | 木 | 金 | 土 |
|---|---|---|---|---|---|---|
| ** | ** | ** | ** | ** | ** | 1 |
| 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 9 | 10 | 11 | 12 | 13 | 14 | 15 |
| 16 | 17 | 18 | 19 | 20 | 21 | 22 |
| 23 | 24 | 25 | 26 | 27 | 28 | 29 |
| 30 | 31 | ** | ** | ** | ** | ** |

| (メモ欄) | |
|---|---|
| 1(土) | 16(日) |
| 2(日) | 17(月) |
| 3(月) ＩＳＯ会議 | 18(火) |
| 4(火) 〃 | 19(水) |
| 5(水) 〃 | 20(木) |
| 6(木) 〃 | 21(金) |
| 7(金) 〃 | 22(土) |
| 8(土) | 23(日) |
| 9(日) | 24(月) |
| 10(月) ＡＮＳＩ会議 | 25(火) |
| 11(火) 〃 | 26(水) |
| 12(水) 〃 | 27(木) |
| 13(木) 〃 | 28(金) |
| 14(金) 〃 | 29(土) |
| 15(土) | 30(日) |
| | 31(月) |

EXAMPLE OF FORTRAN OUTPUT USING KANJI FEATURE

---

Current status of Japan - Standardization

- KANJI feature special committee has made the guideline of KANJI feature on programming language, computer graphics, database, documentation etc.

- Status of standardization on KANJI feature

  FORTRAN: Japanese standard draft has been completed and proposed to 8x.
  C : studying
  LISP : Japanese standard draft has been completed and proposed to ANSI.
  COBOL : studying
  SQL : Proposal to ISO/TC97/SC21/WG3 SQL2 has been prepared.

Current status of Japan - FORTRAN

- We have already had the Japanese FORTRAN standard draft on KANJI feature.
  - The specification of the draft differs from that of currently proposed to 8x by Japan. In the draft, new character type is defined by keyword NCHARACTER, and it confirms the specification which has already been implemented on most compiler systems in Japan.

- Why we don't propose NCHARACTER ?
  - Only one kind of national character can he handled by NCHARACTER. We consider that it will be necessary to be able to handle more than two kinds of national characters in the same program in the future.
  - FORTRAN8x aims at abstracting and unifying of data type.
  ex: DOUBLE PRECISION

---

We understand that the philosophy of FORTRAN 8x is to modernize FORTRAN so that FORTRAN will he more widely used in the future, Therefore we believe that it is necessary for FORTRAN 8x to support the national character feature.