

IBM SYSTEM/360 OPERATING SYSTEM  
PL/I (F) COMPILER  
PROGRAM LOGIC MANUAL

This Technical Newsletter, a part of Release 17, of IBM System/360 Operating System, provides replacement pages for the PL/I (F) Compiler, Program Logic Manual, Form Y28-6800-3. These replacement pages remain in effect for subsequent releases unless specifically altered. Pages to be inserted and/or removed are listed below.

23,24  
39,40  
40.1 (added)  
61  
127,128  
343,344  
405-408  
413-416  
421-428.1

A change to the text or a small change to an illustration is indicated by a vertical line to the left of the change; a changed or added illustration is denoted by the symbol • to the left of the caption.

Summary of Amendments

Additional material on BASED items has been added to Phase FX in Section 2; system generation material for the Model 91 has been incorporated in Appendix G; and the list of diagnostic messages in Appendix I has been updated.

Note: Please file this cover letter at the back of the manual to provide a record of changes.

The compiler completion codes are as follows:

<u>Code</u>	<u>Meaning</u>		
0	No diagnostic messages issued; compilation completed with no errors		
4	Warning messages issued; program compiled		
8	Error messages issued; program compiled but with errors; execution may fail	12	Severe error messages issued; compilation may be completed but with errors, successful execution improbable. If a severe error occurs during compile-time processing, a listing of the PL/I text on SYSUT3 will be printed if the SOURCE option is specified. The compilation will be terminated.
		16	Terminal error messages issued; compilation terminated abnormally

Data Set	SYSIN	SYSLIB	SYSLIN	SYSPRINT	SYSPUNCH	SYSUT1	SYSUT3
<u>Module</u>							
AA <sup>1</sup>			WRITE			OPEN <sup>3</sup>	
AB	OPEN		OPEN	OPEN/WRITE	OPEN	OPEN <sup>3</sup>	OPEN
AC <sup>1</sup>							WRITE/READ
AE <sup>1</sup>		OPEN					
AG							CLOSE/OPEN
BW						CLOSE	
BX	READ						
CI	READ			WRITE		READ/WRITE	WRITE/READ
AS <sup>2</sup>	READ	READ		WRITE			WRITE
FY				WRITE			
UA				WRITE			
UD					WRITE		
UF				WRITE			
XB				WRITE			
AE	CLOSE						CLOSE
AK <sup>4</sup>						CLOSE	
AK <sup>5</sup>		CLOSE	CLOSE	CLOSE	CLOSE	CLOSE	

<sup>1</sup>AA, AC, and AE are modules of the control phase, and contain actual I/O routines which interface with the O/S access methods (BSAM, QSAM). I/O activity shown for other modules indicates that these modules are utilizing the I/O routines.

<sup>2</sup>AS may read from included data sets in addition to those shown in the table.

<sup>3</sup>If the SIZE option results in 1K text and dictionary blocks, SYSUT1 is opened by Module AB. In the case of other SIZE options, SYSUT1 is opened by Module AA when the available main storage is full. The timing depends on the size of program to be compiled.

<sup>4</sup>AK closes only the spill file SYSUT1 when the batch option is used and a further source program is to be compiled.

<sup>5</sup>AK closes all other files upon termination of a single or batch compiler run.

● Figure 4. Input/Output Usage Table

COMPILER CONTROL MODULES

Module AC

In addition to modules AA and AB, further modules, AC, AD, AE, AF, AG, AH, AI, AK, AL, AM, AN, and JZ are used in compiler control. The functions of these modules are briefly described in the following paragraphs.

Module AC controls reading and writing operations on SYSUT3, the intermediate file. It is loaded only if the CHAR48 or MACRO option is specified, and is deleted at the end of the Read-In Phase.

ameter descriptions for entry points and defined items, are placed in the AUTOMATIC chain for the appropriate block.

7. The dictionary entry for any item described by a picture is expanded by the precision and scale or string length, extracted from the picture table entry. Identifiers of different modes sharing the same picture table are now placed in separate tables.
8. The 'dope vector required' bit (see Appendix C.5) is set on where necessary.
9. When a label array is found which has initial label statements for any of its elements, the chained statements are moved into the second file. The original statement is left in the text, to be removed by Phase FV.
10. Dictionary entries similar to label BCD entries are made for all TASK variables.

Phase FV

Phase FV scans the second file and reverses the pointers to the dictionary.

Dictionary entries for DEFINED data are completed (see Appendix C.4 and C.5). Overlay and correspondence defining are differentiated between, as are static and dynamic defining. A preliminary check of the validity of defining is also carried out.

When PROCEDURE and BEGIN statements are encountered, any second file statements associated with data in the AUTOMATIC chain for that block are inserted in the text following such statements.

When ALLOCATE statements are found, any second file statements associated with the item being allocated are inserted in the text following the statement.

When a reference to dynamically defined data is found, the base reference is inserted into the text following the defined reference.

When an initial label statement is encountered in the main text, it is not copied into the output string.

The dictionary reference of a POINTER in a PEXP (pointer expression) second file statement is inserted into the defined slot of the associated based variable. If the

based variable is a structure this reference is propagated throughout the structure. The PEXP statement is then deleted.

A similar procedure is performed for BVEXP (based variable expression) second file statements whereby the dictionary reference of the AREA is inserted into the dictionary entry of the associated OFFSET variable.

ADV second file statements referring to a BASED variable are checked for compliance with the (F) implementation rules. If the rules are obeyed, the dictionary entry of the 'bound' variable is inserted in the appropriate slot in the multiple table entry.

If an MTF statement refers to a based variable the appropriate bound slot is copied from one multiple table entry to the other.

Phase FX

Phase FX is an optional phase entered only if the XREF or ATR (cross reference lister and attribute lister respectively) options are specified. It scans the STATIC, AUTOMATIC, and CONTROLLED chains, and the formal parameter lists.

For each identifier it creates an entry in scratch text storage of the form:

2 bytes	3 bytes	3 bytes
Dictionary reference	Text reference to this item	Text chain

This entry is inserted into a chain of similar entries in the alphabetical order of the BCD of the identifier.

If the XREF option is specified, the text is scanned for dictionary references. When the dictionary reference of an identifier is found in the text, an entry is created in a chain of entries from the dictionary entry of the identifier. If the identifier is that of a BASED item, an entry is also created in a chain of entries from the dictionary entry of the associated pointer.

Each chain member thus represents a text reference to an identifier and has the form:

2 bytes	3 bytes
Statement number	Text chain

Each reference chain for an identifier is in scratch text storage.

The sorted chain of identifiers is then scanned, and for each entry in the chain the following actions take place:

1. The statement number of the DECLARE statement, if any, in which the identifier was declared is printed
2. The BCD of the identifier is printed. For variables having constant dimensions and/or constant string lengths, these dimensions and lengths will be listed.
3. If the ATR option is specified, the dictionary entry of the identifier is analyzed and its attributes are printed
4. If the XREF option is specified, the reference chain for the identifier is scanned, and the statement number contained in each entry is printed

Finally, all scratch storage is released and control is passed to the Pretranslator Phase.

#### THE PRETRANSLATOR LOGICAL PHASE

The purpose of the Pretranslator Phase is to expand those statements in the language that can be broken down into simpler statements, and to insert explicitly generated statements in place of implied ones.

Second level markers (see Appendix D.1) are removed from internal compiler codes, and some of the I/O statements are changed into a form more suitable for the pseudo-code phase.

Argument lists are examined and the matching of arguments with parameter descriptions takes place, with temporary variables being created where necessary, e.g., where data conversions are required.

If the compilation contains ON CHECK conditions the appropriate calls to the library routine are provided.

Any structure assignments containing the BY NAME option are processed.

If any structure assignment statements or structures in I/O lists are detected in the program, they are expanded into scalar assignments and DO groups.

If the program contains any array assignments, or array expressions in I/O

lists, these are expanded into DO loops and scalar assignments or expressions.

If the program contains iSUB references, the subscripts are computed for the base array corresponding to the subscripts given for the defined array.

#### Additions to the Text

In addition to changing the content of the text, the Pretranslator introduces some new symbols and grammatical forms into the source text. These are as follows:

The Umbrella Symbol: this is designated by the symbol code X'5E', which is used to introduce a literal as an operand. It is used only as a bound of a DO loop, or in a call of the dope vector pseudo-variable.

Statements within statements: a list of statements may be introduced within another statement. In this case the inserted list is enclosed in parentheses. Statements in the list are given no statement number field, but they have semi-colons at the end.

I/O statements: the form of I/O statements is changed considerably during the pretranslator phases, as explained in the description of Phase GB.

BUY and SELL statements: special statements are introduced for manipulating temporary storage at object time; they have a form similar to ALLOCATE and FREE statements.

Temporary Storage: Pretranslator phases create temporary variables for function and procedure calls where the arguments do not match the final parameters, where expressions appear as arguments, for control variables for DO loops in array and structure assignments, and for iSUB defined subscript lists. The Pretranslator has no mechanism for evaluating expressions. Therefore, temporaries which have no data type are created for expression arguments with no parameter description. The data type of these temporaries is completed by the Translator generic phase when the resultant data type of the expression has been determined.

When the Pretranslator creates a temporary from an argument which contains any array with adjustable bounds or adjustable string length, compiler functions (see Appendix D.8) are gener-

ated in-line, to set up the adjustable quantities at object time, to enable storage of the correct size to be acquired by means of the BUY statement.

Any initial value statements associated with the ALLOCATE statement are extracted and placed in-line. The initialization statements are then skipped, and the scan continues. The last two steps are also performed for LOCATE (based variable) and ALLOCATE (based variable) statements. Action for a BUY statement is similar to an ALLOCATE statement, with the following exceptions:

1. Bound and string length code is in-line, bracketed between BUYS and BUY statements - there is therefore no look ahead
2. There is no initial value code associated with temporaries
3. A slot in the DSA is updated with the pointer to the allocated storage for a temporary

The action on encountering a FREE statement is to generate code to load a parameter register with the pointer to the allocated storage for the FREE VDA Library call inserted by the pseudo-code.

#### Phase QU

Phase QU scans the pseudo-code text in search of instructions which have misaligned operands. (A misaligned operand has the UNALIGNED attribute and is not aligned on the boundary appropriate to its data type). When such an instruction is found, QU inserts a move character (MVC) instruction in the pseudo-code text to move the operand to or from an aligned workspace area, and substitutes the address of this workspace for the operand address in the original instruction. If the address of a misaligned operand is loaded into a register, a note is made of that register. QU thereafter treats the instructions which refer to it as if they referred to the operand itself, by inserting a move character instruction, and substituting the workspace address for the reference in the instruction.

Phase QU uses storage beginning at offset 32 from register 9 for its workspace.

Whenever a load address (LA) instruction is found which lies within the calling sequence of a library routine and which loads the address of a misaligned argument of that routine, an aligned workspace address is substituted in the instruction, and the requisite move character instruction is stacked. It is not inserted in the output text until the instruction is encountered that loads register 15 prior to

the exit to the library routine, or in the case of EDIT-directed I/O routines, until the appropriate branch-and-link (BALR) instruction is encountered. The stacked move character instruction is inserted into the output before the exit to the routine if the argument in question is an input argument to the routine, and after the return from the routine if it is an output argument.

#### Phase QX

Phase QX is the 'AGGREGATE LENGTH TABLE' printing phase. It is entered only if the ATR (attribute list) or XREF (cross reference list) options are specified. It scans the STATIC, AUTOMATIC, CONTROLLED and COBOL chains, and, for each major structure or non-structured array that is found, an entry is printed in the AGGREGATE length table.

An AGGREGATE LENGTH TABLE entry consists of the source program DECLARE statement number, the identifier and the length (in bytes) of the aggregate. In the case of a CONTROLLED non-BASED aggregate no entry is printed for the DECLARE statement, but an entry is printed for each ALLOCATE for the aggregate, the source program ALLOCATE statement number being printed in the 'statement number' column.

Where the length of an aggregate is not known at compilation the word "ADJUSTABLE" is printed in the 'length in bytes' column of the entry for that aggregate. If an aggregate is dynamically defined, the word "DEFINED" appears in that column. Entries for COBOL mapped aggregates have the word "(COBOL)" appended.

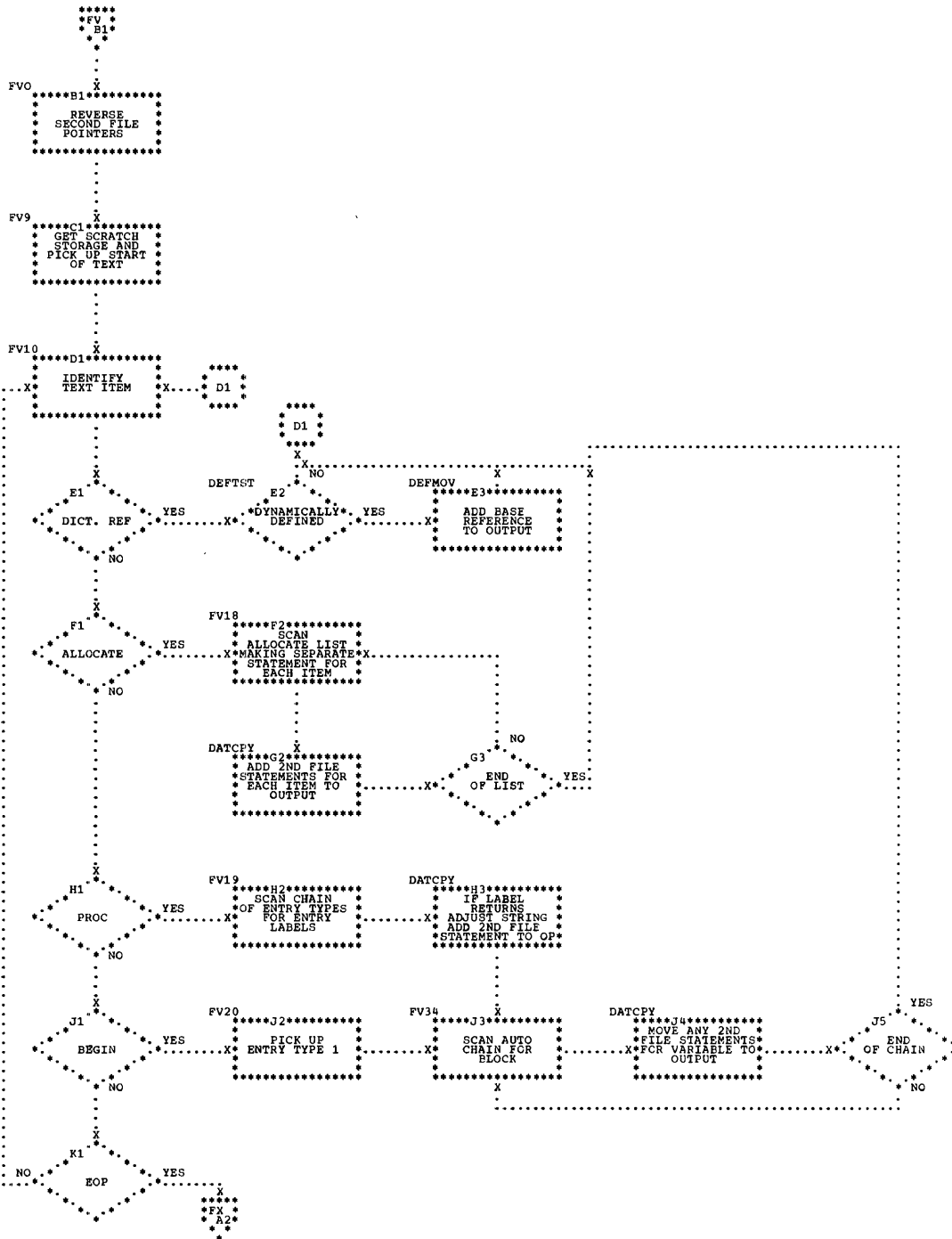
Before printing begins the aggregate length table entries are sorted so that the identifiers appear in collating sequence order.

#### THE REGISTER ALLOCATION LOGICAL PHASE

The purpose of the Register Allocation Phase is to insert into the text the appropriate addressing mechanisms for all types of storage, and to allocate physical general registers where symbolic registers are specified or required as base registers.

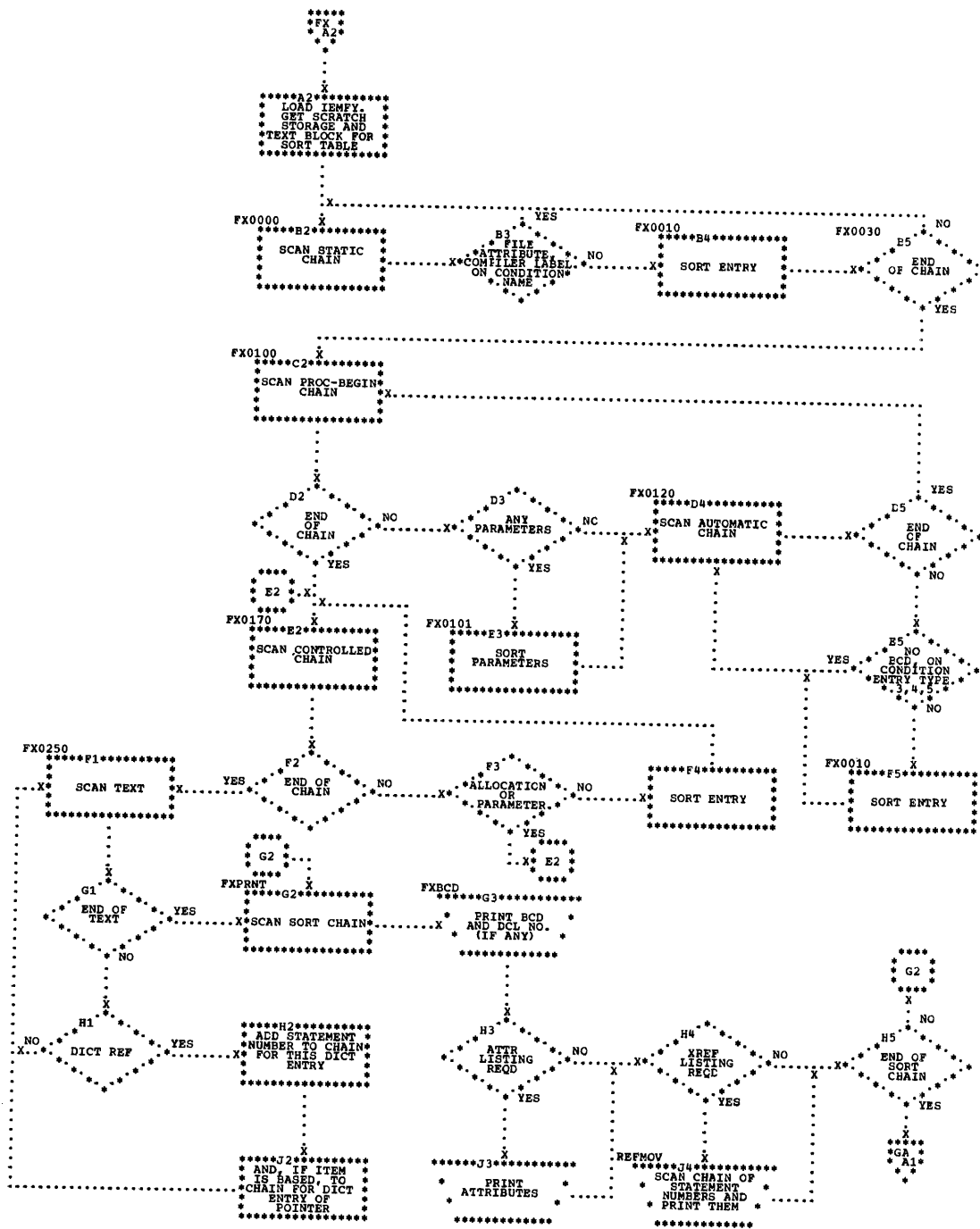
This phase comprises two physical phases, each with a specific function. The first, Phase RA, processes the addressing mechanisms, while the second phase, Phase RF, allocates the physical registers.

Chart FV. Phase FV Overall Logic Diagram





● Chart FX. Phase FX Overall Logic Diagram



PHASE DIRECTORY

Because of the number of phases in the compiler, the phase directory is split into halves. The first half is constructed during the initialization of the compiler; also a list of names of the phases in the second half is kept in Phase AA. This list is used to pass status indications (i.e., whether phases are wanted or not wanted) from the first half to the second half. Phase JZ uses the list to construct a new directory for the second half.

The phase directory is constructed by use of the BLDL macro and a build list. The format of the build list is fully described in the publication IBM System/360 Operating System, Supervisor and Data Management Services, Form C28-6646.

Each entry in the build list is 30 bytes long. On returning from the BLDL macro, two bytes of the name field and ten other bytes of each satisfied entry in the build list are used to construct a 12-byte phase directory entry in the compiler control routines. The build list is destroyed after the initialization process is complete.

The format of a phase directory entry is as follows:

<u>Byte Number</u>	<u>Description</u>
1 - 2	Phase name
3	Status byte
4 - 5	Concatenation number and Library identification
6 - 8	TTR of first text record; where TT is the relative track number, and R is the block number on that track
9 - 10	Total amount of storage required
11 - 12	Length of first text record

Control Code Word -- CCCODE

The format of the control code word (CCCODE), which is four bytes in length, is as follows:

<u>Byte</u>	<u>Bit</u>	
0	0	DUMP 1 wanted 0 not wanted

1		1 abort has occurred
2	LIST	1 not wanted 0 wanted
3	LOAD	1 not wanted 0 wanted
4	DECK	1 not wanted 0 wanted
5	EXTREF	1 not wanted 0 wanted
6	XREF	1 not wanted 0 wanted
7	ATR	1 not wanted 0 wanted
1	0	1 means U-format 0 means F-format records on input
1		1 if track overflow is present
2		Severity code
3		Severity code
4		Severity code
5		Severity code where 0000=FLAGW 0001=FLAGE 0010=FLAGS
6	CHAR 48	1 not wanted 0 wanted
7	MACRO	1 not wanted 0 wanted
2	0	SOURCE 1 not wanted 0 wanted
1	CHK	1 not wanted 0 wanted
2	BCD	1 BCD input 0 EBCDIC input
3	SOURCE2	1 wanted 0 not wanted
4	OPT	1 wanted 0 not wanted
5		1 AE required
6		1 program check has occurred
7		1 means first record has been read
3	0	STMT 1 not wanted 0 wanted

1	MACDCK	1 not wanted 0 wanted
2	COMP	1 not wanted 0 wanted
3		1 macro phase now running
4		1 batch record found 0 batch record not found
5		1 EOF record found 0 EOF record not found
6		not used requirement
7	NEST	1 wanted 0 not wanted

APPENDIX E: STORAGE REQUIREMENTS

The (F) Compiler requires main storage for the following purposes :

- Compiler processing phases
- Print buffers
- Compiler control routines
- Dictionary area
- Text area
- Input/Output buffers
- Input/Output routines (BSAM)

the block size. The space needed for large I/O buffers is subtracted from SIZE and the space remaining is used when determining text and dictionary block size, as detailed in the publication IBM System/360 PL/I(F) Programmer's Guide. A table contained in Phase AB is searched, using the SIZE option as an argument. When the correct entry is found, the block size is extracted.

The first table shows the relationship between the compiler requirements and the text and dictionary block sizes. The second table details the storage allocation in each environment.

The main storage required by each phase of the compiler need be contiguous only for each control section.

During the read-in phases a minimum of two dictionary blocks and two text blocks are available in storage simultaneously.

During the rest of the compilation four dictionary blocks and four text blocks are available in storage simultaneously.

The dictionary and text block size is chosen according to the amount of main storage available to the compiler. The SIZE option, interpreted at invocation time, provides the value used to determine

Compiler Requirements and Dictionary/Text Block Relationship

Environment	Dictionary/Text Block Size	Compiler Requirements
A	1K	44K - 56K
B	2K	56K - 72K
C	4K	72K - 132K
D	8K	132K - 168K
E	16K	168K

Storage Allocation	DURING READ-IN PHASE					AFTER READ-IN PHASE				
	ENVIRONMENT					ENVIRONMENT				
	A	B	C	D	E	A	B	C	D	E
OS Dynamic Storage										
TIOT	228	228	228	228	228	228	228	228	228	228
SPIE	32	32	32	32	32	32	32	32	32	32
LOAD	40	40	40	40	40	40	40	40	40	40
OS Temporary Storage										
End of Volume	800	800	800	800	800	800	800	800	800	800
Data Management	4894	4894	4894	4894	4894	4894	4894	4894	4894	4894
Compiler Control	12266	12266	12266	12266	12266	12266	12266	12266	12266	12266
Phase Area	16384	16384	16384	16384	16384	12288	12288	12288	12288	12288
Text Area	2048	8192	16384	32768	65536	4096	8192	16384	32768	65536
Dictionary Area	2048	8192	16384	32768	65536	4096	8192	16384	32768	65536
Scratch Storage	4096	4096	4096	4096	4096	4096	4096	4096	4096	4096
I/O Buffers	1538	1538	1538	1538	1538	1538	1538	1538	1538	1538
TOTALS	44374	56662	73038	105814	171340	44374	52566	68950	101718	167254

Initially, four text and four dictionary blocks are allocated to the compiler (two each are allocated when only 44K bytes of storage are available to the compiler. This is then expanded to four of each at the end of the read-in phase). If the text and/or dictionary expands to fill these

blocks, more main storage is allocated as blocks. This process continues until the spill point is reached (i.e., until all the main storage available to the compiler has been used). If still more main storage is required, the spill file (SYSUT1) is opened, and blocks are written out.

APPENDIX F: COMMUNICATIONS REGION

The communications region is an area specified by the control routines, and used to communicate necessary information between the various phases of the compiler. The communications region is resident in the first dictionary block throughout the compilation.

Entry to the various compiler control routines is via a transfer vector. Details of the transfer vector and the organization of the communications region appear in this Appendix.

Note: The use of the communications region during compile-time processing is described in Appendix J.

TRANSFER VECTORS

Hex. Offset	Name	Description
8	ZUPL	Print a line
C	ZURD	Read a card
10	ZUGC	Get scratch storage
14	ZUTXTC	Get text block
18	ZURC	Release scratch storage
1C		
20	ZABORT	Dump and go to error message routines
24	ZLOADW	Load and return to caller
28	ZDICAB	Make dictionary entry. Absolute address returned
2C	ZDICRF	Make dictionary entry. Dictionary reference returned
30	ZUERR	Make error message entry
34	ZDRFAB	Convert dictionary reference to absolute address
38	ZLOADX	Load with overlay and return to caller

3C		
40	REQUEST	Give a list of phase names required or not wanted for this compilation
44	RELESE	Release all named phases
48	RLSCTL	Release all named phases and pass to next phase
4C	ZDUMP	Dump specified core and continue
50	ZTXTRF	Convert absolute address to text reference
54	ZTXTAB	Convert text reference to absolute address
58	ZCHAIN	Find next block in chain
5C	ZALTER	Change text block status
60	ZDABRF	Convert absolute address to dictionary reference
64	ZNALRF	Not aligned dictionary entry. Reference returned
68	ZNALDB	Not aligned dictionary entry. Absolute address returned
6C	ZEND	Terminate job
70	ZULF	Write on load file
74	ZUSP	Write on punch
78	ZUBW	Write on backing store
80	RLSCTLX	Release all named phases and hand control to the next phase, after having loaded it with overlay
84	RECONS	Reconstitute instructions in IEMAL
88	DYNAMIC	Pass control to the dynamic dump routines, if required
8C	IEMAL/ IEMAN	Address of second control phase

## COMMUNICATIONS REGION

These tables give the following information for each location of the communications region: name of location; offset (i.e., relative address); use (i.e., stages of compilation during which the location is in use); and a description of the contents. Certain locations are used in one capacity during part of the compilation, and then re-used in a different capacity during another part of the compilation. In these cases, one location will have two table entries: details of alternative usage appear in the columns headed Name<sub>2</sub>, Use<sub>2</sub>, etc.

Table 3. Communications Region. Bit Usage in ZFLAGS

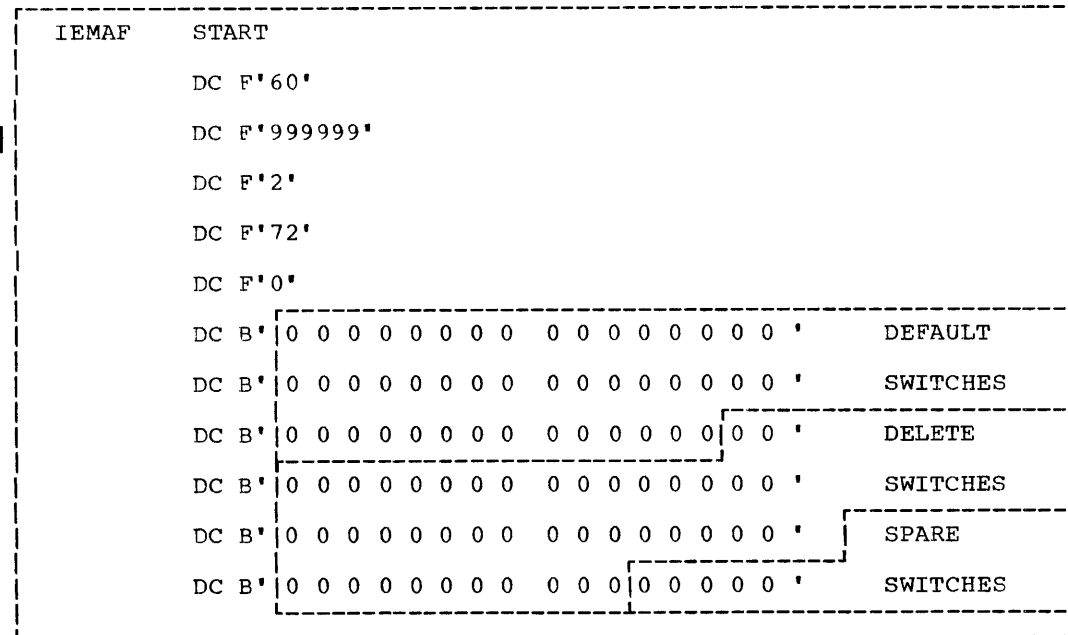
BYTE NAME	OFFSET	BIT (HEX)	BIT NAME	DESCRIPTION Bits are set on, on encountering:-
ZFLAG1	ZCOMM+16	80	ZDEFFL	DEFINED attribute
		40	ZAWAFL	ALLOCATE statement
		20	ZSECFL	Second File statement
		10	ZDIMFL	Dimension attribute
		08	ZCHKFL	CHECK/NOCHECK prefix
		04	ZONFL	ON, SIGNAL or REVERT statement
		02	ZSTRFL	Structure
		01	ZDECFL	DECLARE statement
ZFLAG2	+17	80	ZLIKFL	LIKE attribute
		40	ZINTST	STATIC INITIAL
		20	ZOPCFL	OPEN/CLOSE statement
		10	ZGTPFL	GET/PUT statement
		08	ZGOTFL	GO TO statement
		04	ZTEPFL	TASK/EVENT/PRIORITY options, REPLY statement
		02	ZPICFL	PICTURE attribute/format item
		01	ZISBFL	iSUB defining
ZFLAG3	+18	80	ZCONTG	UNALIGNED(NONSTRING) attribute
		40	ZSETFL	SETS attribute
		20	ZOSSFL	DELAY, DISPLAY, WAIT statement
		10	ZARGFL	Argument list
		08	ZINLFL	INITIAL Label
		04	ZDIOFL	DATA directed I/O
		02	ZRECIO	RECORD I/O
		01	ZINTAC	AUTO/CTL initialization
ZFLAG4	+19	80	ZFREE	FREE statement
		40	STM256	More than 256 statements
		20	FILEFL	Files present
		10		SPARE
		08	ZPUTFL	PUT DATA
		04	ZGETFL	GET DATA
		02	ZPTRFL	Pointer Qualifier
		01	ZRODFL	STATIC DSA Entry
ZFLAG5	+20	80	ZFTASK	TASK/EVENT/PRIORITY option on a CALL statement
		40	ZDENFL	Set by FT
		20	ALCSLM	ALLOCATE, with second level marker
		10		Spare
		to 01		



APPENDIX G: SYSTEM GENERATION

For full details of the system generation process, see IBM System/360 Operating System: System Generation, Form C28-6554.

During the system generation process, a control section named IEMAF is assembled (see Figure 13) containing a table consisting of five fixed-point values aligned on full-word boundaries, immediately followed by a bit string field that is twelve bytes in length. The five fixed-point values are related to the compiler options LINECNT, SIZE, SORMGIN (start), SORMGIN (end), and CONTROL COLUMN (PAGECTI), respectively. Bits 0 to 38, and 42 to 45 in the string are used to specify the default status of the options. Bits 46 to 91 in the string are used to specify if an option keyword is to be deleted or not. A "1" in the bit string means "yes" and a "0" means "no". The remaining bits in the string are spare bits not currently in use. Figure 14 shows the bit identification table associated with the control section.



● Figure 13. The IEMAF Control Section

Bit	Parameter	Bit	Parameter	Bit	Parameter
0	ATR	32	NOOPLIST	64	DELETE=NOXREF
1	NOATR	33	STMT	65	DELETE=SOURCE
2	BCD	34	NOSTMT	66	DELETE=NOSOURCE
3	EBCDIC	35	MACRO	67	DELETE=SOURCE2
4	CHAR60	36	NOMACRO	68	DELETE=NOSOURCE2
5	CHAR48	37	NEST	69	DELETE=OPT
6	DECK	38	NONEST	70	DELETE=LINECNT
7	NODECK	39	Not used	71	DELETE=LINELNG
8	EXTREF	40	Not used	72	DELETE=SIZE
9	NOEXTREF	41	Not used	73	DELETE=SORMGIN
10	FLAGW	42	COMP	74	DELETE=DUMP
11	FLAGE	43	NOCOMP	75	DELETE=STMT
12	FLAGS	44	LIBRARY OPTION, REAL	76	DELETE=NOSTMT
13	LIST	45	LIBRARY OPTION, COMPLEX	77	DELETE=MACRO
14	NOLIST	46	DELETE=ATR	78	DELETE=NOMACRO
15	LOAD	47	DELETE=NOATR	79	DELETE=COMP
16	NOLOAD	48	DELETE=BCD	80	DELETE=NOCOMP
17	XREF	49	DELETE=EBCDIC	81	DELETE=M91
18	NOXREF	50	DELETE=CHAR60	82	DELETE=NOM91
19	SOURCE	51	DELETE=CHAR48	83	DELETE=PAGECTL
20	NOSOURCE	52	DELETE=DECK	84	DELETE=MACDCK
21	SOURCE2	53	DELETE=NODECK	85	DELETE=NOMACDK
22	NOSOURCE2	54	DELETE=EXTREF	86	DELETE=EXTDIC
23	OPT=0	55	DELETE=NOEXTREF	87	DELETE=NOEXTDIC
24	OPT=1	56	DELETE=FLAGW	88	DELETE=OPLIST
25	M91	57	DELETE=FLAGE	89	DELETE=NOOPLIST
26	NOM91	58	DELETE=FLAGS	90	DELETE=NEST
27	MACDCK	59	DELETE=LIST	91	DELETE=NONEST
28	NOMACDCK	60	DELETE=NOLIST	-	Bits 92-95
29	EXTDIC	61	DELETE=LOAD		not used
30	NOEXTDIC	62	DELETE=NOLOAD		
31	OPLIST	63	DELETE=XREF		

●Figure 14. Bit Identification Table

APPENDIX H: CODE PRODUCED FOR PROLOGUES AND EPILOGUES

The mechanism of dynamic storage management is described in the publication IBM System/360 Operating System PL/I Library Program Logic Manual, Form Z28-6591

Part of the code required to implement the storage management is generated as prologue and epilogue code by the compiler. This Appendix contains annotated examples of prologues and epilogues for PROCEDURE, BEGIN, and ON blocks.

PROLOGUES AND EPILOGUES

Example in PL/I

```

A:I: PROCEDURE(X,Y);
    DECLARE Y CONTROLLED;
    .
    .
    ON OVERFLOW C=0;
    .
    .
B: BEGIN;
    .
    .
    END;
    .
    .
AB:IJK: ENTRY(Y,Z)
    .
    .
    RETURN(EXPRESSION)
    .
    .
    END;

```

A	BC	15,16,(0,15)	BRANCH ROUND FOLLOWING CONSTANTS
	DC	A11(1)	LENGTH OF BCD
	DC	C'A'	BCD OF ENTRY POINT
SIZDSA	DC	F' SIZE OF DSA'	ADDRESS OF STATIC INTERNAL CONTROL SECTION
STATIC	DC	A(STATIC CONTROL SECTION)	(ONLY COMPILED FOR EXTERNAL AND ON PROLOGUES)
	STM	14,11,12,(13)	SAVE STANDARD REGISTERS IN SAVE AREA OF
	*		CALLER'S DSA
	LR	10,15	SET UP FIRST PROLOGUE BASE
	BAL	8,GETDSA(0,10)	CALL ROUTINE TO GET DSA
	MVI	SWITCH(13),X'X1'	INSERT RETURN (EXPRESSION) SWITCH
*			(ONLY COMPILED IF THERE IS A RETURN (EXP)
*			AND THE ENTRY LABELS HAVE DIFFERENT DATA
*			ATTRIBUTES)
	BC	15,COPRAM1(0,10)	BRANCH TO COPY OVER PARAMETERS

```

ST      4,84(13)
ST      2,80(13)
ST      2,8(13)
MVI     76(13),X'00'
ST      2,96(13)
BR      14
L       15,32(11)
BR      15

*      END SUBROUTINE

*      STATIC PROLOGUE SUBROUTINE

L       4,PR..IHEQINV(12)
LTR     4,4
BC      11,86(15)
L       7,PR..IHEQLWO(12)
MVC     80(4,3),80(7)
LA      4,1(4)
ST      4,PR..IHEQINV(12)
ST      4,84(3)
MVI     76(3),X'00'
ST      3,8(13)
LR      13,3
L       3,PR..IHEQSLA(12)
ST      3,4(13)
ST      13,PR..IHEQSLA(12)
SR      2,2
ST      2,80(13)
ST      2,8(13)
ST      2,96(13)
BR      14

*      END SUBROUTINE

*      EPILOGUE SUBROUTINE

TM      1(13),X'80'
BC      8,60(15)
L       2,80(13)
LTR     2,2
BC      7,60(15)
C       13,PR..IHEQSLA(12)
BC      7,60(15)
L       13,4(13)
ST      13,PR..IHEQSLA(12)
TM      0(13),X'80'
BC      1,50(15)
L       13,4(13)
B       34(15)
ST      2,8(13)
LM      14,11,12(13)
BR      14

*      END SUBROUTINE

```

APPENDIX I: DIAGNOSTIC MESSAGES

The messages produced by the PL/I (F) Compiler are explained in the publication IBM System/360 Operating System, PL/I (F) Programmer's Guide, Form C28-6594. The following table associates a message number with the particular phase and module in which the corresponding message is generated.

<u>Message Number</u>	<u>Logical Phase</u>	<u>Module</u>			
IEM0001I	Read In	CA	IEM0059I	Read In	CP
IEM0002I	Read In	CA	IEM0060I	Read In	CP
IEM0003I	Read In	CA,CP	IEM0061I	Read In	CP
IEM0004I	Read In	CA	IEM0063I	Read In	CO
IEM0005I	Read In	CA,CL	IEM0064I	Read In	CC
IEM0006I	Read In	CA	IEM0066I	Read In	CG
IEM0007I	Read In	CA	IEM0067I	Read In	CL
IEM0008I	Read In	CA	IEM0069I	Read In	CG
IEM0009I	Read In	CA	IEM0070I	Read In	CG
IEM0010I	Read In	CA	IEM0071I	Read In	CG
IEM0011I	Read In	CA	IEM0072I	Read In	CG
IEM0012I	Read In	CA	IEM0074I	Read In	CG
IEM0013I	Read In	CA	IEM0075I	Read In	CG
IEM0014I	Read In	CA	IEM0076I	Read In	CG
IEM0015I	Read In	CA	IEM0077I	Read In	CG
IEM0016I	Read In	CA	IEM0078I	Read In	CG
IEM0017I	Read In	CA	IEM0080I	Read In	CG
IEM0018I	Read In	CA	IEM0081I	Read In	CG
IEM0019I	Read In	CA	IEM0082I	Read In	CG
IEM0020I	Read In	CA	IEM0083I	Read In	CG
IEM0021I	Read In	CA	IEM0084I	Read In	CG
IEM0022I	Read In	CA	IEM0085I	Read In	CI
IEM0023I	Read In	CA	IEM0090I	Read In	CI
IEM0024I	Read In	CA	IEM0094I	Read In	CI
IEM0025I	Read In	CA	IEM0095I	Read In	CI
IEM0026I	Read In	CA	IEM0096I	Read In	CG,CI
IEM0027I	Read In	CA	IEM0097I	Read In	CI
IEM0028I	Read In	CG	IEM0099I	Read In	CI
IEM0029I	Read In	CA	IEM0100I	Read In	CI
IEM0031I	Read In	CA,CL,CT	IEM0101I	Read In	CM
IEM0032I	Read In	CC	IEM0102I	Read In	CI
IEM0033I	Read In	CC	IEM0103I	Read In	CI
IEM0035I	Read In	CC	IEM0104I	Read In	CC
IEM0037I	Read In	CC	IEM0105I	Read In	CC,CG
IEM0038I	Read In	CC	IEM0106I	Read In	CI,CV
IEM0039I	Read In	CC	IEM0107I	Read In	CI
IEM0040I	Read In	CC	IEM0108I	Read In	CI
IEM0043I	Read In	CC	IEM0109I	Read In	CG,CI
IEM0044I	Read In	CC	IEM0110I	Read In	CI
IEM0045I	Read In	CC	IEM0111I	Read In	CI
IEM0046I	Read In	CC	IEM0112I	Read In	CI
IEM0048I	Read In	CG	IEM0113I	Read In	CG,CM
IEM0050I	Read In	CL,CP	IEM0114I	Read In	CI
IEM0051I	Read In	CL,CP	IEM0115I	Read In	CL
IEM0052I	Read In	CO	IEM0116I	Read In	CI
IEM0053I	Read In	CO	IEM0118I	Read In	CL
IEM0054I	Read In	CO	IEM0128I	Read In	CO
IEM0055I	Read In	CP	IEM0129I	Read In	CL
IEM0057I	Read In	CC	IEM0130I	Read In	CL
IEM0058I	Read In	CC	IEM0131I	Read In	CO
			IEM0132I	Read In	CO
			IEM0133I	Read In	CO
			IEM0134I	Read In	CP
			IEM0135I	Read In	CP
			IEM0136I	Read In	CO
			IEM0138I	Read In	CP
			IEM0139I	Read In	CP
			IEM0140I	Read In	CO
			IEM0141I	Read In	CP
			IEM0142I	Read In	CO
			IEM0143I	Read In	CO
			IEM0144I	Read In	CO

IEM0145I	Read In	CO	IEM0514I	Dictionary	EG
IEM0146I	Read In	CO	IEM0515I	Dictionary	EG
IEM0147I	Read In	CO	IEM0516I	Dictionary	EG
IEM0149I	Read In	CL, CM	IEM0517I	Dictionary	EG
IEM0150I	Read In	CL	IEM0518I	Dictionary	EG
IEM0151I	Read In	CO	IEM0519I	Dictionary	EG
IEM0152I	Read In	CO	IEM0520I	Dictionary	EG
IEM0153I	Read In	CO	IEM0521I	Dictionary	EG
IEM0154I	Read In	CA	IEM0522I	Dictionary	EG
IEM0158I	Read In	CO	IEM0523I	Dictionary	EG
IEM0159I	Read In	CO	IEM0524I	Dictionary	EH
IEM0163I	Read In	CT	IEM0525I	Dictionary	FI
IEM0164I	Read In	CS, CT	IEM0527I	Dictionary	EJ
IEM0166I	Read In	CL	IEM0528I	Dictionary	EH, EI, EJ
IEM0172I	Read In	CL	IEM0529I	Dictionary	EI
IEM0180I	Read In	CT	IEM0530I	Dictionary	EI
IEM0181I	Read In	CL	IEM0531I	Dictionary	EI
IEM0182I	Read In	CL, CS, CT, CV	IEM0532I	Dictionary	EI
		CT	IEM0533I	Dictionary	EI
IEM0185I	Read In	CT	IEM0534I	Dictionary	EI
IEM0187I	Read In	CT	IEM0535I	Dictionary	EI
IEM0191I	Read In	CT	IEM0536I	Dictionary	EI
IEM0193I	Read In	CT	IEM0537I	Dictionary	EI
IEM0194I	Read In	CT	IEM0538I	Dictionary	EJ
IEM0195I	Read In	CT	IEM0539I	Dictionary	EJ
IEM0198I	Read In	CT	IEM0540I	Dictionary	EJ
IEM0202I	Read In	CL	IEM0541I	Dictionary	EJ
IEM0207I	Read In	CG	IEM0542I	Dictionary	EJ
IEM0208I	Read In	CG	IEM0543I	Dictionary	EL, EK, EM
IEM0209I	Read In	CC	IEM0544I	Dictionary	EL, EK, EM
IEM0211I	Read In	CL	IEM0545I	Dictionary	EL, EK, EM
IEM0212I	Read In	CP	IEM0546I	Dictionary	EL, EK, EM
IEM0213I	Read In	CP	IEM0547I	Dictionary	EL, EK, EM
IEM0214I	Read In	CP	IEM0548I	Dictionary	EL, EK, EM
IEM0216I	Read In	CP	IEM0549I	Dictionary	EL, EK, EM
IEM0217I	Read In	CP	IEM0550I	Dictionary	EL, EK, EM
IEM0220I	Read In	CT	IEM0551I	Dictionary	EK, EL, EM
IEM0221I	Read In	CT	IEM0552I	Dictionary	EL, EK, EM
IEM0222I	Read In	CT	IEM0553I	Dictionary	EL, EK, EM
IEM0223I	Read In	CT	IEM0554I	Dictionary	EL, EK, EM
IEM0224I	Read In	CT	IEM0555I	Dictionary	EL, EK, EM
IEM0225I	Read In	CT	IEM0556I	Dictionary	EL, EK, EM
IEM0226I	Read In	CT	IEM0557I	Dictionary	EL, EK, EM
IEM0227I	Read In	CT	IEM0558I	Dictionary	EL, EK, EM
IEM0228I	Read In	CT	IEM0559I	Dictionary	EL, EK, EM
IEM0229I	Read In	CT	IEM0560I	Dictionary	EL, EK, EM
IEM0230I	Read In	CS, CT	IEM0561I	Dictionary	EL, EK, EM
IEM0231I	Read In	CT	IEM0562I	Dictionary	EK, EL, EM
IEM0232I	Read In	CT	IEM0563I	Dictionary	EK, EL, EM
IEM0233I	Read In	CV	IEM0564I	Dictionary	EK, EL, EM
IEM0235I	Read In	CS	IEM0565I	Dictionary	EK, EL, EM
IEM0236I	Read In	CS	IEM0566I	Dictionary	EK, EL, EM
IEM0237I	Read In	CS	IEM0567I	Dictionary	EP
IEM0238I	Read In	CV	IEM0568I	Dictionary	EP
IEM0239I	Read In	CS	IEM0569I	Dictionary	EP
IEM0240I	Read In	CV	IEM0570I	Dictionary	EP
IEM0241I	Read In	CV	IEM0571I	Dictionary	EK
IEM0242I	Read In	CV	IEM0572I	Dictionary	EL
IEM0243I	Read In	CV	IEM0573I	Dictionary	EL
IEM0244I	Read In	CV	IEM0589I	Dictionary	EW
IEM0245I	Read In	CV	IEM0590I	Dictionary	EW
IEM0247I	Read In	CW	IEM0591I	Dictionary	EW
IEM0254I	Read In	CC	IEM0592I	Dictionary	EW
IEM0255I	Read In	CG	IEM0593I	Dictionary	EW
IEM0510I	Dictionary	EH	IEM0594I	Dictionary	EW
IEM0511I	Dictionary	EH	IEM0595I	Dictionary	EW
IEM0512I	Dictionary	EH	IEM0596I	Dictionary	EW
IEM0513I	Dictionary	EG	IEM0597I	Dictionary	EW

IEM0598I	Dictionary	EW	IEM0702I	Dictionary	FI
IEM0599I	Dictionary	EW	IEM0703I	Dictionary	FI
IEM0602I	Dictionary	FV, FW	IEM0704I	Dictionary	FI
IEM0603I	Dictionary	FV, FW	IEM0705I	Dictionary	FI
IEM0604I	Dictionary	FV, FW	IEM0706I	Dictionary	FI
IEM0605I	Dictionary	FV, FW	UEM0707I	Dictionary	FI
IEM0606I	Dictionary	FV, FW	IEM0708I	Dictionary	FK
IEM0607I	Dictionary	FV, FW	IEM0709I	Dictionary	FK
IEM0608I	Dictionary	FV, FW	IEM0710I	Dictionary	FK
IEM0609I	Dictionary	FV, FW	IEM0711I	Dictionary	FK
IEM0610I	Dictionary	FV, FW	IEM0712I	Dictionary	FK
IEM0611I	Dictionary	FV, FW	IEM3713I	Dictionary	FK
IEM0612I	Dictionary	FV	IEM0715I	Dictionary	EJ
IEM0623I	Dictionary	FV, FW	IEM0718I	Dictionary	FO
IEM0624I	Dictionary	FV, FW	IEM0719I	Dictionary	FO
IEM0625I	Dictionary	FV, FW	IEM0720I	Dictionary	FO
IEM0626I	Dictionary	FV, FW	IEM0721I	Dictionary	FO
IEM0627I	Dictionary	FV, FW	IEM0722I	Dictionary	FO
IEM0628I	Dictionary	FV, FW	IEM0723I	Dictionary	FO
IEM0629I	Dictionary	FV, FW	IEM0724I	Dictionary	FO
IEM0630I	Dictionary	FV, FW	IEM0725I	Dictionary	FO
IEM0631I	Dictionary	FV, FW	IEM0726I	Dictionary	FO
IEM0632I	Dictionary	FV, FW	IEM0727I	Dictionary	FO
IEM0633I	Dictionary	EY	IEM0728I	Dictionary	FO
IEM0634I	Dictionary	EY	IEM0729I	Dictionary	FO
IEM0636I	Dictionary	EY	IEM0730I	Dictionary	FQ
IEM0637I	Dictionary	EY	IEM0731I	Dictionary	FQ
IEM0638I	Dictionary	EY	IEM0732I	Dictionary	FQ
IEM0640I	Dictionary	EY	IEM0733I	Dictionary	FQ
IEM0641I	Dictionary	EY	IEM0734I	Dictionary	FQ
IEM0642I	Dictionary	EY	IEM0735I	Dictionary	FQ
IEM0643I	Dictionary	EY	IEM0736I	Dictionary	FQ
IEM0644I	Dictionary	EY	IEM0737I	Dictionary	FQ
IEM0645I	Dictionary	EY	IEM0739I	Dictionary	FQ
IEM0646I	Dictionary	EY	IEM0740I	Dictionary	FQ
IEM0647I	Dictionary	EY	IEM0741I	Dictionary	FQ
IEM0652I	Dictionary	FE	IEM0742I	Dictionary	FQ
IEM0653I	Dictionary	FE	IEM0745I	Dictionary	FQ
IEM0654I	Dictionary	FE	IEM0746I	Dictionary	FQ
IEM0655I	Dictionary	FE	IEM0747I	Dictionary	FQ
IEM0656I	Dictionary	FE	IEM0748I	Dictionary	FQ
IEM0657I	Dictionary	FE	IEM0749I	Dictionary	FQ
IEM0658I	Dictionary	FE	IEM0750I	Dictionary	FQ
IEM0673I	Dictionary	FE	IEM0751I	Dictionary	FQ
IEM0674I	Dictionary	FF	IEM0752I	Dictionary	FQ
IEM0675I	Dictionary	FF	IEM0754I	Dictionary	FQ
IEM0676I	Dictionary	FF	IEM0755I	Dictionary	FQ
IEM0677I	Dictionary	FE	IEM0756I	Dictionary	FQ
IEM0682I	Dictionary	FI	IEM0758I	Dictionary	FQ
IEM0683I	Dictionary	FI	IEM0759I	Dictionary	FQ
IEM0684I	Dictionary	FI	IEM0760I	Dictionary	FQ
IEM0685I	Dictionary	FI	IEM0761I	Dictionary	FQ
IEM0686I	Dictionary	FI	IEM0762I	Dictionary	FQ
IEM0687I	Dictionary	FI	IEM0769I	Pretranslator	GB
IEM0688I	Dictionary	FI	IEM0770I	Pretranslator	GB
IEM0689I	Dictionary	FI	IEM0771I	Pretranslator	GB
IEM0690I	Dictionary	FI	IEM0778I	Pretranslator	GB
IEM0691I	Dictionary	FI	IEM0779I	Pretranslator	GB
IEM0692I	Dictionary	FI	IEM0780I	Pretranslator	GB
IEM0693I	Dictionary	FI	IEM0781I	Pretranslator	GB
IEM0694I	Dictionary	FI	IEM0782I	Pretranslator	GB
IEM0695I	Dictionary	FI	IEM0786I	Pretranslator	GK
IEM0696I	Dictionary	FI	IEM0787I	Pretranslator	GK
IEM0697I	Dictionary	FI	IEM0790I	Pretranslator	GK
IEM0698I	Dictionary	FI	IEM0791I	Pretranslator	GK
IEM0699I	Dictionary	FI	IEM0792I	Pretranslator	GP, GQ, GR
IEM0700I	Dictionary	FI	IEM0793I	Pretranslator	GP, GQ, GR
IEM0701I	Dictionary	FI	IEM0794I	Pretranslator	GP, GQ, GR

IEM0795I	Pretranslator	GP, GQ, GR	IEM1027I	Translator	IA
IEM0796I	Pretranslator	GP, GQ, GR	IEM1028I	Translator	IA
IEM0797I	Pretranslator	GP, GQ, GR	IEM1029I	Translator	IA
IEM0798I	Pretranslator	GP, GQ, GR	IEM1040I	Translator	IM
IEM0799I	Pretranslator	GP, GQ, GR	IEM1051I	Translator	IM
IEM0800I	Pretranslator	GP, GQ, GR	IEM1056I	Translator	IM
IEM0801I	Pretranslator	GP, GQ, GR	IEM1057I	Translator	IM
IEM0802I	Pretranslator	GP, GQ, GR	IEM1058I	Translator	IM
IEM0803I	Pretranslator	GP, GQ, GR	IEM1059I	Translator	IM
IEM0804I	Pretranslator	GP, GQ, GR	IEM1060I	Translator	IM
IEM0805I	Pretranslator	GP, GQ, GR	IEM1061I	Translator	IM
IEM0806I	Pretranslator	GP, GQ, GR	IEM1062I	Translator	IM
IEM0807I	Pretranslator	GP, GQ, GR	IEM1063I	Translator	IM
IEM0816I	Pretranslator	GU, GV	IEM1064I	Translator	IM
IEM0817I	Pretranslator	GU, GV	IEM1065I	Translator	IM
IEM0818I	Pretranslator	GU, GV	IEM1066I	Translator	IM
IEM0819I	Pretranslator	GU, GV	IEM1067I	Translator	IM
IEM0820I	Pretranslator	GU, GV	IEM1068I	Translator	IM
IEM0821I	Pretranslator	GU, GV	IEM1071I	Translator	IM
IEM0823I	Pretranslator	GU, GV	IEM1072I	Translator	IM
IEM0824I	Pretranslator	GU	IEM1073I	Translator	IM
IEM0825I	Pretranslator	GU, GV	IEM1074I	Translator	IM
IEM0826I	Pretranslator	GU, GV	IEM1076I	Translator	JD
IEM0832I	Pretranslator	HF, HG	IEM1082I	Translator	IX
IEM0833I	Pretranslator	HF, HG	IEM1088I	Aggregates	JK
IEM0834I	Pretranslator	HF, HG	IEM1089I	Aggregates	JK
IEM0835I	Pretranslator	HF, HG	IEM1090I	Aggregates	JK
IEM0836I	Pretranslator	HF, HG	IEM1091I	Aggregate Preprocessor	JI
IEM0837I	Pretranslator	HF, HG	IEM1092I	Aggregates	JK
IEM0838I	Pretranslator	HF	IEM1104I	Aggregates	JP
IEM0848I	Pretranslator	HF, HG	IEM1105I	Aggregates	JP
IEM0849I	Pretranslator	HF, HG	IEM1106I	Aggregates	JP
IEM0850I	Pretranslator	HF, HG	IEM1107I	Aggregates	JP
IEM0851I	Pretranslator	HF, HG	IEM1108I	Aggregates	JP
IEM0852I	Pretranslator	HF, HG	IEM1110I	Aggregates	JP
IEM0853I	Pretranslator	HF, HG	IEM1111I	Aggregates	JP
IEM0864I	Pretranslator	HK, HL	IEM1112I	Aggregates	JP
IEM0865I	Pretranslator	HK, HL	IEM1113I	Aggregates	JP
IEM0866I	Pretranslator	HK, HL	IEM1114I	Aggregates	JP
IEM0867I	Pretranslator	HK, HL	IEM1115I	Aggregates	JP
IEM0868I	Pretranslator	HK, HL	IEM1120I	Aggregates	JP
IEM0869I	Pretranslator	HK, HL	IEM1121I	Aggregates	JP
IEM0870I	Pretranslator	HK, HL	IEM1122I	Aggregates	JP
IEM0871I	Pretranslator	HK, HL	IEM1123I	Pseudo-code	LD
IEM0872I	Pretranslator	HK, HL	IEM1125I	Pseudo-code	LD
IEM0873I	Pretranslator	HK, HL	IEM1200I	Pseudo-code	LA
IEM0874I	Pretranslator	HK, HL	IEM1569I	Pseudo-code	LG-ON
IEM0875I	Pretranslator	HK, HL	IEM1570I	Pseudo-code	LG
IEM0876I	Pretranslator	HK, HL	IEM1571I	Pseudo-code	LG
IEM0877I	Pretranslator	HK, HL	IEM1572I	Pseudo-code	LG
IEM0878I	Pretranslator	HK, HL	IEM1574I	Pseudo-code	LG
IEM0879I	Pretranslator	HK, HL	IEM1575I	Pseudo-code	LG
IEM0880I	Pretranslator	HK, HL	IEM1588I	Pseudo-code	MD
IEM0881I	Pretranslator	HK, HL	IEM1600I	Pseudo-code	LS, LT, LU
IEM0882I	Pretranslator	HK	IEM1601I	Pseudo-code	LS
IEM0896I	Pretranslator	HP	IEM1602I	Pseudo-code	LS, LT, LU
IEM0897I	Pretranslator	HP	IEM1603I	Pseudo-code	LS, LT, LU
IEM0898I	Pretranslator	HP	IEM1604I	Pseudo-code	LS, LT, LU
IEM0899I	Pretranslator	HP	IEM1605I	Pseudo-code	LS, LT, LU
IEM0900I	Pretranslator	HP	IEM1606I	Pseudo-code	LS, LT, LU
IEM0901I	Pretranslator	HP	IEM1607I	Pseudo-code	LS, LT, LU
IEM0902I	Pretranslator	HP	IEM1608I	Pseudo-code	LS, LT, LU
IEM0903I	Pretranslator	HP	IEM1609I	Pseudo-code	LS, LT, LU
IEM0906I	Pretranslator	HP	IEM1610I	Pseudo-code	LW
IEM0907I	Pretranslator	HP	IEM1611I	Pseudo-code	LW
IEM1024I	Translator	IA	IEM1612I	Pseudo-code	LW
IEM1025I	Translator	IA	IEM1613I	Pseudo-code	LS, LT, LU
IEM1026I	Translator	IA	IEM1614I	Pseudo-code	LW



IEM1615I	Pseudo-code	ME	IEM1807I	Pseudo-code	OS
IEM1616I	Pseudo-code	MF	IEM1808I	Pseudo-code	OS
IEM1617I	Pseudo-code	MB	IEM1809I	Pseudo-code	OS
IEM1618I	Pseudo-code	MB	IEM1810I	Pseudo-code	OS
IEM1619I	Pseudo-code	MB	IEM1811I	Pseudo-code	OS
IEM1620I	Pseudo-code	MB	IEM1812I	Pseudo-code	OS
IEM1621I	Pseudo-code	MB	IEM1813I	Pseudo-code	OS
IEM1622I	Pseudo-code	MB, ME	IEM1814I	Pseudo-code	OS
IEM1623I	Pseudo-code	MB	IEM1815I	Pseudo-code	OS
IEM1624I	Pseudo-code	MB	IEM1816I	Pseudo-code	NJ
IEM1625I	Pseudo-code	MB	IEM1817I	Pseudo-code	NJ
IEM1626I	Pseudo-code	ME	IEM1818I	Pseudo-code	NJ
IEM1627I	Pseudo-code	ME	IEM1819I	Pseudo-code	NJ
IEM1628I	Pseudo-code	ME	IEM1820I	Pseudo-code	NJ
IEM1629I	Pseudo-code	ME	IEM1821I	Pseudo-code	NJ
IEM1630I	Pseudo-code	MG, MH	IEM1822I	Pseudo-code	NJ
IEM1631I	Pseudo-code	MI, MJ	IEM1823I	Pseudo-code	NJ
IEM1632I	Pseudo-code	MI, MJ	IEM1824I	Pseudo-code	NM
IEM1633I	Pseudo-code	ME	IEM1825I	Pseudo-code	NG
IEM1634I	Pseudo-code	ME	IEM1826I	Pseudo-code	NG
IEM1635I	Pseudo-code	ME	IEM1827I	Pseudo-code	NG
IEM1636I	Pseudo-code	ME	IEM1828I	Pseudo-code	NG
IEM1637I	Pseudo-code	ME	IEM1829I	Pseudo-code	NG
IEM1638I	Pseudo-code	ME	IEM1830I	Pseudo-code	NG
IEM1639I	Pseudo-code	MF	IEM1832I	Pseudo-code	NM
IEM1640I	Pseudo-code	MM, MN	IEM1833I	Pseudo-code	NM
IEM1641I	Pseudo-code	MM, MN	IEM1834I	Pseudo-code	NM
IEM1642I	Pseudo-code	MM, MN	IEM1835I	Pseudo-code	NM
IEM1643I	Pseudo-code	MM, MN	IEM1836I	Pseudo-code	NM
IEM1644I	Pseudo-code	MM, MN	IEM1837I	Pseudo-code	NM
IEM1645I	Pseudo-code	MM, MN	IEM1838I	Pseudo-code	NM
IEM1648I	Pseudo-code	MM, MN	IEM1839I	Pseudo-code	NM
IEM1649I	Pseudo-code	MM, MN	IEM1840I	Pseudo-code	NM
IEM1650I	Pseudo-code	MM, MN	IEM1841I	Pseudo-code	NM
IEM1651I	Pseudo-code	MM, MN	IEM1843I	Pseudo-code	NM
IEM1652I	Pseudo-code	MM, MN	IEM1844I	Pseudo-code	NM
IEM1653I	Pseudo-code	MM, MN	IEM1845I	Pseudo-code	NM
IEM1654I	Pseudo-code	MM, MN	IEM1846I	Pseudo-code	NM
IEM1655I	Pseudo-code	MN	IEM1847I	Pseudo-code	NM
IEM1656I	Pseudo-code	ME	IEM1848I	Pseudo-code	NM
IEM1657I	Pseudo-code	MM	IEM1849I	Constant Conversions	OS
IEM1670I	Pseudo-code	MP	IEM1850I	Constant Conversions	OS
IEM1671I	Pseudo-code	MP	IEM1860I	Pseudo-code	NU
IEM1680I	Pseudo-code	MS	IEM1861I	Pseudo-code	NU
IEM1687I	Pseudo-code	MS	IEM1862I	Pseudo-code	NU
IEM1688I	Pseudo-code	MS	IEM1870I	Pseudo-code	NU
IEM1689I	Pseudo-code	MS	IEM1871I	Pseudo-code	NU
IEM1691I	Pseudo-code	MS	IEM1872I	Pseudo-code	NU
IEM1692I	Pseudo-code	MS	IEM1873I	Pseudo-code	NU
IEM1693I	Pseudo-code	MS	IEM1874I	Pseudo-code	NU
IEM1750I	Pseudo-code	MS	IEM1875I	Pseudo-code	NV
IEM1751I	Pseudo-code	MS	IEM2304I	Storage Allocation	PD
IEM1752I	Pseudo-code	NA	IEM2305I	Storage Allocation	PD
IEM1753I	Pseudo-code	NA	IEM2352I	Storage Allocation	PD
IEM1754I	Pseudo-code	NA	IEM2700I	Register Allocation	RF, RG, RH
IEM1790I	Pseudo-code	OG, OM	IEM2701I	Register Allocation	RF, RG, RH
IEM1793I	Pseudo-code	OE	IEM2702I	Register Allocation	RF, RG, RH
IEM1794I	Pseudo-code	OE	IEM2703I	Register Allocation	RF, RG, RH
IEM1795I	Pseudo-code	OE	IEM2704I	Register Allocation	RF, RG, RH
IEM1796I	Pseudo-code	OE	IEM2705I	Register Allocation	RF, RG, RH
IEM1797I	Pseudo-code	OE	IEM2706I	Register Allocation	RF, RG, RH
IEM1800I	Pseudo-code	OS	IEM2707I	Register Allocation	RF, RG, RH
IEM1801I	Pseudo-code	OS	IEM2708I	Register Allocation	RF, RG, RH
IEM1802I	Pseudo-code	OS	IEM2709I	Register Allocation	RF, RG, RH
IEM1803I	Pseudo-code	OS	IEM2710I	Register Allocation	RF, RG, RH
IEM1804I	Pseudo-code	OS	IEM2711I	Register Allocation	RF, RG, RH
IEM1805I	Pseudo-code	OS	IEM2712I	Register Allocation	RF, RG, RH
IEM1806I	Pseudo-code	OS	IEM2817I	DCB Generation	GA

IEM2818I	DCB Generation	GA	IEM3848I	Compiler Control	AA
IEM2819I	DCB Generation	GA	IEM3849I	Compiler Control	AA
IEM2820I	DCB Generation	GA	IEM3850I	Compiler Control	AA
IEM2821I	DCB Generation	GA	IEM3851I	Compiler Control	AA
IEM2822I	DCB Generation	GA	IEM3852I	Compiler Control	AA
IEM2823I	DCB Generation	GA	IEM3853I	Compiler Control	AA
IEM2824I	DCB Generation	GA	IEM3855I	Compiler Control	AA
IEM2825I	DCB Generation	GA	IEM3856I	Compiler Control	AA
IEM2826I	DCB Generation	GA	IEM3857I	Compiler Control	AA
IEM2827I	DCB Generation	GA	IEM3858I	Compiler Control	AA
IEM2828I	DCB Generation	GA	IEM3859I	Compiler Control	AA
IEM2829I	DCB Generation	GA	IEM3860I	Compiler Control	AA
IEM2833I	Final Assembly	TF	IEM3861I	Compiler Control	AA
IEM2834I	Final Assembly	TF	IEM3862I	Compiler Control	AA
IEM2835I	Final Assembly	TF	IEM3863I	Compiler Control	AA
IEM2836I	Final Assembly	TF	IEM3864I	Compiler Control	AA
IEM2837I	Final Assembly	TF	IEM3865I	Compiler Control	AA
IEM2852I	Final Assembly	TJ	IEM3872I	Compiler Control	AA
IEM2853I	Final Assembly	TJ	IEM3873I	Compiler Control	AA
IEM2854I	Final Assembly	TJ	IEM3874I	Compiler Control	AA
IEM2855I	Final Assembly	TJ	IEM3875I	Compiler Control	AA
IEM2865I	Final Assembly	TO	IEM3876I	Compiler Control	AA
IEM2866I	Final Assembly	TO	IEM3877I	Compiler Control	AA
IEM2867I	Final Assembly	TO	IEM3878I	Compiler Control	AA
IEM2868I	Final Assembly	TO	IEM3880I	Compiler Control	AA
IEM2881I	Final Assembly	TT	IEM3887I	Compiler Control	AA
IEM2882I	Final Assembly	TT	IEM3888I	Compiler Control	AB
IEM2883I	Final Assembly	TT	IEM3889I	Compiler Control	AB
IEM2884I	Final Assembly	TT	IEM3890I	Compiler Control	AA
IEM2885I	Final Assembly	TT	IEM3891I	Compiler Control	AA
IEM2886I	Final Assembly	TT	IEM3892I	Compiler Control	AA
IEM2887I	Final Assembly	TT	IEM3893I	Compiler Control	AA
IEM2888I	Final Assembly	TT	IEM3894I	Compiler Control	AA
IEM2897I	Final Assembly	UA	IEM3895I	Compiler Control	AA
IEM2898I	Final Assembly	UA	IEM3896I	Compiler Control	AA
IEM2899I	Final Assembly	UC	IEM3897I	Compiler Control	AA
IEM2900I	Final Assembly	UC	IEM3898I	Compiler Control	AA
IEM2913I	Final Assembly	UF	IEM3899I	Compiler Control	AL
IEM3088I	Dictionary, Declare Pass 2	EL	IEM3900I	Compiler Control	AB
IEM3136I-	Dictionary, Declare	EL	IEM3901I	Compiler Control	AB
3149I	Pass 2		IEM3902I	Compiler Control	AB
IEM3151I	Dictionary, Declare	EL	IEM3902I	Compiler Control	AB
	Pass 2		IEM3903I	Compiler Control	AB
IEM3153I	Dictionary, Declare	EL	IEM3904I	Compiler Control	AA
	Pass 2		IEM3905I	Compiler Control	AA
IEM3154I	Dictionary, Declare	EL	IEM3906I	Compiler Control	AA
	Pass 2		IEM3907I	Compiler Control	AA
IEM3156I	Dictionary, Declare	EL	IEM3908I	Compiler Control	AA
	Pass 2		IEM3909I	Compiler Control	AL
IEM3162I	Dictionary, Declare	EL	IEM3910I	Compiler Control	AB
	Pass 2		IEM3911I	Compiler Control	AB
IEM3167I-	Dictionary, Declare	EL	IEM3912I	Compiler Control	AB
3173I	Pass 2		IEM3913I	Compile-time Processor	AB
IEM3176I-	Dictionary, Declare	EL	IEM3914I	Compile-time Processor	AB
3190I	Pass 2		IEM4106I	Compile-time Processor	AS
IEM3199I-	Dictionary, Declare	EL	IEM4109I	Compile-time Processor	AS
3213I	Pass 2		IEM4112I	Compile-time Processor	AS
IEM3584I	48 Character Preprocessor	BX	IEM4115I	Compile-time Processor	AS
			IEM4118I	Compile-time Processor	AS
IEM3840I	Compiler Control	AA	IEM4121I	Compile-time Processor	AS, BC, BG
IEM3841I	Compiler Control	AA	IEM4124I	Compile-time Processor	BC, BG
IEM3842I	Compiler Control	AA	IEM4130I	Compile-time Processor	BG
IEM3843I	Compiler Control	AA	IEM4133I	Compile-time Processor	BC
IEM3844I	Compiler Control	AA	IEM4134I	Compile-time Processor	BC
IEM3845I	Compiler Control	AA	IEM4136I	Compile-time Processor	BC
IEM3846I	Compiler Control	AA	IEM4139I	Compile-time Processor	BC
IEM3847I	Compiler Control	AA	IEM4142I	Compile-time Processor	BC
			IEM4143I	Compile-time Processor	BC



IEM4570I Compile-time Processor BG  
IEM4572I Compile-time Processor BG  
IEM4574I Compile-time Processor BG  
IEM4576I Compile-time Processor BG  
IEM4578I Compile-time Processor BG  
IEM4580I Compile-time Processor BG