C.S.I.R.O.

DIVISION OF COMPUTING RESEARCH

MANUAL SUPPLEMENT 33 (EDITION 3)

Supplements 3600 Fortran Reference Manual (Pub. No. 60132900)

KWIKTRAN

A Version of the Control Data 3600 Fortran System
Modified for Fast Translation and Rapid Loading
under the TWO-bank DAD Monitor

by

R.H. Hudson
Division of Computing Research

December 1971

CONTENTS

1 INTRODUCTION

With the Control Data 3600 a normal Fortran program goes through three major phases before entering the execution phase; compilation (FORTRAN), assembly (COMPASSX), and loading (DAD LOADER). The last two phases, because of their extreme - but seldom used - generality, are inefficient for most purposes. The KWIKTRAN system replaces these phases by using a fast one-pass assembler that generates absolute code. The name Kwiktran does not imply that a language with a different syntax is involved. On the contrary, for ease of system maintenance, the identical Fortran source language compiler is used in both KWIKTRAN and FORTRAN-COMPASSX-LOADER. The user is therefore referred to the Control Data Fortran Reference Manual for a description of the language.

There are, of necessity, differences between the two systems: The different interpretations of some of the control statement options are described in section 2; the minor differences concerned with storage allocation and table limits are described in section 3; the difference in speed is described in section 4.

The major difference between the two systems is that with Kwiktran it is not possible to use overlays or snap dumps.

2 KWIKTRAN OPTIONS

The Kwiktran system is loaded when a *KWIKTRAN (or *KTN) control statement is encountered. All fields are free-field, that is, blanks are ignored. The options defined below may appear in any order but must be separated by commas. A terminating period is optional as a field may also be terminated by the end of the record (e.g. a card). Unrecognized options and extraneous characters are ignored. An option may be followed by =n where n represents the logical unit number. If n is 0 or not numerically defined, the option is ignored.

The table below lists the options and also indicates whether the Kwiktran meaning of the option is the same as (s) or different from (d) the Fortran meaning. The N and T options are not available using Fortran.

OPTIONS	OPTION ALONE	n IS NUMERIC
L (s)	List source language program on unit 61.	List source language program on unit n, 1-59, 61.
P (d)	Ignored	Ignored
X (d)	No load-and-go is generated but the option must be present for execution.	No load-and-go is generated but the option must be present for execution
A (d)	Ignored	Ignored
I (s)	Input source from unit 60; same if option is not present.	Input source from n, 1-59, 60.
C (d)	Ignored	Ignored
N	Suppress check references to subscripted variables.	Suppress check references to subscripted variables.
T	Trace program execution on unit 61 (operates only if N absent).	Trace program execution on unit 1-59,61.
B (d)	Unit number, n, must be designated.	Generate KWIK cards on n, 1-59, 62. See explanation below.
* (s)	Compile code for one bank. Same if option is not present.	Compile code for one bank.

R (d)	List all generated symbols and their absolute locations on the unit specified by the L option.	List all generated symbols and their absolute locations on the unit specified by the L option.
D (s)	List compilation diagnostics on unit 61; same if option is not present.	List compilation diagnostics on n, 1-59, 61.
F (s)	Crack format statements at execution time. If option not present formats are cracked at compile time.	Crack format statements at execution time.
Q (s)	Plant actual parameter addresses with Q8QRESID. If option not present in-line code is generated to plant addresses.	Plant actual parameter addresses with Q8QRESID.

If the N option is not present Kwiktran will generate code to check that references to subscripted real or integer variables are within the declared array bounds. Constant subscripts (e.g. A(10)) are checked at compilation time. Subroutine references to arrays given asformal parameters are not checked. If the N option is present the in-line coding is not generated. If not present the checking costs three words and 6.8us per test plus six words per subroutine and one word per statement label to keep track of the current position in the program. The N option should be used for production programs.

If the T option is used (N must be absent), a flow trace of the executing program or subroutine will be printed. Printed output consists of

statement labels as they are encountered

routine names as they are called

the symbol > to indicate return from a routine to where it was called from

Care should be exercised when using this facility as printer output may be excessive particularly in connection with high-order DO-loops. If output is on 61, the trace output and program output will be interleaved.

The KWIK cards produced by the B option are logically equivalent to source decks and any source subroutine may be replaced by its corresponding KWIK deck.

KWIK decks are generally intermediate in size between relocatable binary and source decks. The compilation speed can be up to twice that source statements. Details are given in Appendix 3.1.

As with Fortran, Kwiktran options may be changed from subroutine to subroutine. INTERJOB control cards (e.g. *FTN, *COMPASS, *EQUIP, *LOAD) other than *KTN or *KWIKTRAN should not generally be present between subroutines.

2.1 Examples

(1) *KTN,L,X,N,R

is interpreted as

I: Source program is on logical unit 60, i.e. immediately after the *KTN statement.

L: Listingof source program and diagnostics is to be on logical unit 61.

X: Execution is to be attempted.

N: References to subscripted variables are not to be checked.

R: A symbol table is to be listed on logical unit 61.

(2) *KTN,L=10,B=20,D=11. ANY COMMENT MAY APPEAR HERE

is interpreted as

I: Source program is on logical unit 60.

L: Listing of source program is to be on logical unit 10.

B: KWIK card images are to be written on logical unit 20.

D: Compilation diagnostics are to be written on logical unit 11.

.: Period is an optional record terminator.

(3) *KTN,I=10,N,X

is interpreted as

I: Source program, which may be in either Fortran, KWIK or relocatable binary, is on logical unit 10.

N: References to subscripted variables in either Fortran or KWIK routines are not to be checked.

X: Execution is to be attempted.

3 FORTRAN COMPATIBILITY

A correctly written source program will give identical answers whether run with the FORTRAN-LOADER system or with Kwiktran. However, the program elements will be distributed differently in the store so that malfunctioning programs may produce different results. In particular, Kwiktran divorces all data arrays from actual machine instructions. The library routines are loaded as one absolute block in the high end of store. Data arrays are assigned addresses below the library as they are encountered. Subprogram instructions as they are encountered are assigned ascending locations beginning at location 768. The storage layouts in the two systems are shown in Figure 1. Di, Ci and I_i are respectively the arrays defined by DIMENSION statements, the arrays defined by COMMON statements, and the executable machine instructions in subroutine Pi. The Li are the library subroutines required by the user routines.

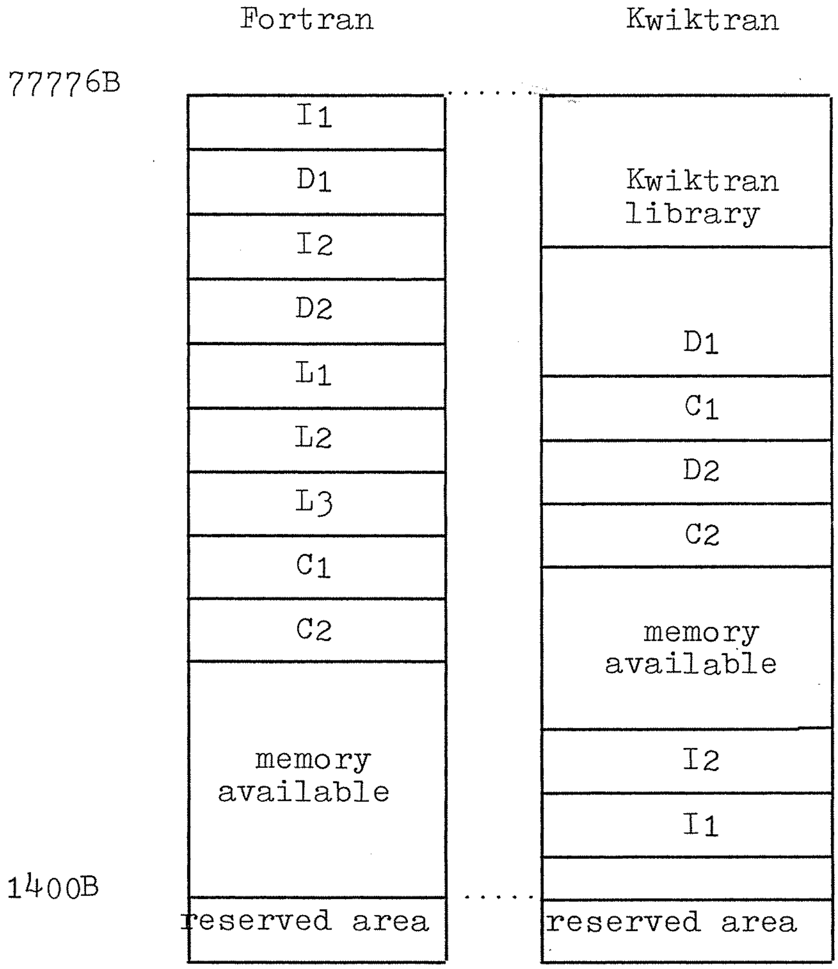

Figure 1. Storage Allocation (Bank 1)

The major consequences of this storage assignment scheme are the following:

(a) The total storage available is less with Kwiktran since all its library subroutines are always loaded (see Appendix 4). The length of the Kwiktran library subroutine block is approximately 9000 words. This penalty is moderated by the fact that a short Fortran program with BCD output requires about 2700 words of library subroutines. The more library subroutines used the smaller the penalty.

(b) If a common block is defined with different lengths in two or more subroutines, the subroutine with the largest block length must appear first in the source deck; otherwise a fatal diagnostic will be printed.

(c) With Fortran the user can assume, although this is bad practice, that contiguous arrays in DIMENSION statements will be contiguous in the store. This is not true in Kwiktran. The correct procedure is, of course, to enforce the assumption with an EQUIVALENCE statement.

(d) All arrays both local and common are set to zero at the start of execution.

(e) Numbered common may be preset. (Note: this is not allowed in Fortran and is therefore bad practice.)

The remaining Kwiktran differences derive from the method of assembly and the method of linking user-defined subroutines. In the assembler the simple method of fixed-length tables has been used since this type of table can be treated much faster than the more general linked table. The sizes of the relevant tables are given below; to date, no user-program is known to have exceeded the limits.

(a) Total number of compiler generated symbols, statement labels and variable names per subroutine: 1536.

(b) Literals (constants not appearing in DATA statements) per subroutine: 512.

(c) Total number of user subroutine entry points: 150.

(d) Maximum number of 'undefined' external symbols during compilation: 110.

(If subroutine ALPHA calls subroutine BETA and ALPHA precedes BETA in the source deck there is one undefined external; if BETA precedes ALPHA there are no undefined externals). All references to library subroutines are defined.

Item (d) above implies two further subtle restrictions. If a user-subroutine has an entry point with the same name as an entry point in the system library routines, it must appear in the source deck before the calling subroutine. Otherwise the address of the library routine entry point will be substituted. In conjunction with this warning the next paragraph is taken verbatim from the 3600 Fortran manual.

The following 3600 functions will be coded in-line rather than called as closed routines. The closed function may be obtained by the appearance of the name in an EXTERNAL statement. If any of these function names appear as actual parameters, they must also appear in an EXTERNAL statement.

ABS or ABSF
SIGN or SIGNF
DIM or DIMF
FLOAT or FLOATF

IABS or XABSF
ISIGN or XSIGNF
IDIM or XDIMF
INT or INTF

DBLE
AIMAG
CONJG
CMPLX

REAL
DABS

The moral to be drawn from the above two paragraphs is obvious: avoid library subroutine names for user-subroutines.

A full list of the system library subroutines and their entry points is given in Appendix 4.

There is one final size restriction. The executable code per subroutine should not exceed approximately 12000 words. Since this will generally allow of the order of 2000 executable source statements per subroutine it is unlikely that the restriction will ever be noticed.

4 TIMING CONSIDERATIONS

Source routines can be compiled and loaded at rates of 2000 to 4000 statements per minute. In favourable conditions the equivalent KWIK decks can be assembled and loaded in half the time. The time required by K iktran to reach the execution phase is generally of the order of 1/5 that required by FORTRAN-COMPASSX-LOADER (1/8 was observed in a job loading a 10,000-word array by DATA statements).

Kwiktran can generally compile and load a Fortran program about twice as fast as the DAD LOADER can link and load the relocatable binary program produced from the same Fortran program by the Fortran compiler. For very large programs the speed advantage is reduced so that the DAD LOADER is about as fast as the Kwiktran compiler and loader.

KWIK decks can be assembled and loaded even faster. In one comparison with a Fortran source deck, the Fortran job terminated in the middle of the loading phase with "time limit exceeded", the job time limit being two minutes. The equivalent KWIK decks began execution after 13 seconds.

For maximum speed, jobs should consist of short subroutines of up to approximately 200 source statements each. For jobs of this type, Kwiktran requires no intermediate drum units (as opposed to the two required by FORTRAN and COMPASSX). For longer subroutines, Kwiktran will use one intermediate drum unit.

As an illustration, a Fortran program was loaded by each of the available methods. The table below compares the number of cards in the deck, the number of drum sectors used and the time from the start of compilation to the end of loading. The program used had a main program and thirteen subroutines and contained 771 statements and 625 comment cards. The total program length, excluding common arrays, was 3824 words.

METHOD OF COMPILATION AND LOADING	TIME (s)	DRUM (sectors)	CARDS
N rmal Fortran	141	61	1396
Kwiktran	22	62	1396
KWIK	14	61	752
Relocatable binary loaded by Kwiktran	8	31	385
Relocatable binary loaded by LOADER	22	31	371
Absolute binary loaded by *LOADMAIN	4	32	-

5 EXAMPLES OF DECK STRUCTURE

```
(1)    *JOB,charge code,ident,time limit
       *KTN,L,X
            PROGRAM
            ...
            ...       Fortran source code
            ...
            END
                SCOPE
       *LOAD
       *RUN,time limit, print limit
        ...
        ..          data
        ..
       *EOD
```

In the above example all the routines to be compiled and run accompany the job. Array bounds checking is to be incorporated.

```
(2)    *JOB,charge code, ident, time limit
       *KTN,L,X
            PROGRAM
            ...
            ...       Fortran source code
            ...
            END
                IDENT
       ...
       ...  KWIK and/or relocatable binary routines
       ...
                SCOPE
       *LOAD
       *RUN,time limit, print limit
       ...
       ...      Data
       ...
       *EOD
```

This example shows how Fortran, KWIK and relocatable binary routines may all be included for Kwiktran loading (and compilation if necessary). The KWIK and relocatable binary decks must be preceded by an IDENT statement punched starting in column 10. If several KWIK or binary decks are entered as a block only one IDENT statement is required.

(3)

```
*JOB,charge code,ident, time limit
*DFCOPDR,10,,FTNSRC
*KTN,I=10,X,B=62,L
*KTN,L,X,B=62
     ...
     ...   Fortran source routines
     ..   and KWIK routines
     ...
         SCOPE
*LOAD
*RUN, time limit, print limit
...
...    Data
...
*EOD
```

In the above example some of the routines to be run are on the disc as a document called FTNSRC and some accompany the job. Both lots are to be compiled and loaded and are to have KWIK decks produced on logical unit 62.

APPENDIX 1

A1 ARRAY BOUNDS CHECKING

For each source language reference to a subscripted variable the Fortran compiler generates an assembly instruction of the form

```
op        address+ca,ir
```

where op is the operation code, address is the location of the array, ca is a constant addend and ir is an index register 1-6. If the subscript in the source statement is a constant, ir is 0.

The Kwiktran assembler utilizes the fact that for a legal reference, i.e. a location inside the array in question, the sum of the constant addend and the currentcontents of the index register must be in the range 0 to n-1 (inclusive) where n is the dimension of the array. If the subscript is a constant the constantaddend must be in the same range.

The above assembly instruction is matched against the following tests.

(a) op is one of the following; STA, ADD, SUB, MUI, DVI, LDA, LAC, FAD, FSB, FMU, FDV, ENA, INA.

(b) The previous instruction was not an AUGMENT (this eliminates double precision and certain subroutine calling sequences).

(c) address appeared in a DIMENSION or COMMON statement and is of dimension n where n is greater than 0 (arrays in formal parameter lists are given dimension 0 in the relevant subroutine).

The action next taken depends on the value of ir. If ir = 0 the subscript is a constant and ca is checked to see that it lies between 0 and n-1. If not, a fatal compilation diagnostic is given. If ir is not equal to 0 the subscript is a variable and must be checked at execution time.

In subroutines a small percentage of instruction sequences which refer to formal parameters cannot be altered by the insertion of instructions to check array bounds. In these situations therefore no array bounds checking is incorporated.

The following five instructions are inserted ahead of the instruction in question when bounds checking is incorporated i.e. when the N option is not specified.

```
RXT       P,D
INI       ca,ir
RGJP,GE   ir,n,AD.CHECK
NOP       xx
INI       -ca,ir
```

P is the location counter, D is the D register, AD.CHECK is the location of a library diagnostic routine and xx is the octal equivalent of the first two characters of the array name. If the register jump is taken the program is terminated with the diagnostic

```
>>>        ppppp   - LOCATION OF ARRAY BOUNDS ERROR
AFTER      lllll   - LAST STATEMENT LABEL
IN      abcdefgh   - (SUB)PROGRAM NAME
        nmopqrst   - CALLING (SUB)PROGRAM
              xx   - ARRAY NAME (1ST 2 CHARACTERS)
         nnnnn10   - DIMENSION
         iiiii10   - SUBSCRIPT VALUE
```

Line 4 of the diagnostic is not printed if the error occurred in the main program. Statement labels greater than 32767 will be ignored.

The information contained in the diagnostic will generally be adequate for locating the point in the program where the error is occurring. If more precise information is required a memory map can be obtained with the R option and the location found from ppppp.

APPENDIX 2

A2 DIAGNOSTICS

Two forms of assembly/loading diagnostics are given. The first is concerned with table limits and hopefully will not be seen by the user. The first form of diagnostic is

name TABLE FULL

The limits of the tables which may produce this diagnostic are

name	Limit	
LITERAL	512	
COMMON	126	(DAD LOADER Block Common Limit)
EXT.	110	
ENTRY	150	
SYMBOL	1536	

The second form which is concerned with source program errors is

PROGRAM ident SYMBOL name message

where ident is the name of the program, subroutine or function, and name and message point to the error.

name	message
name of entry point	MULTIPLE ENTRY POINT
name of undefined variable	IS UNDEFINED
name of common block	COMMON LENGTH ERROR
name of external symbol	IS UNDEFINED EXTERNAL
name of array	ARRAY REFERENCE ERROR
name of variable	IS USED INCORRECTLY
blank	TOO MANY BRT CARDS
octal length of subroutine	SUBROUTINE TOO LONG
first word of last binary card in BCD	BAD BINARY DECK
first word of binary card in BCD	BINARY CHECKSUM ERROR
first word of KWIK card in BCD	KWIK CHECKSUM ERROR
first word of KWIK card in BCD	KWIK SEQUENCE ERROR

In the last three diagnostics ident may be printed as 0 if the error occurs in the first few cards of the subroutine. KWIK card numbers can be obtained by converting the rightmost two characters of the BCD word to their numeric value.

The MEMORY AVAILABLE is always given and is a fatal diagnostic if negative. The execution time diagnostics associated with bounds checking is described in Appendix 1.

APPENDIX 3

A3 KWIK AND RELOCATABLE BINARY DECKS

A3.1 KWIK Decks

For lack of a better name the cards generated by the Kwiktran B option are called KWIK cards. KWIK cards contain the basic fixed format assembly language generated by the Fortran compiler. The primary advantage of KWIK cards over Fortran source cards is speed, since they can be accepted directly by the Kwiktran assembler, thus bypassing the major portion of the normal compilation phase.

When the B option is nominated a KWIK deck is generated for each Kwiktran source program or subprogram. The KWIK deck begins with a BCD card containing the word IDENT beginning in column 10. The IDENT card is followed by a series of binary cards and the KWIK deck is terminated by, but does not include, another IDENT card or a SCOPE card. The last card written on logical unit nominated by the B option will be a SCOPE card. The logical unit is then backspaced over the SCOPE card.

Any Kwiktran source program, subroutine or function may be replaced by its corresponding KWIK deck. Source decks and KWIK decks may be mixed in any order.

If a KWIK deck appears among a series of source decks the control card options are interpreted as follows. The L option is ignored in that an error-free KWIK deck produces no output on the list logical unit. The F and Q options are ignored since they are implicit in the structure of the KWIK deck. In fact, the F and Q options will be those that were nominated when the KWIK deck was generated. All other options are as described in section 2. If the B option is nominated for a KWIK deck an identical deck will be generated on the B logical unit.

If a checksum or sequence error is detected, a fatal diagnostic is printed along with the subprogram name and the card sequence number and the job is terminated. The first binary card after the IDENT card is number 1.

A3.1.1 Binary KWIK Card Formats

Word 1 contents:

bit 47	1
bits 39-46	card checksum folded six times
bits 36-38	5 (binary card 7 and 9 punch)
bits 12-35	first four characters of program name
bits 00-11	card sequence number modulo 2**12

Words 2 - 20 contents:

Nineteen OUTLIS words. The basic fixed format assembly language generated by Fortran is kept in a list called OUTLIS. This list is, if possible, maintained in the core store. For large subroutines OUTLIS is written on to logical unit 51 in 100-word records. Each item (assembly instruction) in OUTLIS consists of two or three words, interpreted as follows.

Word 1:

47

S	U	L	I	F	E	Q		G	T	A	B	C
1	1	1	3	1	1	1	8	7	1	1	7	1

where

FIELD	VALUE	USAGE
S	1	If symbol in word 2 is not in IVLIST, skip this item
	0	No effect
U	1	Force upper
	0	No effect
I	0-7	Value in index field
F	1	G is a numeric function code
	0	Look up G in table
E	1	Word two to be preceded by =S or =H
	0	No effect
Q	1	Word two isbinary
	0	Word two is BCD symbol
T	1	Word three is binary
	0	No word three
A	1	Field B is absolute bank (0-7)
	0	Field B is relative bank

B	0-7 0 0	Bank number if A=1 Local bank if A=0 Set bank to $ if A=0
C		Octal additive or displacement
G	13-80	Function code - see OUTLIS codes

Word 2: BCD symbol orbinary constant

Word 3: Binary constant or BCD symbol for EQU op-code (G=45)

Note that if G=69 (Double Precision Augment) and the S field is non-zero, the CM (Complement) bit will be set in the augment instruction.

OUTLIS CODES (G-FIELD)

13	ENA
14	INA
15	ADD
16	SUB
17	MUI
18	DVI
19	LDA
20	LAC
21	FAD
22	FSB
23	FMU
24	FDV
25	ROP,XOR
26	ENI
27	INI
28	LIL
29	STA
30	AJP,ZR
31	AJP,NZ
32	AJP,PL
33	AJP,MI
34	SLJ
35	IJP
36	BRTJ
37	EXT
38	IDENT
39	ENTRY
40	BLOCK
41	COMMON
42	ORGR
43	OCT
44	BSS
45	EQU
46	END
47	SAU
48	SAL
49	RAD
50	RAO
51	QJP,PL
52	SST
53	SCL
54	SSU
55	LQC
56	STQ
57	LDQ
58	ALS
59	LRS
60	NOT USED
61	SIU
62	SIL
63	LIU
64	ENQ
65	LLS
66	SLS
67	RSO
68	ENO
69	DPA*
70	RTJ
71	ROP,-
72	RGJP,EQ
73	RXT
74	ROP,+
75	BANK
76	ROP,AND
77	ROP,OR
78	UBJP
79	QJP,MI
80	SCM

* Double precision augment

A3.2 Relocatable Binary Decks

The ability to use relocatable binary decks has been added to Kwiktran to make available routines originally written in Compass. If the routines were originally written in Fortran it is suggested that these should be converted to KWIK as this enables array bounds checking to be incorporated.

The relocatable binary decks produced by the Fortran or Compass P option can be used as input to Kwiktran provided that the following conditions are met:

(a) A binary deck must be preceded by a BCD card containing the word IDENT in columns 10-14. IDENT cards between binary subroutines are optional.

(b) The subprogram length is not greater than approximately 12000.

(c) There are no complemented references to external symbols.

(d) Any 'undefined' external symbols as described in section 3 must appear in and be referenced by BRTJ instructions.

If a relocatable binary deck satisfies the above conditions it may be placed in a job anywhere that a source deck or KWIK deck may be used.

Relocatable binary decks produced by Fortran never violate conditions (c) and (d). Arrays in common do not contribute to the length of condition (b).

If a relocatable binary deck appears among a series of source decks the control card options are interpreted as follow. The L option is ignored in that an error-free relocatable binary deck produces no output on the list logical unit. The F and Q options are ignored since they are implicit in the structure of the deck. In fact, the F and Q options will be those that were nominated when the deck was generated. All other options are as described in section 2, however the use of the B option is inadvisable.

APPENDIX 4

A4 KWIKTRAN LIBRARY

ENTRY PTS.	ROUTINE
ABNORMAL	IOPACK.
ABSF	ABSF
ACOSF	ASINF
AD.CHECK	AD.CHECK
AIMAG	CMPLXCVR
ALLOCIN.	IOPACK.
ALLOC.	IOPACK.
ALOG10	ALOG10
AND	MASK32.
ANDD	MASK32.
ASINF	ASINF
ATAN2	ATAN2
ATANF	ATANF
AUTOPLOT	AUTOPLOT
BACKFILE	BACKSKIP
BCDBUF.	IOH.
BCDCKA.	IOH.
BCDERA.	IOH.
BCDERSET	BCDINERR
BFI.	BFI.
BFO.	BFI.
BSPF	BACKSKIP
BSP.	BSP.
BUSY.	IOPACK.
CABS	CABS
CANG	ATAN2
CANGQ8Q	ATAN2
CATAN	CATAN
CCOS	CSIN
CEXP	CEXP
CLOG	CLOG
CLOSEXIT	IOPACK.
CMAGQ8Q	CABS
CMPLX	CMPLXCVR
CONJG	CONJG
COSF	SINF
COTF	COTF
CSIN	CSIN
CSQRT	CSQRT
CUBERTF	CUBERTF
DABS	DABS
DATAN	DATAN
DATAN2	DATAN2
DATE	DATETIME
DBLE	SNGL
DCOS	DSIN
DCUBRT	DCUBRT
DEC.	DEC.
DENS	DENS
DEXP	DEXP
DIMF	DIMF
DISCDOCS	DISCDOCS
DISCDOC.	DISCDOC.
DLOG	DLOG
DLOG10	DLOG10
DMAX1	DMAX1
DMIN1	DMAX1
DMOD	DMOD
DSIGN	DSIGN
DSIN	DSIN
DSQRT	DSQRT
DVCHKF	OVERFLF
EFT.	EFT.
ELB.	IOB.
ELD.	IOH.
ENC.	ENC.
EOFCKF	EOFCKF
EOR	MASK32.
EORD	MASK32.
ERASE	ERASE
EXFLTF	OVERFLF
EXIT	IOPACK.
EXPF	EXPF
FLOATF	FLOATF
IDINT	IDINT
IEQUIV	IEQUIV
INBCDCK	BCDINERR
INBCDCKF	BCDINERR
INTAPE	NEXTCHAR
INTF	INTF
IOCHKF	EOFCKF
IOE.	IOPACK.
IOH.	IOH.
IOP.	IOPACK.
IOR.	IOPACK.
IOS.	IOPACK.
ITOJ	ITOJ
ITOX	ITOX
KTNLIB	KTNLIB
LABEL	LABEL
LENGTHF	LENGTHF
LOGF	LOGF
LUNSET	NEXTCHAR
MACHTYPE	DUMYLINK
MAX0F	MAX1F
MAX1F	MAX1F
MIN0F	MAX1F
MIN1F	MAX1F
MODF	Q8QMODF
NEXTCHAR	NEXTCHAR
NOT	MASK32.
NOTD	MASK32.
OR	MASK32.
ORD	MASK32.

OVERFLF	OVERFLF
PAPERCH	DUMYLINK
PLANT	DUMYLINK
PLOT	PLOT
PLOTCHOP	PLOT
PLOTSET	PLOT
PLTDUMP	PLOT
POWRF	POWRF
PROGLINK	DUMYLINK
PUN.	STH.
Q0Q06200	Q1QDUBLE
Q0Q06300	Q1QCPLEX
Q0Q06500	TYPEBYTE
Q1Q00100	Q1QREINT
Q1Q00200	Q1QDUBLE
Q1Q00210	Q1QDUBLE
Q1Q00300	Q1QCPLEX
Q1Q00310	Q1QCPLEX
Q1Q00320	Q1QCPLEX
Q1Q00500	TYPEBYTE
Q1Q01100	Q1QREINT
Q1Q01200	Q1QDUBLE
Q1Q01210	Q1QDUBLE
Q1Q01300	Q1QCPLEX
Q1Q01310	Q1QCPLEX
Q1Q01320	Q1QCPLEX
Q1Q01500	TYPEBYTE
Q1Q02100	Q1QREINT
Q1Q02200	Q1QDUBLE
Q1Q02210	Q1QDUBLE
Q1Q02300	Q1QCPLEX
Q1Q02310	Q1QCPLEX
Q1Q02320	Q1QCPLEX
Q1Q02330	Q1QCPLEX
Q1Q02500	TYPEBYTE
Q1Q03100	Q1QREINT
Q1Q03200	Q1QDUBLE
Q1Q03210	Q1QDUBLE
Q1Q03300	Q1QCPLEX
Q1Q03310	Q1QCPLEX
Q1Q03320	Q1QCPLEX
Q1Q03330	Q1QCPLEX
Q1Q03500	TYPEBYTE
Q1Q04100	Q1QREINT
Q1Q04200	Q1QDUBLE
Q1Q04210	Q1QDUBLE
Q1Q04300	Q1QCPLEX
Q1Q04310	Q1QCPLEX
Q1Q04320	Q1QCPLEX
Q1Q04330	Q1QCPLEX
Q1Q04500	TYPEBYTE
Q1Q05100	Q1QREINT
Q1Q05200	Q1QDUBLE
Q1Q05210	Q1QDUBLE
Q1Q05300	Q1QCPLEX
Q1Q05310	Q1QCPLEX
Q1Q05320	Q1QCPLEX
Q1Q05330	Q1QCPLEX
Q1Q05500	TYPEBYTE
Q1Q10010	Q1QSTORE
Q1Q10020	Q1QSTORE
Q1Q10030	Q1QSTORE
Q1Q10100	Q1QSTORE
Q1Q10120	Q1QSTORE
Q1Q10130	Q1QSTORE
Q1Q10200	Q1QSTORE
Q1Q10210	Q1QSTORE
Q1Q10230	Q1QSTORE
Q1Q10300	Q1QSTORE
Q1Q10310	Q1QSTORE
Q1Q10320	Q1QSTORE
Q1Q10400	Q1QSTORE
Q1Q10410	Q1QSTORE
Q1Q10420	Q1QSTORE
Q1Q10430	Q1QSTORE
Q1Q10500	TYPEBYTE
Q2Q07000	ITOJ
Q2Q07101	ITOX
Q2Q07110	XTOI
Q2Q07111	POWRF
Q2Q07202	Q2Q07202
Q2Q07212	DPOWER
Q2Q07220	DTOI
Q2Q07221	DPOWER
Q2Q07222	DPOWER
Q2Q07303	Q2Q07313
Q2Q07313	Q2Q07313
Q2Q07323	Q2Q07323
Q2Q07330	Q2Q07330
Q2Q07331	Q2Q07331
Q2Q07332	Q2Q07331
Q2Q07333	Q2Q07331
Q2QDLDA	Q2QDLDA
Q2QLOADA	Q2QLOADA
Q3Q00040	Q1QREINT
Q3Q00140	Q1QREINT
Q3Q00240	Q1QDUBLE
Q3Q00340	Q1QCPLEX
Q3Q00550	TYPEBYTE
Q3Q01040	Q1QREINT
Q3Q01140	Q1QREINT
Q3Q01240	Q1QDUBLE
Q3Q01340	Q1QCPLEX
Q3Q01550	TYPEBYTE
Q3Q02040	Q1QREINT
Q3Q02140	Q1QREINT
Q3Q02240	Q1QDUBLE
Q3Q02340	Q1QCPLEX
Q3Q02550	TYPEBYTE
Q3Q03040	Q1QREINT
Q3Q03140	Q1QREINT
Q3Q03240	Q1QDUBLE
Q3Q03340	Q1QCPLEX
Q3Q03550	TYPEBYTE

Q3Q04040	Q1QREINT	Q8QPOWRF	POWRF
Q3Q04140	Q1QREINT	Q8QRANF	RANF
Q3Q04240	Q1QDUBLE	Q8QRESID	Q8QRESID
Q3Q04340	Q1QCPLEX	Q8QSENLT	Q8QSENLT
Q3Q04550	TYPEBYTE	Q8QSIGNF	SIGNF
Q3Q05040	Q1QREINT	Q8QSINF	SINF
Q3Q05140	Q1QREINT	Q8QSNGL	SNGL
Q3Q05240	Q1QDUBLE	Q8QSOS	IOPACK.
Q3Q05340	Q1QCPLEX	Q8QSQRTF	SQRTF
Q3Q05550	TYPEBYTE	Q8QSTOPS	Q8QPAUSE
Q3Q10040	Q1QSTORE	Q8QTANF	TANF
Q3Q10050	TYPEBYTE	Q8QTANHF	TANHF
Q3Q10140	Q1QSTORE	Q8QTRACE	Q8QTRACE
Q3Q10240	Q1QSTORE	Q8QTRICE	Q8QTRACE
Q3Q10340	Q1QSTORE	Q8QXABSF	ABSF
Q3Q10440	Q1QSTORE	Q8QXDIMF	XDIMF
Q3Q10550	TYPEBYTE	Q8QXFIXF	XFIXF
Q7QLODLC	Q7QLODLC	Q8QXINTF	XFIXF
Q8QABSF	ABSF	Q8QXMODF	Q8QXMODF
Q8QACOSF	ASINF	Q8QXSIGN	SIGNF
Q8QASINF	ASINF	Q8QXTOI	XTOI
Q8QATANF	ATANF	Q9QEVALB	Q9QEVAL
Q8QCHAIN	IOPACK.	Q9QEVALL	Q9QEVAL
Q8QCORE.	IOPACK.	QNDOUBL.	IOPACK.
Q8QCOSF	SINF	QNSINGL.	IOPACK.
Q8QCOTF	COTF	RANF	RANF
Q8QCUBER	CUBERTF	RANFGET	RANF
Q8QDBLE	SNGL	RANFSET	RANF
Q8QDCONS	Q8QDLDA	RDISC	DISCJOCK
Q8QDICT.	IOPACK.	RDRUM	DRUMMER
Q8QDIMF	DIMF	RDRUMINT	DRUMMER
Q8QDLDA	Q8QDLDA	REAL	CMPLXCVR
Q8QDLODA	Q8QDLDA	RELEASE	UNLOAD
Q8QENTRY	IOPACK.	RETURN.	IOPACK.
Q8QERROR	IOPACK.	REW.	REW.
Q8QERSET	IOPACK.	SAVE	SAVE
Q8QEXPF	EXPF	SCALER	SCALER
Q8QFLOAT	FLOATF	SHIFT	SHIFT
Q8QHIST.	IOPACK.	SHIFTD	SHIFT
Q8QIFDIV	Q8QIFDIV	SIGNF	SIGNF
Q8QIFEOF	Q8QIFIOC	SINF	SINF
Q8QIFEXP	Q8QIFDIV	SKIP	BACKSKIP
Q8QIFIOC	Q8QIFIOC	SKIPFILE	BACKSKIP
Q8QIFOVF	Q8QIFDIV	SLI4.	SLIO4.
Q8QIFSLT	Q8QSENLT	SLITE	SLITETF
Q8QIFUNI	Q8QIFUNI	SLITEF	SLITETF
Q8QINP4	Q8INOUT4	SLITETF	SLITETF
Q8QINTF	INTF	SLI.	SLI.
Q8QITOJ	ITOJ	SLO4.	SLIO4.
Q8QITOX	ITOX	SLO.	SLI.
Q8QLDCON	Q8QLOADA	SNGL	SNGL
Q8QLOADA	Q8QLOADA	SQRTF	SQRTF
Q8QLODA	Q8QLOADA	SSWTCHF	SSWTCHF
Q8QLOGF	LOGF	STATUS	STATUS
Q8QMODF	Q8QMODF	STB.	IOB.
Q8QOUT4	Q8INOUT4	STH.	STH.
Q8QPAUSE	Q8QPAUSE	TAN	TANF

TANF	TANF
TANHF	TANHF
TEXT	PLOT
THEND.	IOPACK.
TIME	DATETIME
TIMEF	TIMEF
TIMELEFT	TIMEF
TSB.	IOB.
TSH.	TSH.
UNITSTF	UNITSTF
UNLOAD	UNLOAD
UNSAVE	SAVE
UTILITY	UTILITY
UTILITY.	UTILITY.
WDISC	DISCJOCK
WDRUM	DRUMMER
WDRUMINT	DRUMMER
XABSF	ABSF
XDIMF	XDIMF
XFIXF	XFIXF
XINTF	XFIXF
XMAX0F	MAX1F
XMAX1F	MAX1F
XMIN0F	MAX1F
XMIN1F	MAX1F
XMODF	Q8QXMODF
XSIGNF	SIGNF
XTOI	XTOI
.REPCNT.	IOH.
.TSERR.	IOH.