FORTRAN and the ART of PC PROGRAMMING

TIM WARD, EDDIE BROMHEAD

FORTRAN and the ART of PC PROGRAMMING

TIM WARD
Harrison-Ward Associates Ltd, UK

and

EDDIE BROMHEAD
Kingston Polytechnic, Kingston upon Thames, UK

JOHN WILEY & SONS
Chichester · New York · Brisbane · Toronto · Singapore

Wiley Editorial Offices

John Wiley & Sons Ltd, Baffins Lane, Chichester,
West Sussex, PO19 1UD, England

John Wiley & Sons, Inc., 605 Third Avenue,
New York, NY 10158–0012, USA

Jacaranda Wiley Ltd, G.P.O. Box 859, Brisbane,
Queensland 4001, Australia

John Wiley & Sons (Canada) Ltd, 22 Worcester Road,
Rexdale, Ontario M9W IL1, Canada

John Wiley & Sons (SEA) Pte Ltd, 37 Jalan Pemimpin #05–04,
Block B, Union Industrial Building, Singapore 2057

Library of Congress Cataloging-in-Publication Data:

Ward, Tim.
FORTRAN and the art of PC programming / Tim Ward and Eddie Bromhead.
p. cm.
Bibliography: p.
Includes index
ISBN 0 471 92253 6
1. FORTRAN (Computer program language) 2. Microcomputers–
Programming. I. Bromhead, Eddie. II. Title.
QA76.73.F25W37 1989 89-31593
005.13′3—dc20 CIP

British Library Cataloguing in Publication Data:

Ward, Tim
FORTRAN and the art of PC programming.
1. Computer systems. Programming languages : Fortran language
I. Title II. Bromhead, Eddie
005.2′6

ISBN 0 471 92253 6

Printed and bound in Great Britain by Courier International, Tiptree, Essex

Disk set with manual

Price £25/$40 (including VAT, postage and packing)

The programs described in this book, together with the two utility libraries and the graphics library described in Chapter 15, plus example programs and manual, are available in source code on a disk set for your IBM PC (and most compatibles). They are written in FORTRAN and assembler code. and the assembler routines will also be supplied as object code libraries. You will need a FORTRAN compiler (the authors used Microsoft FORTRAN); the assembler code libraries will have been assembled using Microsoft's MACRO Assembler.

The disk set and manual are *only* available from the authors at the following address:

Tim Ward
Harrison-Ward Associates Ltd
14 St Georges Road
Farnham
Surrey GU9 8NB
UK
(Tel: (0252) 714055)

Cheques should be made payable to Harrison-Ward Associates Ltd. Don't forget to include your own address!

CONTENTS

CHAPTER 1

WHY FORTRAN, WHY PCs?

1.1 WHY FORTRAN?

Traditionally, FORTRAN (FORmula TRANslation) has been *the* language for scientific and engineering software, and has been widely used in other fields. Other computer languages have had vogues, which have passed. These include languages of the same era as FORTRAN, such as ALGOL, which had a far richer syntax, and a better structure. FORTRAN has fended them all off. Even upstart languages of the 1970s and 1980s, such as Pascal and C, are still not used in the main for numeric applications. There are a number of reasons for this. Perhaps the most important is that FORTRAN was developed and supported by the giant computer company International Business Machines (George Backus, IBM, 1956), and so was available on all the most popular computers. Also FORTRAN's relatively primitive syntax, which translates easily and efficiently into the underlying machine language of the host computer, leads to high speed processing, an increasingly important component of any scientific language. Today, over 30 years after FORTRAN was developed, there is a huge investment in FORTRAN code and expertise. This, together with its slow but steady improvement, particularly in the areas in which the language has been traditionally weak—such as character handling—means that FORTRAN will be with us for some time to come. The persistence of FORTRAN has led to the implementation of the language on every mainframe and minicomputer of which we are aware, making FORTRAN source code extremely portable (although implementation specifics sometimes make this less easy than it should be).

Most major analysis systems in the technological arena have been written in FORTRAN because it is so portable and enduring, and because of its built-in mathematical and trigonometrical functionality. Typical of these are large finite element analysis systems. Many widely used systems started life in the mid to late 1960s. The majority of machines available in those days often lacked the memory or processing power of today's microcomputer, and only met the needs of small communities of users who were prepared to suffer a poor turn-round (i.e. slow and unresponsive) batch service, if for no better reason than because they had no choice.

Whether time sharing systems became available to match increases in computer power, or the reverse, FORTRAN users moved away from punched cards and line printer listings (usually in capitals only) to terminal based systems, with much less emphasis on hard copy. Tragically, the user was often faced with very little betterment in the resources made available to him, since multi-user systems require huge amounts of computer resource to be devoted to monitoring all the terminals for the odd keypress, and to scheduling the timeslice for each terminal, and comparatively little of the computer power was available immediately to each user.

1.2 WHY PCs?

The microcomputer, when it first appeared, left many FORTRAN users unimpressed. Often, the 64 kbytes maximum of the CP/M micro, which seemed astronomic in comparison to early 8 or 16 kbyte capacity machines, was still far too small to accommodate the data structures of the average FORTRAN program, and it is noteworthy that those few FORTRAN compilers which made headway in the 8-bit micro world concentrated on the larger, 128 kbyte bank switched machines.

The 16-bit micro changed all that forever. Here was a machine which had (potentially rather than actually at first) the capacity to run almost all applications. And it had processing power to match. Many users found, like the authors, that the combination of that amount of power, built-in graphics, and all for the price of a terminal, an irresistible temptation.

Early experiences were disappointing. The first version of any piece of software, however well tested, normally has a number of 'bugs', not even this knowledge prepared us for the unacceptable quality of the early FORTRAN compilers. This forced many FORTRAN programmers back to BASIC(s!) or to learn assembler or to abandon the microcomputer for ever. The graphics, the potential of which was well illustrated by demonstration software, was frustratingly intractable except to the most persistent computer hobbyist. The computer manuals were unintelligible to all but the person who wrote them (at least, we presume so, but with that degree of impenetrability?), and technical help was by and large non-existent. The only saving grace was that the software which had been developed on the 8-bit microcomputers quickly became available on the 16-bit machines.

Due to their immense potential, microcomputers were produced and bought in large quantities, and good but expensive software became available in increasing quantities. Today our early faith in microcomputers has been amply rewarded. There is almost too large a choice of top quality software some at ludicrously modest prices. Also, as it turned out, the microcomputers were, and still are, remarkably reliable. Even the FORTRAN compilers are now polished professional products with only a few idiosyncrasies reminding us of the earlier versions. Graphics is readily available through such products as GEM (Digital Research) and Windows (Microsoft) to both users and programmers alike, and the Microsoft disk operating system, with GEM or Windows, gives a powerful, and compared to most mainframe computers, accessible and user friendly interface.

Microprocessor power has increased in almost exponential proportions since those early days, with 32-bit processors in microcomputers becoming more affordable, making the distinction between microcomputers and minicomputers almost impossible to define. Part of the reason for this is that the prices of RAM (Random Access Memory) and mass storage devices (Winchester disks) have fallen dramatically. But the main reason is the popularity of the microcomputers themselves, which has allowed mass production with its resulting economies of scale, and at the same time has created a competitive market-place in which there is fierce price competition.

The most popular microcomputers are those which mimic the computers marketed by IBM. Copies abound, with the most notable being manufactured by Compaq, Dell, Tandon, and Amstrad. Although there are a number of Japanese manufacturers of microcomputer parts and whole computers, this is one area of the electronic market that they are yet to dominate. The reason is because the most popular microprocessors are still all made in America.

The only other microcomputers to have a sustained popularity are the Apple Macintosh range. These are built around a 32-bit chip from Motorola, and have an entirely different architecture. The strength of the Macintosh is in its user interface, and application tools for the programmer are not readily available. Although FORTRAN compilers are just beginning to appear on the Macintosh they are still in their infancy, and it is not yet a machine for the FORTRAN programmer.

The IBM microcomputers and compatibles, commonly known under the umbrella term PCs (Personal Computers), coined by IBM for their first microcomputer the IBM PC, use a Disk based Operating System called PC DOS or MS DOS. PC DOS is IBM's proprietary product developed under contract by Microsoft, which with IBM's contract has become one of the three largest software companies in the world. It is worth noting that the other two of these three companies specialize largely in software products for Microsoft's operation system. They are Ashton Tate (dBASE), and the Lotus Corporation (Lotus 1-2-3). MS DOS (Microsoft DOS) was also developed by Microsoft, and is as far as the user is concerned almost identical to PC DOS. Throughout the book we will refer to both these operating systems as DOS unless we wish to distinguish between the two.

As we shall see, DOS has one or two limitations, the most important of which is the restriction on the amount of memory that can be addressed, and the inability to run a number of processes at the same time. These limitations have become increasingly restrictive as the microprocessors have become more powerful. A new operating system was needed, and OS/2 (Operating System 2) is the result. However there are too many DOS users for it to be abandoned altogether, and OS/2 contains many of the familiar DOS commands. Software compatibility is also important, and the new system can emulate DOS. In addition, software is already being developed which will enable applications to be targeted to DOS or OS/2. The latest Microsoft FORTRAN compiler is available as an OS/2 product or a DOS product, with negligible differences in its use under either operating system.

At the time of writing, OS/2 is still in its infancy. This means that it will be some time before the operating system will be fully functional and boast as many applications as DOS—except for those which run under DOS too. Even then, there will still be a vast population of people running DOS on machines which cannot run OS/2. Version 4 of MS DOS has just been released, and so we do not believe that the lessons learnt under DOS will be wasted, rather, to make full use of new systems, additional information will be required, and new techniques will have to be learned.

1.3 WHY, THEN, THIS BOOK?

Purchasers of PCs bought with programming in mind are likely to run through a number of stages in their familiarization with the machine. These are:

- Period of enchantment
- Period of disenchantment
- Period of envy
- Period of utilization

The period of enchantment covers the time in which one unpacks the hardware, boots it up, runs a simple application, begins the process of transferring FORTRAN source (often it is simplest to retype a few small programs initially) to the PC, and then compiles and runs a simple application. Enchantment is still the name of the game when you find that all the FORTRAN code you ever wrote uses up negligible amounts of floppy disk space, and what you thought were quite large programs run easily. That tiny PC really is a useful computer!

The period of disenchantment begins when a fundamental limitation of your FORTRAN compiler comes to light. Or you cannot establish a communications link with the mainframe on which your code is stored. Or a floppy disk fault loses crucial files. Or you forgot that you used some system specific facility like a graphics library on the mainframe, and do not have the equivalent on the PC. Or quite simply you lack the range and quality of peripherals you had grown used to on the mainframe. Hopefully, this book will help you over some of the disenchantments.

The period of envy comes about when you have overcome many of the foregoing problems, and are using the PC in earnest. It is a simple phenomenon, brought on by using other software, not usually written in FORTRAN.

Many PCs offered some bundled software as a sales gimmick. Regardless of how useful it was, it inevitably had a polished appearance. On-line help, properly designed screen layouts, proper use of emboldened, underlined or inverse video text and so on. These had to be simple to program, as the resulting executable files were often small in length, especially compared to the smallest program (even one of the 'Hello World' variety) which could be generated from FORTRAN. It must be possible to program this sort of material, and the purpose of this book is to show how.

Envy is also experienced when reading the microcomputer press. Beware! The microcomputer press is full of tomorrow's products, even if they are available today, in many cases they will not work properly. Do not buy version 1 of *anything*! The astounding success enjoyed by the company Amstrad is founded on giving the customer a product that will be useful at an affordable price—often this is based on yesterday's technology (but manufactured by today's). There is very little innovation here.

The period of utilization comes with experience. Your FORTRAN programs compile and run, applications that you write have friendly user interfaces and exploit the graphical capabilities of your machine, you have mastered the printer and plotter, you fully exploit the power of your microcomputer and peripherals. There are no short cuts to this degree of experience, but with direction a lot of wasted time can be saved. We hope our book will show you the way, and be a companion in the hours of darkness when nothing seems to go right.

The reader of this book is probably a FORTRAN programmer of some years experience, embarking as we did, into unexplored areas with a new PC. He will know that his machine has the power and facilities to do almost everything he needs. He will, however, be frustrated at how unpolished and unprofessional his programs look in comparison to the user interfaces of inexpensive (often free) utilities. This book is an attempt to save him some of the slow learning process which we experienced. The rules are often simple, and with hindsight, glaringly obvious. Once one understands escape codes, for instance, much hitherto impenetrable computer jargon becomes instantly comprehensible (see Chapters 12 and 14). The contents of this book should make it possible for you too to do all (well, most) of these things in FORTRAN.

It was not our aim to convert the micro-wise to the merits of FORTRAN. The best computer language to use is always the one with which you are familiar.

Two things make the Eden of FORTRAN on a PC rather unidyllic. The first of these is the memory organization of the Intel chip family as used in simpler PCs. This poses a considerable challenge to the 'power user' who needs every facility he can lay his hands on, and which he has a right to expect in a megabyte address space. The second is the nonsensical 640 kbytes limit imposed by PC DOS, and hence for reasons of compatibility, on MS DOS. This has led to an industry in overcoming it. Some ideas appear in the book.

1.4 WHAT'S IN THE BOOK?

Each chapter in the book is designed to be self-contained, with a minimum of forward and backward references to other chapters.

Chapter 2 looks at 'Microcomputer Architecture' and the relatively simple 8086 Intel processor. A passing knowledge of how microprocessors compute will help us to understand more about the PC (whether it is based on the 8086 processor or not). It also introduces assembly language programming. FORTRAN programmers do not ordinarily want to write in assembler unless absolutely necessary, but it does give us one of the keys to unlocking the power of the PC. Assembly language routines are surprisingly simple to access from FORTRAN.

Chapter 3 is about the 'Disk Operating System'. All the power of the microcomputer is available through DOS if you know where to look.

In Chapter 4 we turn our attention to 'Programs, Files, and Filenames'. The chapter contains hints on editing and creating our FORTRAN programs, and the names we should give to them; we also consider the files that are created by our FORTRAN programs, and how and where programs and files are stored.

In the early days of microcomputing, FORTRAN compilers tended to be pretty crude, and the environment encountered by the FORTRAN programmer could be very different to the one he was used to on the mainframe computer. Current compilers are much more professionally presented, and less error prone. In Chapter 5 we look at 'Compiling, Linking, and Debugging' on the microcomputer.

A widely held misconception is that microcomputers store numbers to a lower precision than mini or mainframe computers and hence will not give accurate answers. In Chapter 6, 'Precision, Accuracy, and the Right Answer', we put the record straight.

Chapter 7 is about 'Readable and Maintainable Code'. Productivity is greatly enhanced if programs are written so that errors can be quickly spotted.

What about code already written on a mainframe, how can it be transferred to a PC? If code was developed on a PC how can it be sent to other systems? Having moved code or changed compilers what problems will be faced? And what about those old codes written in FORTRAN 66 (FORTRAN IV)? Will FORTRAN 8x have anything new to offer? These and other questions are posed and answered in Chapter 8, 'Code Transfer, and Portable Code'.

In Chapter 9, 'Invoking FORTRAN Programs and Other DOS Interfaces', the execution of the FORTRAN program is discussed, together with how to make use of the operating system commands from inside a FORTRAN program.

One of the differences between most microcomputers and mini and mainframe computers is that they usually have less core memory (RAM). This need not be a problem for all but the largest programs if you get into the habit of 'Using Memory Effectively', the subject of Chapter 10.

In Chapter 11 consideration is given to the problems associated with 'Coping with a Slow Computer' and aspects of code optimization.

One of the joys of FORTRAN programming on the microcomputer comes with the mastery of writing to the screen. In Chapter 12, 'Screen Handling: Text Modes', the ways that text can be manipulated on the screen are discussed; and in Chapter 13, 'Screen Handling: Graphics Modes', the principles involved in graphical output to the screen are considered.

The next problem is to obtain hard copy output of text and graphics. This is a common area of confusion and disappointment: extremely powerful printers can turn out to be next to useless, if your software does not drive them. Chapter 14 is full of tips for the FORTRAN programmer, and if you are yet to buy a printer or plotter, you should read this chapter, 'Printing and Plotting—The Hard Copy Jungle', first.

Storing tried and tested FORTRAN utilities in object code libraries is a must for all serious FORTRAN programmers. Libraries of FORTRAN utilities are also marketed commercially. In Chapter 15, 'FORTRAN Libraries and Utilities', we look at the use of libraries. This chapter also details the libraries of DOS, screen handling and graphics utilities that you will find on the accompanying disk set.

Finally, Chapter 16, 'Choosing a FORTRAN Compiler', takes a brief look at some of the compiler products available to the FORTRAN programmer on the microcomputer.

To accompany this book there is a disk set. It contains all the complete programs that appear in the text, plus screen driver software using ASCII codes, KERMIT the communications product (see Chapter 8), and any other useful assembler or FORTRAN subroutines and libraries that we think might be of use and lie in the public domain.

1.6 WHAT'S *NOT* IN THE BOOK?

This is not a book about FORTRAN programming. There are many good texts on the market, and to add another just for the sake of it would not be a productive use of precious time. In the bibliography, we list a number of such books, and we suggest that, as a serious FORTRAN programmer, you should have more than one of them on your bookshelf.

Nor is this a book about MS DOS, either from the user's or programmer's point of view. There are many books on both of those aspects. Again, we think you will want to have specialist books on these subjects.

Instead, this book is about the in-between area which it is so difficult to find out about. The jargon, for one thing, may well be comprehensible to trained programmers: much of it will certainly be alien to the FORTRAN user. Not only that, but what information there is tends to be hidden away in the most unlikely corners.

1.7 WHAT YOU NEED

If you are yet to buy microcomputer equipment then this book contains a host of information which will be invaluable when you come to make your choice of PC and peripherals, and we hope it will save you a lot of money too.

To get the most out of this book and its contents, you will need access to a personal computer running MS DOS or PC DOS, a text editor capable of producing text files in 'non-document' or 'system' format, any one of the FORTRAN compilers listed in Chapter 16, and an assembler. We have successfully used WordStar, Microsoft FORTRAN, and the Microsoft Macro Assembler on a variety of machines. We have based our discussion of screen graphics around the graphics primitives and drivers provided in GEM, any GEM application contains the necessary environment and the subroutines in the text or on the accompanying disk will allow you to access these facilities (Digital Research's previous product GSX can also be accessed in the same way). A number of other graphics primitives products with bindings for most FORTRAN compilers exist, and the range of facilities provided does not vary a great deal from product to product. So the general points on graphics will still apply if you have one of these products, or indeed the Microsoft Windows ISV (independent software vendor's) tool-kit.

Hardcopy devices are more of a luxury: any dot matrix printer, an Epson compatible preferably, is on the border of necessity. If buying a plotter, get one which offers compatibility with the Hewlett Packard Graphics Language (HPGL).

Both of us have spent many years using FORTRAN, and over five years with our PCs. The book distils some of this experience. However, we cannot hope to know all there is to know, and in any case, the compilers on offer get more sophisticated and generally better as time goes on. This means that the reader will find errors, out-of-date allusions, and matters of opinion no longer tenable, among the more generally usable material in this book. We hope that our readers will write with their own comments and observations, which we will try to include in future editions of the book.

1.8 ANNOTATED BIBLIOGRAPHY

The FORTRAN programmer probably needs a full bookshelf. Since we do not set out to cover FORTRAN programming from first principles, one of the first books he will require is a text on the language. Depending on the level you start from, the choice is varied. For the beginner, there is plenty of choice, but for the 'expert', looking to improve, the number of books is much smaller. We start with the beginner.

All the books described below cover FORTRAN programming, which we do not in this book. This is by no means a complete list of FORTRAN 77 books on the market nor are these necessarily the best of them. However, we think there are some good books here. When assessing a FORTRAN book look particularly at the example programs presented in the text. Are they complete? Are they varied and interesting? Are there many of them?