Converting FIG-Forth Programs to Forth-83#

Table of Contents

Forth-83 is not completely upward compatible with any of the other commonly available Forth dialects: FIG-Forth, polyFORTH, or Forth-79. Some of the differences can be quite subtle, and programs written in these dialects will need careful inspection and editing before they will execute properly on top of a Forth- 83 Standard nucleus.

We provide below a checklist of things to look for, which is not guaranteed to be inclusive. It is, however, based on our experience gained during the conversion of several thousand screens of FIG-Forth source code to Forth-83 during 1983-1984. For further details on any of the Forth words listed below, refer to the 83-Standard document and the Glossary section of your LMI Forth System User Manual. In case of a conflict between the two, the 83-Standard document description may be assumed to be correct.

General Considerations#

1. Due to the runtime requirements of the redefined LOOP, +LOOP, and LEAVE words, the return stack is usually maintained in a different format in Forth-83 than in other dialects. In LMI Forth systems, the DO...LOOP or DO...+LOOP construct requires 3 control words on the return stack. Programs which rely on specialized knowledge of the return stack will require extensive changes.

2. All FIG-Forth "state smart" words such as ' (tick) and ." (dot-quote), which previously had different actions depending on whether they were invoked inside or outside a colon definition, have either been eliminated or redefined in Forth-83.

3. The FIG-Forth words CFA, PFA, LFA, and NFA, which were used to find the addresses of different fields within a dictionary header, have been eliminated. A new set of words, adopted from the Kim Harris's experimental proposal, has been included in the LMI Forth-83 systems: BODY>, >BODY, NAME>, >NAME, LINK>, >LINK, N>LINK, and L>NAME. See detailed explanations below. At present, the only word of this set which is part of the 83-Standard is >BODY.

4. For various reasons the definition of all divide functions general effect is that quotients are floored instead of rounded toward zero. This should cause no problems for most pre-existing application software. The new divide functions are marginally slower than the old (a few machine cycles under most circumstances). The side-effects of the redefinition for floored divide can be counter-intuitive under some circumstances. For example, in FIG-Forth the operation

```
-40 360 MOD
```

would return the obvious answer (-40) on the stack, while 83- Standard Forth will return the answer 320!

5. The true flag returned by all logical operations has been changed from the value 1 (in FIG-Forth) to the value -1 (in Forth-83, all bits set). If your code used the 0 or 1 returned by a comparison in an arithmetic operation, you will need to interpolate the operator ABS after the logical operator. This is a particularly difficult problem to look for in your source code. However, we feel that this mutation in the 83-Standard was beneficial as it allows the returned true/false value to be used as a mask for AND.

6. PICK and ROLL are now zero-based, instead of one-based. The reasoning behind this change by the Forth-83 Standards committee was based mainly on the rather weak argument that programmers frequently use zero as the beginning index for loops.

7. The word LEAVE has become an "Immediate" compiler word with state-checking in LMI Forth-83 systems, to satisfy the 83- Standard's requirements. "LEAVE" in turn compiles the run-time word "leave". If your FIG-Forth code included compiler extensions referencing LEAVE, it may need modification.

In addition, the run-time action of LEAVE is immediate, that is, it causes a direct transfer of control past the end of the currently active innermost loop. Consequently, the new LEAVE is unstructured, so that a loop increment may accidentally be passed outside the loop construct under certain circumstances. This problem is easier to demonstrate than explain; observe the following (admittedly contrived) example:

```
: TEST 1000 0 DO 2 ?TERMINAL IF LEAVE THEN +LOOP ;
```

If ?TERMINAL returns a true flag, after exit from the DO...+LOOP construct 2 will remain on the stack in a 83-Standard system whereas it would have been discarded in FIG-Forth.

8. The definition of DO...LOOP has been altered in a manner that makes it more efficient to implement on most processors. Loops now behave as though the INDEX transits an unsigned "number circle" to reach the LIMIT, and terminate when INDEX crosses the boundary between LIMIT-1 and LIMIT. There are side effects of this definition, however, that cause loops to behave differently description of LOOP and +LOOP below.

Restrictions on a 83-Standard Forth Application Program#

Forth systems, whether complying with any Standard or not, typically contain many environmentally dependent words in addition to the Standard required word set. In order for your application programs to be portable to any 83-Standard Forth system, you should observe the following rules (abridged from the 83-Standard document):

1. A Standard application may reference only the definitions of the 83-Standard Required Word Set and Standard Extensions, and definitions which are subsequently defined in terms of those words.

2. A Standard Program may operate only on data which was stored by the application. The initial contents of variables and arrays created at compilation time are explicitly undefined.

3. A Standard Program may address:

- parameter fields of words created with CREATE, VARIABLE, and user defined words which execute CREATE,
- dictionary space ALLOTted,
- data in a valid mass storage buffer,
- data area of user variables,
- Text Input Buffer and PAD up to amount specified as the minimum for each area.

4. A Standard Program may not address:

- directly into the data or return stacks,
- into a definition's name, link, or code fields,
- into a definition's parameter field if its contents were not stored by the application.

Although the capability of doing these three types of operations is present in LMI Forth-83 systems (SP@, RP@, >NAME, etc.), they should be avoided if you intend to port your program to 83-Standard systems provided by other vendors.

Checklist for Program Conversion from FIG-Forth to Forth-83#

getting your FIG-Forth programs to the point where they will compile in Forth-83 (proper execution, of course, is another problem). Most of the changes indicated were related to adoption of the 83-Standard, other renamings are not properly part of the standard but have been adopted in the Laboratory Microsystems implementations.

1. Search for all instances of R, replace with R@.
2. Search for all instances of -DUP, replace with ?DUP.
3. Search for all instances of WORD. When it occurs as WORD HERE, delete the word HERE. When WORD is not followed by HERE, replace it with WORD DROP.
4. Search for all instances of PICK and ROLL, replace them by 1- PICK and 1- ROLL respectively.
5. Examine all DO...LOOPs. Any loop which might be entered with the limit equal to the index should have DO replaced with ?DO.
6. Search for all instances of LEAVE. Note that the action of LEAVE will be immediate. If it occurs within a IF...ELSE...THEN clause, LEAVE should be the last word before the ELSE or THEN. If LEAVE is used within a DO...+LOOP construct, make sure that the incrementing value will not remain on the stack if LEAVE is executed.
7. Search for all instances of ' (tick) within a colon-definition. If it was not preceded by [COMPILE], replace it with ['].
8. Search for all instances of ." (dot-quote) outside of colon definitions, replace the ." with .(and the closing delimiter " with) to prevent compilation failures.
9. Search for all instances of NFA, PFA, LFA, and CFA. It is best to examine these and recode them individually, but you can make some "brute force" substitutions as follows:

```
'          becomes   ' >BODY  (outside colon definition)
' CFA      becomes   '        (outside colon definition)
' CFA      becomes   [']      (inside colon definition)
NFA        becomes   BODY> >NAME
' NFA      becomes   ' >NAME
LFA        becomes   BODY> >LINK
' LFA      becomes   ' >LINK
PFA        becomes   NAME> >BODY
PFA CFA    becomes   NAME>
```

Bear in mind that the old definitions had the following actions:

```
CFA        ( pfa --- cfa )
NFA        ( pfa --- nfa )
PFA        ( nfa --- pfa )
```

These were predicated on the fact that ' (tick) returned the parameter field address. Since ' and [] now return the code field address, the new words suggested by Kim Harris revolve around that value:

```
>BODY ( cfa --- pfa )
>LINK     ( cfa --- lfa )
>NAME     ( cfa --- nfa )
BODY>     ( pfa --- cfa )
LINK>     ( lfa --- cfa )
NAME>     ( nfa --- cfa )
```

These are appealing and symmetric, but before you get too carried away with them remember that a Standard program can't access any part of a dictionary definition except for the parameter field ("Body"). Two additional words are provided for convenience in traversing the linked dictionary list:

```
N>LINK    ( nfa --- lfa )
L>NAME    ( lfa --- nfa )
```

1. Search for all instances of ENDIF and replace with THEN.
2. Search for all instances of MINUS or DMINUS and replace with NEGATE or DNEGATE respectively.
3. Search for all instances of SIGN and fix up stack logic. Usually the "old" SIGN can be replaced by ROT SIGN.
4. Any use of -FIND will have to be individually recoded. It can usually be replaced by the sequence BL WORD FIND.
5. Search for all instances of ? and replace with @ . .
6. Search for all instances of BLANKS, replace with BLANK.
7. Uses of the FIG-Forth CREATE word in your programs will have to be individually inspected and recoded. The 83-Standard word CREATE has a much different effect.
8. Search for all instances of the construct <BUILDS...DOES> and replace with CREATE...DOES> .
9. Search for all instances of END and replace with UNTIL.
10. Search for all instances of (NUMBER) and replace with CONVERT.
11. Search for all instances of IN and replace with >IN.
12. Search for all instances of FLUSH and replace with SAVE-BUFFERS.
13. Search for all instances of U* and replace with UM*; similarly, replace all occurrences of U/ with UM/MOD.
14. Replace all instances of S->D with DUP 0< (83-Standard) or S>D (LMI Forth-83 systems).
15. Neither SP! or RP! are 83-Standard. They are present in LMI Forth-83 systems but with a different meaning than in FIG-Forth. The FIG-Forth SP! should be replaced with the sequence S0 @ SP!, and the word RP! should be changed to R0 @ RP!.
16. Search for all instances of TIB @ and replace by TIB. Any sequences TIB ! will have to be recoded in an impelmentation-dependent manner.
17. Difficult to find by inspection but very dangerous: use of the boolean flag returned by a comparison in a calculation, such as the sequence 0= ADD . These must be individually examined and recoded.
18. Also difficult to find: use of specialized knowledge of the return stack (such as fetching the index of the third outer loop). These will have to be individually examined and recoded.
19. Search for all VARIABLE declarations and delete leading initializing value (these do no harm, but will be left as residual data on the stack at the end of compilation). Although most implementations do set the initial value of a variable to zero or -1, a Standard application program is of course not allowed to take advantage of this information. It is most correct to initialize variables at run-time (so that the code is reusable), however your old FIG-Forth compile-time initialization of variables can easily be mimicked. For example, the FIG-Forth statement

```
4 VARIABLE XVAR
```

would be changed to

```
VARIABLE XVAR  4 XVAR !
```

1. Search for all instances of +- , replace with 0< IF NEGATE THEN (Forth-83 Standard) or with the word ?NEGATE (LMI Forth-83 systems). Similarly, replace D+- with 0< IF DNEGATE THEN (Forth-83 Standard) or with the word ?DNEGATE (LMI Forth-83 systems).
2. Examine occurrences of MOD . If either argument can take on a negative value, the results may surprise you.
3. Find all M/MOD and replace with MU/MOD (LMI Forth-83 implementations only, may be different with other vendors). Search for all instances of M/ and replace with M/MOD (83-Standard).

Alphabetical Summary of Vocabulary Changes in Forth-83#

```
#TIB            A new 83-Standard user variable containing the
                length in bytes of the valid input stream
                within the terminal input buffer. This is NOT
                the length of the buffer itself. #TIB is set
                by QUERY after calling EXPECT.

'               "Tick" word is no longer immediate, returns
                the CFA of the target name instead of the PFA
                as in our previous version. Outside a colon
                definition, your previous code
                  ' CFA
                should be replaced by
                  '
                while inside a colon definition it should be
                replaced by
                  [']
                See also detailed comments above.

(FIND)          FIG-Forth word that has been deleted. To
                perform the function of the FIG-Forth (FIND)
                use the 83-Standard word FIND.

+LOOP           Termination has been redefined to occur when
                the INDEX crosses the boundary between LIMIT-1
                and LIMIT. This +LOOP is faster but
                behaves somewhat differently, for example:
                  1 1 DO ... 1 +LOOP
                will execute 65,536 times (in FIG-Forth or
                Forth-79, it would have executed only once),
                while
                  1 1 DO ... -1 +LOOP
                will execute once. See also LOOP .

+-              Not in 83-Standard, but can be replaced by the
                sequence 0< IF NEGATE THEN . Present as
                ?NEGATE in LMI Forth-83 systems.

+ORIGIN         Deleted.

-DUP            Renamed to ?DUP .

-FIND           Essentially replaced by the 83-Standard word
                sequence BL WORD FIND .
```

."	"Dot-Quote". Now may be used inside of colon-definitions only. See also .(.
	up to but not including the delimiting) are displayed on the standard output device (usually the operator's console) . May be used inside or outside of a colon definition.
<BUILDS	Replaced by CREATE in Forth-83, which has some slightly different effects.
<CMOVE	Renamed to CMOVE> in 83-Standard FORTH.
>BODY	Converts code field address to parameter field address.
>LINK	Converts code field address to link field address. Not 83-Standard, but present in LMI Forth-83 systems.
>NAME	Converts code field address to name field address. Not 83-Standard, but present in LMI Forth-83 systems.
?	Deleted in Forth-83, can replace with "@ ." sequence.
?DNEGATE	New name for D+- in LMI Forth-83 systems.
?DO	Works like DO except executes zero times if the input INDEX and LIMIT are equal. Not 83-Standard, but present in all LMI Forth-83 systems.
?DUP	Same as old -DUP .
ABORT"	A new word which requires a flag on top of stack, and does nothing if the flag is false or prints a string and executes ABORT if the flag is true. Used like FIG-Forth ?ERROR but with no requirement for a disk access.
AGAIN	Not present in the 83-Standard or even in the controlled reference word set. Probably fated for extinction, so its use should be avoided in new code. Can be replaced by 0 UNTIL .
BLANK	Renamed from BLANKS. See below.
BLANKS	Renamed to BLANK (not an 83-Standard word, but included in the Controlled Reference word set).
BODY>	Converts parameter field address to code field address. Works like the old word CFA . Not 83-Standard.
CMOVE>	Previously known as <CMOVE .

CONVERT	Forth-83 word that essentially works like the old FIG-Forth word (NUMBER) . Caution: converts positive double numbers only. Sign handling must be done outside.
CREATE	Redefined from FIG-Forth. Now used with DOES> in the same manner as the old <BUILDS .
D+-	Not in 83-Standard, but can be replaced by the sequence 0< IF DNEGATE THEN. Present as ?DNEGATE in some systems.
DIGIT	This is not an 83-Standard word, but present in all known systems.
DMINUS	FIG-Forth word renamed to DNEGATE in 83-Standard systems.
DNEGATE	83-Standard word, previously called DMINUS in FIG-Forth.
DOES>	Functionally similar to DOES> in FIG-Forth, but implementation is considerably different and involves a mixture of machine code and high level code in the defining word's parameter field.
EMPTY-BUFFERS	Mark all disk block buffers as un-assigned. Does NOT write blocks marked for UPDATE to the disk. Not an 83-Standard word, but still present in LMI Forth systems. See also SAVE-BUFFERS and FLUSH .
END	Removed. Use UNTIL .
ENDIF	Removed. Use THEN .
EXPECT	Works about the same as in FIG-Forth, but leaves the actual length of the input in the system variable SPAN .
FIND	New word which essentially provides the capabilities of the old -FIND and "state-smart tick".
FLUSH	Writes all UPDATEd blocks to disk, then un-assigns all block buffers. See also EMPTY-BUFFERS, SAVE-BUFFERS .
ID.	Not in 83-Standard. Present as .NAME in some systems..
IN	Renamed to >IN .
LEAVE	In Forth-83 causes an immediate transfer of control to the code just beyond the next LOOP word. For example, in the code: DO IF XXX ELSE LEAVE YYY THEN LOOP if the ELSE path is taken, the word YYY will

	never be executed (unlike FIG-Forth or Forth-79).
L>NAME	Converts link field address to name field address. Not 83-Standard.
LINK>	Converts link field address to code field address. Not 83-Standard.
LITERAL	Compilation only. The actual runtime word compiled may depend on the magnitude of the literal value.
LOAD	Loading from screen 0 is now defined to be illegal.
LOOP	Termination has been redefined to occur when the INDEX crosses the boundary between LIMIT-1 and LIMIT. This LOOP is faster than in the previous version but behaves somewhat differently, for example: 1 1 DO ... LOOP will execute 65,536 times (in FIG-Forth or Forth-79, it would have executed only once).
M*	No longer present in 83-Standard or Controlled Reference Word set, but still present in LMI Forth-83 systems.
M/	Renamed to M/MOD in Forth-83. This is different than the M/MOD that was present in FIG-Forth.
MINUS	FIG-Forth word renamed to NEGATE in 83-Standard systems.
MOVE	FIG-Forth word no longer supported, avoid it.
N>LINK	Converts name field address to link field address. Not 83-Standard, but present in LMI Forth-83 systems.
NAME>	Converts name field address to code field address. Not 83-Standard, but present in LMI Forth-83 systems.
NEGATE	83-Standard word, was previously named MINUS .
NFA	Removed. See >NAME .
NOT	Returns 1's complement.
PAD	Now provides a work area of at least 84 bytes.
PICK	The argument to PICK is now zero-based (was previously one-based). Examples: 0 PICK is equivalent to DUP , 1 PICK is equivalent to OVER .
R@	Forth-83 new name for FIG-Forth R .

R	FIG-Forth word renamed to R@ in 83-Standard.
ROLL	The argument to ROLL is now zero-based instead of one-based. Examples: 0 ROLL is a null operation, 1 ROLL is equivalent to SWAP , 2 ROLL is equivalent to ROT , etc.
RP!	Not 83-Standard, but present in most systems. If available it may take its argument from the top of stack instead of from R0.
S->D	FIG-Forth word not present in 83-Standard, but equivalent word S>D present in LMI Forth-83 systems.
SAVE-BUFFERS	All buffers marked as UPDATEd are written to the disk, but remain assigned. See also EMPTY-BUFFERS and FLUSH .
SIGN	Definition changed, takes its argument from the top of stack rather then the third item on the stack. The word SIGN in your old code can be replace by ROT SIGN .
SP!	Not 83-Standard, but present in most systems. If available, may take its argument from the top of stack, rather than S0 .
SPAN	A new system variable which contains the length of the last input via EXPECT .
STATE	System variable that is zero if the system is interpreting and non-zero if compiling. A 83-Standard application shouldn't modify this variable.
TIB	Definition changed. Formerly returned the address of the location containing the address of the terminal input buffer, now returns the actual address of the buffer. In your existing code, replace all sequences TIB @ by the word in an implementation dependent manner.
U*	FIG-Forth word renamed to UM* in Forth-83.
U/	FIG-Forth word renamed to UM/MOD in Forth-83.
UM*	New Forth-83 name for U* .
UM/MOD	New Forth-83 name for U/ .
VARIABLE	Does not accept an initializing value in Forth-83.
VLIST	Renamed to WORDS LMI Forth-83 systems.
VOCABULARY	The 83-Standard, in an attempt to remedy the 79-Standard's restrictiveness on Vocabulary

	structures, left most details to the imagination of the Forth implementor. A radically different experimental proposal was offered by Bill Ragsdale and printed as an appendix to the Standards document, but was not approved as part of the Standard proper and has not been adopted by all Forth vendors at this point. Beware!
WIDTH	No longer present.
WORD	Definition slightly changed from FIG-Forth. Returns the address of the first byte of the blank delimited token it scanned off the input stream. A token may be up to 255 bytes in length.
WORDS	Displays the names of the definitions in the vocabulary which is first in the current search order. Similar to FIG-Forth VLIST command.
[']	Used inside a colon definition to compile the CFA of the following word, like the old "state-smart" tick word. See also ' and FIND .

Further reading:#

- FORTH-83 Standard, a publication of the Forth Standards Team.
- "When a Page is not a Page: Forth-83 and Vocabularies". Goerge W. Shaw II, Dr. Dobb's Journal, September 1983, page 90.
- "Upgrading Forth-79 Programs to Forth-83", by Robert Berkey, Forth Dimensions, Volume VI, Number 3, September/October 1984, page 26.
- "Forth-83 Program to Run Forth-79 Code", by Robert Berkey, Forth Dimensions, Volume VI, Number 4, November/December 1984, page 28.
- "FORTH-83, Evolution Continues", by C.Kevin McCabe, BYTE Magazine, August 1984, page 137.