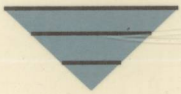
BASIC COMPUTER NUMERICAL CONTROL PROGRAMMING

SECOND EDITION



KENNETH J. LAVIANA

TP31
L411
E.2

9761675

贈閱

SECOND EDITION

Basic Computer Numerical Control Programming

Kenneth J. Laviana

國立中山大學
圖書館藏書

E9761675

Merrill Publishing Company
Columbus Toronto London Melbourne

Published by Merrill Publishing Company
Columbus, Ohio 43216

This book was set in Century Schoolbook

Executive Editor: Stephen Helba
Administrative Editor: Jennifer Jewett
Production Editor: Victoria Althoff
Art Coordinator: Ruth Ann Kimpel
Cover Designer: Russ Maselli
Text Designer: Debra A. Fargo

Copyright © 1990 by Merrill Publishing Company. All rights reserved. No part of this book may be reproduced in any form, electronic or mechanical, including photocopy, recording, or any information storage and retrieval system, without permission in writing from the publisher. "Merrill Publishing Company" and "Merrill" are registered trademarks of Merrill Publishing Company.

First edition copyright 1982 by Ken Tek Engineering.

Library of Congress Catalog Card Number: 89-63412
International Standard Book Number: 0-675-21298-7
Printed in the United States of America
1 2 3 4 5 6 7 8 9—94 93 92 91 90

Preface

In the late 1970s I was working for a division of a Fortune 500 company in Connecticut as a Senior Manufacturing Engineer. One of my major cost reduction and capital projects was the justification and implementation of CNC turning and machining centers in the firm's production facilities. Naturally, one of my concerns was to train the machinists in using this new technology, even though many of them had years of manual machining experience. One immediate problem I encountered was a lack of training materials. The machine makers' manuals usually assumed the user was already conversant in CNC programming theory, and some of the foreign manuals lost meaning when translated.

Although some textbooks were available, they seemed to relate either to older NC or hard-wired systems or to computer-assisted programming systems. Many of the ideas used in presenting technical theory in this book—how and in what order to present it, for example—were developed during in-house training classes that I gave.

In 1981, I left to begin a business of my own, doing contract CNC programming, process planning, and consulting. CNC manual programming classes became a major part of my business as more and more companies bought their first CNC machines. Eventually I began to offer CNC programming courses to companies who wanted to prepare their employees with a general knowledge in this field. Thus I came to begin this book.

This book is intended to help the student or reader understand what CNC programming is and how it works. I present the material in an easy-to-understand format, using analogies where appropriate and a conversational manner to help maintain interest. The presentation becomes more technical as the reader progresses through each chapter.

When presenting new subject matter, I include many program examples, covering several of the many systems currently in use. I am a firm believer in the value of examples rather than theory alone in the teaching of practical applications. In one chapter I list the meanings of many of the common and uncommon G and M codes used in current and past CNC systems. I also list some of the known differences in these codes among different systems, encouraging the reader always to check the machine builder's manual when in doubt.

This book does not become heavily involved with trigonometry but does have formulas and some examples on the use of trigonometry in programming. Math is a separate subject outside the scope of this particular book, which stresses planning, understanding, and the structure and use of code to establish tool motion and sequence.

The chapters on canned cycles for turning and drilling applications are comprehensive. They present many of the so-called standard meanings, along with some that are not standard but are found on some systems with large installed bases. The liberal use of illustrations in these and other chapters should help the student.

The concept of circular interpolation is introduced early and is then amplified in a separate, highly annotated, chapter. This particular function has always seemed the most difficult for some students to grasp, perhaps because of an inadequate math background.

The concept of subprogramming is introduced, and the use of examples should help the reader comprehend its value in helping to reduce the amount of code required to write a part program.

Chapter 11 consists of a test program for turning and one for milling, each partially written, so that the student can fill in the blanks.

There is a glossary of common terms. An appendix on calculating feeds and speeds includes the standard formulas,

此为试读,需要完整PDF请访问: www.ertongbook.com

a data table for 28 commonly used materials, and common formulas to calculate various drill tip heights and chamfer diameters.

An Instructor's Manual with additional programs, test questions, and sample programming forms is also available.

I wish to thank my fellow manufacturing engineer and friend John Donlan for his assistance in writing this book.

I am also grateful to the following reviewers who offered suggestions to improve the manuscript of this second edition: William Blackledge, Augusta Technical Institute; Paul J. Demers, University of Cincinnati; and Richard Neuverty, Hawkeye Institute of Technology.

Kenneth J. Laviana

MERRILL SERIES IN MECHANICAL, INDUSTRIAL, AND CIVIL TECHNOLOGY

ADAMSON	*Structured BASIC Applied to Technology* (20772-X)
BATESON	*Introduction to Control System Technology,* 3rd Edition (21010-0)
COOPER	*Introduction to VersaCAD* (21164-6)
DAVIS	*Technical Mathematics* (20338-4) *Technical Mathematics with Calculus* (20965-X)
GOETSCH/RICKMAN	*Computer-Aided Drafting with AutoCAD* (20915-3)
HUMPHRIES	*Motors and Controls* (20235-3)
KEYSER	*Materials Science in Engineering,* 4th Edition (20401-1)
KIRKPATRICK	*The AutoCAD Textbook* (20882-3) *Industrial Blueprint Reading and Sketching* (20617-0)
LAMIT/LLOYD	*Drafting for Electronics* (20200-0)
LAMIT/WAHLER/HIGGINS	*Workbook in Drafting for Electronics* (20417-8)
LAMIT/PAIGE	*Computer-Aided Design and Drafting* (20475-5)
LAVIANA	*Basic Computer Numerical Control Programming,* 2nd Edition (21298-7)
MARUGGI	*Technical Graphics: Electronics Worktext* (20311-2) *The Technology of Drafting* (20762-2)
MOTT	*Applied Fluid Mechanics,* 3rd Edition (21026-7) *Machine Elements in Mechanical Design* (20326-0)
POND	*Introduction to Engineering Technology* (21003-8)
ROLLE	*Thermodynamics and Heat Power,* 3rd Edition (21016-X)
ROSENBLATT/FRIEDMAN	*Direct and Alternating Current Machinery,* 2nd Edition (20160-8)
WEBB	*Programmable Controllers: Principles and Applications* (20452-6)
WOLANSKY/AKERS	*Modern Hydraulics: The Basics at Work* (20987-0)
WOLF	*Statics and Strength of Materials: A Parallel Approach* (20622-7)

Contents

Introduction

The purpose of this manual is to help you become a good numerical control programmer. The emphasis is on you, the programmer, not the machine, and on the things you must consider in order to write a good program.

What Is a Numerical Control Program?

A numerical control (NC) program is a set of instructions designed by you, the programmer, and fed into a numerical control machine. The machine acts on these instructions to fabricate a workpiece according to a part blueprint.

What Is a Good Numerical Control Program?

A good numerical control program is a set of instructions designed and carefully planned (put into an efficient, workable order) by the programmer and fed into a numerical control machine. The machine acts on these instructions in the fabrication of a workpiece according to a part blueprint. A good NC program takes into account (1) cycle time, (2) machine capacity, (3) tool life, (4) ease of setup, (5) operator accessibility, and (6) flexibility.

The first part of *Basic CNC Programming* deals with just that—the basics. The latter part deals with the refinements you must learn to become a good (maybe even great?) NC programmer.

The machines used most often in this book to illustrate techniques are the computer numerical control (**CNC**) lathe and the CNC machining center or milling machine. The skills

you develop as a programmer, however, can be applied to any machine.

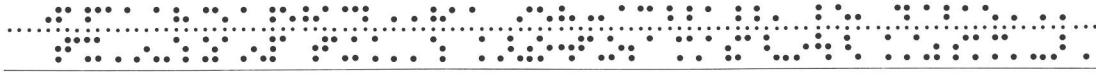
Follow this manual at your own pace. Go back and reread sections you don't understand.

Remember: If you really want to learn, you will do just fine.

Relax and enjoy! Numerical control programming is really quite fun!

此为试读,需要完整PDF请访问: www.ertongbook.com

CHAPTER 1



What Is Numerical Control Programming?

To find out just what numerical programming is, let's listen in on a conversation between a couple of employees at the coffee machine one morning.

"Brian, I'm tired of just using my muscles to work for a living. I'd sure like to use my mind some. I've been watching you do your job these past few weeks—you know, programming? It looks very interesting. Do you really enjoy it?"

"You bet, Ted. It's interesting and challenging, and I have a lot of fun doing it."

"Fun? In a job?"

"Yes, fun. Enjoyment, I mean. There's a great satisfaction that comes from writing a good program, one that enables the CNC machines to cut production costs in half—or even better!"

"If CNC machines are so great, why don't all shops have them?"

"A lot of shops do, Ted, but they are very expensive to buy—sometimes ten times more expensive than manual machines—and they need intelligent and creative people to program them—people who can think things out carefully and make decisions based on experience, on what the best approach would be to solving the machining problems."

"Well . . . I've got some experience in a machine shop, and I can think things out pretty clearly. But what is this stuff about creativity?"

"Ted, almost anyone with average intelligence can write a program, but the really good programmers use their creativity, their imaginations, to write good programs—programs that will make the difference between

profit and loss for their companies. In most cases there are probably a dozen different ways to write a program for a given machining job, but usually only one or two will work best."

"How do you do programming, Brian?"

"The trick to programming is to break all of your machining operations down into their simplest parts. You have to describe to the CNC machine exactly what you want it to do every step of the way. For instance, if you want to tell it to drill a hole in a specific place, you can't just program the words 'drill a hole.' The machine won't know where to do it, how deep it has to be, or how fast the drill has to cut. You have to tell it all of these things in your program, and you have to do this with each tool that is going to machine the workpiece."

"I don't know, Brian. It sounds awfully complicated to me!"

"Ted, when you look at a blueprint and then you look at the piece of raw material that the workpiece is going to be made from, it does look complicated. But if you start off by deciding just which machining operations need to be done and then (*this is very important*) in which order they should be done, that is half the job right there. It is simply a matter of starting with the first tool and telling the machine exactly what you want it to do with that tool. Then you tell the machine what to do with the second tool, and then the third tool, and so on, until the part is completed. The finished workpiece may look very complicated, but programming the CNC machine to fabricate it is really easy, if you take it one tool at a time."

"That doesn't seem too hard. I do that on my manual machine every day—take one tool at a time, I mean. The only difference is that I don't tell the machine what to do with each tool; I just do it."

"That's not quite true, Ted. When you turn down a shaft on a lathe, don't you decide which tool bit you want to use?"

"Yeah."

"And don't you choose the proper speed for the spindle, the one that will work the best and give you good results?"

"Yes I do, but . . . "

"And when you drill holes on a milling machine, don't you tell the drill where to go by moving the dials on the table to the right coordinates?"

"Coordinates? Oh, you mean locations! Yes, I do that, and quite well too."

"Well then, you are telling the machine *which tools to use* and just *what to do with them.*"

"I never thought of it quite that way before, but you're right."

Ted thinks for a minute about the way he goes about machining workpieces on the manual machines and about the steps he takes to do each operation, but then something puzzles him.

"Brian, how do you tell a CNC machine what to do with each tool? I know you don't just walk up to it and say, 'Here, machine this.' How do you talk to it?"

"You tell the machine what to do by using a code. The code is something that both the programmer and the machine can understand, and, with a few exceptions, there is just one meaning for each word in the code. That way there is no confusion about what the programmer is instructing the machine to do.

"The coded instructions are written on a piece of paper (called a *manuscript*). Then they are typed at a terminal and either fed into the machine's control unit (*MCU*) directly or transformed into a punched pattern on a roll of paper tape, which is taken to the machine and fed into a tape reader on the MCU."

"Do all of the CNC machines understand the same code, Brian?"

"Each CNC machine manufacturer has certain code words that pertain only to his machine, but the great majority of code words used can apply to almost any machine. When you learn what they are and how to use them, you can walk up to any CNC machine, and by reading the *printout* (the typewritten copy of the manuscript) of the program that is running, you can easily understand the instructions being given to the machine."

"That sounds fascinating, Brian!"

"It is, Ted. The code words are not hard to learn. You can even use the code words to talk to other programmers, and they will know exactly what you mean.

"But the key to being a good programmer is not how easily you use these words, but how well you plan their use. Any programmer can write a program to drill a hole in a piece of metal, Ted, but a good programmer will tell the machine how to do it without a lot of wasted motion."

Brian draws a simple sketch on a piece of paper, showing two ways in which a CNC lathe might drill a hole. The two possibilities are shown in Figure 1–1.

"Look at these two programming examples, and see if you can figure out which one will end up drilling the hole in the shorter amount of time."

"It seems to me that program B would drill the hole faster."

"That's right, Ted, and notice that there are also fewer moves in program B than in program A."

"Yeah, Brian. Program A has six moves, but program B has only four, and the tool starts closer to the workpiece."

"The fewer moves a program has, the shorter and simpler it is. This is what is known as programming 'smart' by using creativity. Both of these programs will

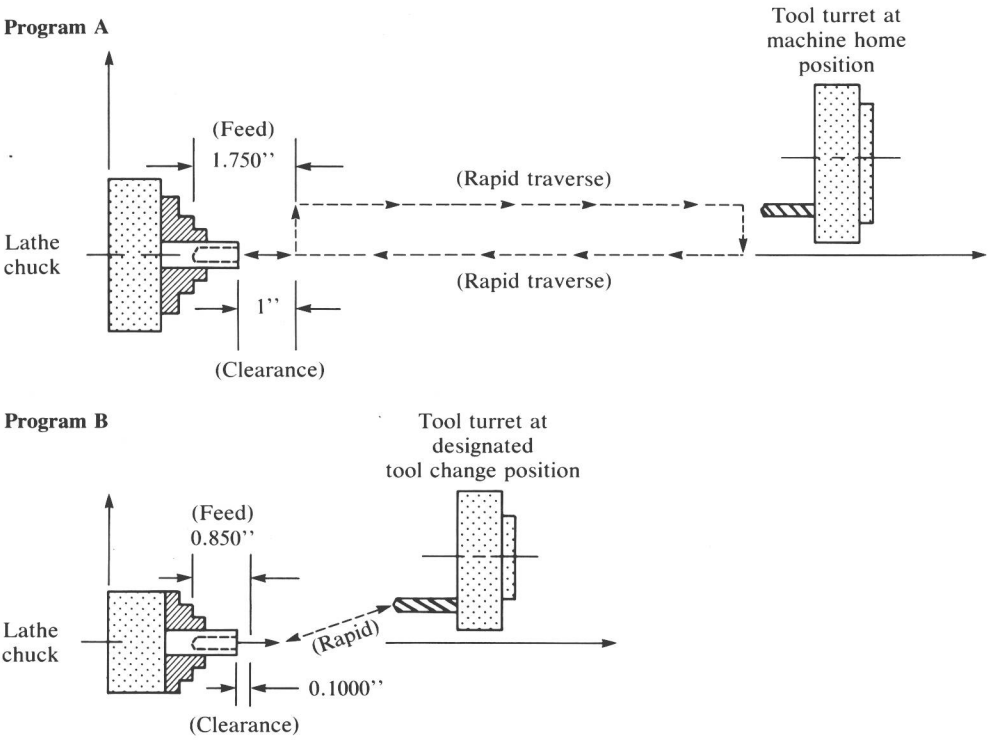


FIGURE 1–1 Lathe Drilling Example

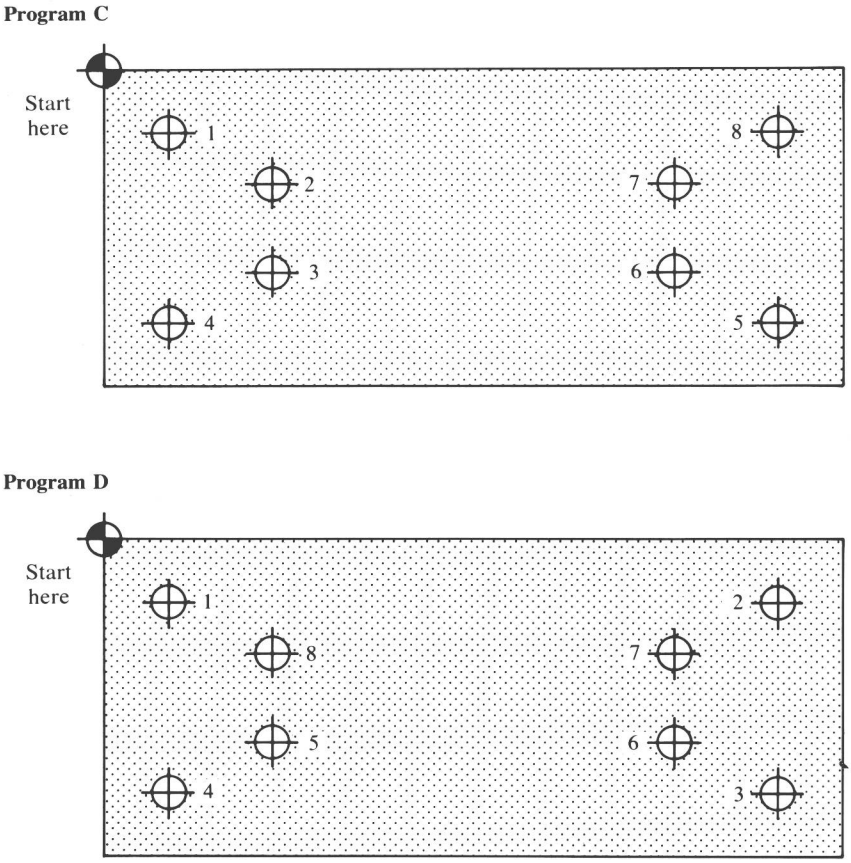


FIGURE 1–2 Machining Center Drilling Pattern

drill the hole equally well, but by thinking of alternatives to the choices in program A, the programmer was able to plan and develop program B, which will take less machining time and make more of a profit for the company. Programming 'smart' is programming for efficiency."

Now Brian draws another sketch on the paper, this time of two ways in which a CNC machining center might drill a hole pattern. These two examples are shown in Figure 1–2.

"Look at these sketches, Ted, and see if you can tell which of the two programs will drill the holes in the shorter amount of time. The numbers next to the hole

此为试读,需要完整PDF请访问: www.ertongbook.com