DEFT Pascal Workbench

User's Guide

TRS-80™ Color Computer Software Series

Version 8 Second Printing

Software License

DEFT Systems, Inc. grants to you, the customer, a non-exclusive, paid-up license to use the **DEFT Systems** software on **one** computer, subject to the following provisions:

1. Except as otherwise provided in the Software License, applicable copyright laws shall apply to the Software.
2. Title to the medium on which the Software is recorded (cassette and/or diskette) or stored (ROM) is transferred to you, but not title to the Software.
3. You may use the Software on one host computer and access that Software through one or more terminals if the Software permits this function.
4. You shall not use, make, manufacture, or reproduce copies of Software except for use on **one** computer and as is specifically provided in the Software License. You are expressly prohibited from disassembling the Software.
5. You are permitted to make additional copies of the Software **only** for backup or archival purposes or if additional copies are required in the operation **of one** computer with the Software, but only to the extent the Software allows a backup copy to be made.
6. You may resell or distribute unmodified copies of the Software provided you have purchased one copy of the Software for each one sold or distributed. The provisions of this Software License shall also be applicable to third parties receiving copies of the Software from you.
7. All copyright notices shall be retained on all copies of the Software.

Term

This License is effective until terminated. You may terminate this License at any time by destroying the Software together with all copies in any form. It will also terminate if you fail to comply with any term or condition of the License.

Warranty

These programs, their instruction manual and reference materials are sold AS IS, without warranty as to their performance, merchantability, or fitness for any particular purpose. The entire risk as to the results and performance of these programs is assumed by you.

However, to the original purchaser only, **DEFT Systems, Inc.** warrants the magnetic diskette on which these programs are recorded to be free from defects in materials and faulty workmanship under normal use for a period of thirty days from the date of purchase. If during this thirty day period the diskette should become defective, it may be returned to **DEFT Systems, Inc.** for a replacement without charge, provided you have previously sent in your limited warranty registration notice to **DEFT Systems, Inc.** or send proof of purchase of these programs.

Your sole and exclusive remedy in the event of a defect is expressly limited to replacement of the diskette as provided above. If failure of a diskette has resulted from accident or abuse **DEFT Systems, Inc.** shall have no responsibility to replace the diskette under the terms of this limited warranty.

Any implied warranties relating to the diskette, including any implied warranties of merchantability and fitness for a particular purpose, are limited to a period of thirty days from the date of purchase. **DEFT Systems, Inc.** shall not be liable for indirect, special, or consequential damages resulting from the use of this product. Some states do not allow the exclusion or limitation of incidental or consequential damages, so the above limitations might not apply to you. This warranty gives you specific legal rights, and you may also have other rights which vary from state to state.

Support

DEFT Systems, Inc. (and not Radio Shack) is completely responsible for the Warranty and all maintenance and support of the Software. Any questions concerning the Software should be directed to:

DEFT Systems, Inc.
P.O. Box 359
Damascus, Md. 20872

DEFT Pascal Workbench User's Guide

DEFT Pascal Workbench

1 DEFT Pascal Workbench

DEFT Pascal Workbench is a set of software development tools designed to support a programmer through the process of creating computer programs; from entering source code through executing the resulting machine program. **DEFT Pascal Workbench** is comprised of the following software packages:

DEFT Pascal
DEFT Edit
DEFT Macro/6809
DEFT Linker
DEFT Debugger
DEFT Lib

DEFT Pascal Workbench requires a TRS-80 Color Computer to be configured with at least 32K of memory, Extended Disk BASIC, and one floppy disk drive. **DEFT Pascal Workbench** utilizes a device independent file structure which is fully compatible with Disk Extended BASIC. Disk and tape files created with **DEFT Pascal Workbench** are of the same internal format as those produced and supported by BASIC.

1.1 DEFT Pascal

The **DEFT Pascal** Compiler is a fully recursive, single-pass Pascal language compiler for the TRS-80 Color Computer. It compiles Pascal programs directly into machine language code that can be executed by the 6809 microprocessor in the CoCo.

DEFT Pascal generally supports most standard Pascal language constructs. In addition, **DEFT Pascal** supports many extensions to the standard language which makes text processing, multi-language and systems type programs easier to write.

1.2 DEFT Edit

DEFT Edit is a screen mode, in-memory, text editor which provides its users with a selectively moveable *window* into a text file. **DEFT Edit** was designed primarily for the development of program source code, but it can also be used for the production of software documentation.

1.3 DEFT Macro/6809

DEFT Macro/6809 is a device-independent software package designed to translate Motorola 6809 Assembler source programs into 6809 micro-processor machine programs in two passes. Program source files may be read from either cassette or disk with the resulting machine program object files written to either cassette, disk, or the serial I/O port. **DEFT Macro/6809** parses and evaluates Motorola 6809 Assembler language statements and declarations, and generates the corresponding 6809 micro-processor machine programs according to Motorola 6809 Assembler language syntactical rules and conventions.

1.4 DEFT Linker

DEFT Linker is a program which reads the program object files produced by both **DEFT Pascal** and **DEFT Macro/6809** and converts them into machine executable binary image files suitable for loading with the Color Computer's LOADM command. **DEFT Linker** can also read multiple program object files and combine them into one larger machine executable binary *Load Module* so as to allow Color Computer users to develop very large programs one piece at a time.

1.5 DEFT Debugger

DEFT Debugger is an excellent tool for debugging machine programs developed in either Pascal or Assembler. **DEFT Debugger** allows you to stop and start a program under test at almost any point. Once the program under test has been stopped, you can display and/or change any memory location or micro-processor register.

When used with **DEFT Pascal**, **DEFT Debugger** provides symbolic access to your program as well as a trace facility for displaying currently active procedures.

1.6 DEFT Lib

DEFT Lib is an excellent tool for the development of object module libraries using object modules produced by either **DEFT Pascal** or **DEFT Macro/6809**. **DEFT Lib** is a device independent software package capable of creating and maintaining up to 50 object module sections in one library file. Once created, these libraries can be used

as input to **DEFT Linker** which will only use those sections which have been referenced by the particular program which is being linked.

2 DEFT Pascal Workbench Users Guide

The **DEFT Pascal Workbench** Users Guide is structured to be helpful in understanding and using **DEFT Pascal Workbench.** The Users Guide is not intended to be a self teaching guide in how to program but rather a tutorial on how to use the programs in **DEFT Pascal Workbench.**

If you already have an understanding of programming, then the User's Guide should contain more than enough information for you to immediately begin programming. If you have only programmed in BASIC, then you should be able to begin programming but you may need a Pascal text book when tackling some of the more advanced portions of the language. In either case, practice makes perfect, and no one should expect too much of themselves without some experience.

2.1 Document Divisions

The **DEFT Pascal Workbench** User's Guide is presented in three parts: *Introduction*, *How To* and *Background*. Each section was written with two specific objectives in mind.

- To support **DEFT Pascal Workbench** users according to their operation of a **DEFT** software product.
- To provide background information for reference.

2.2 Document Section Descriptions

The *Introduction* section informs the reader of two things. First, it describes the contents of the User's Guide itself and second, it describes how, in general terms, to use **DEFT Pascal Workbench** to develop programs.

The *How To* section describes in operational detail how to execute each tool provided in **DEFT Pascal Workbench.** This section starts with a **Familiarization Exercise** designed to be performed by you when you are first becoming acquainted with **DEFT Pascal.** This exercise provides a working example program. Following the exercise are individual sections which describe the operation and use of each program in the **DEFT Pascal Workbench.**

The *Background* section presents the reader with reference information. The first part summarizes the standard language elements of **DEFT Pascal** and includes a brief explanation of each. The second part summarizes the language extensions that are

contained in **DEFT Pascal**, with an explanation of each element. The last part summarizes the language elements of **DEFT Macro/6809** assembly language.

Regardless of how much experience a you may have, we highly recommend that you read the entire User's Guide. Good luck and have fun with **DEFT Pascal Workbench.**

3 Software Development

Developing programs with the **DEFT Pascal Workbench** is somewhat different from the procedure for developing programs in BASIC. With BASIC, you essentially type in the program and then type RUN. Debugging usually consists of hitting the BREAK key at appropriate points, PRINTing variables and turning the trace on and off.

This is a very good environment in which to develop small programs which do not have to execute with exceptional speed. However, as the programs you write become larger and more complex, some of the limitations imposed by the BASIC language will come in to play. These are primarily the small identifier size, lack of program structure, and execution performance of the interpreter.

DEFT Pascal Workbench takes up where BASIC leaves off. It should be seen as a powerful addition to your existing program tools. It is ideal for those programs which become very large, complex, and which execute for relatively long periods of time. All the programs in the **DEFT Pascal Workbench** were themselves developed using the workbench.

In general, the **DEFT Pascal Workbench** allows you to divide and conquer a large problem in smaller pieces. The linkage facilities found in **DEFT Pascal** and **DEFT Macro/6809** provide a very simple and straightforward method for combining the program pieces. This linkage facility is an extra step in the program development process and for small programs may not provide many benefits. However, in larger programs, the ability to modularize and compile or assemble only a small piece of a program at a time can be invaluable.

Since you are producing 6809 micro-processor instructions with **DEFT Pascal**, you will be dealing directly with the CPU when you begin debugging your resulting machine language program. You will use the **DEFT Debugger** to perform this step.

3.1 Program Design Development

Design. This step is one that you consciously or unconsciously perform before typing in a program. At the very least you should:

- Decide exactly what things the program is supposed to do. These are the program's functions.

- Decide how to organize the program around these major functions. This will identify what your major program pieces are.
- Decide how each piece should be organized to perform its function.

For *very* large programs, you may want to go to even more detailed design before beginning your coding. Remember that organizing the program is half the job of solving the problem. This usually involves defining all of the major data elements that you will be using before writing the code that manipulates them.

3.2 Source Code Development

Edit. This familiar step is the entry of a program's instructions which usually begins about halfway through the design stage. At this point, you will be creating *source module files;* that is, each program that is entered is stored in its textual form in a file. This step is performed by the programmer using a text editor such as **DEFT Edit**. The resulting text file containing the program statements is referred to as a *source file* or *source module file.*

This step is very similar to that in BASIC, except that in BASIC once the program is entered, it can then be immediately executed by the BASIC interpreter. With **DEFT Pascal**, the program statements in text form must first be translated into machine instructions for execution by the 6809 micro-processor. This leads us to the next phase of program development.

3.3 Object Code Development

Compile/Assemble. This is a new step for those used to BASIC. This step involves transforming the *source module files* that you created with **DEFT Edit** into *object module files* which contain two things:

- The *machine language* version of your programs
- *Linkage* information that will allow one *object module file* to be combined with others

DEFT Pascal and **DEFT Macro/6809** are both used to perform this step. Both programs prompt the user for both the name of the *source module file* which it uses for input and the *object module file* which it produces.

3.4 Load Module Development

Link. This is the last step before actually executing your program. This step converts the previously created *object module files* into single *binary load module files.*

When **DEFT Pascal** creates its object module files, it includes calls to machine language routines in other object modules which were included on your **DEFT Pascal** diskette. These object modules are in a special file called a runtime library and provide services such as I/O, string and set handling as well as floating point arithmetic. All of these object modules must be combined together and all of the address references between these modules must be adjusted appropriately in order to create a working program.

DEFT Linker performs this whole operation. It prompts you for the name(s) of the object module file(s) to be linked, which it uses for input, and the name of the *load module file* which it produces. This step takes all of those object module files and combines them into a single file that can be loaded via the BASIC LOADM command.

3.5 Program Execution and Debugging

Execute/Debug. This step involves actually testing your program by providing it with test data developed during the design step to determine if the program is producing the correct results. The **DEFT Debugger** permits a programmer to stop and restart a program under test at any point within the program. The programmer may then examine any memory location and/or microprocessor register and change its contents if desired. With the **DEFT Debugger**, the user may specify up to eight program stopping or *break* points at one time.

DEFT Debugger is an object module that is linked into your program's load module by **DEFT Linker** and therefore becomes a part of it. It initially gains control when your program begins execution so that you can use it to control subsequent execution. Once your program is debugged, you can re-link it without the debugger which will make your program smaller and faster.

For most large programs, the first and last steps, design and debugging, take the majority of the total time spent on a program. In fact, in very large projects the first and last steps are broken into a number of sub-steps in order to keep the job to a manageable size.

4 Getting Started

This section of the **DEFT Pascal Workbench** User's Guide is meant to provide you with the operational details required to use **DEFT** software products on the TRS-80 Color Computer. This section is required reading before you should attempt anything with a **DEFT** software product.

4.1 Program Execution

All **DEFT** programs for the TRS-80 Color Computer are binary machine language programs that are loaded into memory with the LOADM command and executed with the EXEC command. Before executing any **DEFT** program or any program that you create with the **DEFT Pascal Workbench**, it is absolutely necessary to protect it from BASIC. This is done with the following set of 4 BASIC Monitor commands. These commands need to be entered only once, just before the first time that you load a **DEFT** program. Subsequent loads of **DEFT** software will not require the re-entry of these BASIC Monitor commands.

1. *NEW* - This command is not necessary if you have just turned on your Color Computer. It is used to initialize the memory area normally used by the BASIC Interpreter in the Color Computer's ROM.

2. *PCLEAR 1* - This command causes Extended BASIC to reserve the minimum number of 1.5K byte pages for graphics. Since no **DEFT** software product uses BASIC's graphics for presentation, this command releases otherwise unused memory for use by the program being loaded.

3. *FILES 0,0* - This command tells BASIC that you do not intend to access any disk files via BASIC. Note that even after executing this command you can still DIR, KILL and RENAME. However, you will not be able to COPY. Since each program of the **DEFT Pascal Workbench** is an independent machine program, none of the BASIC Interpreter's file facilities are required, thereby freeing up even more otherwise unused memory.

4. *CLEAR 16,4999* - This reserves the upper 59K (27K in a 32K system) bytes of memory for use by **DEFT** software products. It will leave a little over 300 bytes of memory for use by BASIC. This Color Computer BASIC Monitor directive must be entered exactly as presented in this example. The first directive argument, *16*, tells the BASIC Monitor how many bytes of

memory to reserve for BASIC strings. Since no **DEFT** software products use the Color Computer's BASIC language, 16 bytes of memory is more than enough. The comma (,) preceding this next number is required. the next number, *4999*, tells the BASIC Monitor the last or highest value "address" in memory that it is allowed to use. This number is expressed in decimal, thereby reserving the rest of the Color Computer's memory, from decimal address 5000 on up, for any **DEFT** software product.

It is absolutely essential that you perform these commands before executing any of the programs in the **DEFT Pascal Workbench.** If you do not, BASIC may "over-write" portions of any program that you may load. If that were to happen, the loaded program's execution will produce unpredictable results.

The BASIC command for executing any of the programs in the **DEFT Pascal Workbench** is *LOADM "<filename>":EXEC* and the possible filenames are:

PASCAL	**DEFT Pascal**
EDITOR	**DEFT Edit**
ASSEMBLE	**DEFT Macro/6809**
LINKER	**DEFT Linker**
LIB	**DEFT Lib**

4.2 64K Operation

Whenever any **DEFT** program first begins execution, it immediately changes the Color Computer's memory map to unmap the BASIC ROM and map in any RAM that may exist in the top 32K of memory. **DEFT** programs are all fully self-contained and so don't need the BASIC ROM to operate.

After changing the memory map, the program will check to see whether you have a 32K or 64K system and then adjust the size of its main data structure to whatever memory is available. The result of this is that these programs can access up to 64K bytes of memory in your Color Computer.

With **DEFT Pascal**, or any other **DEFT** high level language compiler, any programs that you create will be able to use all the available memory in the system for your data variables. The only restriction is that the program instructions (not stack) must fit in the lower 32K of memory since this is loaded via BASIC.

4.3 32K Operation

Some 32K systems may show the same RAM memory size as a 64K system. This will cause all programs to switch to memory map 1 which will cause the system to hang. If you have such a TRS-80 Color Computer, you will want to do the following:

1. Power on your Color Computer.
2. Make a backup of your distribution diskette and put the distribution diskette in a safe place.
3. Put the *un-write-protected copy* of the distribution diskette that you just made into drive 0.
4. Enter the 4 BASIC commands found in the *Program Execution* section.
5. Enter: *RUN"MAKE32K"* <enter>

The program will run for about a minute and after it finishes, the diskette in drive 0 will contain a 32K version of the software.

If you have a 64K system and want to write Pascal programs that access the BASIC ROMs, you can rename *PASBOOT/OBJ* to *PASBOOT/64K* and *PASBOOT/32K* to *PASBOOT/OBJ*. By doing only this, your **DEFT** software will still run using all 64K but any program linked using this new version of *PASBOOT/OBJ* will operate with the BASIC ROMs in place.

4.4 DEFT Files

One of the advantages of using the **DEFT Pascal Workbench** is the device independent file structure which is supported while remaining fully compatible with the TRS-80 Disk Extended Color BASIC System Software. Disk or tape files created with BASIC, **DEFT** software products or programs developed with **DEFT Pascal** are all of the same fundamental format.

When executing **DEFT** software development tools you will have to specify the names of the *source module*, *object module* and *binary load module* files. The file naming conventions used with the **DEFT Pascal Workbench** are only slightly different from that of BASIC and allow complete device independence. The format of the names are as follows:

<filename>/<ext>:<device#>

This is the same format that BASIC uses for Disk files. However, by extending the device numbers, **DEFT Pascal Workbench** also uses it for the keyboard, screen, tape and printer. The <filename> is 0 to 8 ASCII characters. The extension is 0 to 3 ASCII characters. The device numbers range from -3 to 3 with the following meanings:

-3 Keyboard/Screen
-2 Printer
-1 Cassette Tape
0 Disk drive 0
1 Disk drive 1
2 Disk drive 2
3 Disk drive 3

As can be seen, the positive device numbers correspond to BASIC's drive numbers. The negative device numbers correspond to BASIC's device numbers with the exception that the Keyboard/Screen is -3 rather than 0.

All of the fields are optional in different circumstances. When a device number of -3 or -2 is specified, there is no need for a <filename> or <extension>. When a device number of -1 is specified, the <extension> is not required. For device numbers 0 thru 3, a default <extension> is always present depending on the program being run. When a device number is not specified, 0 is assumed. Following are some examples:

:-3	**Keyboard/Screen**
:-2	**Printer**
MYFILE:-2	**Printer (filename ignored but allowed)**
TAPEFILE:-1	**Cassette Tape File**
DISKFILE/ASM	**Assembler source file on disk drive 0**
F2NAME:1	**File is on disk drive 1, default extension used**

4.5 DEFT Pascal Workbench Diskette Contents

The following files are contained on the diskette that you received. You are encouraged to make a copy of the distribution diskette for your own backup purposes and to execute from the backup rather than the original diskette.

1. *PASCAL/BIN* - This file contains the executable image of the **DEFT Pascal** Compiler.

2. *EDITOR/BIN* - This file contains the executable image of **DEFT Edit.**

3. *LINKER/BIN* - This file contains the executable image of the **DEFT Linker.**

4. *ASSEMBLE/BIN* - This file contains the executable image of **DEFT Macro/6809.**

5. *LIB/BIN* - This file contains the executable image of **DEFT Lib.**

6. *PASCALIB/EXT* - This is a Pascal source file which is automatically *copied* by **DEFT Pascal** at the beginning of all programs which it compiles. This file contains the declarations of all of the predefined procedures and functions provided with **DEFT Pascal.** This file must be present on disk drive 0 whenever **DEFT Pascal** is executed.

7. *PASBOOT/OBJ* - This is the object file for the standard *boot* code for all Pascal programs. All programs produced by **DEFT** Software have a *first instruction.* For **DEFT Pascal** programs these first instructions are kept in this file. This object module file contains the machine language routines for Pascal program initialization. This file must be present on disk drive 0 when linking a Pascal program with the **DEFT Linker.**

8. *RUNTIME/LIB* - This is the object module library file which contains all the Pascal Runtime routines for Pascal programs developed with **DEFT Pascal.** Each library section contains machine language routines which are automatically called by **DEFT Pascal** when you use various parts of the language. This file must be present on disk drive 0 when linking a Pascal program with **DEFT Linker.**

9. *DEBUGGER/LIB* - This is the library file which contains **DEFT Debugger** for debugging any program created with **DEFT Pascal Workbench.** This file must be present on disk drive 0 when linking any program which is to include **DEFT Debugger.** See **DEFT Debugger** for more information.

10. *FORMAT/PAS & FORMAT2/PAS* - These are the two source files which contain the *Text Formatter* **DEFT Pascal** program. You will use these source files in the *Familiarization Exercise* part of the HOW TO section, to create your own text processing system.

11. *FORMATSP/ASM* - This is a source file which contains the 6809 Macro Assembler language portion of the *Text Formatter* program.

12. *FORMATSP/OBJ* - This is an object file produced by **DEFT Macro/6809** from the *FORMATSP/ASM* source file. It is included on the distribution diskette in case you do not wish to use the assembler.

13. *FORMAT/TXT* - This is an ASCII file that the FORMAT program uses for input. The FORMAT program will produce a set of instructions describing how to use itself.

14. *PASBOOT/ASM* - This is a source file which contains 6809 Macro Assembler language instructions which are the very first instructions executed by any Pascal program developed via the **DEFT Pascal**.

15. *MAKE32K/PAS* This is a BASIC program that converts a distribution diskette to 32K operation.

4.6 Single Disk Drive Operation

When using a single disk drive system you will have to create a *work diskette* that contains a couple of files from the distribution diskette as well as your own source, object and binary files. To execute a program you will insert the distribution diskette into your disk drive, load the proper binary image, insert your work diskette into the drive and then execute the loaded program.

The files that need to be copied onto your work diskette are:

DEBUGGER/LIB
PASCALIB/EXT
PASBOOT/OBJ
RUNTIME/LIB

You can copy these files by using the COPY command in BASIC. Although single drive operation is not documented, this command works the same way BACKUP does in single drive mode.

On some early versions of Disk Extended Basic the COPY command will not work on a single disk drive. If you have one of these, use BACKUP to create a work diskette and then KILL all the files on the diskette except those named above.