History of FORTRAN standardization

by MARTIN N. GREENFIELD

Honeywell Information Systems, Inc.
Billerica, Massachusetts

ABSTRACT

The history of FORTRAN Standardization, ranging from the original efforts in the early 60s up to the present, is presented. Some of the precedent-setting development during the initial cycle in handling problems common to all language standardization is discussed. The background in introducing some of the features in FORTRAN 77 is covered. The nature and reasoning behind the current activity are described.
There is an interesting and appropriate introduction in my daughter’s college text on FORTRAN. It reads, “After you have learned some of the language, you will show off your sophistication by knocking its lack of elegance. Everybody does. After you learn a little bit more, you will appreciate that it is the way to really get your work done.” FORTRAN has for most of life been the blue-collar worker of the programming language set. What it lacked in savoir-faire and style, it returned in cost effectiveness. Those working with FORTRAN pioneered the way for the acceptance of higher-level languages and their standardization. Those who have influenced its development were continually aware of the underlying fact that the language, first and foremost, must remain an efficient tool for producing results.

FORTRAN standardization dates back to early 1960. The language had just been selected by industry over ALGOL as the language for scientific and engineering work. The major vendors recognized the requirement to provide FORTRAN compilers in order to compete with IBM. The general strategy was to provide a compiler with the functionality of the 704/709 FORTRAN and to add features as a competitive inducement. The impact of these added features was two-edged. Although they contributed to the development of the language, they threatened to splinter it into a myriad of uncontrolled dialects. Adding to the problem, a rigorous definition of the language did not exist, even within IBM.

Fortunately, at that time ASA (subsequently to become ANSI) and BEMA (subsequently, CBEMA) undertook sponsorship of a massive standardization effort covering a broad variety of data processing areas. Someone had the brave idea of including languages. The ASA X3.4 committee conducted a survey of existing programming languages. FORTRAN, COBOL, and ALGOL were selected as the candidates for standardization. X3.4 at their May 1962 meeting established the X3.4.3 committee and directed it to standardize the FORTRAN language.

INITIAL STANDARDIZATION (1962–1966)

Bill Heising, of IBM, was appointed as the initial chairman of X3.4.3. Bill sent invitations to potentially interested groups to attend a formation meeting. Accompanying the invitations was a document written by Bill together with Dick Ridgeway that was proposed as the starting draft for the standardization effort. This Heising-Ridgeway FORTRAN was based upon the forthcoming FORTRAN IV.

The initial meeting of X3.4.3 was held at the BEMA Headquarters in New York City on August 14, 1962. This makes 1982 both the twenty-fifth anniversary of FORTRAN and the twentieth anniversary of the start of its standardization. At this August 1962 meeting, there was a consensus to undertake the standardization work. The scope and criteria of the effort were established.

X3.4.3 assumed the role of parent and policy maker and delegated all the chores below that to two working subcommittees. As such, X3.4.3 met only about twice a year. X3.4.3 originally had about two dozen regular members. All the major hardware vendors were represented. A number of user groups (SHARE, Honeywell Users Association, USE, VIM, IBM 1620 Users, CO-OP) participated. Some software houses (CSC, CUC) and universities (Wisconsin, Penn State, UCSD) had members.

The parent X3.4.3 did thrash out some very controversial issues. One of recall concerned a proposal from those working with the then new character-addressable hardware. They could save much space by not allocating the same space to integer and logical data as was allocated to reals. In fact, they preferred not to have any fixed storage relationship between the data types. Logicals could be packed into one byte or less. Double precisions could occupy just two or three more bytes than reals. Their arguments centered about the concept that a language standard should not be as hardware biased as the word-storage-unit relationship implies. After some impassioned discussions the heavy dependence of FORTRAN on storage association for efficiency and the dominance of word-addressable processors won.

Most of the actual standardization work was handled by the two subcommittees. X3.4.3-IV was responsible for the standardization of the language based on FORTRAN IV, while X3.4.3-II was to do the same for FORTRAN II.

The subcommittees were small compared to the size of groups currently developing draft standards. It was fortunate, because it provided an efficient working arrangement and uninterrupted participation. Little time was lost in having to bring new members up to date. The regular members of X3.4.3-IV were

Martin N. Greenfield, Honeywell, chairman
Richard K. Ridgeway, IBM, editor
Caral Sampson (Giammo), Philco, secretary
Tom Martin, SHARE and Westinghouse
Geraldine Zimmerman (Bowen), UNIVAC
Lou Gatt, CSC
Ken Tiede, CDC
Carl Bailey, CO-OP and Sandia
Bob Mitchell, CO-OP and UCSD

Along with the X3.4.3 chairmanship responsibilities, Bill Heising was a very active participant in the effort of the
X3.4.3-IV subcommittee. Others from X3.4.3 participated from time to time, but the bulk of the effort was done by the group above.

The work proceeded during the following two years. Although some meetings were hosted at the sites of the different members throughout the country, the bulk of the sessions were either at BEMA headquarters or at the IBM program development center in the Time-Life building, both in New York City.

The initial FORTRAN IV compilers were all under development while the work of X3.4.3-IV was in progress. The members of X3.4.3-IV were all either responsible or could direct changes in their language specifications for these implementations. It was a unique situation, where language changes adopted by the subcommittee were incorporated into the compilers almost immediately. I have always felt that the actual standardization of FORTRAN stemmed from the discussions, understandings, and agreements of X3.4.3-IV rather than from formal text that followed some years later.

The undocumented agreement within X3.4.3-IV was that the standard would not incorporate any feature that was not planned for all the implementations. Since the starting point for all of our language specs was the IBM-proposed language, it followed that the draft most closely resembled the IBM implementation. It was by no means a slavish copy. For one thing, there were no rigorous specifications within IBM of much of FORTRAN IV that could have been copied. This was particularly true in the input-output area. There were some features that IBM meant to carry into FORTRAN IV from their FORTRAN II implementations in order to protect their user’s investment. Unfortunately, some of their FORTRAN II implementations contained some objectionable shortcuts. For example, a constant could precede a variable and imply a multiplication operator (SL meant 5 * L). To their credit, there was never much of a hassle with those from IBM in deleting features that were objectionable carryovers from existing implementations of FORTRAN II. I believe they were sincerely motivated in working toward the best long-term interests of the language. Another change of note was that the DATA statement syntax was altered from the way IBM was implementing it. It was originally specified with parentheses rather than slashes as the delimiter for the list of constants.

Having no precedents, X3.4.3-IV had to address numerous problems common to all language standardization. Much of this we take for granted now, but there was nothing to turn to at the time. There were discussions as to whether there should be a standard. There is a penalty. The presence of a standard implies the pressure of conformance over a long period to a static document. This could certainly serve to limit the growth and development of the language. Even if motivated, the implementor, constrained to conform, would be prohibited from adding extensions. Programs requiring nonstandard functionality could not be developed. Unanticipated requirements could not be satisfied until after the many years needed for a new revision had elapsed. The difficulties of specification of a standard could artificially limit the functionality because it might be too difficult or unwieldy to word the true restrictions. Once a feature was standardized, its life would be semi-eternal even if the feature were a mistake. The result is that generally a very conservative posture is assumed in deciding what is to be included. The potentially useful but untested functionality usually doesn’t make it. These are all penalties to be weighed against the advantages of portability and communication that standardization could provide.

A partial answer to these objections to having a standard was worked into the interpretation section of the standard and has been carried into all the subsequent revisions of FORTRAN standards. The standard is to be interpreted as permissive. That is, that the standard serves only to specify a part, not all, of the language. Anything not specified isn’t unclean, bad, immoral, or even not kosher. It is simply not specified. Similarly, things that are prohibited are things that are simply uninterpreted when violated. A standard program must be limited to what is specified in order to conform, but the same is not true for a processor. A processor may provide array processing, but it must handle standard subscripting in the conforming manner. Thus, an experimental extension can be available in a standard processor. The processor must be able to properly interpret standard programs, but may also provide interpretation to a nonconforming program. The choice is then available to conform or not as the economics dictate. Some nonconformance is encouraged.

The subcommittee decided that the target audience for the standard would be compiler implementors or those on users’ staffs who were the FORTRAN support experts. It was felt that this latter group were competent in being able to implement a compiler; so, in effect, there was just the implementor that characterized the audience. It was felt that the standard should specify the requirements for a standard conforming program rather than a compiler, but I don’t believe this was apparent in the document.

The decision was made to use English rather than some metalanguage. This was in the belief that the description of the semantics was the difficult problem. Use of a metalanguage would not help there. A metalanguage was at best only assisting in tackling the easiest part of the description. It was felt that its precision did not compensate for the need to become familiar with the added formality. Interestingly, the one most useful area that could have been served by a precise description using a metalanguage is the FORMAT statement. There was actually an error in the way it was specified in the standard. I am still unaware of a complete and precise description of that statement using some metalanguage.

There were many challenges to our ability to describe. CDC had proposed that the new logical IF be a two-way branch analogous to the arithmetic IF. This would have saved us much descriptive grief in handling the concept of a compound statement that had in this one place crept into the language. For example, we could no longer accurately state that every statement could have a label. It also led to an unduly harsh restriction prohibiting some forms of the logical IF from being the terminal statement of a DO loop.

The greatest challenge to our descriptive capabilities was presented by the extended range of a DO loop. (There are some who would claim that this honor should go to the concept of second-level definition). All the implementations of FORTRAN IV being developed allowed a more liberal extended range than the one appearing in the standard. The committee would have been amenable to a less restrictive extended range if it could only have been appropriately de-
scribed. Everyone tried at least twice. Any definition that included statements about the sanctity of the contents of index registers, although reflecting the real concern, was inappropriate. The definition finally adopted was an accurate subset of what everyone was providing. The definition was felt to be reasonably understandable. Those of you who have struggled with that definition and its prerequisite concept of completely nested nest might quibble about the description being reasonably understandable. This is only because you did not struggle with some of the descriptions that were rejected. This was certainly an instance where the ability to describe limited the technical content. I believe that there is some of this effect in most standards. It is deluding not to admit it.

There were a surprisingly small number of new terms that had to be coined. Terminology common to several manuals was preferred, since it would already be familiar. The, usually missing, rigorous definitions of these terms had to be developed. Among the newly coined terms were definition and undefinition and their related states of being defined or undefined, reference as applied to data and to procedures, and intrinsic function. The term intrinsic function had its birth at a ball session during one of our meetings. We had been discussing the classification of functions, using the then customary terms open and closed functions. Open functions meant in-line code; closed meant some internal procedure. There was the concern that the absolute function (ABS), generally thought of as the obvious prototype of in-line code, was no longer such when the argument was of complex data type. Further, the tightening techniques being developed for some codes might make it attractive to put more formerly closed functions in line for greater speed. Besides, the terminology smacked of a particular implementation consideration. Lou Gatt piped up with the idea that the basic characteristic of these functions was that they were cast into or intrinsic to the processor, and that therefore we should call them intrinsic functions. So credit for this term belongs to Lou.

We were later to find that a subtle side benefit of our standards work was the widespread use of the terminology used in the standard. Our terminology was generally accepted and replaced the proliferation of some terms for certain actions and objects that were in use before without any rigorous and agreed-upon definitions.

The subcommittee gave some consideration to how to enforce the standard through use of acceptance procedures. Two hurdles caused us to turn away from further work in this area. We realized that an exhaustive verification was not possible. It might be misleading to develop some partial verification package that might be construed as being total. Any such official package might be misused as a standard performance benchmark. The second hurdle was simply not having the manpower to do the work. It was hoped that market pressures would lead to some accepted verification means, but we didn’t have the resources.

The subcommittee X3.4.3-II drafting the specification based on FORTRAN II was even smaller than that of X3.4.3-IV. Their membership, as I recall, was

Jack Palmer, IBM, chairman
Irwin Boris, Honeywell

Charles Davidson, University of Wisconsin, 1620 Users Group
Don Laird, Penn State University
Bob Brunelle, Honeywell Users and NIH
Bernice Weizenhoffer, IBM
Robert Hux, RCA

Partly because their target was better defined, X3.4.3-II completed their work and the first draft FORTRAN standard almost a year before X3.4.3-IV finished. They were directed by X3.4.3 to keep the draft on hold until X3.4.3-IV had its draft ready. There was still the hope at that time that a compatible standard representing FORTRAN II and FORTRAN IV could be produced.

Subsequently, X3.4.3 decided that there should be a standard for the full language and a standard that was a proper subset of the full language. It was not possible to use the X3.4.3-III draft as the subset because of the number of totally incompatible differences between FORTRAN II and FORTRAN IV. The result was that the work of the X3.4.3-II was discarded. The subset was created by deleting text from the X3.4.3-IV draft. I hope that the draft produced by X3.4.3-II finds its way into the archives of FORTRAN history. Through no fault of its own, the effort of X3.4.3-II was not incorporated. Their work is historically significant in that it was the first completed draft of any language standard.

In October 1964, the two proposed draft standards were published in the Communications of the ACM. These were the first standards ever proposed for a programming language. They severely taxed the editing and approval mechanisms of ASA and BEMA. Draft standards up to then rarely needed more than a page of text and that page usually had room for the diagrams of the screw thread. The inability to rigorously check for conformance was shattering. It is little wonder that it took almost a year and a half before final approval was obtained in April 1966. The full language standard was designated ASA X3.9-1966 FORTRAN and the subset, ASA X3.10-1966 Basic FORTRAN.

Early in the standardization effort, the European Computer Manufacturers Association (ECMA) submitted a proposed draft of what they felt the full language should contain. Since they were separated from the developments in this country, their proposal fell between the Basic FORTRAN and the full FORTRAN. X3.4.3 voted to standardize on only two levels. When FORTRAN standardization was considered by the International Standards Organization, the ANSI form and content was chosen as the basis. The ECMA subset in ANSI form was added as the intermediate of three levels.

INTERPRETATIONS PERIOD (1967–1970)

Late in 1967, the then disbanded X3.4.3 was recalled primarily through the urging of the National Bureau of Standards. NBS, and in particular, Betty Holberton, was attempting to produce a Federal standard for FORTRAN. Betty’s examination of the X3.9–1966 FORTRAN standard led her to submit a few dozen questions on interpretation. Other clarification inquiries were received from other sources. The FORTRAN
group was revived as the only body that could authoritatively provide the clarifications. This process turned out to be more tedious and demanding than the standardization effort itself. Because we were dealing with an approved standard, not a single comma could be altered without going through the same long approval cycle. Interpretations had to be based on a rationale developed from the standard's actual wording and not from what even the original authors felt it should have been. Two interpretation reports were published, but they took over three years of meetings to produce. The difficulty of that interpretation effort has had its impact on the form of the standard for FORTRAN 77. Those who participated in both efforts took pains to carefully examine every phrase to reduce to a minimum the chance of misinterpretation.

By 1968, enough extensions had appeared in the more current implementations to have the FORTRAN group appoint someone to study whether these extensions should be standardized. Frank Engel was selected as the one to conduct this study. Following Frank's report, in January 1969, the committee voted not to reaffirm X3.9–1966 when its review period came up, but to provide a new draft standard.

The committee had a succession of chairmen during this period. Bill Heising was replaced by Dick Ridgeway. Heising later returned as chairman prior to having Dennis Hamilton assume the position. In September 1970 Frank Engle assumed the chair and was to last throughout the development of FORTRAN 77. Frank's tenure, the longest of any chairman, ended in October 1977 when Jeanne Adams, the current holder was appointed.


By early 1970 the interpretation activity had had it. There were unresolved issues that could not be handled within the wording of X3.9–1966. They decided that since the standard had to be reviewed and replaced or reaffirmed by 1971, it would be more productive to abandon the clarification work and devote their energy to a replacement. It is interesting that the most pessimistic schedule proposed at that time had the draft available by the end of 1971. The initial effort did not sharpen the ability to predict the time required to develop a standard.

Criteria and goals were drawn up for what would become FORTRAN 77. Their jist was to evolve the language, keep it approximately the same “size,” and be sure that its efficiency features would not be impaired. It was important that the standard should be in a much more expository form and be meaningful to a larger and less knowledgeable audience. The form of the revision was chosen to be a single standard containing two subset levels. A later decision removed the intermediate subset. Because of the single standard approach, ASA X3.10–1966 Basic FORTRAN would be discarded (i.e., not reaffirmed).

They further voted that the new draft standard would be an evolutionary development that would not invalidate programs written in the language of the 1966 standards. This position was subsequently modified in two significant areas. The Hollerith data type was deleted because it was replaced by the more functional and machine independent character data type. The zero trip DO loop was specified. Actually, the control conditions for a zero trip DO were conditions that were nonconforming to the 1966 standard. However, since many implementations interpreted these conditions by executing the statements in the range once, many programs would have to be adjusted. There were objections even though the issue related to programs that were technically not standard conforming.

Six years of effort went into FORTRAN 77. That standard represented work on over two hundred technical proposals from all over the world. The cost of the effort was in excess of two million dollars. The text was almost six times the size of X3.9–1966. While some very significant language additions are present, the expansion was largely attributable to the effort to make the document more understandable. The draft had a completely different organization than the 1966 standard. Emphasis was on clarity rather than compactness and nonredundancy. Extensive use was made of word processing, a concordance tool (KWIC), computer graphics, and direct transcription to hard copy and fiche facilities. The very extensive editing, consistency checking, rewriting, and the distribution of the numerous interim drafts were made possible only by some herculean efforts of the two editors, Lloyd Campbell and J. C. Noll. The editorial staff of ANSI was presented with a camera-ready copy of the draft for publication, thus avoiding the errors that might have been introduced by an ANSI stage of processing.

The features of the draft standard were publicly presented by X3J3 members at the West Coast FORTRAN Forum held in Anaheim, California, in February 1976. The following month, the draft standard appeared in a special edition of SIGPLAN Notices. An East Coast FORTRAN Forum was later held at the National Bureau of Standards in Gaithersburg, Maryland. Smaller groups of X3J3 members presented sessions on the new language standard at meetings of professional societies, user groups, and at conferences. The public review was initiated and comments were solicited.

During the period of public comment and review 289 responses consisting of 1225 pages were received. This was probably the largest outpouring to any proposed standard as of that time. It took almost a year for the committee to complete the responses. The number of public comments was evidence of the large, present, and continuing interest in the language and the understandability of the document. Despite the earlier extensive checking by the committee, there were a number of changes and corrections incorporated because of the comments.

The major issue, as measured by the volume of comments received, was to add some facility in support of structured programming. There were a significant number of preprocessors available that enabled FORTRAN programmers to develop programs using statements such as IF ... THEN ... ELSE, DO WHILE, DO UNTIL, CASE statements and the like. These preprocessors would convert the source into valid FORTRAN statements. There was a clear requirement to place some of the facility directly into the language. In responding, the committee felt that although some facility should be added, there were many syntactic variations and an insufficient experience basis to select and standardize many of the constructs. They took an appropriately conservative ac-
tion of adding only the BLOCK IF constructs. This addition, as specified by Walt Brainard, provided most of the important capability requested. It avoided adding and being stuck with some of the other constructs such as DO UNTIL that are already falling into disuse because of superior forms.

The reaction of X313 to the structured programming requests is a good example of how a responsible committee should avoid an over reaction that would prematurely add features that it would shortly regret. Unfortunately, there are counter examples in FORTRAN 77 such as the ENTRY statement and the alternate RETURN that should not have been included.


Next Revision (1978–Present)

Following the approvals of the FORTRAN 77 standards, the expected lull in the standardization activity did not materialize. There was pressure to consider the additions received during the public response to FORTRAN 77 that were rejected as premature. New FORTRAN implementations were incorporating additions such as a free form for statements. CODASYL had established a group (FORTRAN Data Base Language Committee, FDBLC) to provide a foundation for the addition of a major database augmentation to the language. ISA and the Purdue Workshop had developed standards addressing issues of tasking, file synchronization, and event management. An interest in a graphic addition was looming.

The committee devoted its time during 1978 to the planning for the future direction of the language. They solicited the thoughts of many other interested groups such as ISA, CODASYL, IEEE, and SIGNUM who were known to be interested in FORTRAN extensions. The level of interaction with international bodies was dramatically increased. International meetings under the informal structure of ISO FORTRAN Experts Group were convened in Europe during 1977, 1978, 1979, and 1980. All of this activity was in the attempt to obtain a broad basis of experience upon which to develop the successor standard.

X313 felt confident it could manage desired additional language features such as free form for statements, new control and data structures, and even most of the array handling. They even felt comfortable in handling the removal of some of the basic restrictions such as dynamic storage allocation, recursion, identification via storage association, and storage related precision. However, they were unsure of how to cope with major augmentations such as the database and graphics handling. The additions would be expensive, not only in the cost of the processors, but in the breadth of the language that would be impacted. Even those not interested in these features would be paying a price in terms of what they would have to know to work with the language. The committee knew it did not have the expertise to select among the competing forms of database and graphics facilities. It wanted to be able to responsibly control these augmentations and yet didn’t see how a single committee could commandeer all of the expertise needed for this development and management.

The answer is one that is still evolving and is a change in the architecture of the language. It is called the core plus modules approach. The plan for the language revision, called FORTRAN 8X, is to specify a relatively small, general purpose, self sustaining core language. There would be added features that would modernize and streamline the language. The size of this core language would not exceed that of FORTRAN 77 because there would be compensating deletions. The core would be provided with very strong facilities to be able to interface with modules whose use could be selectively chosen. These modules would have to follow some broad conventions established by the committee to qualify as part of the FORTRAN family.

There would be two classes of modules, language extension modules and application modules. A language extension module would be developed by X313 and would represent features that exceed the general purpose scope of the core. It might also consist of features that were desirable for addition, but that had not been subject to sufficient implementation or usage experience. An extension module could not be modified and approved for standardization without reconsideration of the core and all of the other language extension modules.

One special language extension module would be called the Obsolete (Transition) Features Module. This module would contain all of the features needed for compatibility with the previous revision (FORTRAN 77). Features being dropped in a revision would survive for one cycle in this module. When this module was employed, it would override any incompatible features of the current language.

An applications module would probably be specified by some group external to X313 and would address features of some special domain. Examples might be one (or more) of the database facilities, a query capability, or a graphics addition. These would probably take the form of a collateral standard so its maintenance could be managed independently. The hope is that through use of modularity, the heart of what is identified as FORTRAN might remain small.

FUTURES

Over this period of twenty years of standardization we have been through two complete cycles and are in the midst of a third. How long does this go on and when does it end? Jean Sammet once asked me if it weren’t time for the FORTRAN gurus to get together and call an end to the effort so people can get on with the using of the good languages. I have reservations over which of the current choices should be crowned the good languages. There should be something fundamentally different and better to justify dropping the huge
investment in the current languages. The replacement should have features that defy compatible inclusion in what we have.

Until this revolutionary development makes its appearance, interest in FORTRAN will remain. There is the story of the farmer who was asked by one of his eager turks why he didn’t replace his old burro with one of the younger, sleeker, more highly tuned and spirited steeds. He looked at the young hand with wrinkled and wizened eyes and said, “When you have something yeh gotta be sure gets done, yeh goes with what you knows.” So be it with FORTRAN.

REFERENCES