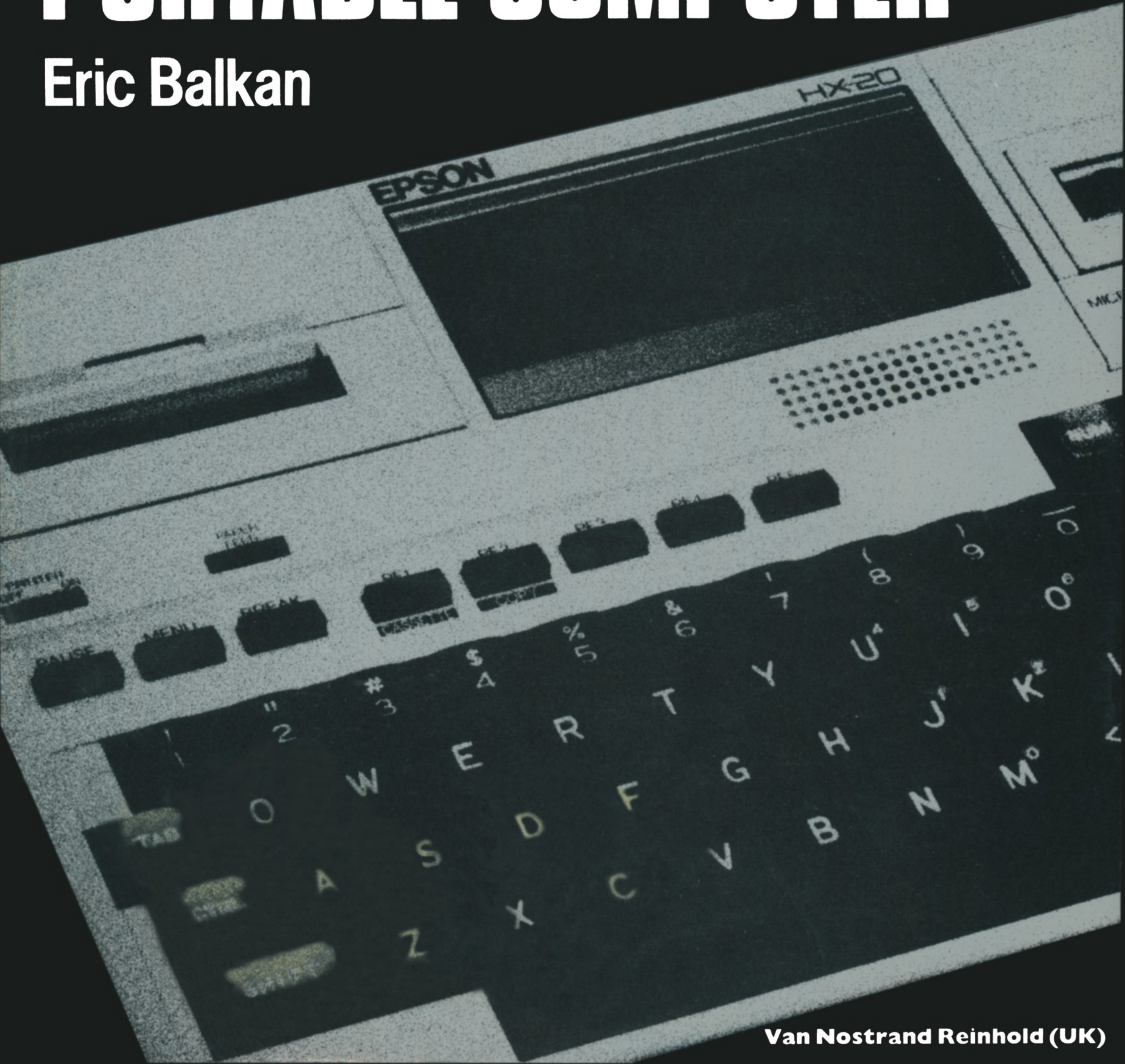
USING & PROGRAMMING THE EPSON HX-20 PORTABLE COMPUTER

Eric Balkan



Van Nostrand Reinhold (UK)

Using and programming the Epson HX-20 portable computer

Using and programming the Epson HX-20 portable computer

Eric Balkan

VNR UK

Van Nostrand Reinhold (UK) Co. Ltd

First published in 1985 by
Van Nostrand Reinhold (UK) Co. Ltd
Molly Millars Lane, Wokingham, Berkshire, England

Typeset in 10 on 11pt Palatino by
Columns Ltd, Reading

Library of Congress Cataloging in Publication Data

Balkan, Eric.
Using and programming the Epson HX-20 portable computer.

Includes index.
1. Epson HX-20 (Computer) I. Title.
QA76.8.E57B35 1985 001.64 85-3274

ISBN-13: 978-0-442-30650-2 e-ISBN-13: 978-94-009-4862-4
DOI: 10.1007/978-94-009-4862-4

CONTENTS

PREFACE

Why this book? Other than the fact that I like writing about computers more than just about anything else, this book fills several real needs. No matter how many manuals a computer manufacturer puts out to accompany a system — and some of Epson America's are very good — not everything can be covered. This book fills in the gaps.

This book is unbiased, having been written independently of Epson. So, I won't be telling you to drop everything and run out to buy an HX-20. The HX-20 is good for some uses, not so good for some others. This book is a guide to getting the most out of the machine and/or pointing you towards a different machine that might better suit your needs.

At the start of this project I had to decide who was my target audience: novices, experts, or those in between? Because HX-20 owners and prospective owners don't fall into neat categories, I tried to 'cover all the bases'. Or at least as many as possible. As with any attempt to do everything, I didn't always succeed. But I did succeed in providing at least something for everyone.

For those who haven't yet bought a portable — or are unsure if buying an HX-20 was the right move — there are descriptions of 20 other portables on the market.

For those who have used other computers before, there's information on how Epson BASIC differs from other BASICs, with tips on converting programs.

For those who want to learn how to control the HX-20 on a machine code level, there's information on a relatively easy way to learn assembly language — and an assembler to practise with.

For those who have never used another computer before, I've provided explanations of basic computer concepts, including an extensive glossary.

For everyone, there are descriptions — and vendor addresses — of many software packages and hardware add-ons available for the HX-20.

I plan to continue to play around with the HX-20 and will probably turn up additional information that HX-20 owners may find useful and which I'll be glad to share. You can contact me at PO Box 30214, Bethesda, MD 20814, USA. Comments, suggestions, and questions are welcomed. (Please include a self-addressed, stamped envelope if you want a reply. Overseas readers, please include US$1 or approximate in local currency.)

ACKNOWLEDGEMENTS

I'd like to thank Mark Weber who contributed several programs to the book, Terry Ronson of the HX-20 Users' Group who very kindly allowed me to reprint material from his newsletters, and Paul Heckel of Quickviews Systems who provided useful contacts at Epson America.

My thanks also to the software publishers who sent us copies of their programs for review: Talbot Computers Ltd, King Software, HCCS, Software Riches. And to those other publishers who sent me complete manuals on their products: Ffoss Ltd, Kuma Computers, Longdin & Browning, Transam Microsystems. And BiTech Enterprises who loaned me a bar code reader. Thanks to Naomi Sigler of The Bohle Company, Epson America's public relations firm, for her assistance with the project, including the loan of an HX-20. Thanks also to Bob Diaz and Ed Urbina of Epson America for being available for technical assistance.

Others deserving of mention: Robert Labenski, Sandra Thorogood of Thomas Nelson Australia, Eric Maloney of 80 Micro for allowing me to reprint a bar code article, Davey S. Thornton for writing the article, Hitachi America for allowing me free use of their data book, Ian J. Phipps of Epson (UK) Ltd, and my wife, Freda, for putting up with me while I was working on the book.

And Michael Balkan, age 4, for stress testing the machine.

Eric Balkan

TRADEMARK ACKNOWLEDGEMENTS

INTRODUCTION

This chapter covers:
An overview of this book
Some applications of portable computers

When science-fiction writers of the 1950s looked into the future, they saw computers as large as entire city blocks. These massive banks of flashing lights were to be cared for by an army of white-coated technicians. Perhaps, on occasion, an ordinary citizen would be gifted with a few nanoseconds of the computer's time.

Well, we know the future didn't work out quite like that. To the surprise of many, the second generation of computers was smaller than the first. And the trend has continued. Maybe we'll all end up with wrist-watch-size computers that we can talk to and that, in turn, can talk via airwaves to all other computers. Maybe we'll all end up with. . .

But, in the meantime, there are desktop computers, briefcase computers, 'lap' computers, and hand-held computers. This book is both a guide through the computer world and a guide to a specific computer, the Epson HX-20, and its future progeny.

This book starts at the beginning: picking a computer. It does not necessarily recommend an Epson computer. Rather we suggest that you look over what you need a computer for and compare your needs against what's on the market. We describe things to look for in a computer system. We also describe some 20 portable and not-so-portable computers that might meet your needs.

We don't expect too many of our readers to become programmers, but knowing even a little BASIC can help out quite a bit when running someone else's program. Or even getting a program written for another computer to work on the HX-20. So, there are two chapters on BASIC. The first describes HX-20 BASIC and how it differs from other BASICs. The second gets into conversion techniques, programming tips, sample programs, and so on. Both chapters were written with the idea that we would add to Epson's own manuals, rather than duplicate their contents.

To understand the HX-20, or any computer for that matter, requires a knowledge of the microprocessor and some feel for what machine code is all about. So we'll spend some time on that.

The book also goes into some typical ways to use the HX-20 or any other portable: communications, word processing, inventory.

Many software packages that are available for the HX-20 are described in this book. These include programs for business, entertainment, engineering; systems for pharmacists, surveyors, the handicapped, etc.

You can read the book sequentially or just pick out the chapters that most interest you. If you come across terms or concepts that are unfamiliar, check the glossary. Or, backtrack to earlier chapters that offer more explanations.

Whether or not the HX-20 becomes popular, portable computers are here to stay. They bring the power of the computer to you, wherever you might be, rather than force you to go wherever the computer is.

SOME APPLICATIONS

What can you do with a portable computer?

Well, some people who might make use of its capabilities are:

- pilots (compute flight plans)
- engineers (calculations)
- salespeople (order entry)
- students (class/library note-taking)
- reporters/writers (when away from their VDT/typewriter)
- insurance agents (for presentations)
- financial analysts (calculations)
- investors (on vacation, at the broker's)
- consultants (time reporting/billing)
- lawyers (courtroom notes)
- executives (scheduling)
- pharmacists (drug label preparation)
- handicapped (voice synthesis, motor aid)
- truckers (accounting)

The list can go on and on. We talked to some of the people that were using portables, and the HX-20 in particular, and would like to pass on the experiences of two of them.

Brad Knapp works for the USDA at a Montana livestock and range research station. He's been using an HX-20 for six to seven months to collect data on animal weights. Connected to electric scales, the HX-20 records the data on tape and provides a printout on site. But while Knapp uses the HX-20, he refrains from endorsing it. His biggest complaint: the 'Charge Battery' message coming up in the middle of doing something. The HX-20, he feels, would have been a better design if it could switch to AC power and/or if the batteries could be changed in the field.

Dan Holoien works for the American Crystal Sugar Research Center. He collects agricultural data for later editing and transmission to another computer site. Like many HX-20 owners, he was attracted by the internal printer and the quality of the keyboard. A Forth user, Holoien also sees the HX-20 as a good Forth teaching tool. He also sees possibilities for the high-speed serial port.

A lot of what the HX-20 can do remains as possibilities. Since portable computer use is not widespread, it takes a certain amount of investigation and experimentation to learn the best ways to use the machine. Magazines like *Portable Computing* often run articles on how people use their portables. We'll give some suggestions here too.

Students

Portable computers have been suggested as ideal for students who have to take notes in class or while doing library research. Well, 'ideal' is not quite correct. There are advantages and disadvantages.

A good typist can type faster than he/she can write. And there's no writer's cramp. Notes that are in machine readable form have the additional advantage of being able to undergo further processing. Notes can be re-organized, formatted, printed on an external printer. This makes later study easier, for individual study as well as group study. Coupled with the search feature of a word processor, specific references can be found that might otherwise be hard to locate.

We mentioned disadvantages above. Here's the biggest: no capacity to include diagrams. Some subjects: chemistry, biology, geology, make heavy use of diagrams. No maths formulas either. Conceivably, the student can do the diagrams or formulas on paper and somehow reference these diagrams back to the taped notes — but this seems clumsy to us. Experimentation is in order. Perhaps, with the right software, the HX-20's graphics capability can overcome this problem.

Another disadvantage: it's conceivable to lose an entire session's notes because, for some reason, the tape can't be read back in. Another subject for nightmares: in the middle of note-taking, the machine hangs, a cold start must be done, and previously entered text is lost. Well, fortunately, this kind of thing doesn't happen too often. But its possibility should be kept in the back of the mind. Perhaps an audio tape recorder should be used as a back-up. You don't want to use an audio recorder as a primary note-taker, because you probably won't have the time to listen to all the class sessions over again. But it can be used as a just-in-case device.

Pre-School children

Children are taking to computers at a rate that astounds many older people. But it's actually been known for quite a while that children love IBM Selectric typewriters. They like keyboards. They like the direct cause-and-effect relationship of hitting a key and seeing a letter appear.

Portable computers like the HX-20 are small. Coupled with appropriate software — and this includes easy-to-use word processing programs like SkiWriter — an HX-20 can become a very practical alternative to the usual video computer system. Built-in printer, no cables, rugged con-

Fig. 0.1 Pre-school use

struction, no tying up of the family TV set — what do you think?

Lawyers

When the Osborne I came out, numerous lawyers saw it as something that could be taken into the courtroom. But the Osborne I and similar transportable computers are really quite bulky to carry around in this fashion. True portables like the HX-20 should do better in this environment.

Do portables have a place in the law library? Perhaps with the proper software. With an IBM 3101 terminal emulation program, they might even be able to be used as LEXIS terminals.

Travellers

As we go to press, neither the Federal Communication Commission nor the Federal Aviation Administration has ruled on whether or not portables can be used on board aircraft. We know for certain that the Osborne I computer causes radio frequency interference (RFI) that interferes with flight instruments, but we can deduce that this stems from the Osborne's disk drives and CRT display. Without disks, without a CRT, portables should produce much less RFI. But if you intend using a portable while flying, you'd best check with the airline.

That still leaves shipboard, train, and car use. A battery-powered computer has a distinct advantage over its line-powered brethren here.

This has been just a smattering of suggestions. If you read the book, and particularly if you look at Chapter 11, Software and Systems, where many available programs are described, you'll likely uncover some way that a portable computer can help you in your business or profession.

1 WHICH COMPUTER?

'To err is human, but to really foul things up requires a computer.'

This chapter covers:

DEVELOPING A SELECTION METHODOLOGY

Creating a formal plan before making a computer purchase can save a lot of time in the long run. The cost of a computer may or may not put a big dent in your budget. But buying a computer is similar to getting married. You'll be spending a lot of time living with your choice, so it's wise carefully to consider your options before plunging.

The first step must be to analyse your business or profession. Rather than go by intuition, try to verbalize what it is you do. Then analyse your needs. Decide what is a requirement and what is a nicety. Put this in writing so that you'll have it later to refer to. Putting things into actual words, by the way, is a good method for focusing thinking. Now, check out the market. Just look around and see what's available. Then go back and adjust your expectations and your list of requirements.

Once you've decided that there are things on the market that can help your business, then you can more seriously check out specific products. Of course, if it looks like current technology is pretty far away from your business needs, save your search for a later date.

Narrow down the list of possible products/vendors. Call dealers and ask to have demonstrations of their equipment. Rather than just walking in off the street, call the dealer in advance and tell him what you're looking for so that he has time to set up something just for you.

When you've narrowed down your search to two or three models, check customer references. Make sure that there are real people out there who have bought the same products you're planning to buy, from the same dealer you're planning to deal with. And that they're happy with their choice.

This whole process is iterative. You may think you've got your selection nailed down, then suddenly discover from talking to a user that your prospective purchase is missing some important element. This takes you back to the beginning again.

When you do make your selection, and it's from a dealer you haven't dealt with before, be sure to have everything in writing, including all salesmen's promises. Not so much because you'll really want to return the machine later on, but to see how much of the salesman's pitch the vendor is really willing to stand behind.

COMPUTERS v. TERMINALS

Up until the late 1970s, if you used a computer you probably did so by connecting to it via a remote terminal. This terminal provided a keyboard for you to enter something into the computer and some way — display screen or 'hard-copy' printout — for you to see the results. Sometimes the connection to the computer was over a telephone line and sometimes via a direct wire.

Terminals have not become obsolete with the advent of the personal computer. In fact, terminal sales are up because many businesses are using terminals to connect to multi-user microcomputers, as well as minicomputers and mainframes. Terminals have improved, too. The simple 'dumb' terminal has grown up and can now edit data, store it in memory, and display it with a variety of options, even colour.

The examination of your business/personal needs may not rule out the use of a terminal instead of a computer. If your primary need, for instance, is to read data stored on a computer system — stock quotes or bibliographic references, perhaps — then a terminal may do the job. Even if you need to save the data for later printout, a terminal with memory such as the double memory units from Teleram can do it.

But if you need to process the data in some fashion, then you'll probably need a computer. While today's terminals have microprocessors inside them, this processor has been programmed to do only one thing — act in the way the manufacturer wants it to. If you as a user want to manipulate the data in some other way, then you need a computer.

PORTABLES v. HAND-HELDS v. TRANSPORTABLES v. PLUG-INS

Computers can be divided into several classes. Besides the desktop models that we've all seen — Apple II, IBM PC, etc. — there are several types that can be called portable, or at least transportable, in one sense or another.

Hand-held is the name originally given to those computers that were intended to be used while being held in the palm of the hand. They look and operate as calculators which have been expanded with an alphabetic button-type keyboard, a couple of thousand characters of user memory, and the BASIC programming language. Relatively few people have found this type of computer meets their needs, but they're worth checking into if you're a heavy calculator user.

Transportable is the name often given to desktop computers that can easily be transported from place to place. These include the best-selling Osborne I and Compaq computers. Often, some design compromise is made between functionality and bulkiness. But, generally, these computers can do nearly everything that ordinary desktop computers can do.

If you don't need true portability, if all of your computing is done in an office, then look into transportables. Comparing transportables to desktop computers, transportables are a better bet even if you expect to use your computer in one place, as there will undoubtedly be times when you'll want to use it elsewhere. Comparing transportables to portables, the size and weight of the transportable makes it something you don't want to carry around as often as you would a true portable.

Plug-ins is the name we've given to computers about the size of portables, but which require a hook-up to an external display screen. A model like the Victor/ACT Apricot, for instance, could be tucked under one arm and transported from place to place. When you arrive at your destination, you just plug it into a power source and a display. Most office locations may not fill the bill, but hotel/motel rooms will. So work can be done in the ordinary fashion during the day, with computer use reserved for when you get back to your room. Of course, if you split your computing entirely between two locations, the most convenient thing to do is buy two compatible computers.

WHY A PORTABLE?

The truly portable computer has advantages over its other computer relatives, among them: size, weight, and freedom (if only temporary) from a power cord. As we noted earlier, it's important to examine your needs so that you'll know what's really important for you.

THE IDEAL PORTABLE

In the mind of every portable computer designer and probably every portable computer user, there exists the 'perfect portable'. This machine will do everything for everybody. It will include:

- high-speed 8- and 16-bit processors compatible with software for all of the other machines on the market;
- sufficient memory (about 500,000 characters) to run any program on the market;
- sufficient secondary storage (built-in) to allow an entire 5–10 million character database to be carried around;
- easy-to-read full-page display, with graphics, on a flat screen that flips out of the way when not in use;
- full-travel keyboard with special user-friendly keys for any conceivable operation;
- input devices (e.g., voice recognition, touch-screen) that allow the user to bypass the keyboard for many functions;
- built-in, fast, letter-quality printer with graphics capability;
- full communications capabilities to allow for high-speed data transfer, as well as the ability to emulate any type of terminal in common use;
- power supply that lasts for as long as you need it to, which will be possible because none of the above components will use any power worth mentioning. Of course, the inevitable recharging process will take no more than a few seconds;
- the size of the entire package will allow it to be carried around (pocket-sized?); weight will be unnoticeable even after you've carried it around all day;
- standard software will include all of the usual functions that everyone needs, in an easy-to-use, easy-to-learn format, completely integrated with one another, as well as the ability to perform more than one task at a time;
- Price? Under $1000 for outright purchase, with rentals available.

Back in the real world, we're not going to see this machine very soon. But then, we don't really need to. What we really needs is something that fits our own personal requirements. If you want a portable note-taker, for instance, you don't need a lot of secondary storage. If you want to retrieve stored data, you don't need a full-page display. That's why we can't overstress how important it is to:

- outline your needs based on what you are going to be doing with the machine;
- develop a checklist of 'must-have' features — hardware and software, both;
- develop a checklist of features in the 'nice to have but not strictly necessary' category.

Some further advice:

- don't buy features you don't need or are unlikely to use;
- if features that are indispensable to you are not present, it may pay to wait for new models rather than buy.

Hardware and software can't be separated. Unless you intend to write your own programs, the software that is commercially available will determine your use of the machine. For instance, even though a particular machine may seem like an ideal choice for text editing, it'll be useless unless somebody writes a text editor for it.

Often, software can overcome hardware deficiencies. A text compression/expansion routine, for instance, can make up for having a small display screen.

WHAT TO LOOK FOR IN A PORTABLE

Size and weight

You're buying a portable because you can carry it around, so size naturally becomes a major criterion. If you could fit the computer into something that you always have with you, like a pocket, that would be ideal. But then you can't get a full keyboard into a pocket device. The next best thing is to look for a computer that can fit into a case which you frequently carry or wouldn't mind carrying.

The way to find out if you really won't mind carrying a particular computer around is to locate something of the same size and weight. Try carrying it around just as you would the computer. Try typing on it, try supporting it in your lap, etc.

What we said for size goes for weight as well. Many Osborne owners have unwittingly found themselves into some heavy computing. Again, locate a dummy object with the same approximate weight and see if it can be manageably carried for the time and distance that you would be carrying your portable.

Size of the display

If there's anything that grabs the attention when comparing one portable to another, it's the size of the display. And it can make a big difference. We've found a small display, for instance, to be really unsuitable for entering/editing text — we wanted to see the previous paragraph but couldn't.

Vendors will tell you that you'll get used to the small screen, but then, vendors will tell you anything. The truth is that if you are used to using a standard size screen, you will probably never get comfortable performing the same functions on a 4 line by 20 column display. But then, you may not intend performing the same functions on your portable as you would on a desktop computer. All the more reason for you to build your own checklist.

What actually will you be doing with the portable? If you intend doing a lot of data retrieval from remote mainframe computers — using the Dow Jones News Service, for instance — it pays to get a portable with a screen the same width as the data you will be receiving. If you don't, you're left with two options: having the data wrap around to a second line which can make it hard to read. (Tabular data is particularly difficult to read that way.) Or, if the portable has virtual screen capability, you can horizontally scroll to read the data. More on horizontal scrolling — which we're not fond of — later.

Doing program development? If you can't fit a line of code onto a line of the display, you'll have trouble following the program logic. If you can't fit a line of code onto the screen at all — BASIC statements can be up to 255 characters — you're really in for it. But if you intend doing your development on another machine, then this is not a big consideration.

Want to do spreadsheets? The smaller the screen the harder it's going to be to grasp the inter-relationship of the data.

Doing inventory tracking? The screen size probably won't matter. Ditto for any application that uses the portable as a sophisticated replacement for a calculator.

Readability of the display

This is often a subjective judgement. But there are definite criteria to rate a display on. Of course, if you're using your portable in an application that doesn't require looking at the display very often, you can skip to the next heading.

An electro-luminescent (EL) display, such as on the GRiD Compass, is probably the clearest one that can be produced by present technology. After that comes CRTs, green or amber long-persistence phosphor preferably. A black/white CRT produces a persistent flicker that you may not notice consciously, but which will tire your eyes with prolonged use.

A liquid crystal display (LCD) such as found on digital watches is the least desirable type of display from a readability standpoint, but is the lightest weight, the cheapest, and uses the least power. Those qualities make it practically ubiquitous on the current generation of portables. Other displays that you may see in the future but which are not now economically or technically feasible are —

plasma — too expensive
EPID (electrophoretic imaging displays) — still under development
flat CRTs — still under development
vacuum fluorescence — presently seen on some consumer products, but so far only suitable for very small displays.

An LCD screen is best seen looked at straight on. If viewed at an angle, the characters will be harder to read; if viewed at a sharp angle, they may be completely invisible. You may find that the position in which you'd like to place your portable does not let you see the display clearly. You may also find, unless you're using the portable like a desktop computer, that you're forced to sit in a fixed position. Or that you're constantly re-adjusting the position of your portable to bring the display back into focus. In both these cases, it would help to have a control that tilts the LCD, such as on the Teleram, and thus makes the characters more readable.

Typically, characters on an LCD screen are made from a matrix of 5 horizontal dots by 7 vertical dots, with an additional dot horizontally and vertically for separating the characters. This compares poorly to the usual 7 × 9 display found on CRTs. (Some CRTs, such as on the Corona PC, may go up to 16 × 13 dots.) Of course, people have used Radio Shack TRS-80 Model I/III computers for years (including your author) and these have only a 5 × 8 matrix. But the more dots the easier the display is to read and therefore the less tiring it is to read it. While on the subject of dots, note that some of the lower case letters on this printed page descend below the line. A 7 × 9 CRT display reserves 2 dots for these descenders. Most 5 × 7 LCD screens don't have any dots available for descenders. Without descenders, the display takes a little longer to read. Some models, like the Radio Shack 100, provide 1-dot descenders, but these share the same dot-line as the cursor. That may not be an advantage.

Yet another drawback of LCDs is their inability to be read in very dim light. If you own a digital watch, you know that there are some times you want to look at it but there isn't enough light to do so. The same thing applies to portables. If you think a power failure will let you continue computing on your battery-powered portable, think again, you'll need one hand to hold the flashlight. A backlit display would be very valuable, but as far as we know, only the Datec portable has it.

If you plan on using your portable in a fixed location very often, consider getting one with a monitor (CRT) or TV attachment. This provides a way for portables with very small screens to display a more reasonable number of characters. If you go back to the same place constantly, just leave your monitor there. If you move around very much, a TV attachment is useful because there are many more TVs around than monitors — such as in motel rooms.

Keyboard type

Desktop computers almost always have the same basic type of keyboard. The feel may vary from one to another, which is something to consider, but you don't see the great variety of keyboard types that you do with portables. So, if you've bought a computer before you may not have paid much attention to this facet of the system. But check it out before you buy your portable.

If you plan to do any kind of text processing, you'll want full-travel, full-size keys. It's just extremely difficult to type rapidly on any other kind of keyboard. If what you will do is mostly calculations, then a button-type, limited-travel keyboard may be OK.

If you plan to work in 'hostile' environments, you'll want a sealed keyboard to keep out dust, liquids, etc. Generally, this means a flat, touch-sensitive, 'membrane' keyboard such as you may have seen on the Atari 400 game computer. A membrane keyboard is best equipped with some sort of feedback — a tone or click — so that you will know that a key has been depressed — since you won't be able to feel the depression. At least one company is working on a full-travel membrane keyboard, but that type has not yet shown up in any commercial products.

Keyboard layout and key assignments

Most portables have their keys laid out in QWERTY fashion — the standard typewriter arrangement. But after that, there are no standards.

Do you need keys for specific functions? Do you want to be able to assign keys to your own functions? 'Soft' keys are user-definable. Usually, a computer will have one or more of these soft, program function (PF) keys. Some portables will have an entirely soft keyboard, allowing the sophisticated user to re-assign any key. (Among desktops, the IBM PC and TRS-80 Model I/III have soft keyboards.) This is particularly valuable where the portable is used for a single purpose, rather than as a general purpose computer. Single keys can be assigned to character sequences, for instance, that would otherwise require lengthy operator typing.

Do you need a numeric keypad? On some portables, the right-hand side of the keyboard contains an integral keypad. Typically, there will be a NUM key or equivalent that changes these keys to numbers only. This is OK but takes getting used to if you're familiar with a standard numeric, calculator-type keypad. The difference comes because these keys are on staggered rows, while a calculator keypad has its keys on aligned rows.

Do you want to be able to enter graphics characters directly from the keyboard? On some portables you can, others require that graphics be created only from within a program.

Graphics

Most computers provide two ways to put pictures on the screen: character graphics and dot graphics. Character graphics is simply the assignment of a picture to a particular code. When this code is generated — either by the keyboard or by a program — this graphics character appears on the screen.

The character that's drawn takes up as much space as an ordinary letter or number would. In most computers, these characters are pre-defined. There may be an assortment of vertical and horizontal lines, boxes, Greek letters, etc. Some computers, like the HX-20, allow the user to create his own graphics characters within the allowable dot matrix of a character space.

A more powerful means of generating pictures is by programming each dot on the screen. By turning each dot on or off, any size and type of graphics may be drawn. The more dots on the screen a computer has, the greater the resolution. Diagonal and curved lines look less jagged, for instance, at higher resolution. For many computers, this dot graphics facility is built right into BASIC and is easy to experiment with.

Of course, if the entire screen is only one or two lines high, then screen graphics don't mean much. But you may be able to use the built-in graphics capability to draw directly on an attached printer, or possibly insert character graphics into a text file for later output on a printer.

Processor

The brain inside any computer system is the processor. The micro revolution, in fact, was begun when a processor was put on a single silicon chip. Since that time many different microprocessors have been developed. Today, the differences among these microprocessors creates a basic incompatibility among software for different computers. (More on software compatibility later.)

There are several processors that can show up in portable computers. First, there are the NMOS processors which have been used in desktop computers for several years. These are fast, but use quite a lot of power. Some portables were designed to use these processors, e.g., the Z-80 and the 8086, but these machines cannot live for long away from a power cord.

Most processors found in portables are CMOS versions of NMOS processors. The 6301 is a CMOS version of the 6801, the NSC800 is a CMOS version of the Z80, the 80C85 is a CMOS version of the 8085. While slower than their NMOS equivalents, CMOS chips use far less power, which makes them more suitable for portables.

The criteria for deciding among the various CMOS processors is: what software can they run, how much memory can they support, and how fast are they. All of these topics are discussed in separate sections in this chapter. At this time, suffice it to say that the 8080, 8085, and the Z80 are part of a processor family that can run the CP/M operating system with all of the business applications developed under it. (Assuming that sufficient memory, etc., is available.) The MS-DOS operating system used by the IBM PC can be run on the 8088/8086 family, assuming that other requirements are met. The 6800/6301/6303 family, on the other hand, is backed up by little software.

RAM

The letters in RAM stand for random access memory, but all computer memory today is organized so that any location can be accessed at any time. What RAM really is, then, is read–write memory. Unlike ROM (read-only memory), the user can store data and programme in RAM. It doesn't take any leap of faith, therefore, to see that the more RAM you have the better off you'll be.

Portables tend to have less RAM than desktops because semiconductor RAM chips only hold their contents as long as power is applied — and power has to be allocated very carefully in a battery-operated unit.

Many people who run spreadsheets and do word processing on 48K (48,000 character) RAM desktop computers find that they run out of space eventually. So, you'll likely run out of space that much sooner on your 16K portable. Also, you may want to have more than one program in memory at the same time — another reason for looking for as much RAM as you can get.

One solution, found on the NEC PC-1801 and a few others, is RAM cartridges. These are independently-powered plug-in modules that expand the capability of the portable without draining the battery. They're also used as secondary storage. That is, you can take them out, put them in your pocket, plug a blank one in — and be able to plug the first one back in later on without losing any of its contents. That works especially well for the situation where you're working on something, want to stop to work on something else, and then go back to working on the first task. It also works well with large databases where speed is important (and tape is too slow) but where the entire database can't fit into memory at one time.

Read-only memory

Many manufacturers stress the amount of read-only memory their systems contain. But this is really inconsequential. By and large most users will not be putting their own programs in this ROM space. So, the ROM is really for vendors. And what programs go into this ROM is much more important than how big it is.

The only advantage in having a lot of ROM space available is that it saves time loading from cassette or across a communications line, which is a worthwhile advantage, by the way. But often this comes at the cost of reducing the amount of RAM available, usually a much more important consideration for the user. The reason this happens is that even though the contents of ROM are burned in, it still requires power to access it.

Another reason: without sophisticated memory management, an 8-bit microprocessor like the Z80 can't access more than 64K of memory (RAM + ROM). A new version of the CP/M-80 operating system, CP/M Plus, does this, but CP/M Plus hasn't turned up on portables yet. Some 8-bit processors like the 6301 may do some memory management of ROM, but what they do is limited. Portables with 16-bit processors, by the way, don't have this problem and usually can give you much more ROM than you can fill.

ROM cartridges — or even just plug-in ROM chips as found in the Panasonic HHC — are a good way of handling the what-do-you-do-with-the-ROM problem. There's no problem with contention for the same memory location between two programs because one can be removed and the other plugged in. Again, think of a ROM cartridge as a sort of permanent, unchangeable, super-speed cassette or disk. It's a form of secondary storage of programs supplied by vendors (or even your own programs, if you have access to technicians who can burn PROMS — programmable read-only memories — for you).

Tapes

Back in the early days of microcomputing, audio cassettes were just about the only means of storing programs and data. Then a few hobbyists started interfacing floppy disks to computers, Vector Graphic built a computer with a floppy disk right in it — and the cassette was dead.

Until home computers. Floppies, even minifloppies, were too expensive for the early home computer buyers, so cassettes underwent a resurgence. Once again, though, as floppy prices continued falling, buyers started opting for the speed and reliability of disks. Now, it's portable computers that are keeping cassettes alive.

A hook-up to an audio recorder that uses a standard-size cassette is available on a number of portable computers. Becoming more common, though, is the built-in microcassette drive. A microcassette drive under software control is easier and more convenient to use than standard cassettes, but not really any faster. Still, the weight of a disk drive makes tape a good choice for a portable.

Floppy disk interface

Floppies and portables don't really go together. They're heavy and should not be moved while in use. But having a connection to a floppy can be useful when you get back to your home base. It allows you to store the data that you've collected onto a permanent, reliable medium.

The use of a disk interface has to be balanced against the alternative of uploading your data to another computer. In the first case, you don't need to buy a second computer — though the cost of the disk drives may be almost as much as a second computer. If you already have another computer, then all you need is some sort of communications link between the two. You may have the necessary software to process your data already on this other computer which means that you are spared the expense of buying and learning another software package to run on the portable.

If you decide you do need a floppy wired directly to the portable, then check the following:

1. Is the interface serial or parallel? A serial interface sends 1 bit at a time and is going to be slower than a parallel interface. This is not a problem with slow devices like printers, for instance, because the rate of printing is determined by how fast the print head can move and not by how fast the data transfer is. But in talking about disk drives, then the data rate to/from the disk becomes important.
2. Is double-density supported? Single density 5.25 in (133 mm) minifloppies typically handle 90K of storage. Part of this 90K will have to be set aside for a disk operating system and for various programs, which leaves little room for data. So, double-density (about 170K) is much preferred. Even better are double-sided disks (340K), though here the diskettes tend to wear out faster. There are also quad density disks — still more storage, but less reliable — and a number of 3–4 in (76–102 mm) microfloppy disks with varying storage capacities. The Sony 3.5 in (89 mm) in diskette with its hard plastic case is a good choice for data reliability. But since we're going to use our floppy on a desktop, there's no particular advantage to a size smaller than the standard minifloppy. And

minifloppies are extremely reliable as long as they're kept away from dust, coffee, etc.

Printers

An integral, built-in or attachable printer is the best solution for hard-copy needs. The convenience of even small printers makes them valuable for those times when a full-size printer is not available, takes too long to hook-up or can't exactly reproduce what you want.

How bulky is the printer? How heavy? Is it something that can be carried around easily? If the answer to this last question is no, you're losing the portability you bought your portable for in the first place.

What kind of print output can you get? Can you reproduce the entire computer screen, graphics as well as text? Can it be done at any time, via a PRINT key, or do you have to go into a special routine which may overwrite the screen?

Print quality varies greatly among printers. Some printers, called letter-quality printers, produce fully-formed characters. Others, the dot matrix type, produce characters made up of dots, just as a display screen does. Dot matrix printers run the gamut from barely-readable up to near-letter-quality. If you need a built-in printer, decide what print quality you're willing to settle for.

If you don't need portable printing, then just make sure that your selected portable has an interface to enable you to hook up whatever printer you'll be using. The most common type of interface is called Centronics parallel, but RS-232 is also acceptable. If you do use the RS-232 port for a printer, and the portable has only that one port, keep in mind that you'll only be able to use a printer simultaneously with another RS-232 device (like a modem) with difficulty.

Speed is another important factor in choosing printers. The printer is typically the slowest part of a computer system and often prevents you from doing anything else while waiting for something to finish printing. (Sometimes RAM — internal or external — can be used to spool the print file and thus free up the computer sooner.) Speed is usually given in cps — characters per second. For comparison purposes, a fast typist can type on an IBM Selectric at about 8cps; the popular Epson MX-80 dot matrix printer is rated by the manufacturer at 80 cps. Unfortunately, all manufacturer-claimed printer speeds are wrong. Like automobile mileage ratings, the speed you get may only be half of what the manufacturer says is possible. So, use speed ratings to provide rough estimates only.

Print line width is another major consideration. Most computer output is 80 columns, some is 132 columns. Anything less than 80 and you'll find some application programs will wrap the print around to the next line. If you're doing spreadsheets, you'll probably want as wide a line as possible. If you're doing text editing, it's not as important. Some programs are starting to come out now that print sideways so that the line width is effectively infinite. If that sounds like just what you need, check to see which computers/printers it's available for.

RS-232

RS-232 is the standard for data communications connections. The things to look for here are: synchronous v. asynchronous, speed, and cable type. All of the technical aspects are explained in Chapter 8, Communications. But for those already familiar with communications, here are the things to look for:

- Does the port accept a male DB25 connector? If not, then special cables/connectors will have to be ordered.
- What speed can the port handle? If you're using a 300/1200 bps modem then there's probably no problem — nearly all computers can handle this. If you want to connect your portable directly to another computer for uploading/-downloading information, then you'll want to be able to use a higher speed.
- Do you want your computer to 'talk' to another computer using a particular synchronous protocol such as 2780 or 3270? Then your RS-232 port will need to be synchronous as well. Otherwise, you'll have to get a separate (and expensive) asynchronous-to-synchronous converter.

If the portable you're looking at has no RS-232 port, then you'll need to buy your modem (and possibly your printer) from the computer manufacturer. Or, possibly, there's a converter on the market that will transform what the computer does have into RS-232. But that approach means carrying a separate box around.

Other serial ports

A serial port is an interface over which bits are transmitted one at a time. RS-232 is the most common serial interface, but sometimes other, non-standard, non-RS-232 serial ports are seen. These can be used for a floppy disk interface, TV interface, or for other special purposes as long as the particular computer manufacturer supplies the necessary hook-up. If not — if it's just been promised for futher delivery but doesn't yet exist — then the port is useless.

Centronics port

Most computer printers will hook up to a parallel connection called a Centronics interface. It's given that name because a New Hampshire printer company designed and popularized that interface. But since then, most printers (and computers) have adopted it as the most desirable way of connecting a printer. Unlike RS-232, there is never a need for the user to be concerned about what signal is on what pin. If the connection can be physically made, then it will work.

If there's a drawback to Centronics connections, it is that the plug usually used has 36 pins, which would take up quite a lot of space on a portable computer. Sometimes other plugs are used, but there's no escaping the fact that a serial printer port needs only one data lead, while an 8-bit parallel port needs at least eight. But that's a problem for designers. The problem for users is generally finding a cable with the right plug on the end.

Typically, computers with a Centronics printer interface also have an RS-232 interface. The main advantage of this is being able to have a device connected to both ports at the same time. For instance, you may want to receive data into your computer from a modem and print the data at the same time — a difficult task if you have only a single I/O interface.

Other parallel ports

If you use data acquisition devices in your work, you'll want to be able to feed directly into the computer. The best way to do that is if the machine has a bidirectional parallel port. You can't usually use a Centronics parallel port, for instance, because Centronics ports are set up for one-way traffic only, i.e., out from the computer. If no parallel port is available, you don't have a total loss because parallel data can usually be converted to serial through a 'black box'. Check Chapter 12, Peripherals, for more information on this.

Built-in modem

Like a built-in printer, an integral modem means one less piece of equipment to carry around. Of course, this only works if the phone you want to use has a modular jack. If it doesn't — like in a telephone booth — you'll still need a separate acoustic coupler. And modems vary from one country to another, so a Bell-compatible modem won't have much use outside the USA. Some integral modems are 300 bps, which is a very common transmission speed. Some others are Bell 212A compatible, which allows transmission at either 300 or 1200 bps. The price of 212A integrated circuits is coming down so we can expect to see an increasing number of portables with this expanded capability.

Speed of operation

Head-to-head timings on similar tasks, called benchmark tests, are really the only way to compare different computers for speed of execution. Even then, you'll find that one machine that does something faster than another may be slower than the other when doing something else.

Every computer system has an internal clock. The faster the clock, the faster things will happen. Because different computers have different architectures, i.e., are designed differently, the clock speed of one machine can't be compared against the clock speed of another. But you can compare two machines that use the same processor. A 1 MHz 6502, for instance, will always do things faster than a 0.5 MHz 6502.

The CMOS processors found in many portables, while keeping down power requirements, are comparatively slow. If execution speed is an overriding consideration, you should look at those machines that use standard (NMOS) processors.

Real-time clock

Every computer has a system clock, but few have a time-of-day clock. This is one of those things that you either must have or will never use. If you're doing data collection, for instance, and must time-stamp the data as it comes in, the best way to do it is with a real-time clock. If you don't have this clock and still must keep track of the time you can still do it with a program — but that way is much less reliable.

A clock/calendar feature may also be nice for keeping track of appointments, though most people will find it more convenient to do this with a pocket calendar.

Sound generator

Many computer manufacturers, following the original Apple lead, have provided built-in tone generators in their machines. These can make a sort of electronic music which can accompany game programs, but we're not sure what else they're good for. Perhaps, beeps of different tones could be used by a program to signal different conditions during times when the operator is not watching the screen. But we haven't seen any software that makes use of this. If you plan on writing your own software, pick a portable whose sound generator is loud enough to catch someone's attention. (The manufacturer may have decided to reduce the power drain on the batteries by keeping the volume of the sound at a very low level.)

Bar code interface

Supermarkets are finding out that having the UPC bar code on products can cut their costs. Scanning groceries at the checkout line eliminates keying errors by the clerks, speeds up checking out, always provides current prices, automatically tracks sales by item and category, and automatically deducts sales from inventory levels. Similarly, many other items that are handled or tracked manually can be bar coded for more efficient processing.

If your portable doesn't have a bar code interface, you can still add bar code capability by using a reader that talks to an RS-232 interface — but you'll be sacrificing portability and paying more to boot. But just having the interface doesn't mean you're all set to go. What the bar code wand reads has to be interpreted by software — of which there is very little on the market. (See Chapter 10, Inventory/Stock Tracking for HX-20 programs.)

You'll also need good quality bar code labels — a poorly imprinted label cannot be read by the reader. (Fortunately, bar codes have built-in error checking, unlike OCR for instance, so the scanner will either be able to read the bar code or it won't.)

Expandability

In selecting a computer, you get the one that fills most of your needs today. If you know what your needs are going to be tomorrow, then those needs should also enter into your purchasing decision.

If you know that your needs are going to change over time, or if you have no way of foreseeing your future needs, then there are two alternatives. You can select a computer for a particular period of time and replace it with something else when it has outlived its usefulness. Or, you can expand and add onto the computer you select today. The path you take depends on the computer you choose. It depends on the price and on how expandable the computer is. So, expandability to meet future needs (even when you don't know those needs) has to be something to take into consideration today.

The one major thing to look for in determining the expandability of a computer is whether or not the internal system bus is brought to the outside. If it is, then additional hardware can easily be added — at some gain in size and weight. If the system bus is closed off — and this applies to desktops as well as portables — then the computer system is really closed off to future additions.

Figuring out what is the system bus on a particular computer may be a job for an engineer. But a short-cut is to look and see what add-ons are offered by the manufacturer of the computer. If all the add-ons connect into standard I/O ports like RS-232 or Centronics, then you can reasonably assume that this is the only means of connection. On the other hand, if you see add-ons like memory expansion units, then you know at least part of the bus can be connected up externally. Keep in mind that there is probably no room inside the portable to add more circuits — unlike Apple or IBM PC desktops, for instance — so that an external hook-up is critical from the standpoint of expandability.

Software compatibility

Though you may not have realized it, we've already talked about hardware compatibility. Things like RS-232 ports and Centronics printer interfaces allow you to use pre-existing peripherals on the market. Suppose, though, that the portable manufacturer didn't supply an RS-232 port but instead wanted you to use his own proprietary design. Then if you wanted to hook up a modem, you'd have to hook up *his* modem or hope that your computer will be popular enough so that someone else will come out with a compatible modem.

What kind of software can the portable run? (Not just what is available from the vendor.) If your portable can run software written for MS-DOS systems, CP/M systems, the Apple II or the TRS-80 Model III, then you have access to a large variety of business and personal programs. Anything else, and you have to hope that:

(a) the manufacturer is going to supply all the software you will need;
(b) the machine will be popular enough to attract a large number of third-party software publishers;
(c) you will be able to write all of your own software.

Having Microsoft BASIC on the machine can help. Even though BASIC is too slow for many applications, there are some where it's useful — and Microsoft BASIC programs are often not hard to convert from one machine to another.

Earlier, we mentioned compatibility. There are several kinds. There's run-time compatibility and media compatibility, to just name two. Run-time compatibility means that a program which will run on one machine will run on the other. It does not mean that you can pick up a disk (or a tape) from one and plug it into the other — that's media compatibility. So even if your portable can run MS-DOS applications (run-time compatibility) you may have no easy way of actually getting the programs into the machine. A couple of years from now, this may not be a problem as computer stores will have equipment to download, i.e., transmit programs right into your machine — but it's something to consider right now.

Bundled software

Every computer comes with some software included. Usually it's in ROM, sometimes it's on a disk. In earlier years what you got was the BASIC language and/or a rudimentary operating system. Now, the trend is to bundle more and more software in with the hardware — on the theory that it costs the computer manufacturer a lot less to buy the programs in bulk than it would cost you the customer to buy them one at a time. The drawback, of course, is that you may end up paying for programs you don't need.

Usually, you'll get an operating system included with the machine. This is the hardest kind of software to compare because the best operating systems are those that you don't see. That is, if you can get your work done without the operating system getting in the way then you've got a good operating system. It's the operating system that takes care of the mechanics of reading/writing to I/O devices from your programs. Most operating systems provide means to keep track of separate files and programs in memory or on secondary storage; a multi-tasking operating system may even let you run several programs at one time. On portables, look for a HELP facility that will guide you as to what to do next, and possibly, for small screen portables, text compaction/expansion for efficient screen usage.

Another feature worth looking for but which has only recently become popular on desktops and has not filtered down to portables, is the ability to add device drivers. Basically, what this means is that as new hardware add-ons come onto the market, the operating system can be adapted to use them. MS-DOS version 2 and DOSPLUS version 3.5 are two such operating systems.

You're likely to get a programming language with the system too. That language will probably be BASIC. BASIC is relatively easy to learn and is especially useful for 'quick-and-dirty' solutions to problems. Microsoft BASIC, in its various flavours, is the one most often seen on microcomputers (exceptions: Apple, Atari). A program written for one Microsoft BASIC machine, say the IBM PC, will run with only minor changes on another Microsoft BASIC machine, say the HX-20 (source code compatibility). That is, everything else being equal: no graphics, no disk access, etc. Chapter 3, HX-20 BASIC, describes the typical Microsoft BASIC set of statements and variations thereupon. Keep in mind that many portables are squeezed for memory, which means that some BASIC features may be left out.

One feature that Microsoft BASIC doesn't have is program overlay control. Since memory space is at a premium, there may not be enough memory to hold your entire program. The solution is to keep part of the program out on tape and just bring it in when the first part of the program has finished. Some BASICs allow for this and it can be very useful.

Are you going to be doing program development work on the portable? Then look for good development/debugging tools. All Microsoft BASICs have a TRON and a TROFF (trace on/off) and will let you hit break at any time to display variables, but you'll want more than that for serious programming.

Sometimes the BASIC offered will include a compiler. Compiled code runs faster than the usual interpreted code. It also has the advantage of being able to be burned into a PROM so that the operator can run the program without messing with disk or tape. But not all compilers have that capability, and not all portables can use PROMs.

Productivity tools are a favourite for bundled software on a portable. Calculators and calendars show up a lot, but this is not what you bought a computer for. If the portable has a real-time clock and calendar, then look for software that puts them to real use rather than trivial things that are better done with a pen and a pocket calendar.

Word processing may be built-in, as it is on the HX-20 and Radio Shack Model 100. In that case, make sure that it does what you want it to do *and* does not take up memory space that you would have wanted to use for something else. Remember, if a program is on disk or tape it can be removed from the machine in favour of something else. But a factory-installed ROM, unless specifically built to be removable, is not going to be easy to plug/unplug frequently.

If communications is a required function, then having the vendor supply a program to do it is well worth it. Communications programs are usually written in assembly language which means that few people will tackle it. (The HX-20 communications program in this book is written in BASIC, but it's s-l-o-w.)

Data management is another thing to look for. It's possible to cheat and use the word processing program to manipulate data, but a real data management program is better. If you're buying your portable to keep track of information, then look for this program. As a minimum, a good data manager should allow you to perform the following functions:

- search for specific data;
- review data;
- insert/delete/update data.

It's very rare to find vertical application software, that is, programs for your specific business/profession, bundled in with the hardware. An exception to this is the Computone SST for insurance agents. The next best thing is an application generator. This is a program that lets you write programs, theoretically with less work than trying to do it in BASIC. We say theoretically, because application generators have produced mixed results.

Documentation

The quality of the documentation is something that must be judged individually. There is much concern right now in the industry about making machines more 'user-friendly'. Often this takes the approach of making the user manuals more oriented to the novice. That's great, if you're a novice. If you're familiar with computers, however, you'll find yourself wasting time wading through a lot of unnecessary verbiage. Probably the best approach for a vendor to take is the one used by Epson America with its BASIC manuals — separate tutorial and reference volumes.

How do you rate documentation? One thing to try is to sit down with a manual and the computer, without the salesman hovering over you, and try to follow the manual's explanation of some feature. If you can't figure out how to use the feature without calling the salesman over, it's certain you'll have the same problem back in your office.

Ease of use

This is one of those criteria that can be judged only after the computer has been used for a while. But perhaps you can borrow a machine from a dealer or a business associate. Because if the computer is a pain to use, all of the previous criteria will not matter.

How will you be using the computer, physically speaking? On your lap? On a desktop? Can you place the machine so that it is comfortable to use and the display is easy to see?

How easy is the software to use? How easy to program if you plan on doing any programming?

Price

The price range for what we have been calling portables runs from under $400 to over $8000. That huge differential makes it all the more important that you identify your needs and do not spend $4000 when something for $1000 would do, or conversely, that you don't try to save money by buying the $1000 unit when a careful analysis of your requirements would show that only a $4000 unit could do the job.

Few people can afford to buy what they really want, so which features are you prepared to trade off?

Picking a vendor

From whom do you buy your computer? Do you go for the lowest price, or the dealer you guess would support you the best? If you are going to need help in getting started with the computer, pick a local dealer. If you want a computer for a special purpose, look for a dealer who has helped others accomplish that same purpose.

Picking a manufacturer

We're down to the home stretch. There are a few things that relate to the manufacturer himself that we must consider. After all, we're not just buying a piece of equipment, we're also buying the company that stands behind the product. (We hope).

Does the manufacturer support independent software developers? If he doesn't, and you can tell this by seeing what percentage of program titles come from independent publishers, then available software is going to be limited.

Is there a hotline for technical support? If you have a technical question, how fast can you get an answer? Keeping in mind that most dealers will not have trained their staff to be portable experts, you'll have to go to the manufacturer. How will the manufacturer receive you?

Is there a publication dealing with the machine? Are there users' groups where you can go for information and solace?

Perhaps the biggest question is: how many computers is the manufacturer selling? The more units sold, the more of a support industry will grow up around the machine. Selecting a popular machine is more than just going along with the crowd, it's having a crowd coming along with you.

SUMMARY

We can take everything we've covered and put it all into one large shopping list. Then we'd select from the list below those must-have and nice-to-have features we'd want. And then match the various portable computers against the list.

Size
Weight
Display size
Display readability
Keyboard type
Keyboard layout
RAM
ROM
Floppy disk interface
Built-in printer
Serial and parallel ports
Built-in modem
Speed of operation
Graphics
Real-time clock
Sound generator
Bar code interface
Expandability
Software compatability
Bundled software
Documentation
Ease of use
Price
Picking a vendor
Picking a manufacturer

On the following pages are descriptions of 20 more-or-less portable computers and almost-computers. Factual information is drawn from specifications provided by the manufacturers and is believed to be accurate, but its accuracy cannot be guaranteed. Also, product specifications and prices are always subject to change by the manufacturers.

Just for comparison purposes, we're providing a list of these models in order of price, display and weight. This is not a ranking, inasmuch as other features, like software, may be more important to you personally. As additional portables come out, they can be inserted into these lists. **Note** Retail discounts may be available for some models and not others.

Table 1.1
By approximate price

Model	Approx price ($)	Display size	Microprocessor type	Data cartridges?	Weight (lb)	Internal modem
TI CC-40	250	1 × 31	proprietary	no	1.4	no
Panasonic HHC	400	1 × 26	6502	yes	1.4	no

Table 1.1 *continued*

Model	Approx price ($)	Display size	Microprocessor type	Data cartridges?	Weight (lb)	Internal modem
Casio FP-200	500	8 × 20	CMOS 8085	no	3.3	no
Microwriter	500	1 × 16	?	no	1.1	no
MSI-88	775	2 × 16	?	no	1.4	no
HX-20	800	4 × 20	CMOS 6801	no	3.8	no
TRS Mdl 100	800	8 × 40	CMOS 8085	no	3.9	300
NEC PC-8201	800	8 × 40	CMOS 8085	yes	3.8	opt 300
Workslate	900	16 × 46	CMOS 6803	no	3	300
HP-75C	1000	1 × 32	proprietary	-1-	1.6	no
Toshiba T100	1500	8 × 40	Z-80A	yes	16	no
Typecorder	-3-	1 × 40	?	no	3	no
Sharp PC-5000	2000	8 × 80	8088	yes	11	no
Xerox 1810	2200	3 × 80	CMOS Z-80	no	5	300/1200
MOST 100	2500	8 × 80	CMOS Z-80	yes	5	300
Husky	3000	4 × 32	CMOS Z-80	no	4.4	no
Teleram	3000	4 × 80	Z-80L	no	9.75	no
Datec	3500	2 × 40	CMOS Z-80	no	4	no
UDI-500	4000	8 × 40	CMOS Z-80	-2-	12.8	opt 300/1200
Gavilan	4000	8 × 80	8088	yes	9	300
GRiD	8200	24 × 80	8086-8087	no	10.8	300/1200

-1- Uses magnetic cards for data storage.
-2- Contains two disk drives.
-3- Price may be dropped from $1500 to $800.

Since the above chart was originally produced, many prices have changed. Also, some new models have come to market, while some older models have been discontinued. The general trend is that more performance is available for less money than previously, though the relative order of the companies listed above has stayed very much the same. As we've suggested before, probably the best use of this and the following charts is as guidelines for you to make up your own comparison charts.

Table 1.2
By weight

	lb	kg
Microwriter	1.1	0.5
Texas Instruments CC-40	1.4	0.6
MSI-88	1.4	0.6
Panasonic HHC	1.4	0.6
Hewlett-Packard HP-75C	1.6	0.7
Sony Typecorder	3	1.4
Converg. Tech. Workslate	3	1.4

Table 1.2 *continued*

	lb	kg		lb	kg
Casio FP-200	3.3	1.5	MicroOffice 100	5	2.3
EPSON HX-20	3.8	1.7	Gavilan	9	4.1
NEC PC-8201	3.8	1.7	Teleram T-3000	9.75	4.4
Radio Shack Model 100	3.9	1.8	GRiD Compass	10.8	4.9
Datec Electronic Notebook	4	1.8	Sharp PC-5000	11	5.0
Husky	4.4	2.0	Universal Data UDI-500	12.8	5.8
Xerox 1810	5	2.3	Toshiba T-100	16	7.3

Table 1.3
By display size

GRiD Compass	24 × 80	(GRiD)	Casio FP-200	8 × 20	(Casio)
Converg. Tech. Workslate	16 × 46	(Wksl)	Husky	4 × 32	(Husky)
Gavilan	8 × 80	(Gav)	EPSON HX-20	4 × 20	(HX-20)
MicroOffice 100	8 × 80	(MO)	Datec Electronic Notebook	2 × 40	(Datec)
Sharp PC-5000	8 × 80	(Sh)	Sony Typecorder	1 × 40	(Sony)
NEC PC-8201	8 × 40	(NEC)	MSI-88	2 × 16	(MSI)
Radio Shack Model 100	8 × 40	(TRS)	Hewlett-Packard HP-75C	1 × 32	(HP)
Toshiba T-100	8 × 40	(Tosh)	Texas Instruments CC-40	1 × 31	(TI)
Universal Data UDI-500	8 × 40	(UDI)	Panasonic HHC	1 × 26	(HHC)
Teleram T-3000	4 × 80	(Tele)	Microwriter	1 × 16	(MW)
Xerox 1810	3 × 80	(Xerox)			

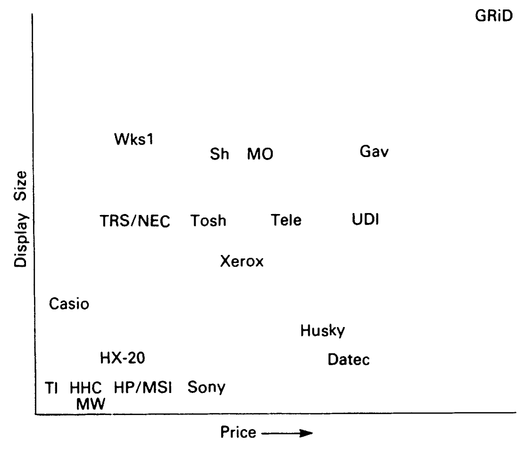


Fig. 1.1 Display size v. price (not drawn to scale)

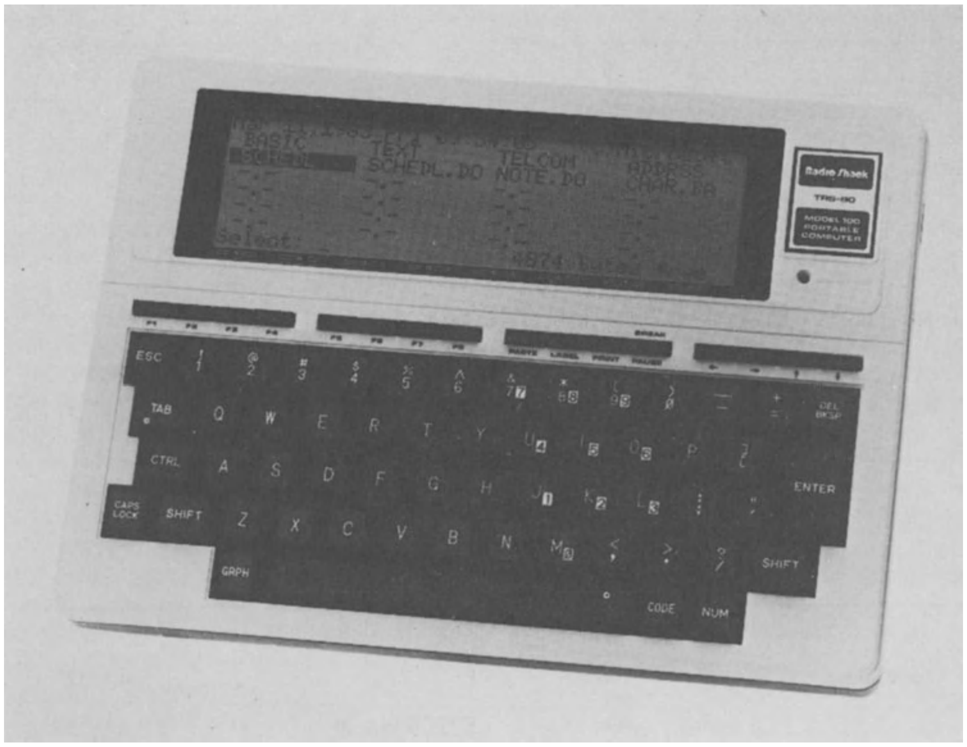


Fig. 1.2 Radio Shack TRS-80 Model 100

RADIO SHACK TRS-80 MODEL 100

Specs at a glance

Price: Under $1000
Size: 8.25 in × 11.6 in × 2 in (210 mm × 295 mm × 51 mm)
Weight: 3.9 lb (1.8 kg)
Keyboard: Full-travel, 72 keys including 8 PF keys, 41 special graphics characters, 32 graphics block characters, 10 key integrated numeric pad
Physical display: LCD, 8 lines by 40 columns
Virtual display: None
Processor: CMOS 8085
RAM: 8–32K
Secondary storage: External cassette
Graphics: 240 × 64 dots
Clock: Yes
Sound: 5 octaves with half tones
RS-232: 75–19,200 bps
Integral modem: 300 bps
Integral printer: No
Bar code reader interface: Yes
Power: 20-hr AA disposable batteries or AC
Software: In ROM: extended Microsoft BASIC, text editor, scheduler, address list, communications

The Model 100 is manufactured by Kyoto Ceramics (Kyocera) of Japan to Tandy's specifications for marketing in the US. (A very similar model is marketed by NEC.)

The Model 100's outstanding features are its relatively large display and its built-in 300 bps modem, which are excellent features for a machine in this price range. This makes the Model 100 a good choice for taking notes on-site for later uploading to another computer. The large screen also makes the Model 100 a better choice for program development than other machines with small screens where so little of the program can be seen at one time. While Model 100 BASIC lacks a built-in editor, programs can be entered and edited with the built-in full-screen text editor.

A Centronics parallel part allows easy hook-up to nearly all of the desktop printers on the market. The serial RS-232 port is also noteworthy: it allows bps rates up to 19.2 kbs. A

ROM cartridge slot is also present, though the necessary cartridges are not being sold by Radio Shack at this time.

Hardware

The disadvantages, hardware-wise, of the 100 are no RAM cartridges, no microcassette drive, in fact, no built-in means of saving data on removable media at all. You can save data onto a separate cassette recorder, but this method has proven notoriously unreliable in the past, both with other Radio Shack machines and with equipment from other vendors. Also: no printer, no virtual display, no provision for hook-up to a larger screen.

The keyboard is only fair, with non-recessed keytops. For touch typists, finding the home keys without looking can be a problem. But many writers seem to have adopted this machine in preference to others on the market.

Software

The extended BASIC looks very nice, providing control of interrupts which is unusual in a BASIC. Radio Shack's four other ROM programs are really only two programs, a text editor and a modem auto-dialler. The text editor is adequate for common usage, keeping in mind that it comes free with the machine. Probably the most marked thing about the Model 100 is that a lot of programmers seem to be interested in writing programs for it. By the time you read this, many of those programs should be out. Look for them in mail-order ads in the micro magazines. (Radio Shack stores sell very little software written by outside developers; other computer stores rarely sell software for Radio Shack machines.)

A hands-on review of the Model 100 appeared in the magazine *Byte* (5/83) and also in *Personal Computer World* (8/83).

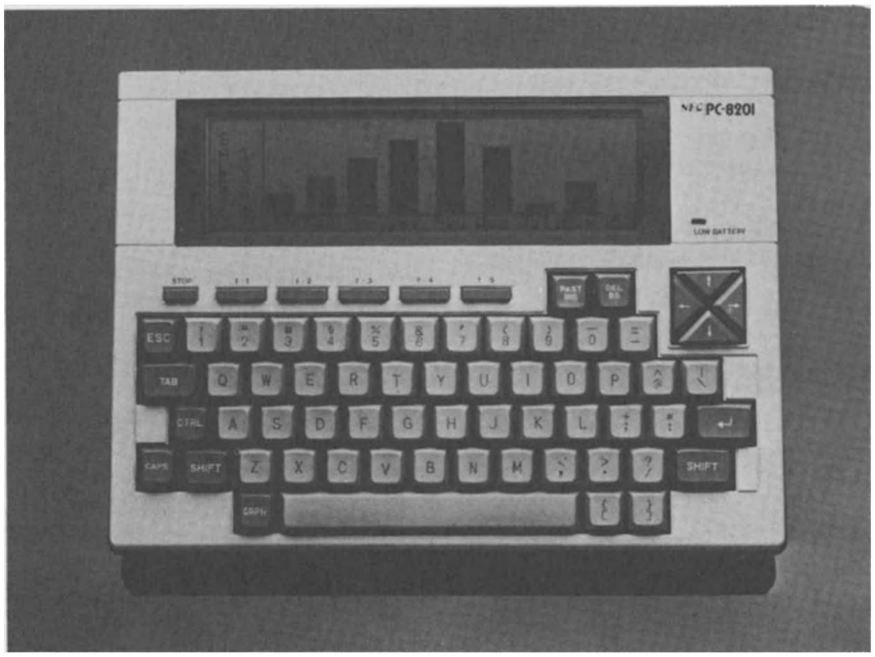


Fig. 1.3 NEC PC-8201

NEC PC-8201

Specs at a glance

Price: Under $1000
Size: 8.25 in × 11.6 in × 2.5 in (210 mm × 295 mm × 64 mm)
Weight: 3.8 lb (1.7 kg)
Keyboard: 67 key typewriter-style, with 5 shiftable PF keys
Physical display: LCD, 8 lines by 40 columns, characters user-definable, interface for TV/-monitor
Virtual display: No
Processor: CMOS 8085

RAM: 16–64K
Secondary storage: RAM cartridges, external cassette, optional floppy disk interface
Graphics: Yes
Clock: Yes
Sound: Yes
RS-232: Yes
Integral modem: Optional 300
Integral printer: No
Bar code reader interface: Yes
Power: AA batteries or AC
Software: ROM: Terminal emulation, extended Microsoft BASIC. On tape: text formatter, investment portfolio, loan evaluator, appointment scheduler, bar code reader, calculator, two games, music composition, RAM bank switch utility, terminal mode selection utility

While NEC has never admitted it, this unit is probably manufactured by Kyoto Ceramics (Kyocera) to NEC's specifications just as Kyocera builds the Radio Shack Model 100 to Tandy's specifications. What NEC has done with the basic hardware, however, is to correct many of the deficiencies in the Radio Shack package.

The biggest plus is the RAM cartridges. Being able to save data and programs on something you can easily plug in and out is an extremely useful option for any computer.

The PC-8201 also has ports for floppies and a larger CRT display, interfaces missing from the Model 100.

Fourteen programs are bundled into the price of the PC-8201, which makes the computer at least somewhat usable right from the beginning. We haven't seen these programs but would guess, given the time frame over which they were written, that they're not very sophisticated. However, programs written for the Model 100 — and we expect to see quite a few — should be runable on the PC-8201 with no more than minor changes. But, you'll have to type them in by hand or download them as the two machines reportedly use different cassette formats.

A full review of the PC-8201 appeared in the magazine *Byte* (6/83).

CASIO FP-200

Specs at a glance

Price: Under $500
Size: 8.6 in × 12.2 in × 2.2 in (220 mm × 310 mm × 56 mm)
Weight: 3.3 lb (1.5 kg)
Keyboard: 69 key full-size keyboard, with integral numeric pad and 5 shiftable PF buttons; external 10 key numeric pad is optional
Physical display: LCD, 8 lines by 20 columns
Virtual display:
Processor: CMOS 8085
RAM: 8–32K
Secondary storage: Interfaces for cassette, disk drive
Graphics: 160 × 64
Clock:
Sound:
RS-232: 300 bps
Integral modem: No
Integral printer: No
Bar code reader interface:
Power: AA batteries or AC
Software: ROM: application generator, BASIC

If you took the NEC notebook computer, cut down on some of the features (and the price as well), you'd end up with the FP-200. The screen is half the size of the NEC or the Radio Shack 100, though twice as large as that of the HX-20. Unlike the NEC, additional RAM cannot be added to the FP-200 in the form of cartridges — at this time, anyway. Like the NEC, a floppy disk or cassette recorder can be used for data/program storage. However, the floppy is just single-sided, single density (70K); the cassette runs only at 300 bps.

Additional peripherals include a four-color mini plotter–printer and a graphics printer.

The FP-200 comes with 'CETL', an interactive table language that is claimed to enable users with no knowledge of BASIC programming to develop their own applications. As of the time we write this, there is no other software available for the machine. Casio's choice of the same 8085 processor used in the NEC/Radio Shack portables should enable it to pick up some of the software written for those machines. But the smaller screen and undoubtedly different I/O interfaces might make the conversion task nontrivial.

All in all, we'd have to say that there is no particular advantage to the Casio unit over the others, except price. If price is the major consideration — and it might be just that for students — then look at the FP-200.

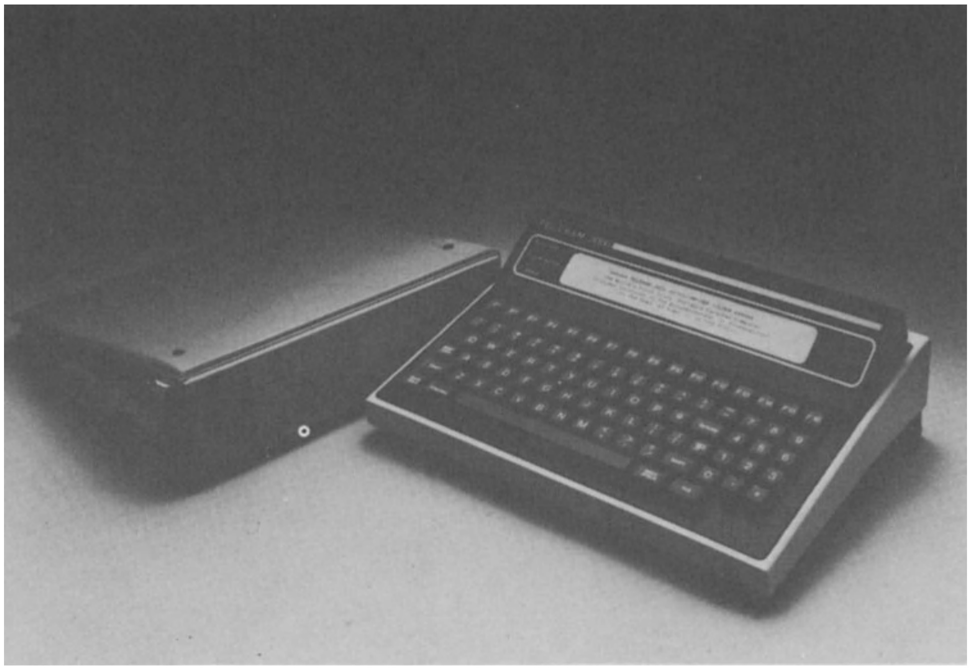


Fig. 1.4 Teleram T-3000 (mock-up)

TELERAM T-3000

Specs at a glance

Price: Under $3000
Size: 9.75 in × 13 in × 3.5 in (248 mm × 330 mm × 89 mm)
Weight: 9.75 lb (4.4 kg)
Keyboard: Full-travel, 83 keys, separate numeric pad, 16 PF keys
Physical display: Tiltable LCD, 4 lines by 80 columns, 160 displayable characters
Virtual display: 24 lines by 80 columns
Processor: Z-80L (low-power NMOS Z-80)
RAM: 64-320K
Secondary storage: optional office station with floppy disk drives
Graphics: No
Clock: No
Sound: No
RS-232: 75-19,200 bps
Integral modem: No
Integral printer: No
Bar code reader interface: No
Power: 5-hr rechargeable batteries or AC or auto cigarette lighter
Software: CP/M 2.2, Crosstalk communications program.

Three things about the T-3000 differentiate it from its lower-priced portable relatives: a full-width (80 column) screen, the popular CP/M operating system, and up to 256K of bubble memory, which can be configured as a pseudo-disk.

Teleram was the first to offer an 80 column screen and this could be an important advantage over many of its competitors — for people who really need a screen that wide.

The CP/M capability opens up a very large variety of possible packaged software. However, you'll have to download these programs into your T-3000 yourself. Since this computer doesn't have removable storage media, the communications port is the only way to get binary data into the machine (except for buying the Office Station).

Teleram's Office Station is a desktop device that provides additional capability, such as a graphics/text CRT, printer port, floppy disk drives, additional memory, more RS-232 ports, networking, etc.

The use of magnetic bubble memory is a nice touch. Programs like Wordstar, for instance, that would normally do a lot of disk I/O on a desktop computer, can be run without any disk delays at all. The reason is that bubble memory, while not as fast as semiconductor memory, is considerably faster than floppy disks.

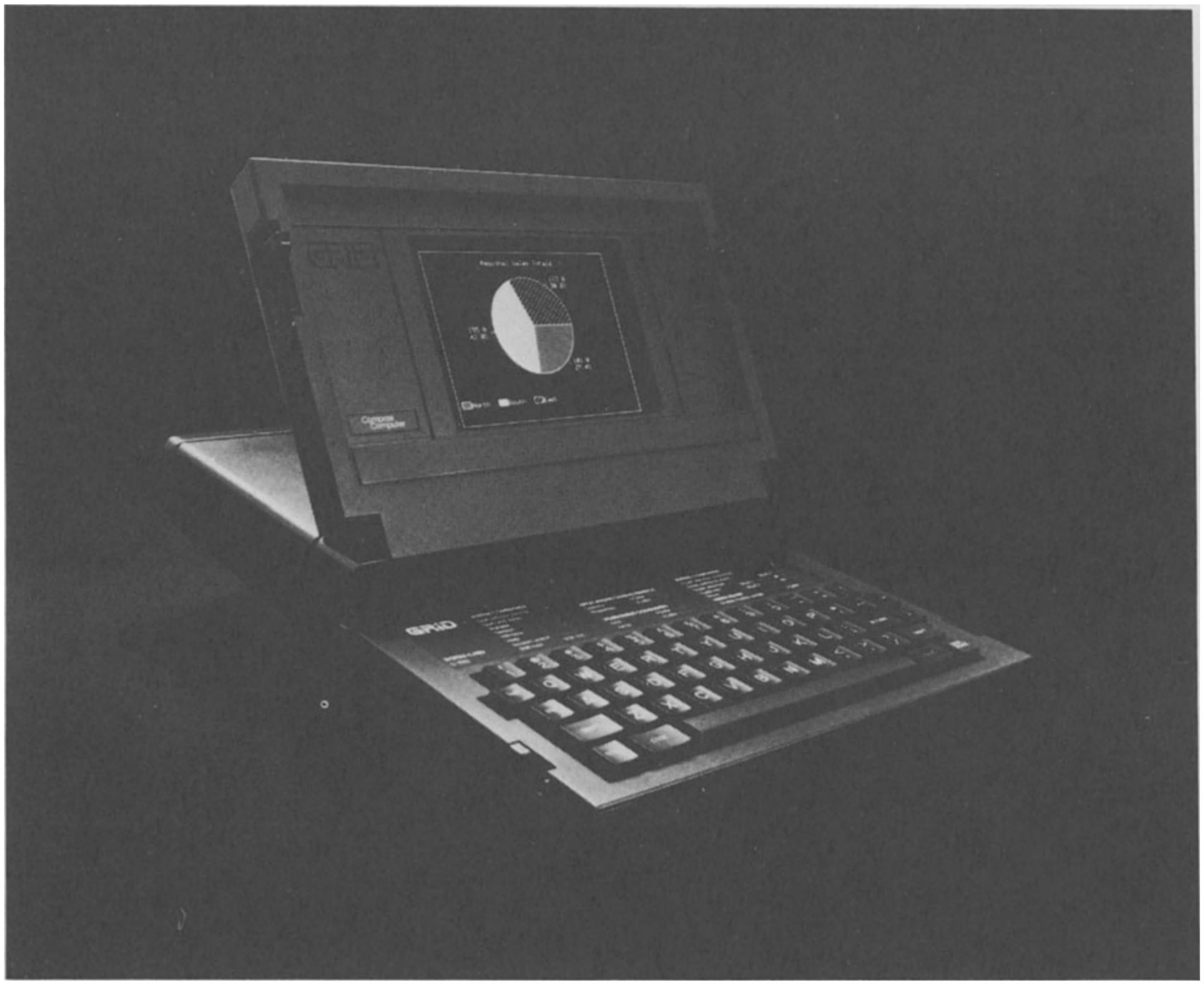


Fig. 1.5 GRiD Compass

GRiD COMPASS

Specs at a glance

Price: $8150
Size: 11.5in × 15 in × 2 in (292 mm × 381 mm × 51 mm)
Weight: 10.8 lb (4.9 kg)
Keyboard: 57 full-travel keys
Physical display: Flip-up EL, 24 × 80
Virtual display: None
Processor: 8086, 8087 arithmetic co-processor
RAM: 256K
Secondary storage: 384K internal bubble memory, floppy/hard disk interface
Graphics: 320 × 240 dots
Clock: Yes
Sound: Yes
RS-232: Yes
Integral modem: 212A (300/1200 bps)
Integral printer: No
Bar code reader interface: No
Power: AC
Software: Optional: MS-DOS compatible operating system, management tools (word processing, spreadsheet, data management, graphics), Microsoft-compatible BASIC, communications (TTY, 3101, VT100), program development tools

The Compass has become generally recognized as the Rolls Royce of portables. For those to whom price is no object, e.g., upper level management at large corporations and government agencies, there is nothing comparable.

Advantages include:

- an electro-luminescent (EL) display, the clearest and easiest-to-read of any display technology;
- a central support system at GRiD headquarters (GRiD Central) that can download software to your Compass;
- networking via a Compass Central desktop system
- high-speed microprocessors
- a variety of interfaces, including RS-422 and GPIB

That's all in addition to the usual features of a portable, such as light weight, convenient size, full keyboard, etc.

There are disadvantages, of course. This is not a field machine — it requires line power (AC) at all times. Even if you were to connect a battery with a DC–AC converter, it would drain in no time at all.

The overall appearance is aesthetically pleasing — even snazzy. I can't think of anyone who uses a computer who wouldn't like to have one — even people who don't use computers are impressed when they see it. But at a price of over $8000, it's a purchase most managers will find hard to justify. If your company bought you an HP calculator watch for $750 when they first came out, maybe you can put one of these on your expense account. If not, . . .

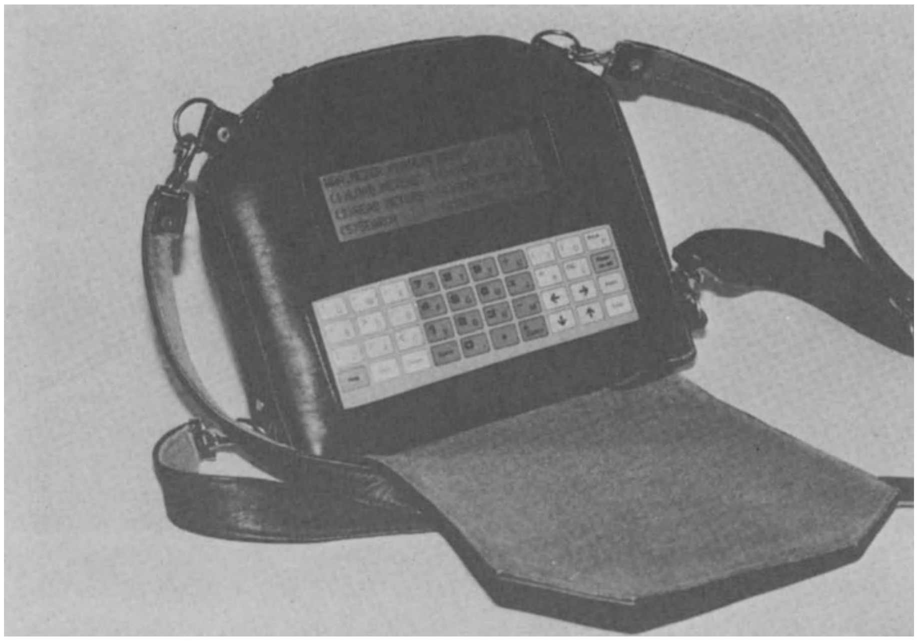


Fig. 1.6 Husky

HUSKY

(UK: DVS Microelectronics; US: Sarasota Automation)

Specs at a glance

Price: Under $3000
Size: 8 in × 9.5 in × 1.75 in (203 mm × 241 mm × 44 mm)
Weight: 4.4 lb (2 kg)
Keyboards: Membrane-type with audio feedback, 40 'soft' keys
Physical display: LCD, 4 lines by 32 columns
Virtual display: No
Processor: NSC800 (CMOS Z-80)
RAM: 16–144K
Secondary storage: None
Graphics: No
Clock: Yes
Sound: No
RS-232: 50–1200 asynch./synch.
Integral modem: No
Integral printer: No
Bar code reader interface: Yes

Power: 14-hr rechargeable NiCad batteries or 45-hr C cells
Software: BASIC, communications protocols (including TTY, 2780), bar code reader (Code 39, EAN 8/13)

A unit designed for rugged use, the Husky comes in an unbreakable, waterproof, shock-proof cast alloy case. This is not a general purpose office computer, but a machine for dedicated applications.

The Husky can be customized to the user's needs. The keyboard is completely redefinable, for instance. An optional parallel port for electronic instrumentation is available. Custom software is offered by both DVW and by Sarasota Automation.

One thing worth noting that might be otherwise missed is the capability for synchronous transmission — a feature not found in most other portables.

The Husky has found use in Britain for scientific data collection, vehicle tracking, inventory control, and meter reading, among other uses. The British Army and the Royal Air Force have also found it useful for collecting maintenance data.

How rugged is it? One user reports that his Husky was squashed flat by an airport truck — but it continued to work without any data loss (PCW 10/83).

Sarasota Automation, which distributes the Husky in the US for DVW Microelectronics, intends to begin the manufacture of the unit themselves.

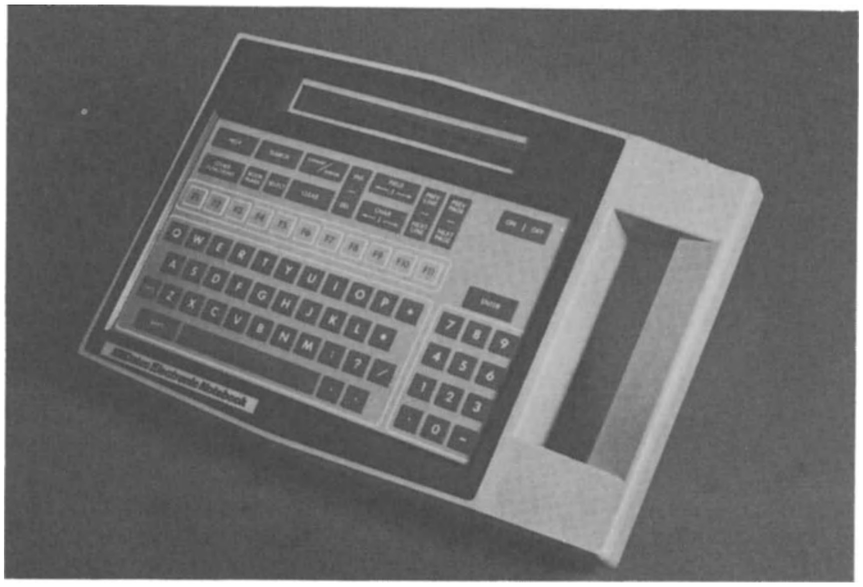


Fig. 1.7 Datec Electronic Notebook

DATEC ELECTRONIC NOTEBOOK

Specs at a glance

Price: Under $3500
Size: 9 in × 14 in × 2 in (229 mm × 356 mm × 51 mm)
Weight: 4 lb (1.8 kg)
Keyboard: Raised membrane with audio feedback, 79 keys including numeric pad and 11 PF keys.
Physical display: LCD, 2 lines by 40 columns, optional backlighting
Virtual display: None
Processor: NSC800 (CMOS Z80)
RAM: 48–504K
Secondary storage: None
Graphics: None
Clock: Yes
Sound: No
RS-232: 2 ports, asynch./synch., 300–9600 bps
Integral modem: No
Integral printer: No
Bar code reader interface: Yes
Power: 16-hr rechargeable NiCad batteries or various external power sources, AC and DC

Software: Multi-tasking OS, data entry executive, application generator (usable only with a separate terminal), BASIC interpreter only with a separate terminal)

The Datec unit is geared towards a particular market niche: outdoor, 'hostile' environment use. The ABS plastic case with built-in handle is claimed to be a completely sealed, immersible unit: resistant to shock, vibration, moisture, dirt, sand, grease, oil, etc. For customers needing this type of capability, Datec will provide a development package and will even write customized software. The keyboard can be customized as well, with a 50 PF key layout substituting for the normal one. Normal operating temperature is 0 °C to 50 °C (32 °F to 122 °F), and can be optionally extended.

The EPROM based operating system provides, among other things: multi-tasking, intertask communications, interrupt handling, realtime clock control, file and record management, operator-definable transmission protocols.

The unit also comes with a Data Entry Executive which includes a TTY compatible terminal simulator, a HELP function, a SEARCH function, date editing functions, display compaction/expansion of text, relational data management, data and program transmission to another computer.

The application generator permits a novice user to define a data entry application by answering questions about data format and relationships.

The amount of memory that the unit can hold is quite high, which could make it better for fast access to larger databases than on other portables. However, at the time of writing, the cost for each additional BK of RAM is $215.

The Datec Electronic Notebook is an enhanced version of a prevous unit from Data Entry Systems Corp. (Descor) called the Electronic Book.

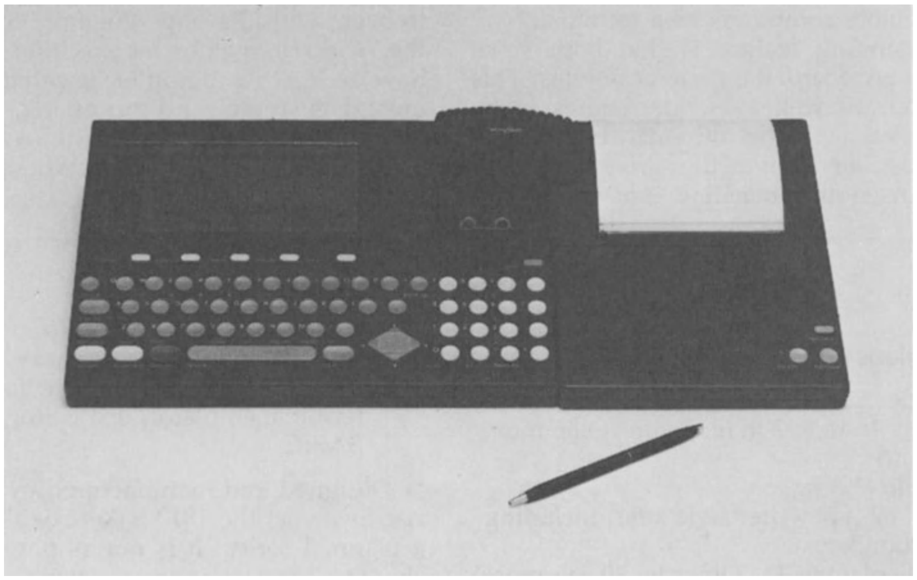


Fig. 1.8 Convergent Technologies Worklsate

CONVERGENT TECHNOLOGIES WORKSLATE

Specs at a glance

Price: Under $1000
Size: 8.5 in × 11.25 in × 1 in (216 mm × 286 mm × 25 mm)
Weight: 3 lb (1.4 kg)
Keyboard: 60 button-type keys including separate numeric pad
Physical display: LCD, 16 lines by 46 columns
Virtual display: Yes
Processor: 6303 (CMOS 6803)
RAM: 16K
Secondary storage: Microcassettes
Graphics: No
Clock: Yes
Sound: Yes
RS-232: Optional external 9600

Integral modem: Auto-dial, auto-answer 300 bps
Integral printer: Optional
Bar code reader interface: No
Power: AA batteries or A/C adaptor or optional NiCad rechargeable batteries
Software: ROM: Terminal emulation, worksheet program with spreadsheet, memo pad, appointment calendar, phone list functions

The Workslate, first of a series of portable computers from Convergent Technologies designed to meet specific needs, can be described as a computerized tool for business-people.

Rather than being a true general purpose computer, the Workslate, as it has been described by one company spokesman, is a replacement for a computer. The Workslate is an easy to use tool for doing spreadsheets, keeping track of appointments, taking notes at meetings, holding addresses and phone numbers, and tying into remote computers as a terminal.

One outstanding feature is that both voice and data can be recorded on microcassettes. This allows spoken as well as written notes to be saved. Coupled with the auto-answer modem, it's even possible to use the Workslate as a telephone answering machine: answering the phone and playing back a recorded message. With the auto-dial function, you can even initiate calls, using the Workslate as a speaker phone.

The built-in clock with alarm capability allows tasks to start up automatically at particular times of the day. The optional battery-powered printer attachment gives 4 color printing with 40 or 80 column width, plus a 9600 bps serial interface.

Taskware microcassette tapes provide predesigned worksheets, similar to spreadsheet templates. These help in setting up worksheets for personal tax, sales reporting, financial statements, cash management, etc.

The only disadvantage that we can see in the design of the Workslate is the limited amount of user memory. The Workslate has a lot of functionality built into it, but does not appear to have enough memory to support very many functions simultaneously. Using memory for appointments or memos will subtract from the amount of memory available for spreadsheets — and 16K does not allow much of a spreadsheet to begin with. Perhaps the most common use of the Workslate will be for executives who already have access to a computer, use that computer for most data storage and processing, and just carry around part of their data in the Workslate.

A hands-on review of the Workslate appeared in the magazine *Byte* (11/83).

XEROX 1810

Specs at a glance

Price: Under $2200.
Size: 9 in × 16 in × 2 in (229 mm × 406 mm × 51 mm)
Weight: 5 lb (2.3 kg)
Keyboard: 67 typewriter style keys, including 10 function keys
Physical display: LCD, 3 lines by 80 columns, interface for TV or monitor
Virtual display:
Processor: NSC800 (CMOS version of Z-80)
RAM: 64K
Secondary storage: microcassette tape recorder, plug-in ROM modules, optional base station with floppy disk drives
Graphics: Yes
Clock: Yes
Sound: No
RS-232: Yes
Integral modem: 300/1200 bps
Integral printer: No
Bar code reader interface: No
Power: 10-hr NiCad batteries or AC
Software: Calendar, four-function calculator, terminal emulator, text editor, Microsoft BASIC.

Designed and manufactured by the new Sunrise Systems, the 1810 is Xerox's first portable in a planned series. It is not as physically impressive on first glance as some of the other machines on the market, probably due to the relatively small screen. But behind the wide packaging are hidden some very nice features.

The built-in microcassette recorder can handle either data or voice. Up to 200K bytes of data can be saved, or 30 minutes of ordinary voice recording. Besides recording to tape, the external microphone can also be used to talk on the built-in phone. With an RS-232 port plus a modem plus a phone, the 1810 really has communications covered.

Theoretically, with its Z-80 compatible pro-

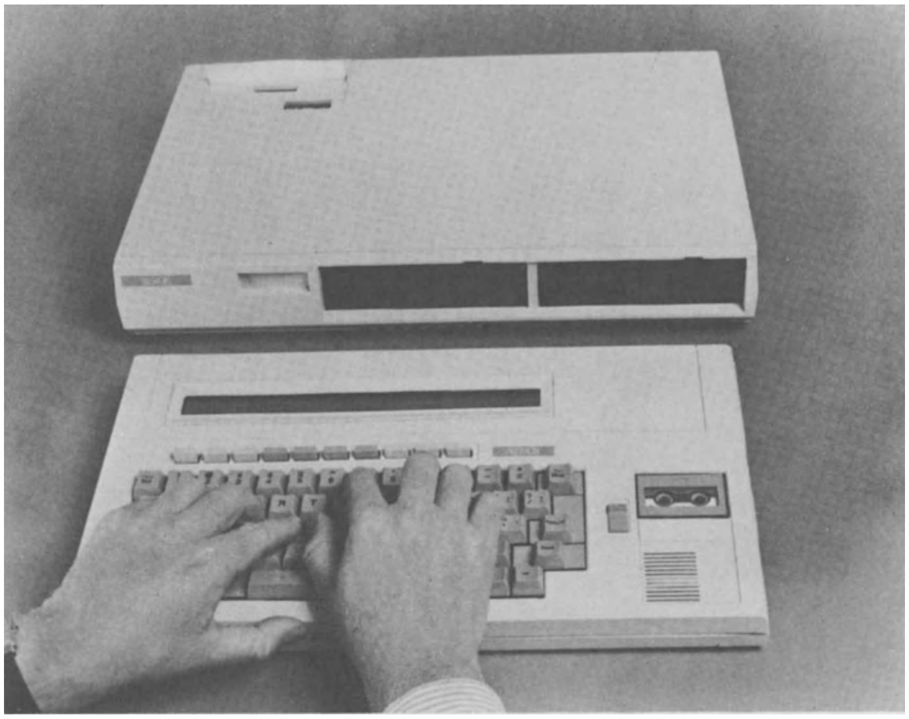


Fig. 1.9 Xerox 1810, with the 1850 base station

cessor and full 64K of RAM, the 1810 is compatible with the universe of CP/M software. We say theoretically, because most CP/M software expects a cursor-addressable 24 line screen. How this will be handled on the 1810 remains to be seen, literally.

With ROM modules, users have the advantage of having programs always ready to use, i.e., no time-consuming loading of tapes nor worries about tape readability.

A base station, the 1850, is available as an accessory to the 1810. The 1850 is a dual processor (Z80 and 8088), dual floppy box with 128K of RAM expandable to 512K. Additional I/O ports allow an RGB color monitor, printer, and more communications options. Add a display and you have a computer capable of running CP/M, CP/M-86 or MS-DOS software. The price for the 1850 is $2495.

The 1810 is also available without a display for $1595 as the 1805.

There are now several companies, including Teleram and Toshiba, making computers that can be used as either straight portables or as full desktop computers, with no sacrifice of function in either case. This could be the way to go for anyone who splits his computing between in-office work and out-office work.

A hands-on review of the Xerox 1810 appeared in the magazine *Byte* (6/83).

GAVILAN

Specs at a glance

Price: Under $4000
Size: 11.4 in × 11.4 in × 2.75 in (290 mm × 290 mm × 70 mm)
Weight: 9 lb (4.1 kg)
Keyboard: Full-travel, 60 keys including numeric pad, plus touch pad
Physical display: Flip-up LCD, 8 lines by 80 columns, zoom feature, video monitor interface
Virtual display: None
Processor: 8088
RAM: 32–160K
Secondary storage: 3 in (76 mm) microfloppy disk drive or RAM cartridges
Graphics: 64 × 400

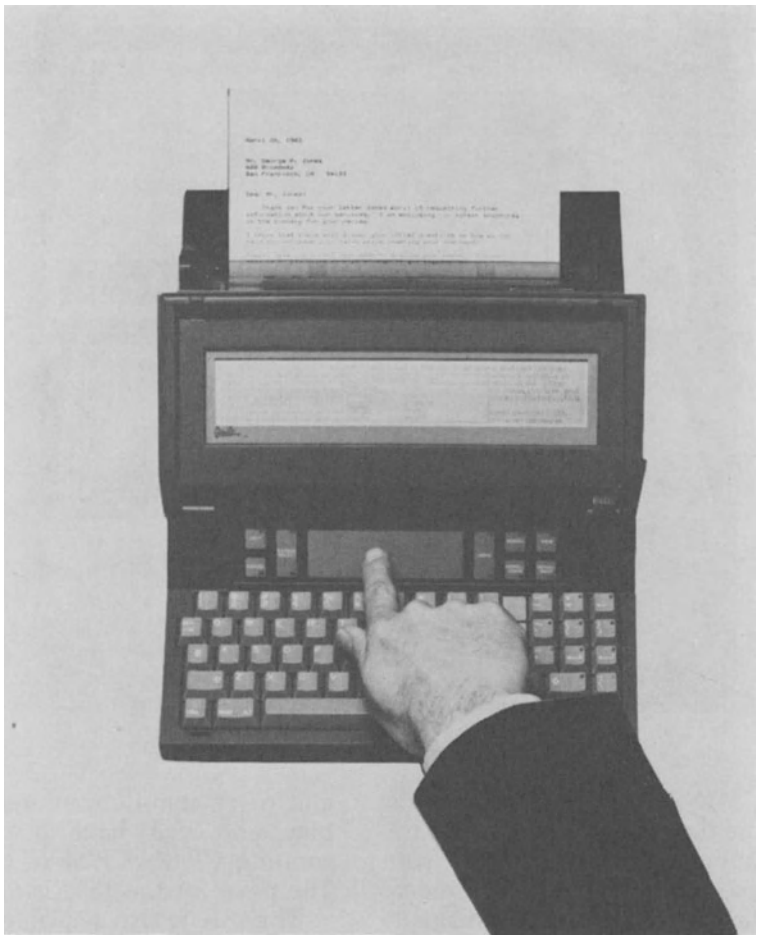


Fig. 1.10 Gavilan

Clock: No
Sound: No
RS-232: Up to 9600 bps
Integral modem: 300 bps
Integral printer: Optional
Bar code reader interface: No
Power: 8-hr rechargeable batteries or AC or auto cigarette lighter
Software: MS/DOS operating system, BASIC language, word processing, calculation, communications, forms processing, portable secretary. Optional: application development system, other applications packages, Pascal language

The Gavilan is a cross between notebook units like the HX-20 and briefcase types like the Osborne. At the same time, it offers some unique features not found even on many larger machines. The touch pad, for instance, is a Lisa-like device that provides fingertip control of a simulated desktop. The LCD display has a reverse zoom feature that allows the user to look at the format of an entire text page, then move the physical window to edit any part of that page. An in-context help command provides instructions on what to do next or tells you what just happened.

The 8088 processor allows the Gavilan to run MS/DOS, the same operating system that runs on the IBM PC and many other machines. Note, however, that it may be hard to locate MS/DOS application software on the 3 in (76 mm) Hitachi-type diskettes. Also, the limited amount of memory for a 16-bit machine plus less-than-standard screen size will prevent some MS/DOS applications from running.

A special portable printer plugs into the back of the unit. (The 50 cps printer weighs 5 lb (2.3 kg) and has its own battery.)

While Gavilan Computer Corp. is a new company, it is run by some old hands, including the former head of Zilog, maker of the ubiquitous Z-80 microprocessor. Gavilan has apparently had no trouble obtaining venture capital funding, and as a consequence, should be around for a while.

All in all, this machine looks like a winner — incorporating many features that non-technically oriented managers and professionals will appreciate. If the sticker price doesn't scare you away, we'd recommend you make a careful examination of the machine.

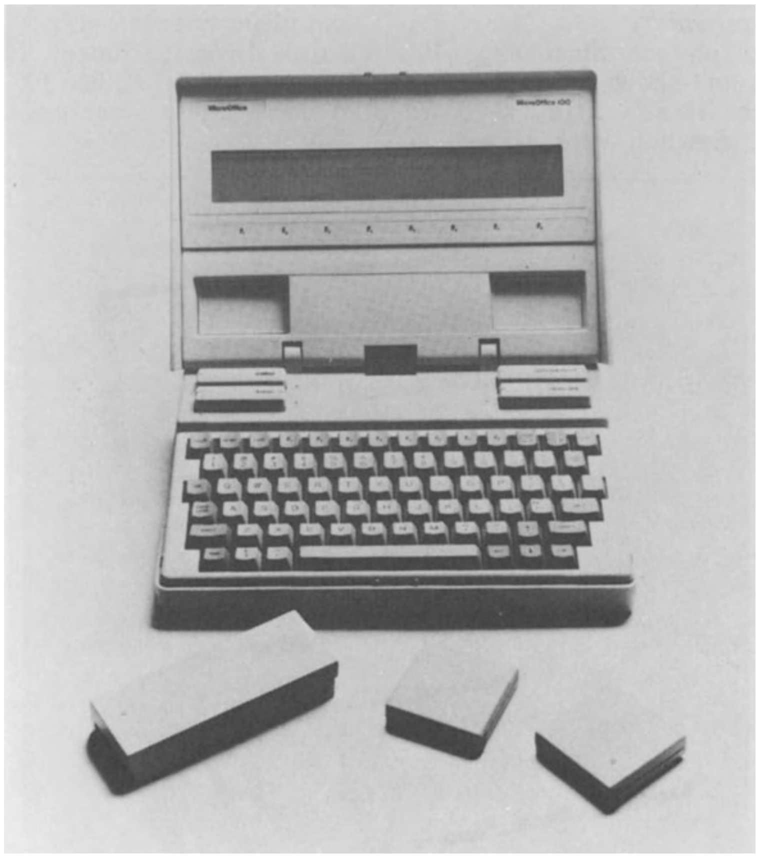


Fig. 1.11 MicroOffice 100

ROADRUNNER/MICROOFFICE 100

Specs at a glance

Price: Under $2500
Size: 7.8 in × 11 in × 3 in (200 mm × 285 mm × 74 mm)
Weight: 5 lb (2.3 kg)
Keyboard: 73 full-size keys, with 8 PF keys
Physical display: Tiltable LCD, 8 lines by 80 columns
Virtual display:
Processor: NSC800 (CMOS Z80)
RAM: 48K
Secondary storage: Four battery-powered cartridges, each holding 32K ROM or 8–16K RAM
Graphics: 64 × 480
Clock: No
Sound: No
RS-232: Yes
Integral modem: 300 bps
Integral printer: No
Bar code reader interface: No
Power: 8-hr rechargable, removable NiCad battery pack

Software: CP/M-compatible operating system
  Optional ROM packs:text editor, spreadsheet, Microsoft BASIC, communications, scheduling, SuperCalc spreadsheet, more promised

This is another in the office-away-from-the-office class of portables. MicroOffice Systems Technology Inc. is a new company that seems to have put the right pieces together with this portable, their first product.

Opening the lid on the MicroOffice 100 powers the unit on and shows a compact, but well-laid out design. There's a full keyboard with programmable function keys as well as special function keys — HELP, UNDO, etc. — plus separate cursor keys. Cartridge slots for holding data and programs are easy to get to. The screen is the state of the art at the time the product was announced (in September 1983) — 8 × 80. It's readable and appears to use the typical 5 × 7 dot matrix without lower case descenders.

A 37-pin connector is present to allow for future expandability. The manufacturer suggests that this connection may eventually be used for a disk drive attachment, but stresses that the owner of a MicroOffice 100 really doesn't need to mess with disks or tapes as the cartridges take their place.

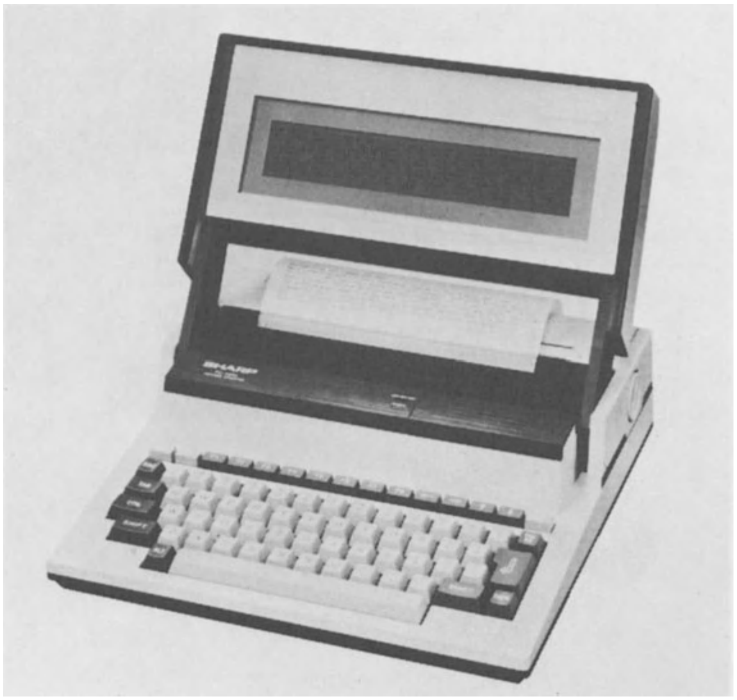


Fig. 1.12 Sharp PC 5000

SHARP PC-5000

Specs at a glance

Price: Under $2000
Size: 12 in × 12.8 in × 3.4 in (305 mm × 326 mm × 87.5 mm)
Weight: 11 lb (5 kg)
Keyboard: 71 full-travel keys, including 8 PF keys
Physical display: Flip-up LCD, 8 lines by 80 columns
Virtual display:
Processor: 8088
RAM: 128–256K
Secondary storage: 128K bubble memory cartridges, 64K RAM cartridges, disk interface (320 Kbyte minifloppies), cassette interface
Graphics: 640 × 80 dots
Clock: Yes
Sound: Yes, 3-octave range
RS-232: Yes
Integral modem: No

Integral printer: Optional
Bar code reader interface: No
Power: Rechargeable NiCad batteries or AC
Software: BASIC, MS-DOS. Optional: word processing, communications, spreadsheet, executive planner, data manager

As with their pocket computers, Sharp seems to have put a number of good features in a reasonably-sized package.

Memory is no problem with the Sharp PC-5000. You can put data and programs on bubble memory cartridges or on RAM/ROM cartridges. That's in addition to having up to 256K of RAM built-in.

The Sharp PC-5000 is physically larger and heavier than most diskless portables. And that's without adding the optional 80-column thermal/plain paper printer. But the availability of the printer (and the disk interface) means that you have nearly full desktop capability in a package considerably lighter and less bulky than the Osborne-style portables.

No information is available on how long the batteries will go between recharges, but we'd guess: not very long. The 8088 processor is not a CMOS processor, which means that it will drain power considerably. All that memory will also be a drain. But that 8088, and that memory, and MS-DOS, gives the PC-5000 an advantage over other portables: the capability of running software written for IBM PC class machines.

The PC-5000 has another interesting feature: an open bus structure. This doesn't provide any immediate benefit to most users, but it gives hardware designers a means of interfacing additional functions right into the machine. That turns into a long-range benefit for users, assuming that the PC-5000 becomes popular enough to interest the add-on hardware companies.

Sharp's optional modem is worth a look. It not only has autodial via software in the PC-5000, it can also be dialled via a keypad built into the modem. There is also a speaker and microphone for conference calls.

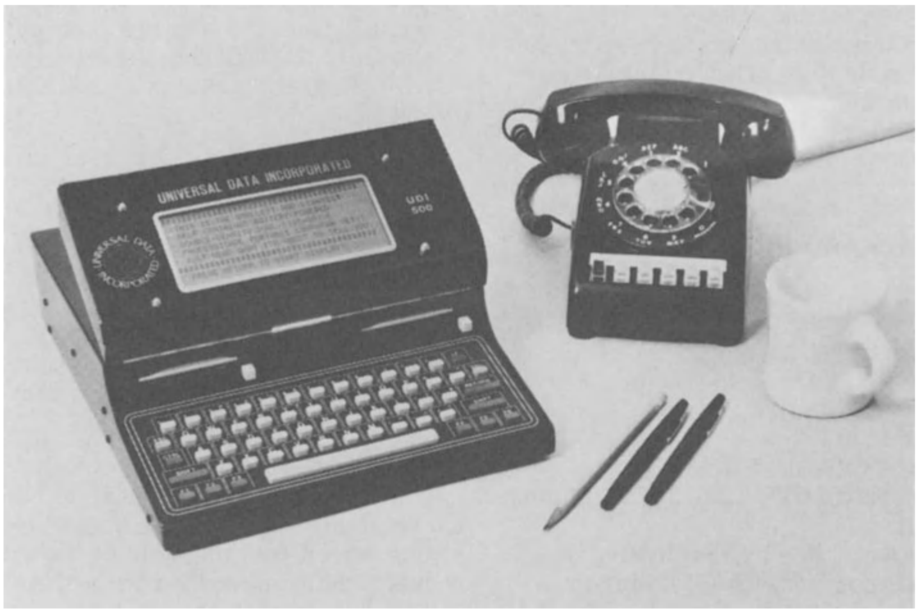


Fig. 1.13 Universal Data Inc. UDI 500

UNIVERSAL DATA UDI-500

Specs at a glance

Price: under $4000
Size: 11 in × 13 in × 3.1 in (280 mm × 330 mm × 79 mm)
Weight: 12.8 lb (5.8 kg)
Keyboard: 59 button-type keys, including 6 PF keys
Physical display: LCD, 8 lines by 40 columns
Virtual display: Not at the present time
Processor: swappable — CMOS Z80 and

CMOS 1805 currently available
RAM: 64–256K
Secondary storage: dual 3.5 in (90 mm) Sony microfloppy disks, double-sided disks (1.5 Mbyte total capacity) optional
Graphics: not at present time
Clock: No
Sound: No
RS-232: 50–19, 200 bps
Integral modem: Optional 300 or 1200 bps
Integral printer: No
Bar code reader interface: No
Power: 12-hr (average disk use) NiCad batteries or AC
Software: With Z80 card — optional software includes Perfect software series (word processing, spelling checker, data filing, spreadsheet), communications, Microsoft or Digital Research BASIC, CP/M operating system. With 1805 card: DOS and BASIC.

With two built-in disk drives, the UDI-500 is more like a desktop 'transportable' than the others we've discussed, but its relatively light weight and freedom from a power cord put it squarely in the portable class.

The UDI-500 is the first portable we've heard of with 2 microfloppies. That reduces weight in comparison to an Osborne-type unit. Unfortunately, microfloppy disks are even less standardized than minifloppies — at least minifloppies are all 5.25 in (133 mm) — so there's no telling whether your local dealer will ever have the software you want on 3.5 in (90 mm) diskettes. While UDI promises to copy customer's programs onto the microfloppies, there is no guarantee that this is a permanent policy. Of course, software can always be downloaded.

This is another computer with a CMOS version of the popular Z-80 microprocessor. This provides compatibility with the CP/M world while still preserving the low-power requirements of a portable.

Like other true portables, memory contents are retained even with the power switch off — the advantage of CMOS chips again. If programs and data are used without disks, then up to 80 hours of computer time can be obtained without recharging.

The UDI-500 comes with two slots for circuit boards. A microprocessor card takes up one slot, another can be used (in conjunction with the 1805) for any standard RCA accessory card.

Universal Data, which has been in the hand-held terminal business for several years, also manufactures a hand-held computer somewhat similar to the MSI unit described in this chapter, with 32–176K memory and BASIC programmability.

HEWLETT-PACKARD HP-75C

Specs at a glance

Price: Under $1000
Size: 5 in × 10 in × 1.25 in (127 mm × 254 mm × 32 mm)
Weight: 1.6 lb (0.7 kg)
Keyboard: 64 calculator-style keys
Physical display: LCD, 1 line by 32 columns, TV interface
Virtual display: 1 line by 96 columns
Processor: Proprietary CMOS, similar to HP Series 80
RAM: 16–24K
Secondary storage: Magnetic card reader, external digital cassette, plug-in ROM modules
Graphics: On optional external plotter
Clock: Yes
Sound: Beeper
RS-232: Optional external
Integral modem: No
Integral printer: Optional external
Bar code reader interface: Optional
Power: Rechargeable NiCad batteries
Software: Internal ROM: BASIC language
Optional ROM modules: Surveying and maths, VisiCalc, text formatting, data communications, others

The HP-75C was one of the first portable computers, coming out at a time when true portability meant being hand-held and a one-line screen was the state of the art. Technology has rapidly moved on from there, but the HP-75C still has some virtues not found in other portables.

Hewlett-Packard designers have never worried about being compatible with the rest of the world. That shows in the HP-75C. But it's something that HP's customers — mostly engineers — don't seem to mind. The HP-75C comes with an HP-IL connection rather than the usual RS-232 port, though external conversion is possible. But this makes it compatible with other

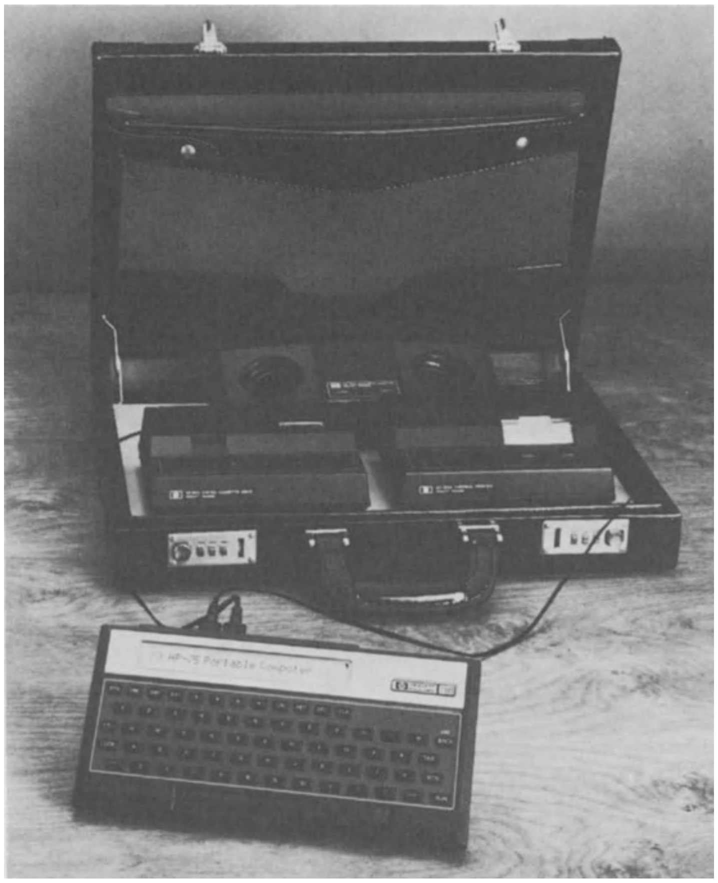


Fig. 1.14 Hewlett Packard HP-75

HP peripherals such as plotters, printers, modems.

The magnetic cards, each holding 1.3K, can be used to store data or programs. No other portable manufacturer has opted to use mag cards — they were last commonly seen about 10 years ago on IBM word processing typewriters — but they serve the purpose of providing convenient and reliable storage.

HP-produced software is provided on digital cassettes or on ROM capsules, as well as on mag cards. The ROM modules easily plug in and out of three separate ports, allowing up to 48K of additional read-only memory.

A hands-on review of the HP-75C appeared in the magazine *Byte*(9/83).

TEXAS INSTRUMENTS CC-40

Specs at a glance

Price: $250
Size: 5.75 in × 9.25 in × 1 in (146 mm × 235 mm × 25 mm)
Weight: 1.4 lb (0.6 kg)
Keyboard: 69 limited-travel keys, including a separate numeric pad and 10 integrated PF keys
Physical display: 1 line by 31 columns, optional TV/monitor interface (24 × 40)
Virtual display: 1 line by 80 columns on LCD
Processor: Proprietary CMOS (99xx series)
RAM: 6–18K
Secondary storage: External wafer tape drive
Graphics: No
Clock: No
Sound: No
RS-232: Optional external
Integral modem: No
Integral printer: No

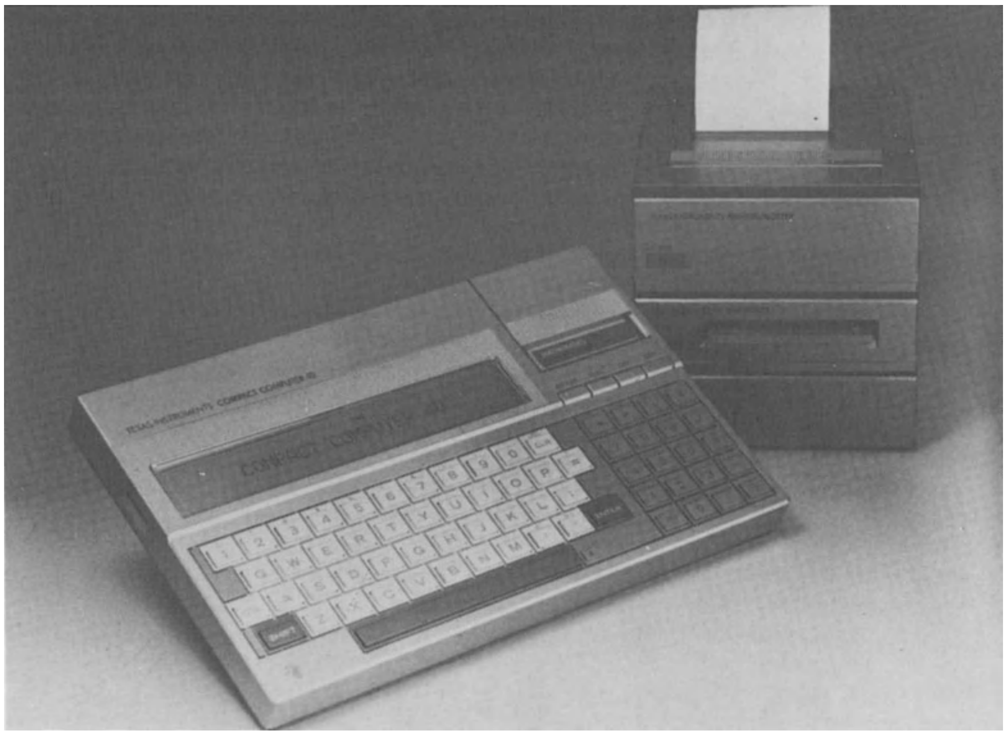


Fig. 1.15 Texas Instruments CC-40

Bar code reader interface: No
Power: 200-hr AA batteries or optional AC
Software: BASIC, optional applications software on ROM cartridges and wafertapes

Like HP and Panasonic, TI has opted to come out with a compact unit that can be hooked into a large number of separate TI peripherals. This means of connection is called the Hex-bus. Existing peripherals include: video interface, modem, printer/plotter, wafertape digital tape drive, RS-232 interface. Except for the video interface, all can be battery powered. But, of course, all add to the weight of what starts out to be a very lightweight unit.

One outstanding feature of the CC-40 is the use of the digital tape. This unit, which is similar to the well-regarded but little known Exatron Stringy Floppy, holds up to 48K. Software packages available on 'Wafertape' include photography, maths, engineering, statistics, inventory, and more.

ROM cartridges can add up to 128K of preprogrammed software. ROM programs available from TI include word processing, an assembler, and games.

In our opinion, there are three reasons to buy this computer: there is a program available for it that you need and can't find elsewhere, you already own Hex-bus peripherals or your budget is extremely limited. The CC-40 holds little attraction for anyone else.

MSI-88

Specs at a glance

Price: $775
Size: 7.75 in × 3.6 in × 2.25 in (197 mm × 91 mm × 57 mm)
Weight: 1.4 lb (0.6 kg)
Keyboard: 28 key, calculator-style
Physical display: 2 lines by 16 columns
Virtual display: None
Processor:
RAM: 16–128K
Secondary storage: None
Graphics: No
Clock: Yes
Sound: No

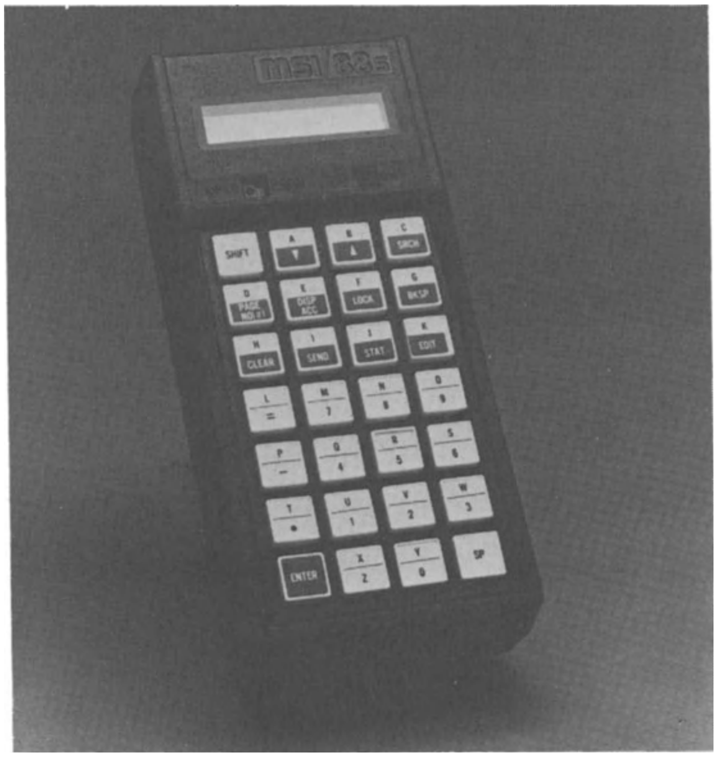


Fig. 1.16 MSI/88s

RS-232: Up to 4800 bps
Integral modem: Optional
Integral printer: No
Bar code reader interface: Yes
Power: Disposable or rechargeable batteries (rechargeable via AC or auto)
Software: Communications, program loading
Optional: bar code including UPC, EAN, Interleaved 2 of 5, 3 of 9, Codabar, Code 11

MSI has long been a manufacturer of hand-held terminals for inventory control. The Model 88, introduced in 1980, was the company's first step towards a computer, but it's not a complete step. While the Model 88 can be downloaded with a program from another computer, it does not support stand-alone user programming. Unlike general purpose computers that are designed as an extension of the office, the MSI-88 is intended to be used by its operator for a specific, pre-assigned task such as inventory control, meter reading, route accounting, etc.

The arrangement of the keyboard in a 7 × 4 matrix keeps the size of the unit down. This does not aid data entry, though, as the operator must press SHIFT for any alphabetic characters.

Standardized and/or custom applications are available from MSI.

PANASONIC HHC

(Manufactured by Matsushita. Similar models distributed by Olympia and Quasar.)

Specs at a glance

Price: Under $400, but options can bring price to over $2000
Size: 3.75 in × 9 in × 1.2 in (95 mm × 228 mm x 30 mm)
Weight: 1.4 lb (0.6 kg)
Keyboard: Calculator-style, 65 keys, including 7 PF keys
Physical display: LCD, 1 line by 26 columns, external color TV adaptor available
Virtual display: Up to 80 columns by a

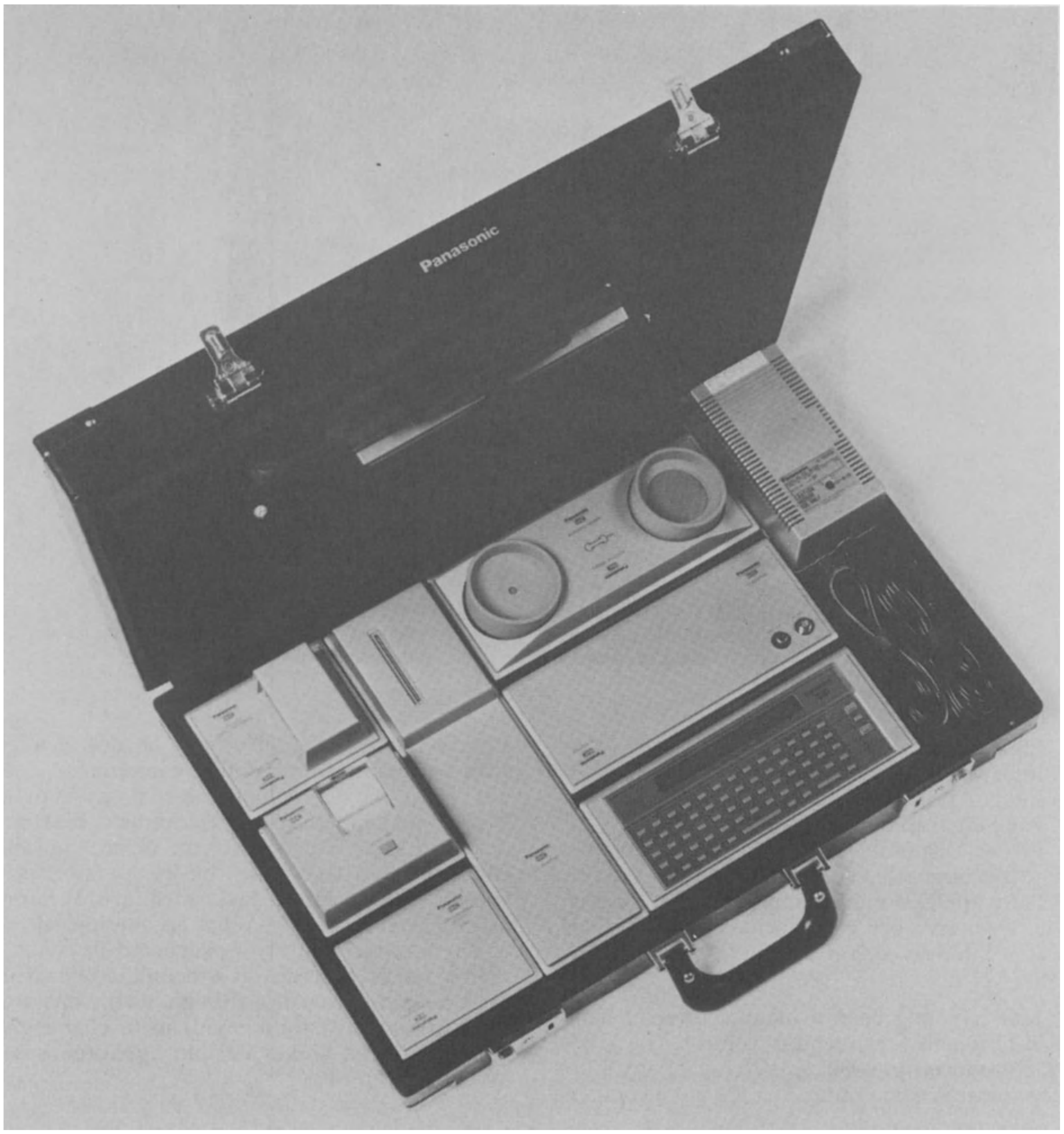


Fig. 1.17 Panasonic HHC

variable number of lines (limited by memory)
Processor: 6502
RAM: 8K – 52K externally via cartridges
Secondary storage: RAM cartridges
Graphics: No
Clock: Yes
Sound: No

RS-232: External adaptor, 50–9600 bps asynch
Integral modem: Optional external
Integral printer: Optional external
Bar code reader interface: No
Power: rechargeable batteries or AC adaptor
Software: Operating system with file management
Opt: ROM capsules for communications,

SNAP FORTH, scientific calculator, BASIC compiler/interpreter, financial calculator, statistics, application generator, time billing, plotter graphics, text editing

We've included this hand-held computer in our survey because it can be purchased with enough peripherals to fill a suitcase. In fact, you can buy a specially designed attaché case to hold color plotter, RS-232 interface, printer, modem with cassette adaptor, RAM cartridges, TV adaptor, I/O adaptor, AC adaptor.

The HHC allows the use of battery-powered RAM cartridges as well as ROM 'capsules'. As we pointed out earlier, the RAM cartridges help keep the HHC user from running out of memory. Unlike users of other portables who eventually must store the contents of memory on cassette or upload it to another system, the HHC user can simply plug in a new RAM cartridge for each new function.

The plug-in ROM capsules are also a nice touch: you can set up your machine so that it runs the software you select rather than wasting memory on something you don't really need. It also saves loading time, of course, to have your program always in ROM rather than having to be loaded from cassette. If you have access to an Apple, the EPROM Writer package plus an EPROM 'burner' will allow you to store your own applications in ROM.

Despite more software and peripherals available than any other under $1000 portable, sales of the HHC are reportedly slow. Perhaps it looks and feels too much like a calculator and not enough like a computer. But with the amount of functionality provided, plus customization available, this unit should be well suited to a number of different applications.

In mid-1983, Panasonic announced a new lap-size computer. However, production plans for this machine have currently been shelved.

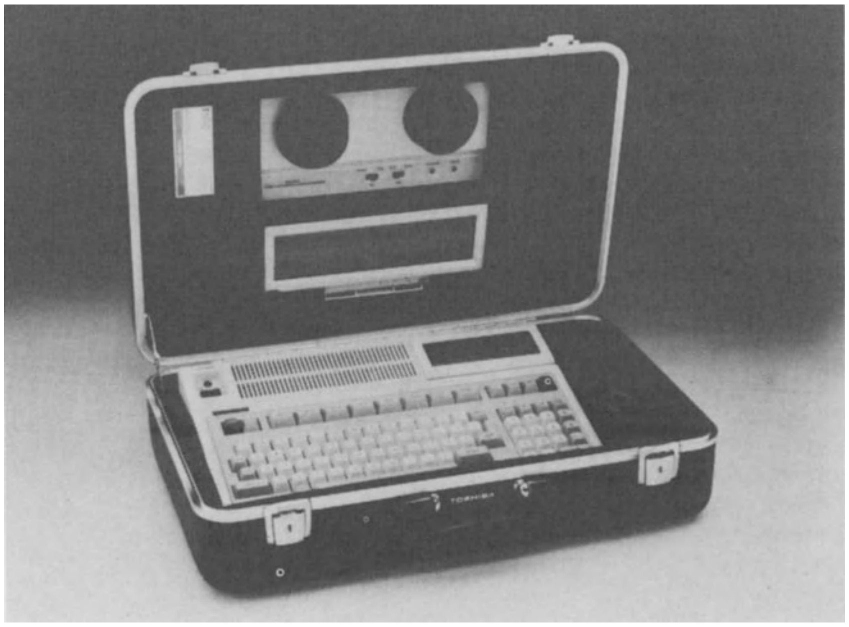


Fig. 1.18 Toshiba T-100

TOSHIBA T100

Specs at a glance

Price: Under $1500
Size: 11 in × 16.5 in × 4 in (279 mm × 419 mm × 102 mm) + display: 5.5 in × 10 in × 1.5 in (140 mm × 254 mm × 38 mm)
Weight: 16 lb (7.3 kg)
Keyboard: Full-travel, 89 keys including separate numeric pad, 8 PF keys
Physical display: Detachable LCD, 8 lines by 40 columns, video monitor interface
Virtual display not known
Processor: Z80A
RAM: 64K

Secondary storage: RAM/ROM cartridges, cassette interface, floppy disk interface
Graphics: 64 × 320
RS-232: Yes
Integral modem: No
Integral printer: No
Bar code reader interface: No
Power: AC
Software: BASIC language, CP/M with optional disks

The Toshiba approach is quite a bit different from that of other companies making 'lap' portables. Rather than designing a portable from scratch, they took their desktop computer and made a portable out of it.

The advantage is: you can take your desktop computer with you, without having to cart around something with the bulkiness and weight of a briefcase-sized 'transportable'. (The disks and the CRT can be left back in your office.) Many portables are designed to be used with another computer — the T100 is all-purpose.

The disadvantage is: AC line power is required. Unlike battery-powered computers, no attempt was made with the T100 to reduce electrical usage: the T100 uses no CMOS chips and runs its Z80A processor at the full rated speed.

But the Toshiba does have one leg up on many portables: RAM and ROM cartridges. RAM cartridges, which can hold up to 32K, allow the user to save programs and data without bothering with tape. The 32 Kbyte ROM cartridges provide a means to run application or system software easily, though such software is not currently available.

A Centronics parallel port, in addition to an RS-232 port, allows hook-ups to external printers and modems.

A hands-on review of the T100 appeared in the magazine *Creative Computing* (11/83).

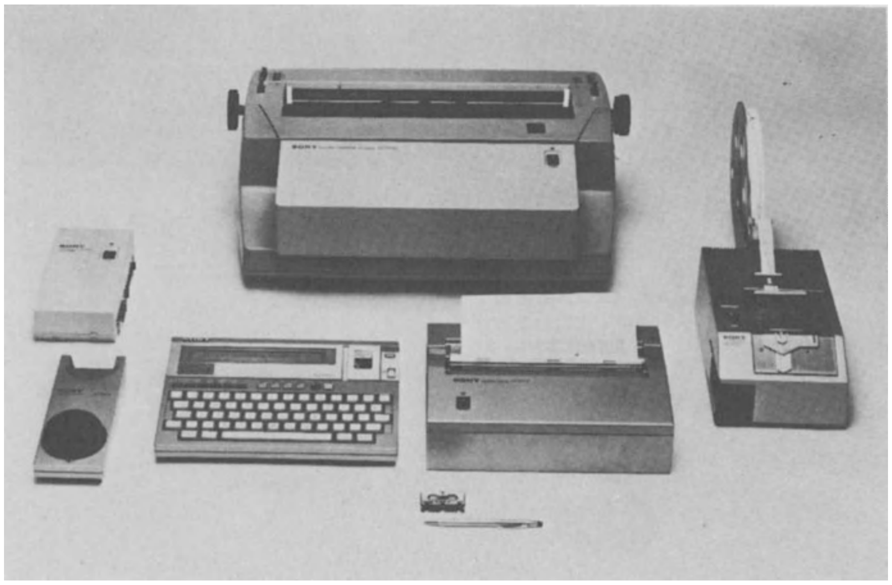


Fig. 1.19 Sony Typecorder and attachments

SONY TYPECORDER

Specs at a glance

Price: Under $1500, but may be reduced to $700
Size: 8.5 in × 11 in × 1.5 in (216 mm × 279 mm × 38 mm)
Weight: 3 lb (1.4 kg)
Keyboard: 70 typewriter-style keys
Physical display: LCD, 1 line by 40 columns
Virtual display: 1 text page
Processor not known
RAM: 2K
Secondary storage: Microcassette drive

Graphics: No
Clock: No
Sound: No
RS-232: Yes
Integral modem: No
Integral printer: No
Bar code reader interface: No
Power: 5.5-hr AA batteries or rechargeable batteries or car battery or AC
Software: Word processing only

The Sony Typecorder is not quite what you think of when you think of a portable computer, but it's certainly more than a terminal.

Outwardly, the Typecorder resembles an HX-20. It might not be too much a stretch of the imagination to say that the HX-20's designers were influenced by the physical design of the Typecorder. But the Typecorder is a relatively old machine, having been introduced in December 1980. As a consequence, this single-purpose note-taking machine lacks many of the features found in today's generation of portable computers. The only reason to look at one now is that it comes with a built-in voice-recording facility. Yes, the same microcassette tape that can handle typed input will also record voice. That's probably a useful enough feature for some people for them to overlook the Typecorder's other deficiencies. (Check the descriptions of the Convergent Technologies and Xerox portables for a similar capability.)

The Typecorder provides the following text functions: margins, tabs, erase line, delete/insert character, scrolling, store/retrieve, 'steno' keys, and a page/line number display.

Add-ons are available: acoustic coupler, portable printer, office printer, electric typewriter actuator, telex tape puncher.

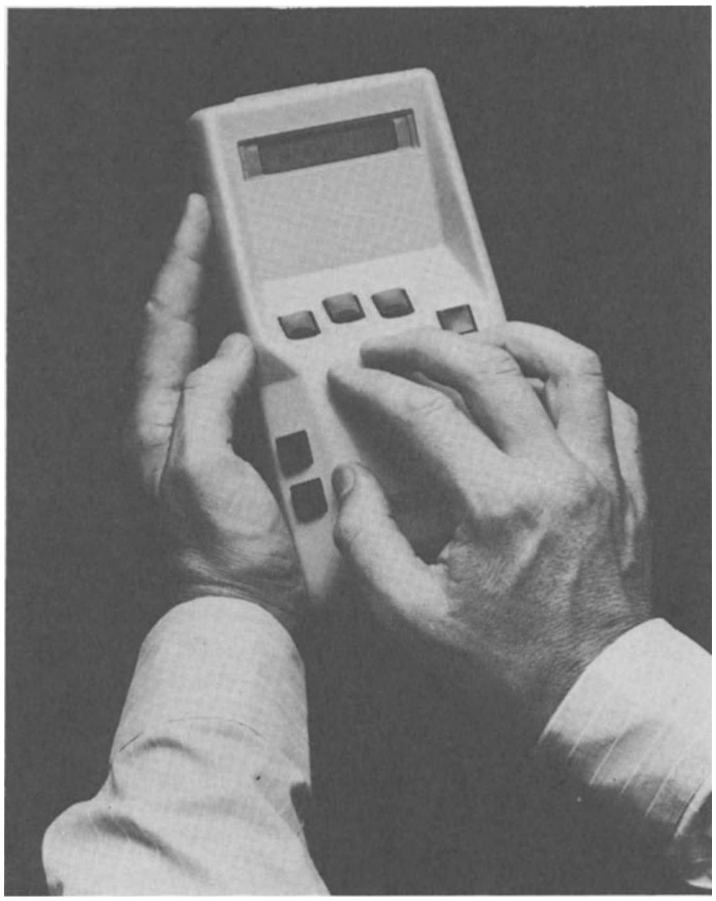


Fig. 1.20 Microwriter

MICROWRITER

Specs at a glance

Price: Under $500
Size: 4.5 in × 9 in × 2 in (229 mm × 114 mm × 51 mm)
Weight: 1.1 lb (0.5 kg)
Keyboard: 6 button keys
Physical display: LCD, 1 line by 16 columns
RAM: 8K
Secondary storage: cassette interface
Graphics: No
Clock: No
Sound: No
RS-232: Yes
Integral modem: No
Integral printer: No
Bar code reader interface: No
Power: 30-hr rechargeable NiCad batteries
Software: Word processing only

And now for something. . . The layout of the universally used QWERTY keyboard was said to derive from the designer's wish to slow down the typist. A good typist could cause the early typewriters to jam by typing too fast. So, a keyboard was laid out that required much finger movement. Now, with electronic keyboards, there has been more and more talk of replacing QWERTY with something else. Microwriter is definitely something else.

Microwriting is based on the shape of the letters of the alphabet, which the user forms with the fingertips of one hand. A sixth key, hit with the thumb, adds editing controls.

Like most portables, the idea is to enter text for later transmission/printing. For some, it succeeds admirably. Others will look for a larger screen and the ability to run a variety of programs.

Designed by an American (in 1978), it was first manufactured and marketed in Britain (in 1982) by Microwriter Ltd.

2

THE HX-20

'Any given program will expand to fill all of available memory.' (*With apologies to Parkinson*)

This chapter covers:
- Introduction to The HX-20
- The Hardware Components: Processor, Memory, Display, Keyboard, Printer, Real-Time Clock, Tone generator, Power Supply, Bar Code Interface, RS-232 port, High-Speed Serial Port, Microcassette Drive, Cassette Interface, PROM Cartridge Interface, Floppy Disk Interface, TV Interface, Memory Expansion Unit
- Environmental considerations
- Documentation and technical support
- Warranty
- Prices

INTRODUCTION TO THE HX-20

Specs at a glance

Price: $800
Size: 8.5 in × 11.375 in × 1.75 in (216 mm × 290 mm × 44 mm)
Weight: 3.8 lb (1.7 kg)
Keyboard: Full-travel, 68 keys including 5 shiftable PF keys, 42 special graphics characters, integral numeric pad
Physical display: Tiltable LCD, 4 lines by 20 columns
Virtual display: Yes, maximum in any one dimension is 255 characters, total size limited by available memory
Processor: Dual 6301s (CMOS 6801)
RAM: 16–32K
Secondary storage: Built-in microcassette
Graphics: 32 × 120 dots
Clock: Yes
Sound: 4 octaves, with half tones
RS-232: 110–4800 bps
Integral modem: No
Integral printer: Yes
Bar code reader interface: Yes
Power: 50-hr rechargeable NiCad batteries
Software: In ROM: extended Microsoft BASIC, text editor (US)

The first thing that strikes the observer is that the HX-20 is an all-in-one unit. Keyboard, display, tape, printer — all in one compact package. While a plethora of other portables have entered the market after the HX-20, none of them has this combination.

Our HX-20 has held up well — no signs of wear even after a year of use. Like other Epson products, reliability is first class. We can report no major electronic or mechanical breakdowns. And only one minor one — a looseness in the tilt knob for the LCD.

PROCESSOR

The heart of any computer system is its microprocessor(s). The HX-20 comes with two: dual Hitachi 8-bit 6301s. The 6301 is a CMOS (low-power) version of the 6801 processor, with some additional features. The 6801 in turn, was an enhanced version of the original Motorola 6800, a chip used in many early hobbyist microcomputers.

The main microprocessor (mpu) controls the keyboard, LCD, RS-232 transmission, high-speed serial interface, bar code reader interface, and clock. The slave mpu controls the external

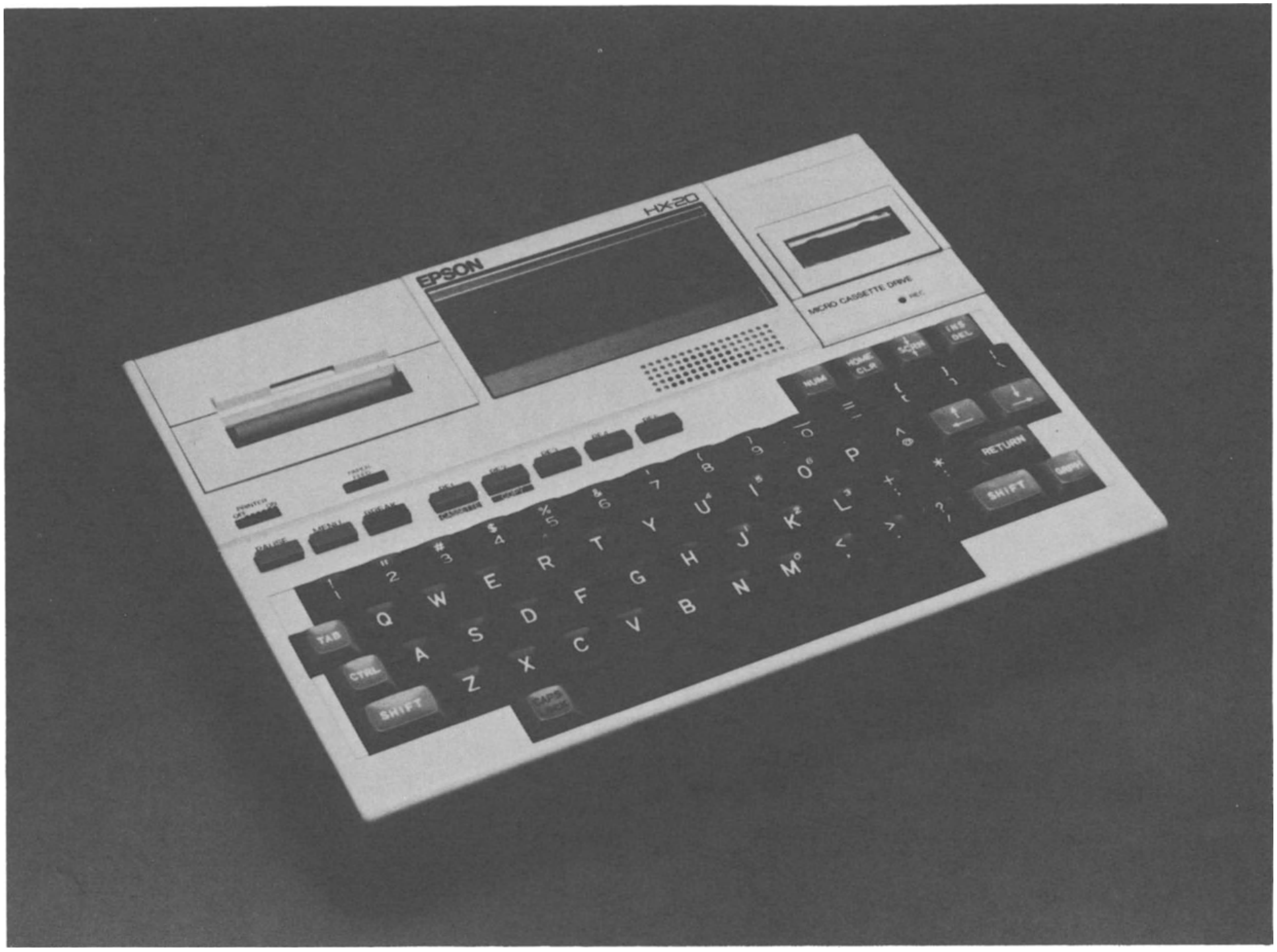


Fig. 2.1 Epson HX-20

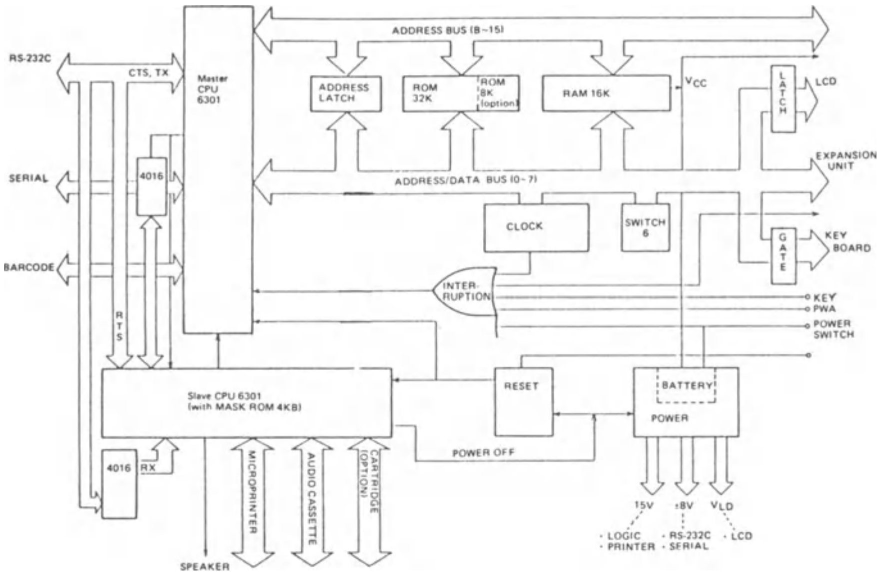


Fig. 2.2 Main circuit board
Reprinted courtesy of Epson Corp

cassette interface, microprinter, speaker, and RS-232 reception. Control of the microcassette or PROM cartridge is handled jointly by both processors. The two mpus 'talk' to each other over a serial bus, separate from the main system bus.

More information on the HX-20's internal operations can be found in Chapter 5, The 6301 Microprocessor.

MEMORY

The HX-20's internal memory, also made up of CMOS chips, is divided into ROM and RAM. The operating system takes 16K of ROM, BASIC takes another 16K of ROM. An additional 8K ROM chip can be plugged in internally. In the US, this slot is occupied by the SkiWriter word processing package. Of course, other programs can be placed in this slot. Epson America recommends that a dealer makes this installation because CMOS chips are very sensitive to static electricity. But a careful user can make the installation himself. Check your warranty terms, though, before opening the case.

Besides the up to 40K of ROM, the HX-20 also contains 16K of RAM. But a little more than 2.5K of this is reserved for data used by BASIC and the operating system, so the user is really only left with 13.5K. Additional memory, ROM and/or RAM, can be added via an expansion unit, described later in this chapter.

User memory in the HX-20 is cut up four ways. The size of each of these sections can be reduced to zero (or close to it) or expanded so far that they will crowd out the other sections.

One section is reserved for use by machine language programs. This area runs from the low end of the user memory up to a value that you set, via the MEMSET statement in BASIC. (Use of the MEMSET statement tells BASIC not to use this part of memory.)

Another section is reserved for what Epson calls 'RAM files'. Unlike data stored in BASIC variables which are cleared out before a BASIC program is re-run, data stored in RAM files remains there until you purposely clear it. (**Note** The HX-20's memory is always provided with power, even when the 'power' switch is 'off'.) RAM files can also be shared among several BASIC programs — more on that in Chapter 3, HX-20 BASIC.

The screen display also steals from user memory (RAM) and the larger the virtual screen display the less RAM is available for other things. (The concept of the virtual screen is explained later in this chapter.)

The fourth use of memory is, of course, by BASIC programs. This includes both the program itself and any data it acts upon.

The limitation of 16K internal RAM on the

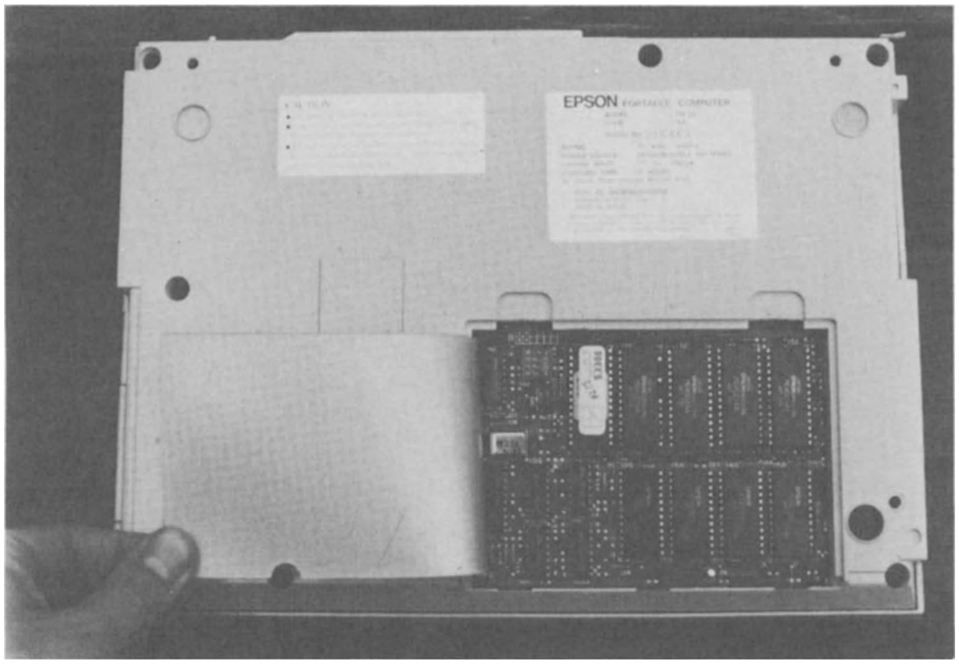


Fig. 2.3 Internal RDM in the HX-20. On this unit, the spare ROM slot is occupied by HCCS Forth. The 6301 processors sit in the covered area to the left of the exposed ROM section

HX-20 is one that users will have to work around. Undoubtedly, future models will have more RAM, but, in the meantime, HX-20 users will typically find that they are reloading programs frequently.

DISPLAY

When the HX-20 was first introduced, an LCD display of 4 lines by 20 characters represented a big improvement over previous computers. Only the Teleram T-3000, first shown at about the same time, used an LCD screen to provide a larger display, though at a much higher price. But LCD technology has improved so rapidly that larger and larger displays are becoming commonplace.

The small size of the screen is partially overcome via the concept of the virtual, or logical, screen. (Software text compression can also be used.) On the HX-20, the virtual screen is an area of memory set aside for display purposes. The physical screen then acts as a window over this larger area.

If you've used a spreadsheet program, you're familiar with the window concept. The part of the spreadsheet you can see is a window on the entire spreadsheet. If display windows don't mean anything to you, just think of looking out of a car window at a scenic view. As you drive the car along, you can see more of the scenery, but you can't see any more than what is framed in the window.

You may feel like sticking your head out the window to see more, but there's no way you can do that on an HX-20. But you *can* drive the car back and forth and even up and down. You could set the virtual display, for instance, to 255 lines by 20 columns. Then, using the SCRN, HOME or cursor keys you can look at different 4 × 20 character 'pages' of the display. You could also, for instance, set the virtual screen size to 24 lines by 80 columns, the size of a standard terminal.

This introduces the concept of horizontal scrolling — where the screen must be moved sideways in order to see an entire line. This is hard to get used to. If you're like most people, you'll set the width to 20 and just change the number of virtual lines.

Each character on the display is made up of a pattern of dots, up to 5 dots across by 7 dots down, with 1 dot separating each character from the next. Despite the fact that this is really a minimum resolution — lower case characters do not have descenders, for instance — it's really not hard to read at all.

Besides 'printable' ASCII characters, the screen also displays 32 special pre-defined graphics characters. An additional 32 characters can be defined by the user, usually via a program that must be reloaded periodically. Each graphic character is composed from a 6 × 8 dot matrix. This is not a lot of dots, so each graphic character can't be too fancy, but the supplied ones are mostly recognizable.

Each of the dots mentioned above is individually programmable. This provides a graphic display of 120 × 32 dots. On the HX-20, graphics is written to a different hardware buffer than text. Both can be superimposed on the same physical screen and both can be acted upon — even cleared — independently of the other. (Not so with the TV display mentioned below.)

Internal switches allow various international character sets to be displayed — the British pound sign, instead of the US pound (#) sign, for instance. Again, this is designed to be a vendor operation, but a careful user can make the change. Or, you can change character sets temporarily by changing certain specified memory locations, e.g., using the POKE statement in BASIC. Note that adding these special characters causes the loss of other characters. The German character set, for instance, will lose the at sign, square brackets, tilde, etc. But the user always has the option of defining his own characters, as mentioned above. The character set options are: USA, France, Germany, England, Denmark, Sweden, Italy, Spain.

All characters are generated via software, from data stored in the ROM. At this time, only one other ROM is offered: for the Katakana character set.

A somewhat fragile adjustment knob allows the LCD display to be slightly tilted to provide an optimum viewing angle. This is an important feature for any LCD display, as this type of display is only viewable when looked at head on.

Provision has been made in the basic unit for an external display, via a high-speed serial port. When available, this interface could drive a TV, producing a 16 × 32 text display, 128 × 96 monochrome graphics, or 128 × 64 color graphics (4 colors).

Other displays for the HX-20 may be introduced in the future. Displays are available mono and color from Oval Automation Ltd, which can produce 80-column monochrome alphanumerics, a keystroke, line, block and symbol graphics only. It also offers a 40-column mode and an HO-20 emulating 32-column mode.

KEYBOARD

The keyboard is the usual QWERTY arrangement with separate keys for special functions. The keys have recessed tops, a good feel, and are full-travel. It's easy to touch type on this keyboard. An 8-character type-ahead buffer helps. Each key has auto-repeat after a delay, which is convenient and doesn't cause any problem with keybounce, i.e., no duplicate letters are unintentionally produced on a single key press as on some cheaper keyboards.

A NUM key turns part of the keyboard into a numeric keypad. But the keys are not lined up as they would be in a real keypad, so this will take some getting used to. Still, it's a space saver — an important consideration in a portable. The NUM key also locks out the alpha keys, to reduce data entry mistakes.

A GRPH key turns the keyboard into a graphics unit. This allows the 32 pre-defined characters to be typed directly from the keyboard. The numeric keys can also be assigned by the user to graphics characters via software, giving a total of 42 graphics produced with the GRPH key. An additional 22 characters can also be assigned via software and produced via CTRL key combinations.

The 5 user programmable function keys are shiftable, providing 10 combinations. These keys are pre-defined with BASIC operations, but can be changed by the user, e.g., via the KEY statement in BASIC. Five additional combinations can be produced with the CTRL key and the PF keys. One is assigned to initiating manual mode on the microcassette. Another is assigned to copying the screen onto the microprinter. (This copy can be done at any time, even in Monitor mode.) The other three combinations are available for use via a machine language program.

The rate at which the computer operates can be changed via the keyboard, by hitting the PAUSE key to freeze the display and then pressing a number. This is particularly useful in slowing down the screen display to a more readable speed. The PAUSE key can be used any time to freeze the display temporarily.

A MENU key returns the system to the main start-up menu, from wherever it happened to be. This is not quite as useful as it sounds, as most programs don't like it when you leave them in the middle without exiting in a more normal fashion. However, a machine language program can intercept the usual handling of the MENU key and take some other action, such as putting up its own menu.

There is a CAPS LOCK key which puts alphabetic characters into upper case — not quite like a typewriter's SHIFT LOCK key, but quite common on computer keyboards. Unfortunately, there is no indicator to let you know when it's on.

One bad point about an otherwise very nice keyboard is that only two keys are reserved for cursor movement. In order to get vertical movement of the cursor, the user has to hit SHIFT and an arrow key. This is clumsy for word processing or any other full-screen application. The HX-20 designers tried to make up for this by giving us CTRL and arrow key combinations, but we would have opted for two more cursor keys.

The operation of the DEL key is a little unusual. Rather than working like a backspace key and deleting the character immediately under the cursor, it deletes the character in the left of the cursor. This takes some getting used to. You can produce your own backspace key by using a PF key — this is discussed in Chapter 3, HX-20 BASIC — but then one PF key is wasted.

When a key is pressed on the keyboard, a bit is set in a map in memory. A scan routine in the ROM looks at these bits and converts them into characters. This is different from a totally hardware solution which sends a particular ASCII code back to the computer. A software driven keyboard is more flexible, but also more prone to timing mistakes. Hitting the letters I N V quickly, for instance, will produce the letters I N Q. U R E can become U R B. (Unfortunately, this is common to many similar keyboards.)

A good assembly language programmer can write his own keyboard driver, bypassing the system keyboard interpreter. In this manner any special keyboard configuration could be produced. One important example is the Dvorak keyboard layout which has been designed ergonometrically to enable faster typing.

All 128 possible ASCII combinations can be produced by the keyboard, either with a single key or a two key combination. For instance, ESC, a common ASCII code, is produced by SHIFT/PAUSE. In addition, other key combinations can produce 65 other 8-bit codes.

PRINTER

The built-in dot-matrix printer operates at 42 lines per minute. A straight conversion to characters per second, the more usual way to rate printer speed, yields 17 cps. But a good part of the printer time is spent feeding from one line to the next, so the 17 cps would only apply if each line had a full 24 characters.

The printer uses a special Epson cartridge ribbon and plain adding-machine-type paper. Purple and black ribbons are available. The ribbon is estimated to last for about 10,000 lines. Both paper and ribbon are easy to change.

The main feature of the printer is that it prints by dots, rather than by characters. This means that anything that can be displayed on the screen can be printed on the printer. This is both good and bad: it means graphics as well as text can be transferred to the printer. But it also means that there are no lower case descenders on the printout as there are none on the screen. Like the screen, the dot matrix is 5 × 7. Since the printer width is 24 characters, 144 dots/line is possible.

The printer has its own on/off switch. This comes in handy when using programs that write to both the screen and the printer — the printer's being off won't halt the running of the program.

Basically, the printer is slow, noisy, and consumes power. To reduce the power drain, Epson recommends pulling the paper up by hand after printing, rather than using the Paper Feed button. But there's not much that can be done about the speed or the noise.

Having a built-in printer is handy, though. There are many times when you'll want hard copy of something on the screen or in memory and there won't be another computer or printer around to send the data to. In addition, there are some applications for which a printer is essential, such as in real-time process monitoring.

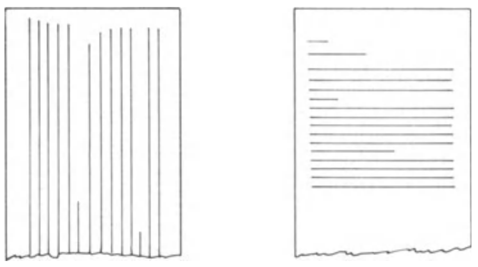


Fig. 2.4

Transam in the UK have produced a utility which allows text to be printed sideways rather than the 24 character lines across the paper. It gives 15 lines of 80 columns which can then be pasted together to give a conventional page layout. This is especially useful when word processing or in similar applications (Fig. 2.4).

REAL-TIME CLOCK

Every computer has a clock. But this system clock just controls the computer's internal operations, it's not a 'time-of-day' clock. The HX-20 includes a time-of-day clock, as well as a calendar feature. The time and date can be reset by the user, though normally this need only be done on a cold start (a complete re-initialization).

The Hitachi real-time clock provides seconds, minutes, hours, days of the week, date, month, year, AM/PM modes, automatic leap year recognition, automatic end-of-month recognition, and time-of-day alarm. It can also be programmed to advance and retard at the beginning and the end of daylight saving time (British Summer Time).

TONE GENERATOR

The speaker can be programmed to produce tones over a four octave range. The duration of each tone can also be set. Probably to avoid battery drain, the volume is set quite low and the sound could go unnoticed in a noisy environment.

The typical application for a tone generator is for entertainment, as musical accompaniment to games. However, different tones could also be used by more serious programs to denote different types of error conditions, for instance. Tone duration could be used likewise, though the only non-entertainment programs we've seen to use tone duration are Morse Code trainers.

POWER SUPPLY

The HX-20 is powered by four nickel–cadmium (NiCad) rechargeable batteries. If using the processor and memory only, the machine is claimed to go 50 hours between charges. (We haven't tested this.) If you use the printer, tone generator, cassette, RS-232 interface, etc., you'll have to recharge more frequently.

The processor can sense a low-power condition (less than 4.5 volts) and will set a flag in memory to indicate this. If you're running a BASIC program, the interpreter will interrupt whatever you're doing to flash repeatedly a 'Charge Battery' message on the screen. The only way to continue using the HX-20 at that point is to plug in the AC recharger. Though part of the Epson documentation leads the user to believe the batteries will be damaged if the computer is run while the charger is plugged in, it's been the general experience that the only consequence is that recharge-time is increased from the normal 8 hours.

Epson recommends recharging immediately upon seeing the 'Charge Battery' message. Otherwise, the company warns you can lose data and shorten the life of the batteries. Simply 'turning off' the HX-20 won't help, because the power on/off switch doesn't do any more than set a flag in memory for the software to check. The HX-20 is always on, even when the display is blank, always feeding power to the memory.

Another way to ruin the batteries, which are not easily replaced, is to overcharge them. This is not hard to do: it will happen if they are left charging for 2-3 days. You might think that the HX-20, since it has a clock, would be smart enough to turn off the charger, but no such luck. It would also be nice if the HX-20 could tell us of the state of its battery at any time, but it can't.

BAR CODE INTERFACE

Most people who use office computers wonder about the presence of bar code reader interfaces on many portables. But if you remember that one of the forebears of the portable computer was the portable terminal, then the answer is clear. Portable terminals are heavily used in inventory control applications and other situations where bar codes have proved to be a handy method of tracking items. The designers of portable computers obviously felt that their machines could cash in on part of the same market.

Note that only a hardware interface is supplied, not the actual capability. The wand that actually reads the codes is available separately. The software required to interpret the codes must also be purchased separately.

For those who need to know, the connector

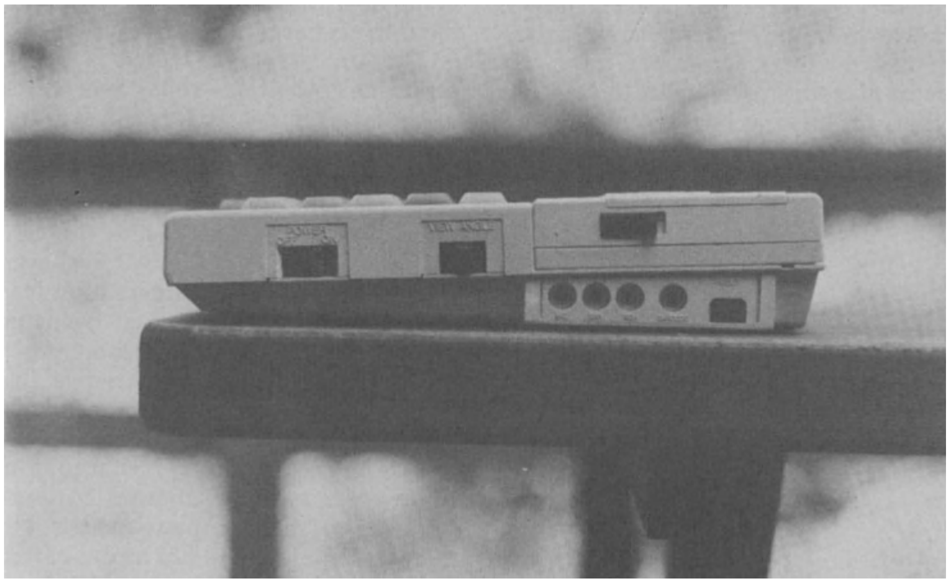


Fig. 2.5 Right side of the HX-20, showing power switch, LCD tilt adjustment knob, microcassette ejection lever, cassette interface sockets, bar code reader socket, reset button

used in the Epson is the HSJ0861-01-440 (Seidenki) with a TTL input level. For (much) more information on bar codes, check Chapter 10, Inventory/Stock Tracking.

RS-232 PORT

For 'talking' to other computers or to external printers, an asynchronous RS-232 port is available. This requires an 8-pin DIN plug — Epson supplies the necessary cables. The permissible speeds are 110, 150, 300, 600, 1200, 2400, and 4800 bps. More on this subject in Chapter 8, Communications.

HIGH-SPEED SERIAL PORT

Many buyers of the HX-20 who have noticed an extra DIN plug on the back of the machine have wondered what it's for. This is actually part of the bus between the two microprocessors brought to the outside. This allows for some nifty link-ups that would not normally be expected from a serial port. Of course, adaptors are required to interface with devices that normally expect parallel input. What is available right now is a floppy disk unit and a TV adaptor. (More information on these may be found in Chapter 12, Peripherals.)

This asynchronous port can communicate at four speeds: 150, 600, 4800, and 38,400 bps. If the system clock of the HX-20 were run at a higher rate, then a higher speed on this port would be possible. But, apparently to keep power requirements down, the system clock is kept at a lower rate.

MICROCASSETTE DRIVE

The early HX-20s in America were sold without a microcassette drive. In the rest of the world it was included, as it now is in America also.

Having a means of saving programs and data is an important feature of a portable computer. Some portables make up for it by giving you extra memory, e.g., Teleram. Some use RAM cartridges, which is probably the best solution. Some others seem to be designed not to be used for very long between connections to a remote computer.

A microcassette drive is not the ideal storage medium. For those used to working with floppy disks, it seems interminably slow. But, unlike ordinary cassette recorders, the microcassette is fairly reliable. To ensure reliability further, the HX-20 writes out every data block on the cassette twice. If the first can't be read, it will automatically try the second.

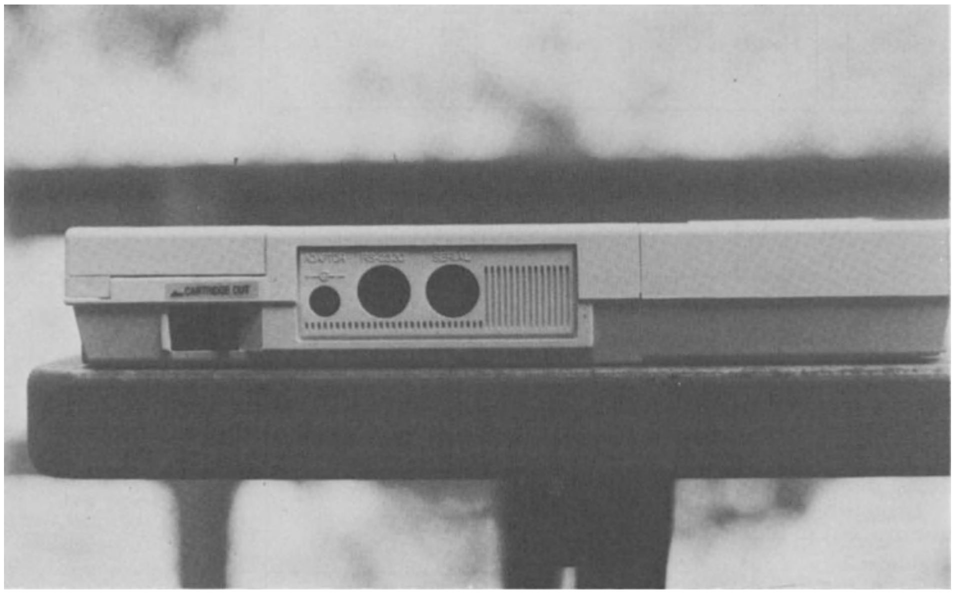


Fig. 2.6 Rear of HX-20, showing cartridge/cassette lever, AC recharging socket, RS-232 socket, high-speed serial socket

The HX-20's microcassette drive can be controlled through function keys on the keyboard. Stick-on labels tell you which PF keys do what — as long as you remember that it is CTL/PFK1 that puts you in microcassette manual mode.

The microcassette can also be controlled through software. The WIND statement in BASIC, for instance, will fast-forward down to a specific point on the tape. Generally this feature is used to store different files on the same tape, manually or automatically keeping track of which file is where. One warning: the tape can slip quite a bit on its reel, which can throw off the tape counter by quite a bit (up to 50% with some cheap Radio Shack tapes).

One software company, Ffoss Ltd, has used this capability to turn the microcassette into a pseudo-disk drive, providing random-access to what is usually thought of as strictly a sequentially accessed medium. Sophisticated error-checking is said to make this usage quite reliable. (Epson Corp. itself does not support non-sequential access.)

The drive operates at approximately 1300 bps. For comparison purposes, a Radio Shack TRS-80 Model I runs at 500 bps, a Model III at 1500 bps.

If an entire side of a '60-minute' tape is written, approximately 50 Kbyte can be saved — this is a little less than 5900 on the tape counter — keeping in mind that tape lengths can vary. Also available are 30 minute and 90 minute tapes.

CASSETTE INTERFACE

For those who must use ordinary cassette recorders, the HX-20 has a cassette port. Miniature phone jacks for input and output and a subminiature jack for remote control are located in the side of the machine. The use of conventional cassette recorders means that longer tapes can be used and so more data stored.

Data is recorded on both cassettes and microcassettes in the same format. Keeping in mind that every block is recorded twice with a short gap between duplicate blocks, then the contents of every file are: 80-byte header, long gap, one or more 256-byte blocks of data with long gaps between blocks, followed by an 80-byte end of file block.

PROM CARTRIDGE INTERFACE

The microcassette drive interfaces to the computer via a port that can also be used for PROM cartridges. If you slide off your microcassette drive, you'll see that the HX-20 has been designed to allow a device with a 12 pin connector and similar physical size to slide into

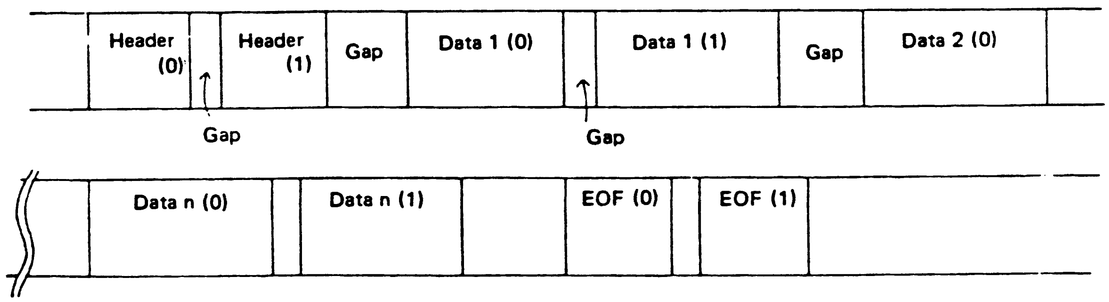
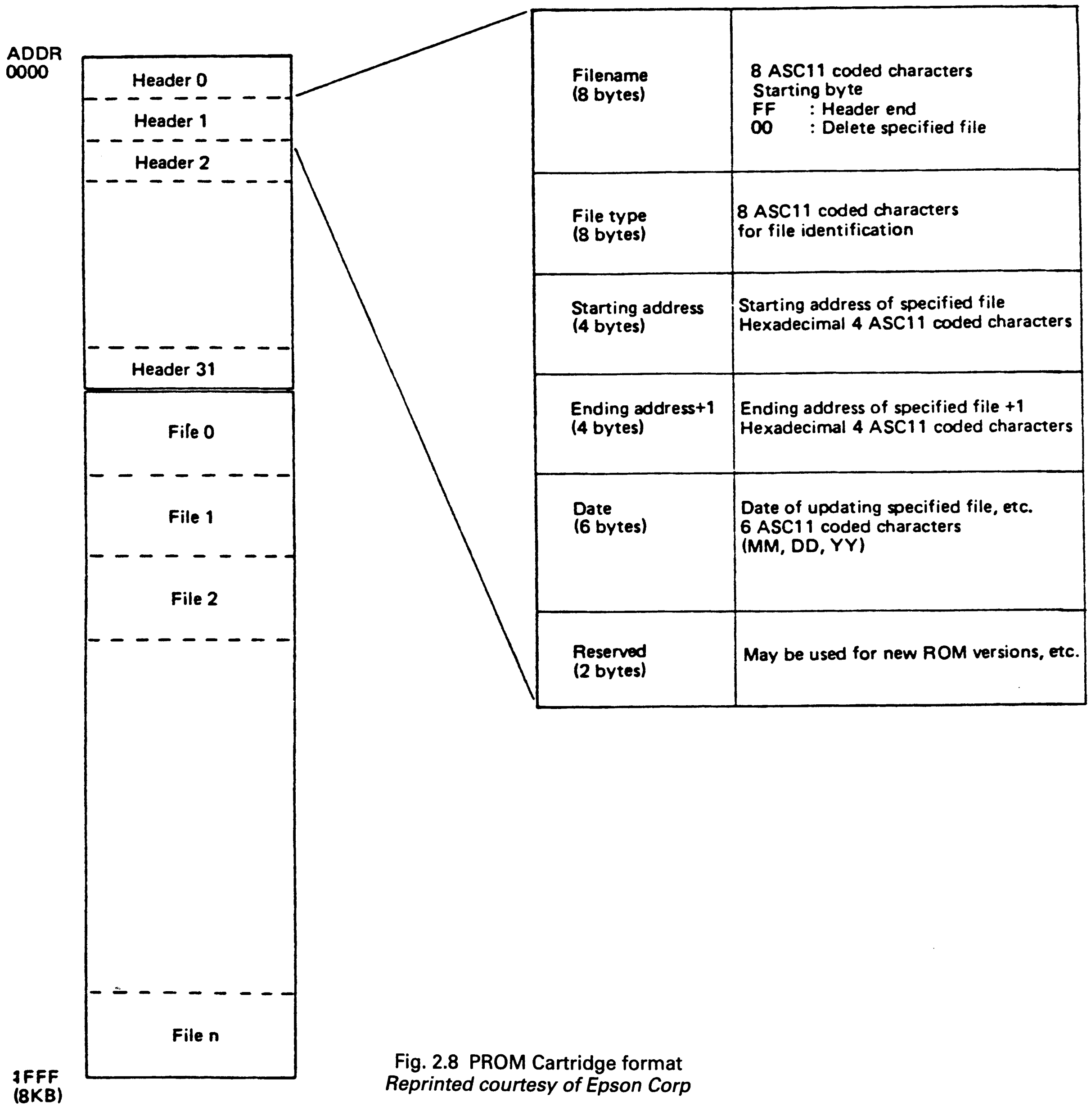


Fig. 2.7 Microcassette/cassette file format
Reprinted courtesy of Epson Corp



Filename (8 bytes)	8 ASC11 coded characters Starting byte FF : Header end 00 : Delete specified file
File type (8 bytes)	8 ASC11 coded characters for file identification
Starting address (4 bytes)	Starting address of specified file Hexadecimal 4 ASC11 coded characters
Ending address+1 (4 bytes)	Ending address of specified file +1 Hexadecimal 4 ASC11 coded characters
Date (6 bytes)	Date of updating specified file, etc. 6 ASC11 coded characters (MM, DD, YY)
Reserved (2 bytes)	May be used for new ROM versions, etc.

Fig. 2.8 PROM Cartridge format
Reprinted courtesy of Epson Corp

that spot. The operating system software, as well, will allow the use of a PROM cartridge on that interface.

This PROM cartridge does not quite work the way you think it might, however. The cartridge does not hook directly into the memory bus of the HX-20. Instead, like a cassette, its contents must be read into memory. So, PROM carts on the HX-20 do not provide any additional memory — what they do is provide a fast means of loading programs and data.

Each PROM can hold up to 32 sequential files, with headers identifying the files — again, like the microcassette.

FLOPPY DISK INTERFACE

The TF-20 disk unit can be attached to the HX-20 via the high-speed serial port. This disk is described in Chapter 12, Peripherals. A future possibility for the HX-20 is Epson's new 3.5 in (90 mm) disk.

There are some features in the HX-20 that were designed for disk use: some of the BASIC commands accept disk parameters, for instance. But true use of a disk will require a real disk-based version of BASIC.

TV INTERFACE

A color TV can be attached via a special adaptor box to the high-speed serial port. When connected, this video screen can display 32 columns by 16 lines, as a window on a virtual screen. Two sets of 4 colors each can be set via software: green-yellow-blue-red or white-cyan-magenta-orange. In monochrome, 128 × 96 points can be controlled; in 4-color mode it's 128 × 64 points. Oval Automation have developed a color/mono adaptor, mentioned previously.

EXPANSION UNIT

The expansion unit adds more memory to the basic unit. It also increases the size of your HX-20 by about 25% and adds another pound. In case you're thinking of just using one on an occasional basis, be aware of the following: in order for the HX-20 operating system to recognize the existence of the extra memory, a cold start must be done. Ditto for when you remove it. Also, Epson recommends permanently attaching brackets (which are supplied) onto the HX-20 in order to hold the expansion unit securely.

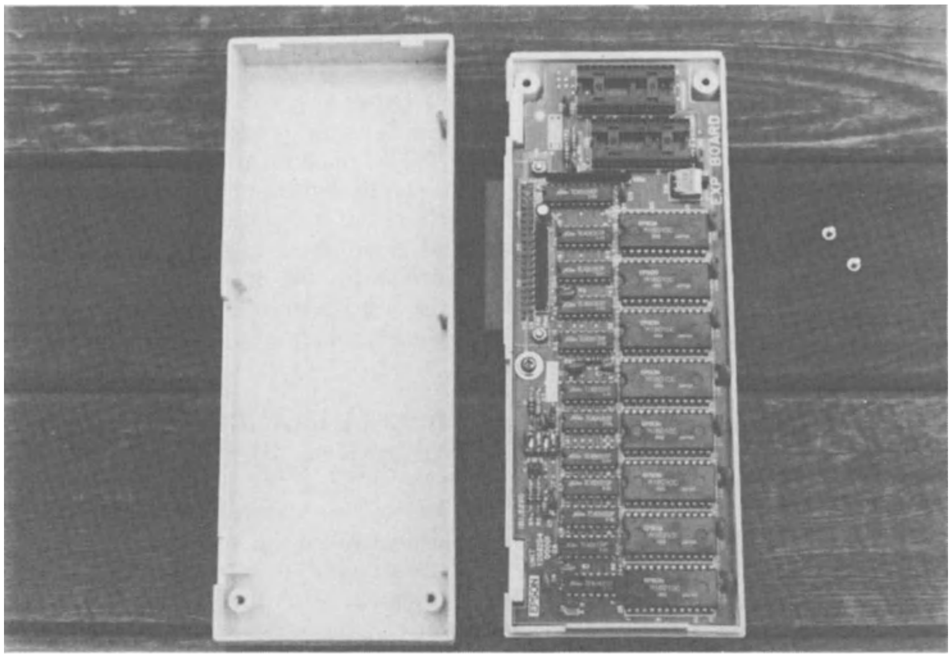


Fig. 2.9 Expansion unit, showing RAM chips (16K) and two ROM slots

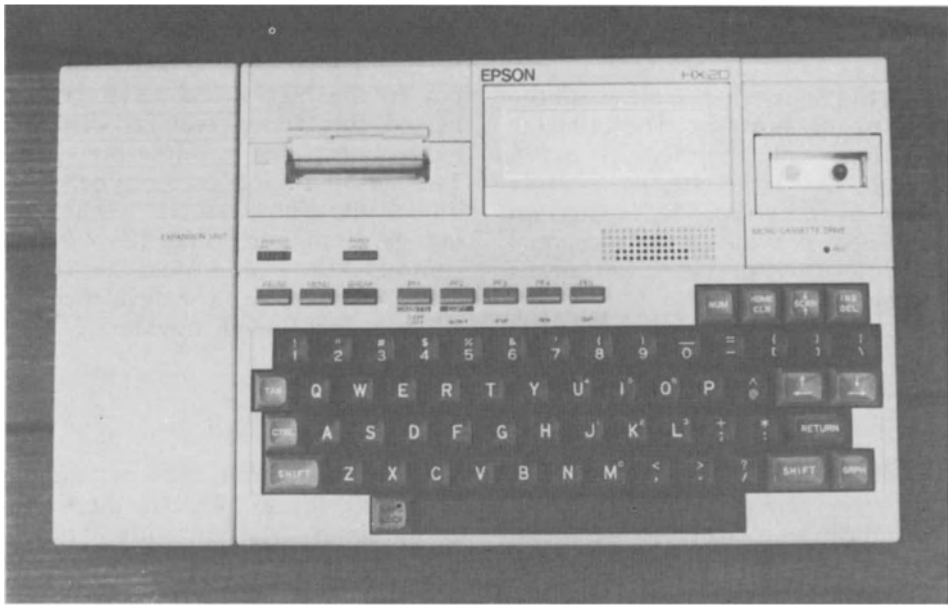


Fig. 2.10

With the expansion unit installed, you get 29,482 bytes of free space, instead of 12,898 normally. Of course, it would also add onto the 1201 that SkiWriter normally leaves you with.

Apparently because of ROM bank switching via the Menu — we have not tested this — you can have an optional ROM in the expansion unit that uses the same memory addresses as BASIC or a program in the internal ROM slot, like SkiWriter. The operating system takes up $C000–$FFFF (16K); BASIC takes $8000–$BFFF (16K), SkiWriter takes $6000–$7FFF (8K), ROM in the expansion unit can go from $4000–$BFFF (32K). However, any RAM in the expansion unit overrides ROM at the same location.

The location for added RAM partially conflicts with SkiWriter, or any optional internal ROM program, so only 8K of RAM (at $4000–$5FFF) can be used. There's no such problem with BASIC. In any case, 0–32K of ROM can be used.

There are numerous combinations of RAM/ ROM on the HX-20 with the expansion unit. These are all diagrammed in a 15-page manual accompanying the expansion unit.

ENVIRONMENTAL CONSIDERATIONS

The HX-20 seems more like an extension to an office computer than anything else. However, it's sufficiently rugged to be used for some types of out-of-doors applications.

According to the specs, the HX-20 will operate at 5–35°C (41–95°F). *Breakthrough Newsletter*, a publication for users of 'field' portable computers, says that the LCD display will not operate at freezing temperatures, nor at very high temperatures. Operation of the printer or microcassette drive is also doubtful at temperature extremes.

Data integrity — maintaining data stored in memory without dropping bits — is claimed for a range of −5–40 °C (22–104 °F).

Other specs Humidity range is 10–80% non-condensing, maximum vibration is 0.25 G at 55 Hz; maximum shock is 1 G for 1 ms.

Note that none of these numbers is as good as those for computers designed specifically for use in 'hostile' environments. But they'll present no problems for most people. (We've used our units both indoors and outdoors with no hardware problems as yet).

DOCUMENTATION AND TECHNICAL SUPPORT

In the US, Epson America has provided what is probably the best set of manuals for novices to come with any personal computer. This includes: a 2-volume BASIC set (1 tutorial, reference), an operations manual, a SkiWriter manual and a microcassette manual. Overseas, documentation is not quite as plentiful.

What is missing in the US at this time is a technical reference manual. This is a definite handicap for those who want to develop sophisticated applications, but will not be missed by most HX-20 owners. Technical documentation is available in the UK, however, from Epson (UK) Ltd dealers.

Epson America has established three regional service centres (New York, Los Angeles, Dallas) to handle hardware problems.

A technical hotline has been set up to handle software problems, but unfortunately it is not (yet) manned by technically knowledgeable people. Generally, the hotline refers callers to the Epson distributor (there are 12 nationally) in that caller's geographic area. But distributors, by and large, are neither close enough to handle problems face to face nor are they located where they would have direct contact with those few Epson America employees really familiar with the HX-20.

Epson (UK) Ltd has a Customer Service Desk primarily to support dealers but, when necessary, to help users as well.

WARRANTY

Epson America provides a 90-day limited warranty with the HX-20. This warranty provides complete coverage for parts and labour, except for the batteries. The warranty does not apply, according to Epson, if you remove the serial number or damage or misuse the computer in any way. This misuse includes opening the compartment in the back, to reset switches or plug in a new ROM. The reason for this is that CMOS chips are very static-sensitive and it was

Table 2.1

		US List ($)	UK List (£)
HX-20	(with mc drive)	795	411.00
HX-20	(without drive)		
H20MC	micro cassette drive	149	75.00
CX-20	acoustic coupler	169	
H20RC	ROM cartridge (no ROM)	59	45.00
H20EU	Memory expansion unit (w/16K RAM)	159	80.00
Cables			
702	audio cassette cables	10	5.70
705	RS232 to DB25		15.00
706	RS232 to CX20	29	
707	High speed serial port to monitor/ floppy controller	9	Oval Automation
714	RS232 to Epson printer		15.00
715	RS232 to printer	29	15.00
716	RS232 to another HX-20 RS232 (null modem)	29	15.00
717	Serial port to another HX-20	29	15.00
Supplies			
HOORFS	roll paper (5 rolls)	4.95	2.60
HOOCR	ribbon cartridge	4.95	2.20
HOOCT	3-pack microcassette tape	12.95	4.80
HOOSC	carrying case	24.95	8.00
HOOAA	battery charger (110 V)	14.95	
HOOAAU	spare mains adaptor		8.50
HOORB	battery pack	39.95	
Transam also offers:			
Adhesive labels			4.00
American Tourister case		150	

apparently felt that novice users would not know to earth themselves before touching internal parts.

Epson (UK) offers 12 month warranty on all Epson products including the HX-20.

PRICES

Prices change constantly, particularly in the microcomputer field. They also vary by dealer. But Table 2.1 gives our best guess on US and UK list prices. (The US prices from MidAtlantic Computer Products, the UK prices from Transam Microsystems.)

3

HX-20 BASIC

'Variables won't, constants aren't.'

This chapter covers:

- Why and how to learn BASIC
- What you get in HX-20 BASIC
    - Commands
    - Variables
    - Functions
    - Statements
    - Operators
    - Special characters
- What HX-20 BASIC is missing

WHY LEARN BASIC?

The owner of a typical desktop computer has hundreds if not thousands of packaged programs to select from. But as the owner of an HX-20, you're not so lucky. You'll find that to put your computer to best use you'll have to write some of your own programs or rewrite programs originally written for other machines.

Another thing — most vendors won't admit it, but all new software has bugs — the little errors that the developer didn't uncover but that always seem to creep into a program after you've bought it. If you know some BASIC, even if you're not a programmer, you can correct those errors and keep running.

Many packaged business programs require that you change the way you operate. If you know BASIC, you may be able to customize this software to fit more nearly the way you do business.

LEARNING BASIC

Fortunately for HX-20 owners, Epson has supplied an excellent pair of manuals. The first 240-page volume is a tutorial, the second is a complete reference book. Because of the quality of these books by Kenneth Skier, we'll keep to a minimum the amount of duplication here.

However, if you haven't got your HX-20 yet and don't have access to these manuals — no problem. HX-20 BASIC is mostly the typical, widely-used Microsoft BASIC. Books on learning Microsoft BASIC (used in the TRS-80, IBM PC, etc.) can be found in many general bookshops. If you want to go beyond the beginners' books on BASIC, but are not quite ready to dive into books of programs, then try the Learning Lab.

The Learning Lab was developed by Kriya Systems and is marketed by Epson America. It's a 200 page manual plus 11-program cassette that will guide you, in tutorial fashion, through some of the more interesting topics in BASIC. This includes: using random numbers, calculating interest, doing probability problems, plotting dots, graphing, displaying gravity and ballistics curves, music, sorts, and writing a word game. Each topic includes several example routines, many with flowcharts. The manuals also include full explanations of the cassette programs.

WHAT YOU GET IN HX-20 BASIC

The HX-20's version of Microsoft BASIC includes all of the 'standard' features plus many new extended capabilities. A list of all of the statements, functions, variables, and commands follows. You can use this list to get an idea of what you can do in an HX-20 BASIC program, as an aid in comparing the HX-20 with other portables, and as an aid in converting programs written for other computers so that they'll work on the HX-20.

While we're on the subject of what you get, though, we should mention the screen editor that is built into HX-20 BASIC. This screen editor makes entering and changing BASIC program lines much easier than the more typical line editors that come with Microsoft BASIC.

First, if you're not very familiar with BASIC, a few definitions are in order.

Statement — an instruction to the computer. When you write a BASIC statement, you are telling the computer to do something.
Command — an instruction to the BASIC interpreter. Of course, everything you type in while in BASIC is processed by the interpreter. But a command usually operates on the program itself, rather than on the data that the program is going to process. Sometimes, this distinction may seem arbitrary, but don't worry about it. We're using this division here simply for convenience in categorizing features.
Functions — built-in subroutines, usually returning a value. Having an already debugged and working routine can save a lot of programming time.
Variables — locations in memory that contain changeable data, stored there by BASIC or by the hardware without any further programming requirement.

There are some other concepts you'll need to know to understand the list of BASIC features. (These and other terms are also defined in the glossary.)

Window — what you see in the display screen is only part of what the computer has stored in that part of memory set aside for display purposes. The physical screen can be 'moved' across this memory, much as a window in a moving car lets you see different scenery. The total area that can be displayed (though never all at once) is called the virtual screen.

RAM Files — part of memory can be reserved strictly for data storage. This data can be accessed much as a coherent collection of items, i.e., a file. Multiple RAM files are allowed, but the HX-20 user must keep track of the beginning, length, and format of each file.

Program Area — one of five partitions that can hold an HX-20 BASIC program.
Menu — the main menu visible on system restarts can be extended with the name(s) of program(s) in the program areas.

In the following list, extensions to the universal set of Microsoft BASIC capabilities are marked with an (E). Unless otherwise noted, all actions occur in the currently logged in program area. For the included examples, assume that the variables used contain the following values:

ABCD$='ABCD'	A$='A'	B$='B'
ONE=1	TWO=2	THREE=3
FOUR=4	FIVE=5	TEN=10
TEN$='10'		

Functions

Functions can be used in different ways. For instance:

```
A = INT(5.4)
A = INT(10/3)
IF INT(A/B) > 10 THEN . . .
```

ABS — absolute value of an expression
Example: ?ABS(FOUR-FIVE) prints 1
ASC — ASCII value, in integer format, of a character
Reverse is CHR$ Example: ?ASC(A$) prints 65
ATN — arctangent in radians, single precision
Example: ?ATN(1) prints .785398
CDBL — integer to double-precision number conversion Example: ?CDBL(TEN/THREE) prints 3.333333253860474 (The result is only valid to 7 digits)
CHR$ — numeric to ASCII conversion
Example: ?CHR$(65) prints A
CINT — round off an integer Example: ?CINT(4.6) prints 5
COS — cosine in radians, single precision
Example: ?COS(1) prints .540302
CSNG — return a single-precision number
Example: ?CSNG(3.33333333) prints 3.33333
CSRLIN — locate the cursor's vertical position on the virtual (not physical) screen (E)
Example: ?CSRLIN might return 19
EOF — test for end of file Example: IF NOT EOF(1) THEN INPUT#1,A$
EXP — raise e to an exponent Example: ?EXP(TWO) prints 7.38906
FIX — drop (not round-off) the fractional part of a number Example: ?FIX (4.6) prints 4
FRE — return amount of available memory
Example: ?FRE(X) prints total available memory
?FRE(X$) prints available string space
HEX$ — decimal to hex string conversion. (E) There is no function for the reverse Example: HEX$(65) equals 41
INKEY$ — return the key (if any) that is being

pressed Example: 10 IF INKEY$=" " THEN 10
INPUT$ — obtain raw (unedited) input from the keyboard, device or a file. (E),
Examples: ?INPUT$(5) will get five characters from the keyboard and display them ?INPUT$(3,#1) will get three characters from file #1 and display them
INSTR — search a character string for an embedded, smaller string and return its starting location (E)
Example: ?INSTR(ABCD$,B$) prints 2
INT — returns a number which is the largest whole number not greater than the original number
Example: ?INT(4.6) prints 4
LEFT$ — obtain leftmost part of a character string
Example: ?LEFT$(ABCD$,2) prints AB
LEN — obtain the length of a character string
Example: ?LEN(ABCD$) prints 4
LOF — obtain the number of characters in the communications input area (E)
Example: ?INPUT$(LOF(1),#1) prints any characters received
LOG — natural logarithm Example: ?LOG(TEN) prints 2.30259
MID$ — select a string of specified length, starting at a specified point, from within another string
Example: ?MID$(ABCD$,2,3) prints BCD
OCT$ — decimal to octal conversion (E)
Example: ?OCT$(TEN) prints 12
POS — locate the next byte in a file or the horizontal position of the cursor (E) Example: ?"HI";POS(0) would print 2
RIGHT$ — pick off the rightmost characters of a string Example: ?RIGHT$(ABCD$,1) prints D
RND — obtain a single precision random number between 0 and 1 Example: ?RND(1) might print .207991
SGN — determine whether a number is positive, negative or zero Example: ?SGN(FOUR) prints 1
SIN — sine in radians, single precision
Example: ?SIN(1) prints .841471
SPACE$ — produce a blank string of specified length (E) Example: ?SPACE$(20) prints twenty blanks
SPC — display (on the screen or microprinter) a specified number of blanks (E) Example: PRINT "LEFT" SPC(11) "RIGHT"
displays: LEFT   RIGHT
SQR — square root Example: ?SQR(FOUR) displays 2
STR$ — number to character string conversion
Example: ?LEFT$)STR$(TEN),2) displays [blank]1 (The first character of the new string is always a blank)
STRING$ — produce a string of specified length, using like characters Example: ?STRING$(20,"–") prints twenty hyphens
TAB — jump to a horizontal position on the screen or microprinter Example: ?"GOOD";TAB(10);"BYE" will print GOOD in column 0 and BYE in column 11
TAN — tangent in radians, single precision
Example: ?TAN(1) prints 1.55741
TAPCNT — the value of the microcassette's tape counter (E) Example: ?TAPCNT would display 0 after a WIND
TIME$ — time of day (HHMMSS) Example: ?TIME$ may print 13:55:42
USR — call a machine language subroutine
Example: DEF USR(0)=&H0B00 X=USRO(5) will pass the number 5 to a routine at $0B00
VAL — numeric character string to number conversion Example: ?VAL(TEN$) prints 10
VARPTR — the address of the first data byte of a variable Example: ?VARPTR(A$) may print 2732

Commands

Commands are mostly intended to be used when in the 'READY' mode of BASIC. However, all can be coded in a program.

AUTO — automatic line numbering
Example: AUTO 100,10
The first line would be 100, the next 110, etc.
CLEAR — set variables to zero, clear the string area, optionally reserve space for "RAM files"
Example: CLEAR 1000, 5000 will give 1000 bytes of string space and a 5000 byte RAM file area
CONT — resume program execution after a break
Example: CONT
DELETE — delete program line(s)
Example: DELETE 10-100 removes lines numbered from 10 to 100
FILES — list the files on the specified device (E)
Example: FILES "CASO:" will list the files on a microcassette tape
LIST — display program lines on the screen
Example: LIST 10-100 will display lines numbered from 10 to 100
LLIST — print program lines on the microprinter
Example: LLIST 10–100 will print lines numbered from 10 to 100
LOAD — get a BASIC program from tape, disk, PROM cartridge or RS-232 port, and optionally RUN it Examples: LOAD "COMM20.BAS"
LOAD "COMO: (37E12)",R
LOAD? — checks that a cassette file is OK
Example: LOAD? "COMM20.BAS"
LOADM — get a machine code program from tape, disk or PROM cartridge and optionally execute it Example: LOADM "TEST.OBJ"
LOADM "PACO:TEST.OBJ",R
LOGIN — select another program area and optionally RUN the program found there (E) Example: LOGIN 5,R
MEMSET — reserve memory for machine-language programs (E) Example: MEMSET &HB80
MERGE — Retrieve a BASIC program from tape, disk, RS-232 port or PROM cartridge and combine its lines with those of the currently logged-in program (E) Example: MERGE "CAS1:TEST.BAS"

MON — enter the debugging monitor (E)
Example: MON
NEW — clear the program area and all variables
Example: NEW
PEEK — inspect memory Example: PEEK(&H140)
POINT — determine status (on/off, color) of a point on the screen (E) Example: POINT(10,20)
RENUM — renumber program lines (E)
Example: RENUM 1000, 220, 100 where 220 is the first line to renumber, 1000 is the number to give it, and 100 is the increment for succeeding lines
RUN — execute a BASIC program after optionally loading it. Example: RUN "COMM20.BAS",R
The "R" will keep open any currently open files or devices
SAVE — copy a BASIC program to tape, disk or RS-232 port Example: SAVE "COMM20.BAS",A
The "A" will save it as an ASCII file
SAVEM — copy memory to tape or disk (E)
Example: SAVEM "TEST.MEM",2900,3200,3100 where the contents of memory locations 2900-3200 will be saved; 3100 is an entry point for later use with the LOADM command
TITLE — add a program to the HX-20's menu (E)
Example: TITLE "COMMUNICATIONS"
TRON TROFF — display the line numbers of BASIC statements as those statements are being executed.
Examples: TRON
TROFF

Statements

CLOSE — end operations on a file or device
Example: CLOSE #1
CLS — clear screen Example: CLS
COLOR — (on external TV set) set foreground and background colors, and the color set (E)
Example: COLOR 3,0,1
COPY — print screen image on the printer (E)
Example: COPY
DATA — store data for access by READ statement
Example: DATA CURRENT(1984),FIRST (1985), SECOND (1986)
DEFDBL — declare double-precision variables
DEFINT — declare integer variables
DEFSNG — declare single precision variables
DEFSTR — declare string variables
Example: DEFINT A–E, I All variables starting with the letters A through E and I will be integer variables
DEFFIL — specify a RAM file record length and starting displacement (E) Example: 51,200
DEFFN — define a user-written function
Example: DEFFN RAD(DEG)=DEG/57.2958
A=SIN(FNRAD(90))
DEFUSR.— define up to 10 machine language subroutines Example: DEFUSR(1)=&HOB20
DIM — define a table (array) Example: DIM(40,10) defines a two-dimensional array
END — end program execution Example: END

ERASE — undefine an array (E)
Example: ERASE A$
ERROR — simulate an error or define your own program error code Example: ERROR 11
EXEC — call a machine language subroutine at a specified address (E) Example: EXEC &HE000
FOR. . .NEXT
FOR. . .STEP . . .NEXT — loop control
Example: FOR x=1 TO Y STEP 1:x=x*x:NEXT x
GCLS — clear graphics only (E) Example: GCLS
GET% — retrieve data from a RAM file (E)
Example: GET%R,NAME$,AMOUNT Retrieves a customer name and purchase amount from the record number stored in R
GOSUB. . .RETURN — execute a subroutine
Example: GOSUB 1000
GOTO — branch to a specified program line
Example: GOTO 1000
IF. . .THEN
IF. . .THEN. . .ELSE
IF. . .GOTO
IF. . .GOTO. . .ELSE — change program flow based on the evaluation of an expression Examples: IF X=Y THEN 150 ELSE . . .
IF TRUE GOTO 200 ELSE
INPUT — enter data into a running program from the keyboard Example: INPUT "Quit (Y/N", A$
INPUT# — enter data into a running program from a file Example: INPUT#1,A$
KEY — assign a character string to a function key (E)
Example: KEY 10, "KEYLIST"
KEY LIST — list the function keys on the screen (E)
Example: KEY LIST
KEY LIST — list the function keys on the printer (E)
Example: KEYLIST
LET — assign a value to a variable
Example: LET A=4 is equivalant to A=4
LINE — draw or erase a line on the screen (E)
Example: LINE(0,10)–(20,20)
LINE input — enter data into a running program from the keyboard, ignoring delimiters
Example: LINE INPUT "First Address Line: ",A$
LINE INPUT# — enter data into a running program from a file, ignoring delimiters (E)
Example: LINE INPUT #1,A$
LOCATE — set the cursor to a particular location on the virtual screen (E) Example: LOCATE 0,4,1 would put it in the first (0th) column, on the fifth line, and make the cursor visible
LOCATES — Shift the LCD window to a new area of the virtual screen (E) Example: LOCATES 0,4,0 would locate the physical screen at the fifth line (first column) of the virtual screen and turn off the cursor
LPRINT
LPRINT USING — write data on the microprinter
Example: M$="\ \" LPRINT USING M$;A$
MID$ — exchange characters within a string with other characters Example: A$="1984-1987"
MID$(A$,4,1)="5" changes string to "1985-1987"

MOTOR — turn the external cassette drive motor on/off (E) Example: MOTOR ON
ON ERROR GOTO — specify error-handling routine(s) Example: ON ERROR GOTO 10000
ON. . .GOSUB
ON. . .GOTO — conditional branch depending on value of an evaluated expression Example: ON VAL (A$) GOTO 1000,2000,3000
OPEN — allow I/O to a file or device Example: OPEN "I",#1,"COMO:(37E12)"
OPTION BASE — "to declare the minimum value for array subscripts" (E) Example: OPTION BASE 1
PCOPY — copy a BASIC program into another program area (E) Example: PCOPY 4
POKE — move a byte to a specified location in memory Example: POKE &H0A40,65 will put an "A" in location $A40
PRESET — erase a dot (E) Example: PRESET (10,15)
PRINT
PRINT USING — display data on the screen Example: PRINT "HELLO"
PRINT#
PRINT# USING — write data to a file or device Example: PRINT #1, USING N$;A$
PUT% — store data into a RAM file (E) Example: PUT%3, ABCD$+TEN$ will put ABCD10 into record #3.
RANDOMIZE — reset the random number generator Example: RANDOMIZE RIGHT$(TIME$,1)
READ — input values previously entered in DATA statements Example: READ NAME$,ADDRESS$,AMOUNT
REM — insert comments in a program Example: 100 REM MAIN LOOP is the same as 100 ' MAIN LOOP
RESTORE — set the READ pointer to a specified DATA statement Example: RESTORE 2200
RESUME — restart program execution after an error Examples: RESUME to retry the same statement RESUME NEXT to restart with the next statement RESUME 1000 to restart at line 1000
SCREEN — select the LCD or TV screen for graphics or text output (E) Example: SCREEN 0,1 will put text on the LCD and color graphics on the TV
SCROLL — set screen movement options (E) Example: SCROLL 9,0,4,4 would be the fastest scroll speed, no horizontal scrolling, move 4 spaces on CTL/arrow, and move 4 lines on SCRN keys
SOUND — produce a tone from the built-in speaker (E) Example: SOUND 13,10 would display an 880 Hz "A" note for 1 second (10 tenths)
STAT — provide information about the status of program area(s) (E) Example: STAT ALL
STOP — return to BASIC command level from a program Example: STOP
SWAP — interchange the values of two variables (E) Example: SWAP HI$,LO$
WIDTH — set the dimensions of a device (E) Examples: WIDTH 20,40,3 sets the virtual screen to 20 columns wide, 40 columns high, and provides a scroll margin of 3 columns WIDTH "LPTO:",20 will set the microprinter width to 20 columns
WIND — position the microcassette to a specified location (E) Example: WIND 1000

Variables

DATE$ — current date in MM/DD/YY form Example: ?DATE$ may print 11/24/84
DAY — numeric value of the day of the week (1-7) (E) Example: ?DAY may print 6
ERL — line number of the last program error
ERR — error code of the last program error

General Operators

=	assignment or equality test
>	greater than
<	less than
=>, >=	greater than or equal
<=, =<	less than or equal
<>	not equal

Numeric Operators

+	addition Example: 2 + 3 is 5
−	subtraction or negation Example: 4 − 5 is −1
*	multiplication Example: 4 * 5 is 20
/	division (floating point result) Example: 10 / 3 is 3.33333
	division (integer result) Example: 10 \ 3 is 3
^	exponentiation Example: FOUR^3 is 64
MOD	modulus (the integer remainder of a division) (E) Example: 11 MOD 4 is 3

Logical Operators

If the expressions being operated upon are equal to 0 (false) or −1 (true), then

AND	if all expressions are true, then the result is true
OR	if any of the expressions are true, then, the result is true
NOT	reverse the logical value of an expression
XOR	if any one expression is true, then the result is true except that if all expressions are true then the result is false (E)
IMP	if the first expression is true and the second expression is false, then the result is false. In all other cases the result is true (E)
EQV	if all expressions are logically alike — all true or all false — then the result is true, otherwise the result is false (E)

Examples:
A$="HI"
B$="BYE"

In the statement:
  If A$="HI" and B$="OOPS" THEN. . .
the first expression evaluates to −1 (true), the second to 0
You can check this by entering:
PRINT A$ = "HI"
PRINT B$="OOPS"

In the statement:
  IF A AND B THEN. . .
where the values of A or B are not −1 or 0, then the results of the operation will depend on the bit configuration of the contents. See p.487 of the *Epson America BASIC Manual* for examples. Note that:
  IF A THEN PRINT "OK"
will result in printing OK whenever A is not equal to 0, not just when it is −1.

The results of operating on various combinations of A and B are shown in the following truth table.

Table 3.1

A	B	A AND B	A OR B	A XOR B	A IMP B	A EQV B
T	T	T	T	F	T	T
T	F	F	T	T	F	F
F	T	F	T	T	T	F
F	F	F	F	F	T	T

Operator precedence

expressions in parentheses ( )
exponentiation (^)
negation (–)
multiplication, division with floating point result (*, /) division, integer result (\ )
modulus (MOD)
addition, subtraction (+,−)
relational operators (=,<,>,<=.=<, =>,>=,<>)
NOT
AND
OR
XOR
IMP
EQV

Special characters

Special characters are used in print formatting. This varies with the national character set selected. The following example is for the US

#	numeric digit
.	decimal point
+	print the sign of a number
–	print a trailing minus sign if the number is negative
**	fill leading spaces with asterisks
$	put a dollar sign in front of the formatted number
**$	combines ** and $
,	put in commas every third digit
^^^^	exponential (nnE+nn) format
——	signifies next character is a literal
%	when printed, signifies overflow error

Other special characters

?	print
&H	hex
&O	octal
%	integer variable
!	single precision variable
#	double precision variable
E	floating point (exponential) constant with single precision
D	exponential constant with double precision

WHAT ARE WE MISSING?

We noted above that Epson HX-20 BASIC adds to the fundamental Microsoft set of features. But other Microsoft BASIC machines also have extended features. If you want to compare HX-20 BASIC to other computers or if you want to convert programs written for other computers to HX-20 BASIC, the following list should be of help. These are extensions that are either missing from the HX-20 or are HX-20 features with different names.

Note While disk drives are defined as devices in the HX-20's operating system, there are really no disk functions present in HX-20 BASIC: no random access operations, no sector indexing, no renaming files, etc. So, any disk-based BASIC will have features not present in HX-20 BASIC.

CALL — call a machine language routine and pass variables to it. The EXEC and USR functions on the HX-20 can be used to call machine language routines, but the ability to pass parameters is limited

CHAIN — call a program and pass variables to it. You can simulate this by storing the variables in a RAM file and doing a RUN 'next-program',R

COM ON — enable/disable communications interrupts. Interrupt handling is not available from BASIC on the HX-20

COMMON — common list of variables to pass to a CHAIN'd program. Use a RAM file instead

CVI — convert a 2-character string to an integer

CVS — convert a 4-character string to a single precision number

CVD — convert an 8-character string to a double precision number

DAY$ — obtain alpha day of the week. You can do this on the HX-20 with a simple table look-up

EDIT — BASIC line editor. Epson's screen editor is the replacement, but some uses can't be duplicated e.g., having the program change itself

FILES — display files stored in RAM

HIMEM — returns the address of the highest memory location available. The HX-20 replacement is STAT ALL, which shows how much memory is left. Also, a PEEK of locations &H12C-12D will show the highest address plus 1

INP — input a character from a port. Epson replacement is INPUT#, which allows a string to be read in from any valid HX-20 port

IPL — run a BASIC program on power up. Epson replacement is to use keystack command from monitor

KEY ON/OFF/STOP — enable/disable function key interrupts. HX-20 function keys don't cause interrupts

KILL — erase a RAM file. The HX-20 version (CLEAR) operates on the entire RAM area

LCOPY — same as COPY

LPOS — returns column position of printhead

MAXFILES — returns # of allowable RAM files. There's no limit on the HX-20, except the limit of available memory

MAXRAM — returns highest available RAM file memory address. HX-20 users will have to keep track of this themselves

MDM ON/OFF/STOP — enables/disables the ON MDM interrupt. The HX-20 does not have a built-in modem

MENU — returns to main menu. On the HX-20, use EXEC &HD23B or manually hit the MENU button

MKI — convert an integer to a 2-character string

MKS — convert a single precision number to a 4-character string

MKD — convert a double precision number to an 8-character string

NAME — renames a RAM file. The HX-20 doesn't give names to its RAM files, unfortunately

NULL — set the number of nulls printed after each line

ONCOM — call subroutine when data is received at the RS-232 port. The HX-20's RS-232 port is not interrupt driven, unfortunately, so the programmer must continually execute an LOF function to accomplish the same thing

ON KEY GOSUB — call subroutine when function key is depressed

ON MDM GOSUB — call subroutine when data is received by modem

ON TIME$ GOSUB — call subroutine when clock reaches a specified time. The Epson has this feature available hardware-wise via an interrupt, but it is not implemented in BASIC

OUT — output a character to a port. PRINT# is the Epson replacement

POWER — set power-off timer

POWER CONT — prevent automatic power down

POWER OFF (RESUME) — turn power off immediately

PRINT@ — print data at a specified screen position. Epson requires that the LOCATE command be first used to set cursor; than the next PRINT statement will print at that position

RSET — right justify a string within a larger string

RUNM — run a machine language program, as opposed to a subroutine

SCREEN — turns on/off screen function key labels. Epson uses keyboard overlays for function key descriptors. Those with custom programs are on their own

SOUND ON/OFF — starts/stops tone when loading or awaiting carrier

TIMES$ ON/OFF/STOP — enable/disable time interrupt

WAIT — suspend program execution until a specified action has occurred at a specified I/O port

WHILE/WEND — Loop through statements as long as a condition is true

WRITE — output to the screen or a file. Use PRINT# on the HX-20

4

USING AND WRITING BASIC PROGRAMS

'If carpenters built buildings the way programmers wrote programs, the first woodpecker to come along would destroy civilization.' (Weinberg)

This chapter covers:

- Finding and entering BASIC programs
- Tips on entering programs
- Tips on converting programs written for other machines
- Programming tips
- Some BASIC programs and subroutines
  - Searches
  - Sorts
  - Graphics
  - Finance
  - Electronics

Sometimes programmers don't mind 're-inventing the wheel' because they find it fun to work out their own solutions to a problem. But if you have something you need to get done, this approach isn't for you. Using programs and subroutines already developed could save you a considerable amount of time against doing it over again yourself. If you wanted to build a house, for instance, you would buy ready-made glass for the windows rather than trying to make your own glass.

There are a number of books on the market that consist of collections of programs and subroutines that are appropriate for general business and utility use. One is *Simple BASIC Programs for Business Applications* by J.R.F. Alonso (Prentice-Hall, 1981). Another is *Subroutine Sandwich* by Grillo and Robertson (John Wiley, 1983). The same authors have also put out a sequel — *More Subroutine Sandwich.* Of course, there's also the Kriya package mentioned previously. There are others, but these are the ones we've seen and can recommend.

TIPS ON ENTERING PROGRAMS

Before you start keying in any program from a listing — especially a long one — think about typing short-cuts and think about avoiding typos. Here are a few tricks plus a few things to look out for:

1. Use the HX-20 screen editor to repeat similar lines. It's easier — and provides less chance of error — to edit an existing line than to type in a new one from scratch. Remember, to get a new line, all you have to do is type over the line number of the old one.
2. Set the screen width to 20 and the vertical height to some large number. This will let you see the whole line at once.
3. If there are statements that repeat often throughout the program, assign them to the PF keys.

BASIC PROGRAM CONVERSION TIPS

Undoubtedly, the most marked difference between running a program on the HX-20 that has been written for another machine is the way the screen has been handled. There are two problems: graphics and size.

There is no standard implementation for graphics on computers running Microsoft

BASIC. This means that any program in which graphics is an integral part will likely cause you problems in trying to convert it over to the HX-20.

Not only does the way that the graphics works vary, but the small size of the HX-20 screen will prevent most graphic displays from running. We don't mean to discourage you from trying to convert programs, but just want to point out the difficulties.

Text messages — prompts and instructions for instance — also need to be changed to fit on the HX-20's screen. But this just takes a little reformatting and shouldn't slow you down much. The assembler found later in this book was written for the TRS-80 Model I and converted to the HX-20. For the messages, we just shortened each down to 20 characters and made use of the HX-20's vertical scrolling ability to allow the user to see what's scrolled off the screen. True, you could use horizontal scrolling to avoid having to change the message sizes at all, but we don't advise that. Most people find it uncomfortable to read. It was with that fact in mind that SkiSoft's SkiWriter package was given a fixed limit of 20 columns.

Disk programs present only a couple of problems to the HX-20 prospective program converter, as long as random access is not done. A sequential file is a sequential file on any type of device. But the way in which data is stored by disk BASICs varies from the way HX-20 BASIC stores it. Disk files are often made up of fixed length fields, unlike tape files which are often purely strings and variables. But everything that can be written to disk can be written to tape. Again, take a look at our assembler for a sample of how to read/write to tape.

Programs that use random access disk represent a special problem for conversion to tape use. Typically, disk files are of three types: sequential, indexed, and relative. The sequential type we're familiar with. The indexed type consists of a pointer to a particular record on a disk. If you had a client file, for instance, and wanted to access the record for John Smith, your program would first look in the index. There it would find a location on the disk where the information for Mr Smith could be found. It would then go directly to that place on the disk and read the data.

Relative files are also random access. But instead of an index, the key that we want to find in the file has a number in it that is the relative sector number (sometimes byte number) into the file. For instance, most implementations of Forth number the screens. The eighth screen might be saved in the eighth disk sector. Or, maybe each screen is two sectors and so the ninth screen would be in sector 18. Another example: an inventory can be set up where there are three items per sector and part number xxxx-24 would always be found in sector 8. This saves look-up time (no index), but costs space — relative files tend to have a lot of holes in them.

We've explained a little about random access files as a necessary background for converting programs that use disk files. The way to do it is with assembly language subroutines plus the WIND command. Actually, the best way is to use the RAX ROM described in Chapter 11, Software and Systems.

Just think of a microcassette tape as a diskette. Each record on the tape is equivalent to a record on a disk.

For indexed files, the index would have the tape location of the record. For relative files, the tape location could be computed from the item description. *However*, you can't just spin the tape down to the appropriate point and read the information. BASIC doesn't like to do that. It'll give you an I/O error if you try to read a block that's not the next sequential block in the file. But the data *has* been read and if you PEEK into location $37C (&H37C), the microcassette buffer area, you'll see it there.

For instance, let's say you created a tape with:

```
10 CLEAR 1000
15 A=65
20 OPEN "O", #1, "CASO:TEST"
30 A$=STRING$(255,A)
40 PRINT#1,A$
50 A=A+1
60 GOTO 30
```

This just writes out successive records of As, Bs, Cs, etc. Now WIND back to the starting point and read it back in with:

```
100 OPEN "I",#1,"CASO:TEST"
110 PRINT TAPCNT
120 INPUT #1,A$
130 GOTO 100
```

Now we have the starting point for each record. And so we can replace 110 with:

```
110 WIND nn
```

where nn is the tape location of the record we want. We will get our I/O error on line 120 but if we look in memory at $37C we'll see our 255 Cs or Ds or whatever.

In the above case we cheated a little by using a string that occupied a full 256 byte cassette block (255 characters plus the CR that BASIC sticks in after each PRINT# statement). If we had written a smaller string, say 63 characters, then each physical 256-character block on the tape would be made up of four 63 byte strings (with a CR at the end of each record).

Just as with real disk records, it's easiest if the data you write fills an entire physical block. Otherwise, your strings will be broken up — part in one block and part in another. If we had wanted to write 99 byte strings, for instance, only 56 bytes of the third string would have fitted in the first block, the other 43 bytes would start off the second block.

Up above we mentioned getting an I/O error. That error will prevent you from getting a second record from the file without closing and re-opening the file first (e.g: ON ERROR . . . RESUME 1000 . . . 1000 CLOSE . . . OPEN . . .). Working with machine language we don't have that problem, which probably means that there is some byte you can POKE somewhere to keep going. (We might have that information by the time you read this, so drop us a note.)

If we're going to keep closing and re-opening the file, we may want just to create a myriad of little files. Instead of each data record being part of one large file, we can have each record as its own file. Just do an OPEN before the PRINT# and a CLOSE after it. Again, we would keep track of the value in TAPCNT and use WIND to spin down to the file (i.e., record) that we wanted.

Epson America, by the way, doesn't approve of this pseudo-random access means of using the tape. Epson says that a low battery condition, concurrent use of the RS-232 port or any other condition that produces less than normal voltage may throw off the tape speed and give you an erroneous tape counter reading. Our own experience, using Radio Shack microcassette tapes, is that TAPCNT is none too reliable. But the solution is straightforward. Each block on the tape is numbered, and this number is read into memory at location $379. If you PEEK this location, you can figure out where in your file you are. If creating a new file for each block, give each file a unique name/number.

ADDITIONAL CONVERSION TIPS

There are some additional things to keep in mind when converting programs from other machines to the HX-20.

1. The length of any one string can be no more than 255 characters.
2. The usable length of the name for a variable is 16 characters.
3. All numeric variables that don't have explicit type specifiers are assumed to be single precision.
4. The <RETURN> key inputs a CHR$(13).

BASIC PROGRAMMING TIPS

Whether you write programs for your own use or programs for others to use, the following section should provide some assistance.

What makes a program good? How about:

- it performs its stated function(s);
- it's easy to operate;
- it's easy to learn how to operate;
- it's easy to modify;
- it executes quickly;
- it uses little memory.

You may have come across the term 'structured programming' in various software publications. Structured programming refers to a way of systematically writing a program so that it is easy to debug and easy for others to understand the logic flow. We won't get into all that structured programming entails, as entire books have been written on just that subject, but we will pick up a few good points that we can use in writing our own programs.

1. The most infamous way to make a program hard to understand is to use GOTO statements indiscriminately. This results in 'spaghetti' code, where the flow of control passes in, out, around, and all over the code. Keep the use of GOTOs to a minimum and always try to branch downward in a program.
2. Use subroutines. Ideally the first part of the program should look like:

```
GOSUB initialization
GOSUB main processing
GOSUB termination
```

In the main processing routine, you might have:

```
100  FOR A = 0 to 1 STEP 0:
       GOSUB read record
       IF A=0 THEN GOSUB process record
         GOSUB print data
     NEXT A
200  RETURN
```

In the read-record subroutine, you might have:

```
IF NOT EOF(1) THEN
  INPUT #1,B$,C$
ELSE A=1
RETURN
```

The lines above are indented for clarity, something not easily accomplished (if at all) on the HX-20's screen. But the idea still holds. The use of subroutines makes programs easier to read and also isolates logically separate sections of code from one another. The print routine, for instance, is not dependent on anything that happens in the read routine. Once we know the print routine works OK, then we don't have to worry about it any more. There will never be any branches into the middle of it, it won't stop working because we changed something else, etc. Note also that each subroutine has one entry point and one return point.

3. Use extended variable names. Early BASICs only allowed a couple of characters for the name of a variable, so programs tended to look cryptic. But the HX-20 will allow up to 16 characters for unique names. Isn't DAILYSALES easier to understand than DSO? Just look out for using a BASIC reserved word. You can check the reserved word list each time you create a variable name, or just leave out the vowels, that will usually get you through, e.g., MNTHLYSALES instead of MONTHLYSALES because MON is a reserved word.
4. Use device names rather than specific keywords for I/O. For instance, write to the printer with PRINT# rather than LPRINT. This allows you to change devices within the program (or outside of it, for debugging) by just changing the OPEN statements. No change to any of the I/O statements is required.
5. Use prompts to explain exactly what input is requested. Don't make the operator guess as to which response you had in mind.
6. During routines that process for a long time, print a counter or sound a tone so that the operator will know the program is still running and still getting somewhere.
7. Arrange line numbers into groups, by subroutine. Keep similar line numbers for similar functions, e.g., nn90 could always be a RETURN.
8. Keep processing on errors. The operator may press the wrong key, an illegal arithmetic condition may occur — such as a division by 0 — or a computed point on the graphics screen may be larger than 0–119, 0–31. None of these conditions should cause your program to abort. Error-trapping is particularly important for programs to be used by computer novices who might not know what to do when an error occurs. An alternative is to supply a printed set of instructions that cover actions to be taken in different situations.
9. Offering the operator a default value can make the program easier to use. This default value, the value used by the program when the operator declines to enter a new value, can be pre-defined at the start of the program or can be picked up from the last operator entry. For instance, the city-state in a mailing address file can be picked up from the previous entry if the operator only hits RETURN when prompted for a new city-state.
10. Provide for single-character operator input, wherever possible, to eliminate the need for hitting RETURN. But be consistent. Don't require the use of RETURN sometimes and not other times, or the operator will get frustrated.
11. Use comments, i.e., REM statements, to indicate what subroutines do and what variables are used for.
12. Lay out text on the screen so that it is easily readable, e.g., don't spread single words over two lines if at all possible.
13. If the documentation can't be built right into the program, write two separate sets of instructions: one for operation, one for programming. (You'll probably need it yourself 6 months later when you want to make a change and can't recall why you wrote the code the way you did.)
14. Don't require the operator to re-enter data that the program already has access to or can figure out. If the program has a list of possible values, one of which must be

chosen, it may be better to present the list to the user and ask that he select one, rather than requiring him to retype it. If multiple values may be selected, you can present the list one at a time and ask for a Y/N confirmation.

If there are too many entries to list, then allow the operator to 'abbreviate to uniqueness'. That is, if you have part numbers in an inventory like:
HX-20
QX-10
Coleco Adam
Commodore 64
the operator should only need to enter H to see the HX-20 entry or COM to see the Commodore 64 entry. If what the operator enters is ambiguous, such as CO, then you can sound the beeper as a signal for him to enter more characters.

15. Accept input in either upper or lower case. Not only is it easy to hit accidentally the CAPS key on the HX-20, but going in and out of NUM will turn CAPS on even if it was originally off.
16. Make data files compatible with files used by other programs that the operator may be using.
17. Use the same sort of prompts and responses that are used by other programs the operator may be using.
18. Use text compression to save space on the screen. See the Software & Systems chapter for a discussion of the QuickView technology.

Dave Smith of the Xerox Palo Alto Research Center, the developers of the technology that foreshadowed Apple's Lisa, says that software should be 'familiar, concrete, visual'. Paul Heckel of Quickview Systems has written an excellent book on the theory behind writing easy to use programs — *The Elements of Friendly Software Design* (Warner Books, 1984).

EFFICIENCY TIPS

Unfortunately, speed and memory are two resources we don't have much of on the HX-20. There are a number of ways to make programs run more quickly and use less memory, but these come at the expense of readability. Keep that in mind, if you're going to be making changes to your program later on, you may not want to incorporate the following suggestions.

Before getting into how to tighten a BASIC program, it's worth learning something about what the program looks like in memory.

If you examined memory where your program was stored, you'd find that it looked pretty much the way you typed it in. There are two major differences. One is that each program line is chained to the previous line. That is, the first line of your program contains the address of the next program line and so on. This is so the interpreter can find each line without having to read the entire program. (Of course, some sort of indexing scheme would have been faster yet, but Microsoft BASICs just aren't done that way.)

The second difference is that all BASIC reserved words have been 'tokenized'. That's what makes them reserved words. Every time the interpreter sees you type in 'IF', for instance, it stores an 89 in that program line. This saves both space and time.

Now that we know these two things we can deduce the following:

1. Putting the most used subroutines near the front of the program will speed execution. Why? Because the interpreter will bounce down, from the beginning, to find the line number specified on a GOSUB.
2. To save memory, delete all unnecessary spaces and remarks. Ouch! If you really do this, you'll make the program unreadable. For instance,
   IF A = 12 THEN A$ = "CAT" would run faster as:
   IFA=12THENA$="CAT"
   We haven't done that to any of the programs in this book because we wanted you to be able to understand them. But. . . sometimes this type of compression is necessary. As a matter of fact, we expect someone eventually to market a program that does this compression on the HX-20 automatically, as has been marketed for other computers.
3. Use multiple statements per line. This saves both space and time. Space, because unnecessary program line numbers (and pointers) are not being stored. Time, because the interpreter won't spend time looking at statements that can't be executed until prior statements have been executed.
4. Use short variable names. Ugh again! But it'll save space and a little processing time. To be used only in emergencies.

5. Use NEXT statements with no variable specified, e.g.,
   NEXT
   instead of
   NEXT A
   Various benchmark tests run on Microsoft BASICs have found a speed improvement with operand-less NEXTs.
6. Don't use END statements. They're not necessary for program execution and just take up space.
7. Null out unused string constants. If you've set:
   A$="Good morning"
   and you don't need that string any more, setting:
   A$=" "
   will save you 12 bytes.
8. Define the most used variable names first. The same deal for program lines applies to variables. The ones defined first are kept in memory first. If you reference a variable that was the 18th variable you defined, then the interpreter will have to check the previous 17 variables to get to the 18th.
9. Re-use variable names. This saves space, and some time. Each variable takes up memory, so the fewer variables you have, the more memory is available for other things.
10. ERASE unneeded arrays. You will get back the entire space allocated to an array if you get rid of it when you're finished. Often, BASIC programmers will define their arrays (and their variables) in the beginning of the program. We've learned from the above that it makes more sense to define them at the end. But now we've learned that it is even better not to define an array until you need it.
11. In *Simple BASIC Programs for Business Applications*, J.R.F. Alonso notes that using variables instead of constants makes programs more efficient. By the way, this book is a collection of often-used BASIC business programs, very well explained, and with tips on program efficiency and conversions from one BASIC to another.
12. Use flags for often-tested conditions. If, for instance, you have a variable PRINT$ set to "YES", set PRINT = −1 so that you can say:
    IF PRINT THEN . . .
    rather than
    IF PRINT$ = "YES" THEN . . .
    This is clearer as well as faster.
13. In a sequence of tests (IF conditions), test for the likeliest possibilities first. This cuts down on the number of IFs the program has to execute. Even better if the conditions are mutually exclusive and numeric (or can be converted to numeric) because then you can say ON nn GOSUB. . . which saves space as well as time.

ADDITIONAL HX-20 BASIC TIPS

Most of the tips we've mentioned above are useful on any Microsoft BASIC machine. Here are some that are HX-20-specific.

1. Watch the width. Using WIDTH 44,n is a known bug and can cause problems.
2. Disable the BREAK key. There are times when we don't want the BREAK key hit, accidentally or on purpose. Perhaps some critical operation is running, such as writing data to a cassette. Well, you can make the BREAK key inoperative with:
   POKE 125,4
   (This tip courtsey of T.L. Ronson of the HX-20 Users' Group.)
   To re-enable, do:
   POKE 125,0
   If you set this location ($7D) with MON, then return to BASIC by typing B.
   Location 125 will return to 0 when you go back to the main menu.
3. Return to the Menu. If you want your program to return automatically to the menu, such as at the end of execution, code:
   EXEC &HD23B
   (Tip courtesy of T.L. Ronson of the HX-20 Users' Group.)
4. Define a new DEL key. Don't like the way the DEL key works? (It deletes the character to the right of the cursor instead of the 'more natural' deletion of the character immediately above the cursor.) Well, you can make a new DEL key:
   KEY 1,CHR$(28)+CHR$(8)+CHR$(18)
   This assigns PF Key 1 to a combination of control functions that will do the job. (But note that it doesn't repeat.) This tip provided by the HX-20 Users' Group.
5. Entended insert mode. Want to move the cursor to the left or right but still stay in insert mode?
   KEY3, CHR$(29)+CHR$(18) ' Cursor left
   KEY4, CHR$(28)+CHR$(18) ' Cursor left
   This tip provided by the HX-20 Users' Group
6. Re-arrange dates. Don't like the American

way of printing dates? You can re-arrange the output of DATE$ with the following:
D$=MID$(DATE$4,3)+LEFT$(DATE$,3)
+RIGHT$(DATE$,2)
This gives you, for instance, 20/08/83 instead of 08/20/83.
Tip provided by the HX-20 Users' Group.
7. Produce a random number between 1 and N
R=INT(RND*N)+1

Scrolling v. paging

Just from reading the manual, and without thinking about it, you'd figure that the best, if not only, way to display information on the screen is to let it scroll by. Certainly the easiest way is to PRINT a string, then PRINT another string, and so on. If information rolls off the screen, the operator can always use the SCRN or cursor keys to view the lines he missed (assuming that you as the programmer have used the WIDTH statement to create a virtual screen more than 4 lines long). This is actually what we do with our mini-assembler program. Another alternative is to slow down the scroll rate (with the SCROLL statement) to where a human can read it.

But there's another way to do things, which you might have thought of if you use IBM 3270s or other terminals that display information a screen-full at one time. With this technique, which we'll call paging, the operator will see just a small part of our information — then we wait for him to signal that he's ready to view more — and we give him the next page, and so on. There are three advantages to this method.

First, information that is presented on the screen does not move once it's displayed, whereas scrolled lines that start at the cursor position will move up the screen as each additional line is added. This lack of movement makes the material easier to read.

A second advantage is that the virtual screen size does not have to be more than 4 lines — a saving of memory, which can be important in a 16K RAM computer. Additional memory savings can come from re-using the same strings each time, then rather than having a separate string for each. Of course, if you just say PRINT "xxx" then you're not using any string space at all, though you do use memory, of course.

Third, if different 'pages' are assigned to different keys on the keyboard, the operator can selectively retrieve different sections of the material, without having to sit through the beginning lines each time.

Try this:

```
1000  CLS:F$=A$+SPACE$(20-LEN(A$))
+CHR$(13)+B$
        +SPACE$(20-LEN(B$))+CHR$(13)
        +C$SPACE$(20-LEN(C$))
        +CHR$(13)+D$:PRINT F$;
```

Assign A$, B$, and C$ to any strings. Assign D$=SPACE$(16)+". . .". Now RUN. A$ prints on the first line, B$ prints on the second line, etc. We put the ". . ." in to tell the operator that there's more data to come and he should hit a key to view the next page.

If you add:

```
400   I$=INKEY$
500   IF I$<> " " THEN GOSUB 1000
1100  RETURN
```

then the operator will execute our print routine only when he hits a key (any key). In between prints, i.e., in some other part of the program, we can change the contents of A$, B$, and C$.

We can go a step further. We can 'assign' certain keys to certain pages. For instance:

```
500  IF I$=" " THEN 400
510  IF I$="1" THEN A$="First Line": B$="Second
     Line":
     C$="Third Line": D$=SPACE$(16)+". . .":
     GOTO 590
520  IF I$="2" THEN A$="Fourth Line":B$="Fifth
     Line":
     C$="Sixth Line":D$=SPACE$(16)+". . .":
     GOTO 590
530  A$="Hello": B$="Good-bye":C$="That's
     all,":D$="Folks"
590  GOSUB 1000
600  GOTO 400
```

One more thing on this subject — you don't need to rewrite the screen every time if most of the information remains the same. If you wanted to blank out the third line of the display, for instance, you could code:

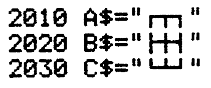
```
LOCATE 0,2:PRINT SPC(20):
```

In the more general case, you would code:

```
LOCATE X,VAL(Y$):PRINT E$;
```

where Y$ is a number entered from the keyboard (Y$=INKEY$) or passed from the main program. E$ can be equal to spaces (to blank out the selected line). X can be 0 to replace an entire line or some other number for only a partial replace.

This technique is also applicable to graphics. For instance:



```
2010 A$="┌┬┐"
2020 B$="├┼┤"
2030 C$="└┴┘"
```

Fig. 4.1

Then when we execute our 1000 subroutine, a composite shape will be displayed.

Tables of shapes can be built by assigning each screen line to a string and then assigning a string to the entire shape. The shape can then be assigned to a key and recalled with a keystroke.

We can control this even further: a draw program could be written that would ask for the location of each new shape to be placed in the virtual or physical screen. (Remember: INKEY$ will not echo back to the screen what the user types in, so what's already on the screen will not be overwritten.)

SOME BASIC SUBROUTINES AND PROGRAMS

The routines that follow have been written to satisfy some of the typical business, engineering, and general needs that you might have. You can run them as they stand or incorporate them within your own programs. Or just use them as sparks for your own ideas. Like everything else in this book, they may be freely copied for your own use, but not resold.

String-to-ASCII conversion

Put the ASCII values of the string WORD$ into an array XX.

```
FOR CTR = 1 TO LEN)WORD$)
XX(CTR) = ASC(MID$(WORD$,CTR,1))
NEXT
```

Hex conversion to integer

Epson/Microsoft BASIC supplies a means to convert an integer into a hex value: HEX$. But there is no means to go the other way. Surprisingly enough, one of the ROM routines will do a hex-to-integer conversion. Or we can use a BASIC subroutine. Here's a simple way to take a hex number in a variable A$ (up to 255) and put its decimal equivalent into variable B:

```
2000  CB$="0123456789ABCDEF"
2010  B=0
2020  FOR CX=1 TO 16
2030  IF MID$(A$,2*LL+1,1)=MID$(CB$,CX,1)
  THEN B=(CX−1)*16 ELSE NEXT
2040  FOR CX=1 TO 16
2050  IF MID$(A$,2*LL+2,1)=MID$(CB$,CX,1)
  THEN B=(CX−1)+B ELSE NEXT
2060  RETURN
```

Searching

Locating a particular item out of a set of similar items is a function found in most applications that do any data management: inventory, mailing lists, etc.

The most commonly used technique (and the easiest to code) is the serial search: start at the beginning and examine each item one by one till a match is found.

Serial search — unsorted table

This example assumes that our information is in integer format in an array TBLE, but that it is not in any particular order.

Comments:
NUM is the number of items in the table
ARG is the element we're looking for
FOUND is set true or false depending on the result, i.e., in the mainline program you can code:

```
IF FOUND THEN . . .
FOUND=0
FOR INDX=1 TO NUM
IF  ARG=TBLE(INDX)
THEN FOUND=−1 ELSE NEXT
RETURN
```

Serial search — sorted table

If the items in the table are in order, then our search will be faster because we can stop as soon as the next table element to be examined is higher than our argument. (The mini-assembler program in this book uses this technique, but with strings.)

```
FOUND=0
FOR INDX=1 TO NUM
IF ARG=TBLE(INDX) THEN FOUND=INDX
  ELSE IF ARG > TBLE(INDX) THEN NEXT
RETURN
```

Upon completion, FOUND will be set to the number of the array element if matched.

Here's a simple inventory application of this technique:

```
TBLE(1)=1001
TBLE(2)=1014
TBLE(3)=1231
TBLE(4)=2019

TBLE$(1)="BIG ONES"
TBLE$(2)="MEDIUM-Sized Ones"
TBLE$(3)="Little Ones"
TBLE$(4)="tiny ones"

ARG=1014
NUM=4
GOSUB . . . 'SEARCH
IF FOUND THEN PRINT TBLE$(FOUND)
  ELSE PRINT "Unknown Item"
```

(**Note** Any non-zero value is considered by the BASIC interpreter to be 'true'.)

Binary search — sorted table

If execution speed is important, and with the HX-20 it often is, then a binary search is best. Instead of starting with the first item, a binary search starts with the middle item. Once you know that the argument is higher, for instance, than the middle item, you can disregard the first half of the table. Then you check the middle item in the remaining half, and so on.

```
1010 LOW=0
1020 HIGH=NUM+1
1030 FOUND=0
1040 IF ARG < TBLE(1) OR ARG > TBLE(NUM) THEN
     RETURN
1050 IF HIGH >= LOW THEN HALF=INT((HIGH+
LOW)/2):
         IF TBLE(HALF) <> ARG
           THEN IF TBLE(HALF) > ARG
             THEN HIGH = HALF-1:GOTO 1050
           ELSE LOW=HALF+1:GOTO 1050
         ELSE FOUND=HALF
       ELSE FOUND=HIGH
1060 IF ARG <>TBLE(FOUND) THEN FOUND=0
1070 RETURN
```

Serial proximity search — sorted table

A different type of search returns not a matching value, but the range of values into which the argument fails. This is particularly useful when you're not sure how to spell the search argument.

The first example uses numeric (integer) values, the second works with strings.

For the first example, the data is assumed to be in TBLE(1) to TBLE(NUM). ARG is the value to search on. WDTH is the number of table elements on each side of the argument to return. NUM is the number of elements in the table.

Sample application:

```
 110 TBLE(1)=11
 120 TBLE(2)=20
 130 TBLE(3)=32
 140 TBLE(4)=45
 150 TBLE(5)=70
 160 TBLE(6)=120
 170 TBLE(7)=140
 180 TBLE(8)=300
 190 ARG=75
 200 WDTH=2
 210 NUM=8
 220 GOSUB 1000
 230 PRINT TBLE(LOW);"-";TBLE(HIGH)
 240 STOP
1000 IF ARG<TBLE(1) THEN LOW=1:HIGH=1:
     GOTO 1100
1010 FOR INDX=1 TO NUM
1020 IF ARG>TBLE(NUM) THEN LOW=NUM:
     HIGH=NUM:GOTO 1100
1030 IF ARG<=TBLE(INDX) THEN 1060
1040 NEXT INDX
1050 INDX=NUM+1
1060 IF ARG=TBLE(INDX)
     THEN HIGH=IND+WDTH ELSE HIGH =
     INDX+WDTH-1
1070 LOW=INDX-WDTH
1080 IF HIGH>NUM THEN HIGH=NUM
1090 IF LOW<1 THEN LOW=1
1100 RETURN
```

```
 110 TBLE$(1)="SMITH"
 120 TBLE$(2)="SMITTY"
 130 TBLE$(3)="SWARTZ"
 140 TBLE$(4)="TELEPHONE"
 150 TBLE$(5)="ZENITH"
 190 ARG$="SZERCSKI"
 200 WDTH=1
 210 NUM=5
 220 GOSUB 1000
 230 PRINT TBLE$(LOW);"-";TBLE$(HIGH)
 240 STOP
1000 IF ARG$<TBLE$(1) THEN
     LOW =1:HIGH=1:GOTO 1100
1010 FOR INDX=1 TO NUM
1020 IF ARG$>TBLE$(NUM) THEN
     LOW=NUM:HIGH=NUM:GOTO 1100
1030 IF ARG$<=TBLE$(INDX) THEN 1060
1040 NEXT INDX
1050 INDX=NUM+1
1060 IF ARG$=TBLE$(INDX) THEN
     HIGH=INDX+WDTH
     ELSE HIGH=INDEX+WDTH-1
1070 LOW=INDX-WIDTH
1080 IF HIGH>NUM THEN HIGH=NUM
1090 IF LOW<1 THEN LOW=1
1100 RETURN
```

Sorts

There are all kinds of ways to sort data into order. The most popular way is the 'bubble' sort, but this method, while simple, is very slow. An exchange sort is not any more complex and is quite a bit faster. According to *Basic Statistical Computing*, by Cooke, Craven, and Clarke (Edward Arnold, 1982), benchmark times for exchange sorts compare favourably with other methods of sorting as long as the number of items is 25 or less. At 100 items, the time required is about 4 times that of the fastest method, the 'quicksort'. At 500 items, the time is 16 : 1.

An exchange sort will take the smallest item in the list and exchange it with the first item, and then do the same for the second item, and so on.

A quicksort would keep breaking the list down into two groups. Basically, it would find the midpoint, put all less than that one midpoint in one group, all greater than or equal to the midpoint in the other group, then break down each group again and so forth, until each group is of a size that could be managed by an exchange sort.

The following is an example of an exchange sort. We'll leave it to the reader to write the quicksort program.

Sample data for an exchange sort:
SMITH, J.
COLLINS, E.
QUIP, G.
FRANK, A.
STORK, Y.
SMITH, U.
BART, C.

Smith, J. and Bart, C. will be exchanged, then QUIP and FRANK, etc.

Note If sorting names, you should remove any spaces in the name (use INSTRING$). Or, be certain that names are always defined the same way. Otherwise:
SMITH, U. will sort out ahead of SMITH, J. because a space is lower in the ASCII sorting order than a J.

```
10  ' Exchange Sort — ASCII
100 ' Program takes data from an array, but can be
      converted to take data from RAM, keyboard or
      cassette
110 ' ITEMS$ is the array in which the data has been
      stored
120 ' NUMITEMS is the number of data items
130 ' ITEMNO is data item being processed
140 ' LOW is the number of the lowest item found in
      the remainder of the list, i.e., the one that will
      be used in the exchange
150 ' SEARCHITEM is the number of the next item to
      be looked at in the search process
200 FOR ITEMNO = 1 TO NUMITEMS — 1
210 LOW = ITEMNO
220 FOR SEARCHITEM = ITEMNO + 1 TO
    NUMITEMS
230 IF ITEM$(LOW) > ITEMS$(SEARCHITEM)
    THEN LOW = SEARCHITEM
240 NEXT SEARCHITEM
250 SWAP ITEMS$(ITEMNO), ITEM$( LOW)
260 NEXT ITEMNO
270 STOP
```

Titles

Centring a string on a line is an often-used function for creating titles, e.g., report headings, graph labels. Here's an easy 1-liner to do it. (If you're going to a printer, replace each 20 in the program with the width of the print line, e.g., 24 for the microprinter, 80 for an MX-80.)

```
500 TITL$=SPACE$(INT(20-LEN(X$))/2)
      +X$+SPACE$(INT(20-LEN(X$))/2)
```

Graphics

Find the greatest common divisor among a group of numbers

In doing bar graphs, you might prefer the divisions to come at integer intervals. Let's say your data looks like this:

#1 = 117
#2 = 234
#3 = 252
#4 = 81

Each division could be 10, for instance — but then you won't be able to tell what the actual amounts are just by looking at the graph. But if we knew that some other number, 9 for instance, went evenly into each amount — we could make the divisions multiples of 9. The following subroutine will find this greatest common divisor.

As before, the data is assumed to be in an array TBLE, starting with item #1. NUM is the number of items. GCD will be the greatest common divisor, the number we're looking for.

```
510 N1=TBLE(1) 'Initialization
520 GCD=N1
530 FOR I=2 TO NUM 'Main Loop
540 N2=TBLE(I)
550 IF N1=0 or N2=0 THEN GCD=0:GOTO 580
    'Test for a 0 entry
560 QUOTIENT=INT (N1/N2):
    RMDR=N1-2*QUOTIENT: IF RMDR<>0 THEN
    N1=N2:N2=RMDR:GOTO 560 ELSE GCD=N2
570 NEXT I
580 RETURN
```

Line graph

The values to be plotted are in two arrays: X and Y. NUM is the number of pairs of values. The data must be normalized first, to enable it to fit on the screen.

```
CLS
YLOW=Y(1)
YHIGH=Y(1)
FOR INDX=2 TO NUM
IF Y(INDX) < YLOW THEN YLOW=Y(INDX)
IF Y(INDX) > YHIGH THEN YHIGH = Y(INDX)
NEXT INDX
SCALE=YHIGH-YLOW
IF SCALE=0 THEN SCALE=1
PSET (X(1),Y(1))
FOR INDX=2 TO NUM
LINE-(X(INDX),Y(INDX)),PSET
NEXT INDX
RETURN
```

To make this line graph into a scatter plot, just change the third line from the bottom to:

```
PSET(X(INDX),Y(INDX)
```

Data Normalization

The following routine will keep your data proportional — a process called normalization.

MAX = top end (e.g. 0–120) for the values

```
2000 SUM=0
2010 FOR INDX=1 TO NUM
2020 SUM = SUM + TBLE)INDX)
2030 NEXT INDX
2040 FOR INDX=1 TO NUM
2050 TBLE(INDX)=INT(TBLE(INDX)/SUM*MAX)
2060 NEXT INDX
```

Bar chart

To create a bar chart after normalizing the data.

Input is TBLE
NUM = # of values (1–9)

```
1000 CLS
1005 IF NUM=<3 THEN SPAC=8 ELSE IF NUM=4
     THEN SPAC=6 ELSE IF NUM=5 THEN
     SPAC=5 ELSE IF NUM=6 THEN SPAC=4 ELSE
     IF NUM=7 THEN SPAC=4 ELSE IF NUM>=8
     THEN SPAC=3
1010 FOR X=4 TO 100 STEP 5
1020 PSET (X,31)
1030 NEXT
1040 FOR X=9 TO 100 STEP 10
1050 PSET (X,30)
1055 NEXT
1060 FOR X=4 TO 100 STEP 5
1070 PSET (X,0)
1080 NEXT
1090 FOR X=9 TO 100 STEP 10
1100 PSET (X, 1)
1110 NEXT
1120 FOR INDX=1 TO NUM
1130 LINE (0,INDX*SPAC)-(TBLE(INDX)-1,
     INDX*SPAC),PSET
1140 NEXT
1150 RETURN
```

Graphics demonstration program

This is a cute little demonstration of the HX-20's graphics capability. You'll understand why it's called 'New York' after you see it run. Original version courtesy of the HX-20 Users' Group.

```
 10 TITLE "NEW YORK"
 20 RANDOMIZE VAL (RIGHT$(TIME$,2))
 40 CLS
 50 LINE(0.31)-(120,31),PSET:LINE(0,30)-
    (120,30),PSET
 60 FOR A=0 TO 120
 70 B=20+(INT(RND*24))
 80 FOR C=B TO 32)
 90 PSET(A,C):PSET(A+1,C):PSET(A+2,C)
100 NEXTC
110 NEXTA
120 G=INT (RND*35)+1
130 IF G<34 THEN 120
140 D=INT (RND*120)+1
150 H=INT(RND*20)+1
160 IF H=2 THEN 230
170 FOR E=0 TO 32
180 PSET (D,E)
190 PRESET (D,E-2)
200 IF E=31 THEN SOUND 15,1
210 NEXT E
220 GOTO 120
230 FOR E=0 to 32
240 line (D,E-2)-(D+2,E-2),PSET
250 LINE(D,E-4)-(D+2,E-4),PRESET
260 IF E=31 THEN SOUND, 3,4
270 NEXTE
280 LINE(D,E-2)-(D+2,E-2)PRESET
290 LINE(D,29;-(D+2,29),PRESET
300 LINE(D,31)-D+2,31),PSET
310 LINE(D,30)-D+2,30),PRESET
320 GOTO 120
```

Graphic character definition

Epson's documentation discusses how to assign graphics characters to the numeric keys. But the HX-20 Users' Group finds the example limited and offers the following:

```
10 MEMSET &H0A80
20 POKE & H011E,&H0A 'Give starting address
30 POKE &H011F,&H40 ' of character list
40 POKE &H0A40,92 ' Design an omega pattern
50 POKE &H0A41,98
```

```
60 POKE &H0A42,2
70 POKE &H0A43,98
80 POKE &H0A44,92
90 POKE &H0A45,0
```

Each of the 6 bytes that defines a character describes a single vertical column. For instance, decimal 92 is hex 5C or binary 0101 1100. This means that the uppermost dot in this column is off, the next dot is on, the next off, the next on, the next on, the next off, the next off. If you want to keep a 1-dot space around your character, make sure that, as in the above example, the lowest bit in each column is off and also that the last byte has each bit turned off.

Note that the program need only be run on a cold start (i.e., CTL/@) or when changing MEMSET. The data can also be typed in via MON, of course.

See Chapter 11, Software and Systems, for information on programs from King Software and Kuma Computers that make character definition easier.

Alarm clock program

This program was provided by the HX-20 Users' Group. You're asked for the time you want to set (use double quotes around the time) and a short tune will play when that time is reached. The clock continues to display.

```
 10 TITLE "ALARM"
 20 CLS:PRINT "Time now is ";TIME:PRINT " Input
    time wanted":PRINT " e.g ' ' '";
    TIME$;" ' ' ":INPUT "   ";AA$
 30 DISPLAY$=TIME$:CLS:PRINTTIME$
 40 IF DISPLAYS$=AA$ THEN GOSUB 70
 50 IF DISPLAYS$=TIME$ THEN 50
 60 GOTO 30
 70 FOR I=1 to 16
 80 READ A,D
 90 SOUND A,D
100 DATA 2,3,4,5,5,6,7,8,1,2,4,7,1,2,8,5,9,2,4,3,5,4,2,
    1,6,5,7,5,2,8,3,9,2,3  ' don't leave any blanks
110 NEXT I
120 RETURN
```

STATISTICS/FINANCE

This routine will find an average:

```
SUM=0
FOR INDX=1 TO NUM
SUM=SUM+TBLE(INDX)
NEXT INDX
AVG=SUM/NUM
```

A weighted moving average is useful for stock tracking, because it assigns a higher priority to the more recent readings.

```
10 ' Compute weighted moving averages
120 PRINT "WEIGHTED M.A.s"
200 MA=0
210 INPUT "Number of readings:"; N
220 FOR A = 1 TO N
230 INPUT "DATA: ";I
240 MA=MA+(I*A)
250 NEXT
260 PRINT N;"day weighted"
270 PRINT "moving average is:"
280 FOR A=1 TO N
290 B=B+A
295 NEXT
300 PRINT MA/B
```

A smoothed exponential mean is a shortcut to computing a weighted moving average. It's probably most handy when you don't have all the previous readings. The first computation only requires one reading, the current day's and the previous day's.

```
10 ' Program to compute smoothed exponential
     mean
140 INPUT "Smoothing constant: ";S
150 INPUT "Previous SEM ";P
160 INPUT "Current data ";C
170 SEM=S*(C-P)+P
180 PRINT "New SEM is: ";SEM
```

The smoothing constant referred to above can be computed by dividing the length of the term into 2 and adding 1. For instance, the smoothing constant for a 10 day average would be 2/(10+1) or 0.18. Reference: *Winning Stock Selection Systems* by Gerald Appel (Boardroom Reports, 1979).

Reduction loans

Loans in which each payment includes part of the principal (as opposed to 'balloon' loans) are quite common.

For this routine, enter a number for all known amounts and a '?' for the value you wish to determine: payment amount, amount that can be borrowed, number of payments.

```
10 '  General reduction loan formula
20 '  Author: Eric Balkan
30 '  P=A*(I/(1-(1+I)^-N))
40 '  P=periodic payment
50 '  A=amount borrowed
60 '  I=interest in decimal (e.g. .10 rather than 10%)
70 '  N=number of payments
80 '  E.g., 30 yr mortgage loan for $100,000 at 12.5%,
90 '  solved for P:
110 ' P = 100000 * ((1-(1+.125)^-360)/.125
120  PRINT "GENERAL LOAN PROGRAM"
130  PRINT "Enter number or ?"
210  LINEINPUT "Payment Amount"; P$
220  IF P$="?" THEN FP=1 ELSE S=INSTR(P$,","):
     IF S=0 THEN P=VAL(P$)
     ELSE P=VAL(LEFT$(P$,S-1)+MID$(P$,S+1)
240  LINEINPUT "Amount Borrowed? ";A$
250  IF A$="?" THEN FA=1 ELSE
     S=INSTR(A$,","):IF S=0 THEN A=VAL(A$)
     ELSE A=VAL(LEFT$(A$,S-1)+MID$(A$,S+1))
270  LINEINPUT "# of Payments? ";N$
280  IF N$="?" THEN ZN=1 ELSE N=VAL(N$)
300  LININPUT "Interest Rate? ";I$
310  IF I$="?" THEN PRINT "NOT AVAILABLE":
     GOTO 430
320  IF RIGHT$(I$,1)="%"
     THEN I$=LEFT$(I$,LEN(I$)-1)
330  I=VAL (I$)/1200
400  IF FP THEN P=A*(I/(1-(1+I)[-N):
     PRINT "PAYMENT: ";P:GOTO 430
410  IF FA THEN A=P*(1-(1+I)[-N)/I):
     PRINT "AMOUNT: ";A:GOTO 430
415  N=LOG (1-I*A/P)
418  N=-N/LOG(1+I)
422  PRINT "# OF PAYMENTS: ";INT(N)
430  END
```

ELECTRONICS

Decibel/Power/Voltage conversion program

This program converts decibels to voltage (current)* and power ratios and vice versa. These parameters are used when measuring circuit gain or attenuation.
***Note** Voltage gain = current gain

```
10 '  "DB Conversion" by Mark Weber
20 '  Version 1.2, 10/21/83
30   CLS
40   PRINT"        *  *  *"
50   PRINT"        Decibel"
60   PRINT" Voltage/Power Gain"
70   PRINT" Conversion Program";
80   FOR T=0 TO 800:NEXT
90   CLS
100  PRINT 'Convert from:"
110  LOCATE 4,2
120  PRINT "(Choose One)"
130  FOR T=0 TO 400:NEXT
140  CLS
150  PRINT "1) Voltage to dB"
160  PRINT "2) Power to dB"
170  PRINT "3) dB to Voltage"
180  PRINT "4) dB to Power";
190  A$=INKEY$
200  IF A$="1" THEN B$="volts":C$="Voltage":
     GOTO 250
210  IF A$="2" THEN B$="watts":C$="Power":
     GOTO 250
220  IF A$="3" THEN C$="Voltage":GOTO 580
230  IF A$="4" THEN C$="Power":GOTO 580
240  GOTO 190
250  CLS
260  INPUT "Initial value";VI
270  IF VI <= 0 THEN 250
280  CLS
290  INPUT "Final value";VF
300  IF VF <= 0 THEN 280
310  DB=20*LOG(VF/VI*.4342943
320  CLS
330  PRINT "Do you want a"
340  PRINT "printout?"
350  PRINT "        1) Yes"
360  PRINT "        2) No";
370  D$=INKEY$
380  IF D$="1" THEN E=1:GOTO 410
390  IF D$="2" THEN E=0:GOTO 410
400  GOTO 370
410  CLS
420  IF INKEY$="1"THEN 140
430  PRINT "Initial value:"
440  LOCATE 6,1:PRINT VI;:PRINT B$
450  PRINT"Final value:"
460  IF VF >= 10000 THEN H=4 ELSE H=5
470  LOCATE H,3:PRINT VF;:PRINT B$;
480  IF D=1 THEN COPY
490  FOR T=0 TO 400:NEXT
500  CLS
510  IF INKEY$="1" THEN 140
520  PRINT C$ + " Gain:"
530  LOCATE 6,1:PRINT DB;:PRINT "DB"
540  IF D$="2" THEN LOCATE, 1,3: PRINT "(" +
     CHR$(34) + "1" + CHR$(34) + " to restart)";
550  IF D$="1" THEN COPY: GOTO 140
560  FOR T=0 TO 400: NEXT
570  GOTO 410
580  CLS
590  INPUT "Enter dB value:", DB
600  IF A$="3" THEN RATIO=EXP(DB/8.685889)
610  IF A$="4" THEN RATIO=EXP(DB/4.3429445)
620  CLS
630  PRINT "Do you want a"
640  PRINT "printout?"
650  PRINT "        1) Yes"
660  PRINT "        2) No";
670  D$=INKEY$
680  IF D$="1" THEN E=1:GOTO 710
690  IF D$="2" THEN E=0:GOTO 710
700  GOTO 670
710  CLS
720  IF INKEY$="1" THEN 140
730  PRINT C$ + " Gain:"
740  LOCATE7,1:PRINT DB;:PRINT "dB"
750  PRINT C$ + "Ratio:"
760  IF RATIO < 1 THEN RATIO = 1/RATIO
770  IF RATIO> 1E+6 THEN H=2 ELSE H=6
780  IF DB < 0 THEN LOCATE H,3: PRINT
     "1:";:PRINT RATIO;
```

```
790  IF DB >= 0 THEN LOCATE H,3:PRINT
     RATIO;:PRINT":1";
800  IF D$="1" THEN COPY:GOTO 140
810  FOR T=0 TO 500 :NEXT
820  CLS
830  IF INKEY$="1" THEN 140
840  LOCATE 1,2:PRINT "(" + CHR$(34) + "1"
     + CHR$(34) + " to restart)";
850  FOR T=0 TO 300:NEXT T
860  GOTO 710
```

Reactance Chart Program

This program simulates a standard reactance chart. It will calculate the capacitance, capacitive reactance, inductance or inductive reactance, at a given frequency, given the other corresponding paramenter. These calculations are used to design frequency-sensitive circuits.

```
10   CLS
20   ' "Reactance Chart" by Mark Weber
30   ' Version 1.1 — 10/21/83
40   PRINT "        * * *"
50   PRINT "  Reactance Chart"
60   PRINT "     Program"
70   PRINT "        * * *"
80   FOR T=0 to 500: NEXT
90   CLS
100  PRINT "What is the unknown?"
110  LOCATE 4,2:PRINT "(Choose One)"
120  FOR T=0 to 300:NEXT
130  CLS
140  PRINT "1) Capacitance"
150  PRINT "2) Inductance"
160  PRINT "3) Cap. Reactance"
170  PRINT "4) Ind. Reactance"
180  A$=INKEY$
190  ON VAL (A$) GOTO 210,210,210,210
200  GOTO 180
210  CLS:SCROLL 9,0
220  PRINT "Do you want a"
230  PRINT "printout?"
240  PRINT "           1) Yes"
250  PRINT "           2) No"
260  D$=INKEY$
270  IF D$="1" OR D$="2" THEN GOTO 300
290  GOTO 260
300  CLS
310  PRINT "Enter the frequency"
320  PRINT "in Hertz (cycles)"
330  PRINT "per second)."
340  INPUT F
350  IF F<=0 THEN GOTO 300
360  ON VAL (A$) GOTO 380,700,1020,1350
370  GOTO 370
380  CLS
390  PRINT "Enter the capacitive"
400  PRINT "reactance in ohms."
410  INPUT XC
420  IF XC <= 0 THEN 380
430  C=1/(6.28318*F*XC)
440  W$="farads":CA=C
450  IF C < .1 THEN CA=C*1E6:W$="uf"
460  IF C < 1E-9 THEN CA=C*1E12:W$="pf"
470  GOSUB 3000
500  XCA=XC:WC$="ohms"
510  IF XC>= 1E3 THEN
     XCA=XC/1E3:WC$="kilohms"
520  IF XC >= 1E6 THEN
     XCA=XC/1E6:WC$="megohms"
530  CLS:SCROLL 9,0,10,4
540  IF INKEY$="1" THEN 130
550  PRINT "The capacitance is:"
560  LOCATE 2,2:PRINT CA;W$
570  GOSUB 3100
580  GOSUB 2000
610  IF INKEY$="1" THEN 130
620  PRINT "For a frequency of:"
630  LOCATE 3,1:PRINT FA;WF$
640  SCROLL 9,0:PRINT "and cap. react. of:"
650  LOCATE 3,3: PRINT XCA;WC$;
660  IF D$="1" THEN COPY:GOTO 130
680  FOR T=0 TO 400:NEXT
690  GOTO 530
700  CLS
710  PRINT "Enter the inductive"
720  PRINT "reactance in ohms."
730  INPUT XL
740  IF XL <= 0 THEN 700
750  L=XL/(6.28318*F)
760  W$="henrys":LA=L
770  IF L < 1 THEN LA=L*1E3:W$="mH"
780  IF L < 1E-3 THEN LA=L*1E6:W$="uH"
790  GOSUB 3000
820  XLA=XL:WL$="ohms"
830  IF XL >= 1E3 THEN
     XLA=XL/1E3:WL$="kilohms"
840  IF XL >= 1E6 THEN
     XLA=XL/1E6-WL$="megohms"
850  CLS:SCROLL 9,0
860  IF INKEY$="1" THEN 130
870  PRINT "The inductance is:"
880  LOCATE 2,2:PRINT LA;W$
890  GOSUB 3100
900  GOSUB 2000
930  IF INKEY$="1" THEN 130
940  PRINT "For a frequency of:"
950  LOCATE 3,1:PRINT FA;WF$
960  SCROLL 9,0:PRINT "and ind. react. of:"
970  LOCATE 2,3:PRINT XLA;WL$;
980  IF D$="1" THEN COPY:GOTO 130
1000 FOR T=0 to 400:NEXT
1010 GOTO 850
1020 CLS
1030 PRINT "Enter the capa-"
1040 PRINT "citance in farads."
1050 INPUT C
1060 IF C <= 0 THEN 1020
1070 XC = 1/(6.28318*F*C)
```

```
1080 W$="ohms":XCA=XC
1090 IF XC >= 1E3 THEN
     XCA=XC/1E3:W$="kilohms"
1100 IF XC>= 1E6 THEN
     XCA=XC/1E6:W$="megohms"
1110 FA=F:WF$="Hz"
1120 IF F>= 1E3 THEN FA=F/1E3:WF$="kHz"
1130 IF F >= 1E6 THEN FA=F/1E6:WF$="MHz"
1140 CA=C:WC$="farads"
1150 IF C < .1 THEN CA=C*1E6:WC$="uf"
1160 IF C< 1E-9 THEN CA=C*1E12:WC$="pf"
1170 CLS:SCROLL 9,0
1180 IF INKEY$=1" THEN 130
1190 PRINT "The capacitive"
1200 PRINT "reactance is :"
1210 PRINT XCA;W$
1220 GOSUB 3100
1230 GOSUB 2000
1260 IF INKEY$="1" THEN 130
1270 PRINT "For a frequency of:"
1280 LOCATE 3,1:PRINT FA;WF$
1290 SCROLL 9,0:PRINT     "and capacitance of:"
1300 LOCATE 2,3:PRINT CA;WC$;
1310 IF D$="1" THEN COPY:GOTO 130
1330 FOR T=0 TO 400 :NEXT
1340 GOTO 1170
1350 CLS
1360 PRINT "Enter the induc-"
1370 PRINT "ance in henrys."
1380 INPUT L
1390 IF L<=0 THEN 1350
1400 XL=6.28318*F*L
1410 W$="ohms":XLA=XL
1420 IF XL>=1E3 THEN XLA=XL/1E3:W$="kilohms"
1430 IF XL>=1E6 THEN
     XLA=XL/1E6:W$="megohms"
1440 GOSUB 3000
1470 LA=L:WL$="henrys"
1480 IF L < 1 THEN LA=L*1E3:WF$="mH"
1490 IF L < 1E-3 THEN LA=L*1E6:WL$="uH"
1500 CLS:SCROLL 9,0
1510 IF INKEY$="1" THEN 130
1520 PRINT "The inductive"
1530 PRINT "reactance is:"
1540 PRINT XLA:W$
1550 GOSUB 3100
1560 GOSUB 2000
1590 IF INKEY$="1" THEN 130
1600 PRINT "For a frequency of:"
1610 LOCATE 3,1:PRINT FA;WF$
1620 SCROLL 9,0:PRINT     "and inductance of:"
1630 LOCATE 2,3:PRINT LA;WL$;
1640 IF D$="1" THEN COPY:GOTO 130
1660 FOR T=0 TO 400 :NEXT
1660 GOTO 1500
2000 IF D$="1" THEN COPY
2010 FOR T=0 TO 600:NEXT
2020 CLS:SCROLL 9.0
2090 RETURN
3000 FA=F:WF$="Hz"
3010 IF F >= 1E3 THEN FA=F/1E3:WF$="kHz"
3020 IF F >= 1E6 THEN FA=F/1E6:WF$="MHz"
3090 RETURN
3100 IF D$="2" THEN LOCATE 1,3:PRINT "(" +
     CHR$(34) + "1" + CHR$(34) + " to restart)";
3190 RETURN
```

Filter design program

This program is a design aid for active multiple feedback type filters. The multiple feedback filter has several advantages over other types of filters and is used extensively in audio applications.

```
10   ' "Multiple Feedback Filter Design" by Mark
     Weber
20   ' Version 1.0 — Oct 31, 1983
30   CLS
40   PRINT "     * * *"
50   PRINT "  Filter Design"
60   PRINT "     Program"
70   PRINT "     * * *";
```

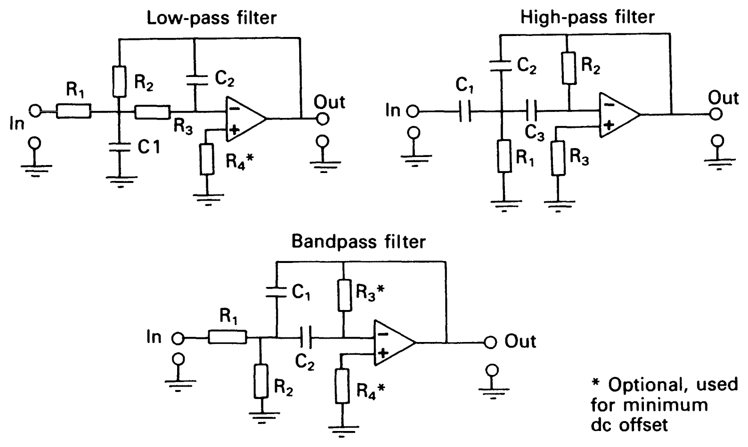


Fig. 4.2 Filters

```
80   FOR T=0 TO 500:NEXT T
90   CLS
100  PRINT"Type of filter:"
110  PRINT"  1) Low-pass"
120  PRINT"  2) High-pass"
130  PRINT"  3) Band-pass";
140  A$=INKEY$
150  IF A$="1" THEN 190
160  IF A$="2" THEN 510
170  IF A$="3" THEN 820
180  GOTO 140
190  GOSUB 1330
200  F$="high":GOSUB 1380
210  GOSUB 1430
220  GOSUB 1530
230  GOSUB 1590
240  C2=C1/(4*Q^2*(H+1))
250  R2=1/(C12.5664*F*Q*C2)
260  R1=R2/H
270  R3=R2/(H+1)
280  R4=(R1*R2)/(R1+R2)+R3
290  CA=C1:GOSUB 1640:C1=CA:WC1$=WC$
300  CA=C2:GOSUB 1640:C2=CA:WC2$=WC$
310  GOSUB 1690
320  IF P$="1" THEN PRINT:PRINT:PRINT:
     PRINT"Low Pass Filter:";:COPY:CLS
330  IF P$="1" THEN PRINT "Gain=";H;" Q=";Q
340  IF P$="1" THEN PRINT "Freq=;F;"Hz"
350  PRINT"C1=";C1;WC1$
360  PRINT"C2=";C2;WC2$
370  IF P$="2" THEN GOSUB 1880
380  IF P$="1" THEN COPY
390  FOR T=0 TO 500:NEXT T
400  GOSUB 1690
410  PRINT"R1=";INT(R1);"ohms"
420  PRINT"R2=";INT(R2);"ohms"
430  PRINT"R3=";INT(R3);"ohms"
440  PRINT"R4=";INT(R4);"ohms"
450  IF P$="1" THEN COPY:GOTO 480
460  FOR T=0 TO 500:NEXT T
470  GOTO 310
480  GOSUB 1840
490  GOSUB 1700
500  GOTO 490
510  GOSUB 1330
520  F$="low":GOSUB 1380
530  GOSUB 1430
540  GOSUB 1530
550  GOSUB 1590
560  C2=C1/H
570  C3=C1
580  R1=1/(6.2832*F*Q*C1*(2*H+1))
590  R2=(Q*(2*H+1))/(6.28318*F*C1)
600  CA=C1:GOSUB 1640:C1=CA:WC1$=WC$
610  CA=C2:GOSUB 1640:C2=CA:WC2$=WC$
620  GOSUB 1740
630  IF P$="1" THEN PRINT:PRINT:PRINT:
     PRINT"High Pass Filter:";:COPY:CLS
640  IF P$="1" THEN PRINT "Gain=";H;" Q=";Q
650  IF P$="1" THEN PRINT "Freq=";F;"Hz"
660  PRINT "C1=";C1;WC1$
670  PRINT "C2=";C2;WC2$;
680  IF P$="2" THEN GOSUB 1880
690  IF P$="1" THEN COPY
700  FOR T=0 TO 500:NEXT T
710  GOSUB 1740
720  PRINT"C3=";C1;WC1$
730  PRINT"R1=";INT(R1);"ohms"
740  PRINT"R2=";INT(R2);"ohms"
750  PRINT"R3=";INT(R2);"ohms"
760  IF P$="1" THEN COPY:GOTO 790
770  FOR T=0 TO 500: NEXT T
780  GOTO 620
790  GOSUB 1840
800  GOSUB 1750
810  GOTO 800
820  GOSUB 1330
830  CLS
840  PRINT"Which do you want":PRINT"  to
     specify:"
850  PRINT"1) -3dB freq pts"
860  PRINT"2) Q & center freq";
870  A$=INKEY$
880  IF A$="1" THEN 990
890  IF A$="2" THEN 910
900  GOTO 870
910  GOSUB 1480
920  IF (2*Q^2-H<=0 THEN SOUND 10,4:CLS:
     PRINT"Q too small. Must":PRINT"be greater
     than":PRINT SQR(H/2):FOR T=0 TO 700:NEXT
     T: GOTO 910
930  IF Q>10 THEN CLS:PRINT"Value of Q too":
     PRINT"large, must be 10":PRINT"or less.":FOR
     T= 0 TO 600: NEXT T:GOTO 910
940  CLS
950  PRINT"What is the"
960  PRINT"center frequency,":PRINT"in Hertz."
970  INPUT FC
980  IF FC<=0 THEN SOUND 10,4:GOTO 940 ELSE
     1060
990  F$="high":GOSUB 1380:FH=F
1000 F$="low":GOSUB 1380:FL=F
1010 IF(FH-FL)<=0 THEN CLS:SOUND 10,4:
     PRINT"YOU HAVE JUST":
     PRINT"DESIGNED A NO-PASS":
     PRINT"FILTER. Try again.":FOR T=0 TO 700:
     NEXT T:GOTO 990
1020 FC=(FH-FL)/2+FL
1030 Q=FC/(FH-FL)
1040 IF Q>10 THEN CLS:SOUND 10,4:
     PRINT CHR$(34)+"Q"+CHR$(34)+"OUT OF
     RANGE OF":PRINT"THIS FILTER.":PRINT"
     Try again.":FDR T=0 TO 700:NEXT T:GOTO 990
1050 IF (2*Q^2-H)<=0 THEN SOUND 10,4:CLS:
     PRINT"Bandwidth too large.":PRINT"Must be
     less than":PRINT FC/SQR(H/2);"Hz":FOR T=0
     TO 700:NEXT T:GOTO 990
1060 GOSUB 1530
1070 GOSUB 1590
1080 R1=Q/H*6.2832*FC*C1)
```

```
1090 R2=Q/((2*Q^2-H)*6.2832*FC*C1)
1100 R3=Q/C1*3.1416*FC)
1110 CA=C1:GOSUB 1640:C1=CA:WC1$=WC$
1120 GOSUB 1790
1130 IF P$="1" THEN PRINT:PRINT:PRINT:
     PRINT"Bandpass Filter:";:COPY:CLS
1140 IF P$="1" THEN PRINT "Gain=;H;" Q=";Q
1150 IF P$="1" AND A$="1" THEN PRINT
     "Fcent=";FC;"BW=";FC/Q
1160 IF P$="1" AND A$="2" THEN PRINT
     FL=";FC-FC/(2*Q);"FH=";FC+FC/(2*Q);"Hz"
1170 PRINT "C1=";C1;WC1$
1180 PRINT "C2=";C1;WC1$;
1190 IF P$="2" THEN GOSUB 1880
1200 IF P$="1" THEN COPY
1210 FOR T=0 TO 600:NEXT T
1220 GOSUB 1790
1230 PRINT"R1=";INT(R1);"ohms"
1240 PRINT"R2=";INT(R2);"ohms"
1250 PRINT"R3=";INT(R3);"ohms"
1260 PRINT"R4=";INT(R3);"ohms";
1270 IF P$="1" THEN COPY:GOTO 1300
1280 FOR T=0 TO 600:NEXT T
1290 GOTO 1120
1300 GOSUB 1840
1310 GOSUB 1800
1320 GOTO 1310
1330 CLS
1340 PRINT"What value of":PRINT"passband gain":
     PRINT"would you like"
1350 INPUT H
1360 IF H<=0 THEN SOUND 10,4:GOTO 1330
1370 RETURN
1380 CLS
1390 PRINT"Enter the ";F$:PRINT"cutoff
     frequency,":PRINT" in hertz."
1400 INPUT F
1410 IF F<=0 THEN SOUND 10,4:GOTO 1380
1420 RETURN
1430 CLS
1440 PRINT"Would you like
     a":PRINT"maximally-flat":PRINT"Butterworth
     filter?"
1450 PRINT"1) Yes. 2) No.";
1460 A$=INKEY$
1470 IF A$="1" THEN Q=SQR(2)/2:RETURN
1480 IF A$="2" THEN CLS ELSE 1460
1490 PRINT"What value of
     "+CHR$(34)+"Q"+CHR$(34):PRINT "would
     you like"
1500 INPUT Q
1510 IF Q<=0 THEN SOUND 10,4:GOTO 1480
1520 RETURN
1530 CLS
1540 PRINT "Do you want a":PRINT "printout?"
1550 PRINT "        1) Yes"
1560 PRINT "        2) No";
1570 P$=INKEY$
1580 IF P$="1" OR P$="2" THEN RETURN
     ELSE 1570
1590 CLS
1600 PRINT "Pick a likely value":PRINT"for C1, in
     farads."
1610 INPUT C1
1620 IF C1<=0 THEN SOUND 10,4:GOTO 1590
1630 RETURN
1640 CB=CA:WC$="farads"
1650 IF CA<.1 THEN CB=CA*1E6:WC$="uf"
1660 IF CA<1E-9 THEN CB=CA*1E12:WC$="pf"
1670 CA=CB
1680 RETURN
1690 CLS
1700 D$=INKEY$
1710 IF D$="1" THEN 230
1720 IF D$="2" THEN 90
1730 RETURN
1740 CLS
1750 D$=INKEY$
1760 IF D$="1" THEN 550
1770 IF D$="2" THEN 90
1780 RETURN
1790 CLS
1800 D$=INKEY$
1810 IF D$="1" THEN 1070
1820 IF D$="2" THEN 90
1830 RETURN
1840 CLS
1850 PRINT CHR$(34)+"1"+CHR$(34)+"to try
     new C1."
1860 PRINT CHR$(34)+"2"+CHR$(34)+"to restart."
1870 RETURN
1880 PRINT:PRINT CHR$(34)+"1"+CHR$(34)+" to
     try new C1."
1890 PRINT CHR$(34)+"2"+CHR$(34)+" to restart.";
1900 RETURN
```

Additional progress in BASIC can be found in Chapter 6, Assembly language, and Chapter 8 Communications.

5

THE 6301 MICROPROCESSOR

'The aim of learning is not knowledge, but action'

This chapter covers:
- Basic concepts
- Introduction to machine architecture
- Programming the 6301:
  - Registers
  - The stack
  - Addressing modes
  - Instructions
- Interrupts
- The real-time clock

WHY GO INSIDE THE HX-20?

Even if you do no more than run pre-programmed, canned software packages, an understanding of what happens when a program is run will be of value. And, of course, having this understanding will open up possibilities for the HX-20 that you may not have previously thought of.

Lest you be scared off by what follows, allow me to say that no knowledge of maths or engineering is required. If it were, this writer — who majored in psychology in college — would not have been able to spend much of the last 10 or 15 years doing assembly language programming.

What is Memory?

You've heard, no doubt, that at the lowest level a computer uses the binary number system to store data. This simply means that a computer's memory is made up of a large number of 'devices' that can be on or off. (Binary is a number system with only two digits — 0 or 1 — making it a natural for computers. Humans, on the other hand, have adopted a decimal number system with digits 0–9.)

On the oldest computers, these devices were vacuum tubes — banks of lights, each one going on or off depending on whether or not an electrical signal was fed into them. (Not exactly portable, and the tubes burnt out regularly.)

Later, tiny electro-magnetic rings were used. An electrical impulse on a wire going through the ring would magnetize the ring to a '1' or a '0' condition — the terms on and off no longer really applied, but continued being used. This 'magnetic core' memory was much more reliable than the old tubes. It also was able to hold its contents even when the power was shut off.

But then came the transistor. Fast, cheap, small. A '1' condition became a particular voltage passing through the transistor; the '0' condition became a different voltage.

The next step was the invention of the integrated circuit, which could reproduce the electrical effect of many transistors onto one piece of silicon. Current computers used LSI chips — large-scale integration — to back the equivalent of thousands of transistors onto one chip. It was the invention of LSI that made microprocessors possible, as well as allowing you to hold a device in your hand that can store thousands of binary digits (1/0), or bits.

How memory is used

Throughout all these years, the simple binary on–off, 1/0 system has been used. But it's hard for people to work with numbers like 1001 — that's 9 in decimal. The logical next step was to group several bits together. Taking three bits together could give you a number as high as 7 (111). That allows you to use the octal numbering system, with eight digits, 0–7.

Even better, you could group four bits together to give you a number as high as 15 (1111). That would result in a hexadecimal system with 16 digits. But there are no names for the digits that run from 10 to 15. So names were made up: 10 is called 'A', 11 is 'B' and so on until 15 which is 'F'. In other words, the number 16 in decimal looks like 10 in hex; the number 30 in decimal looks like 1E in hex.

From this point on, most of what we discuss will use the hex number system. But keep in the back of your mind that when we use a digit like 'A', we really mean four bits — 1010. To avoid confusion between hex and decimal — which share some of the same digit names — we will preface all hex numbers with a $ whenever there's any doubt as to which number system we're using. That means $40 is 0100 0000 or 64 decimal.

At this point you may be thinking how much easier it would have been if computers used decimal. And there are a few that do. But the on–off binary system naturally fitted the machine and makes hardware design much simpler. The hex numbering system, in particular, works well on any machine that uses an 8-bit byte. (Rather than accessing memory a bit at a time, most computers will operate with 8 bits or a multiple of 8 bits at one time. This unit is called a byte.)

One little complication before we go on. We've not talked about negative numbers so far. Manufacturers could adopt the decimal practice of putting a minus sign in front of a number, but it's much easier if each number could tell you whether it was positive or negative. And so negative numbers on the HX-20 are stored in something called two's complement form. We'll give you one quick example and then refer you to books like *Basic Microprocessors and the 6800* by Ron Bishop (Hayden Books, 1979) for further information. By and large, you won't run into negative binary numbers very often, so don't worry if you don't understand the example.

To find the two's complement of a number, reverse all the bits and add 1. (Computers find it much easier to add or reverse bits than to subtract them.) For instance, a byte with the binary number 1 in it ($01) would look in memory like:

0000 0001

Invert the bits and you get:

1111 1110

Add 1 and you get:

1111 1111

So, a −1 in memory looks like: 1111 1111 (Remember back in BASIC, that a TRUE condition is equal to −1?)

So we have a computer memory filled with 1's and 0's, or for short, a lot of $1E's and $6F's and $A9's and $FF's and so on. How do we find anything?

Each byte in memory has an address, just like your house has a street address. This number is not kept in memory, but is computed by the hardware when that address is referenced. The first address is $01. The next is $02. $FFFF is the highest possible address in the HX-20, because the HX-20 only allows 16-bits for an address. (Want to be really confused? What people call an 8-bit computer uses 16-bits for an address, 16-bit computers use 20–32 bits for addressing, depending on the design.)

An address of 1600 Pennsylvania Avenue, Washington DC, would have as its contents a white house. Similarly, an address of $FFFE on the HX-20 has as its contents $E0. (Later on we'll show you how to use the Monitor to look at memory.)

One of the trickiest things for novices to pick up is the difference between an address and the contents of that address. What can make it confusing is that a memory location can itself have an address in it, which could point to another memory location which might have another address in it, and so on. See Figure 5.1.

In assembly language programs, as we'll see next chapter, we can give easier-to-remember names to each of these addresses.

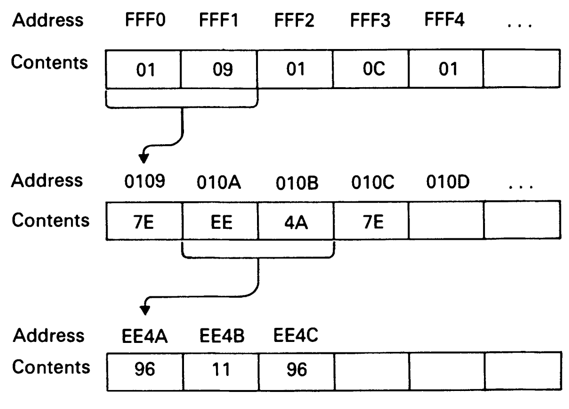


Fig. 5.1 Address pointers in memory

In the HX-20, some memory is in RAM, some in ROM. Some is built right in to the 6301 microprocessor chip.

BACKGROUND ON THE 6301 MICROPROCESSOR

The HX-20 uses dual 6301s. The choice of the 6301 chip was undoubtedly made because of the lack of other CMOS (low power) processors around at the time. There's nothing wrong with the 6301, it's just not compatible with most of the other microprocessors, used in other machines. And microprocessor incompatibility means that assembly language programs written for Z80s or 8088s have to be totally rewritten to work on a 6301 machine.

But the 6301 is not a totally new processor. It's an enhanced, CMOS version of the 6801 which in turn was an enhanced version of the 6800.

The first true microprocessor was Intel's 8008. Like most pioneering designs, it was quite primitive by today's standards. Intel followed quickly with the 8080. Another semiconductor/electronics company, Motorola, also saw the potential of microprocessors and developed the 6800. The first generation of microcomputer systems was based on either the 8080 or the 6800. The next generation grew from the Z-80, an enhancement of the 8080 by some ex-Intel engineers, and from MOS Technology's 6502, an enhancement of the 6800. The Z-80 became the heart of most business computers, while the 6502 found its way into many personal computers including the Apple, Atari, and Commodore. The original Motorola 6800, along with even lesser known processors from Texas Instruments, RCA, etc., became an also-ran.

There were some interesting applications developed for 6800 systems, though. There was briefly, for instance, a militarized, battery-powered, transportable version, intended for use in combat situations where army field commanders needed to track troop movements. Designed to be carried in a jeep, this 1000-lb (455-kg) computer was not quite as portable as today's HX-20.

None of the 6800-based systems ever really enjoyed widespread popularity and today, systems based on the 6800 probably make up less than 0.1% of micros in use.

Unfortunately, for Motorola — and unlike the 8080/Z80 relationship — 6800 programs would not run on the 6502. The design philosophy was similar, the implementation totally different.

In succeeding years, Motorola developed new processors, including the popular 68000. But only the 6801/6802/6803, none of which was used in any commercial micro system, were compatible with the 6800. Then Hitachi developed the 6301 as a CMOS version of the 6801 and Epson selected that chip for its HX-20.

MACHINE ARCHITECTURE

Were we to cover in detail how the 6301 processors fit in with the other internal components of the HX-20, we'd be getting very technical very fast. Instead, we'll just select those points that are most useful and/or interesting. Readers who want more detail are encouraged to contact Epson America for its technical notes and Hitachi America for its *Microcomputer Data Book.*

Besides memory, the real-time clock, and the I/O interfaces that are discussed elsewhere, the major internal components of the HX-20 are the two 6301 processors and the high-speed serial bus that connects them.

All processing functions are divided up between the two 6301s. One 6301 is called the master and the other is called a slave. When the operating system, the control program that sits in ROM, wants to take an action that falls into the slave's province, it sends a command to the slave.

For instance, the slave is assigned the task of listening on the RS-232 port. When you as a user wish to OPEN the RS-232 port, your program or BASIC statement in effect has the master tell the slave to start listening. When you do an INPUT from the RS-232 port, the master tells the slave to send him the next byte from the holding area. All of this inter-processor communication is done via a high-speed serial (1 bit at a time) path between the processors. This is the same path that also has an outlet as a DIN socket on the back of the HX-20 and to which a disk drive can be hooked.

By using the serial path, it is not necessary for the slave to have access to any of the HX-20's main memory. Let's say this another way: any program that you run in the HX-20's RAM or ROM is processed by the master only. When the master needs the slave to do something, it sends a command over the serial path.

At this point you may be wondering how the slave knows what to do with the commands it gets. Since the slave doesn't have access to the HX-20's operating system in ROM, the functions can't be stored there. The answer is that the slave has its own memory — right on the processor chip.

Hitachi's 6301 processor is very flexible as far as memory management and can be set up in different ways for different uses. In the HX-20, the slave is set up to have a 4K internal ROM in which Epson has stored all the machine code the slave needs to execute the commands it receives from the master. See Figs 5.2 and 5.3.

The master is wired to operate in memory mode 4, the slave in memory mode 7. So even though we have two processors that are the same, they operate in different fashions. This is a sophisticated design, called multiprocessing, and it allows for the two processors to be doing different things at the same time. In the example above, for instance, the slave can be filling its buffer with data received over the RS-232 port while the master is doing something entirely different.

PROGRAMMING THE 6301

So, now we've figured out that everything in the computer is just bits. And, that every 8 bits is called a byte and has an address. But how do we get from there to executing a program?

When the HX-20 is turned on, the 6301 master

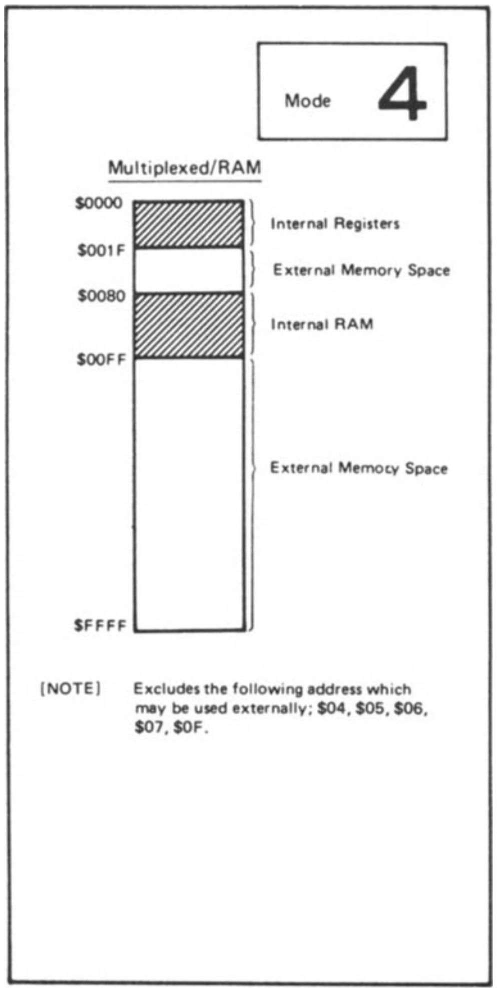


Fig. 5.2 6301 Mode 4, used by the master processor.
Reprinted courtesy of Hitachi America

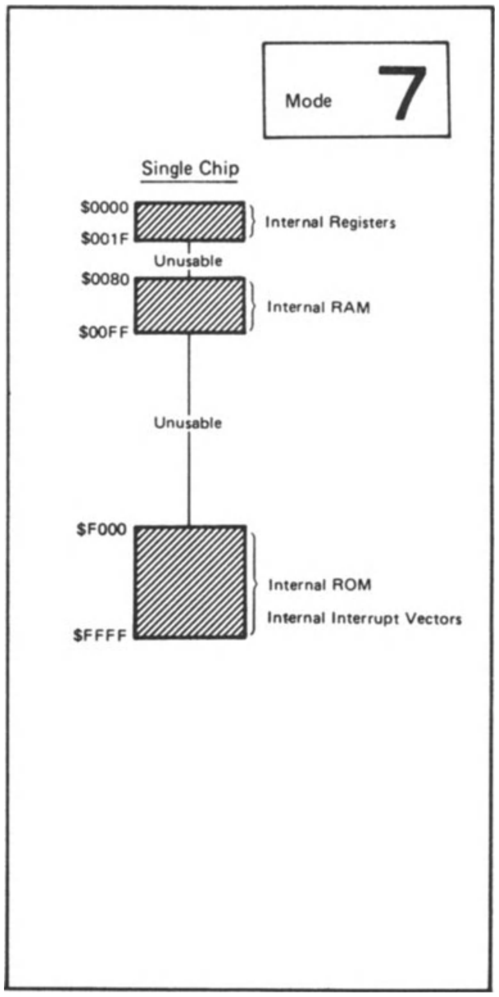


Fig. 5.3 6301 Mode 7, used by the slave processor.
Reprinted courtesy of Hitachi America

cpu fetches the byte at location $115. This byte is a $7E. What $7E means to the cpu is: jump to a specified location. (Similar to a GOTO in BASIC.) $7E is the operative part of an instruction, an operation code or op code for short. An instruction tells the cpu to do something. Each unique op code tells the cpu to do something different. There are over 200 op codes, i.e., 200+ combinations of 8 bits, that the cpu will interpret as instructions when fetched. To repeat, when the computer fetches a byte from memory, it interprets it as an instruction and tries to execute it. This is what is known as machine code.

Now, of course, not all bytes in memory are instructions. Some are data. But the cpu, not without being told, can't tell the difference. If the cpu fetched a $31 from memory it wouldn't know that this was the letter 'A'. It would think it was an INS instruction and would try to execute it. If a $00 was fetched from memory, as another example, the cpu would try to execute it and fail, causing a 'trap interrupt' to occur, because there is no such op code as $00.

The missing link in all this is that the cpu doesn't try to fetch every byte. It will fetch only the one byte pointed to by a device called the program counter. After the op code byte is fetched, interpreted, and executed, the program counter automatically increments to the address of the next op code to be executed — either the next sequential instruction or an instruction that is to be branched (jumped) to.

An instruction may be more than 1 byte long — in fact, most are. All the op codes on the 6301 are just 1 byte, but many tell the cpu that the next byte or next two bytes are to be acted upon as data. For instance, a jump instruction at location $0109 may look like: 7EEE4A. The cpu will fetch the 7E and put the EE4A in the program counter as the location of the next instruction to be executed. As another example, a store instruction at $D026 may look like B7013A. When the B7 is executed, the cpu knows to take the contents of a temporary holding area called accumulator A and store it into $013A. The program counter is then automatically incremented by 3 bytes so that it points to location $D029, where it will find the next instruction.

The cpu knows that a $B7 instruction means store the contents of accumulator A at a specified address and increment the program counter by 3 bytes. This is simply because it has been microcoded that way by the manufacturer. All of the possible instructions a cpu can execute are collectively called the instruction set which is designed into the processor chip by the manufacturer. Sometimes the same processor is microcoded different ways for different purposes. For instance, a Motorola 68000 can be microcoded to run the instruction set of an IBM 370 mainframe computer. But the HX-20's 6301 is very simple: the instruction set that it runs is the same as that of every 6301 in the world. It's even simpler than the ubiquitous Z-80 processor, because the Z-80's designers, after using all 256 possible 8-bit combinations, were forced to design in some 2-byte op codes.

To highlight what we've covered so far: every legal 8-bit op code will cause the 6301 to take a unique action.

Registers

On board every microprocessor are special processing areas called registers. The design of the mpu chips allows the processor to manipulate easily and speedily the contents of these registers.

Some of these registers are set aside for special purposes. Others can be used by the programmer for multiple purposes. In general, the more registers a processor has, the easier it is to program. In that respect, programming the 6301 takes more work than programming the Z-80, but less than 6800.

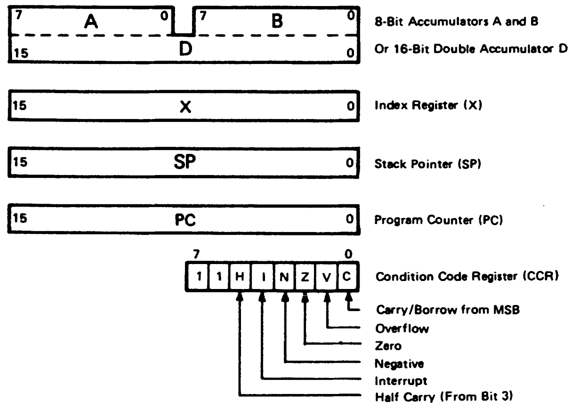


Fig. 5.4 The 6301's registers

There are six registers that the programmer can examine and/or alter: (Don't worry about understanding all of this right away.)

1. Accumulators A and B (ACCA, ACCB). These are two 8-bit registers that hold operands and results from the arithmetic logic unit (ALU) of

the microprocessor. Some instructions can operate on the two registers as a pair, in which case they are referred to collectively as accumulator D (ACCD).

2. The index register (X) is a 2-byte (16-bit) register that can store data or be used to hold a memory address for instructions in the indexed mode.
3. The program counter (PC) is a 2-byte register that contains the address of the next instruction to be executed.
4. The stack pointer (SP) is a 2-byte register that contains the address of the next availabe location in a push-down/pop-up stack located in RAM. The HX-20's operating system uses a stack starting at one location, BASIC uses another, and any other program can use its own stack just by loading SP.
5. The condition code register (CCR) is an 8-bit register that indicates the results of ALU operations. Each bit in the CCR indicates a different type of result:
   C — carry;
   V — two's complement overflow;
   Z — zero;
   N — negative;
   H — half-carry;
6. In addition, the CCR contains an interrupt mask. The other two (high-order) bits of the CCR are not used.

The stack

Many of the 6301's instructions are oriented towards use of a stack. A stack is simply a way of assigning an area of memory for keeping track of specific types of information. (The FORTH language is stack oriented, as we'll see in a future chapter.) A stack works much like a pile of TV dinners at your local supermarket. When you buy a TV dinner, you probably pull off the top one. The next shopper probably does the same, pulling off the new top one. When the supermarket clerk goes to replenish the pile, if he's lazy, he'll just put each new one on top of the old. This is called a LIFO (last in, first out) stack; also known as a push-down/pop-up or push/pull stack.

Look at Figure 5.5. A PSHA instruction is about to be executed. The program counter (PC) points to the PSHA instruction. Accumulator A contains an F3. The stack pointer (SP) points to the next free slot in the stack, which is called 'm'. This 'm' might be $100, for instance. (Think

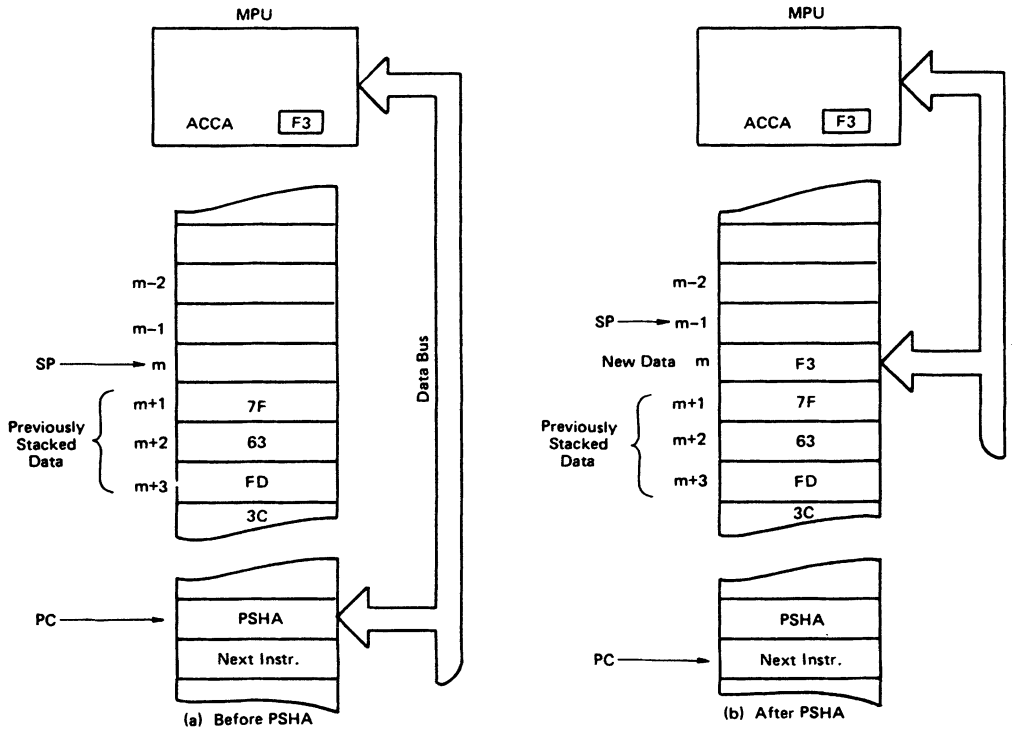


Fig. 5.5 Stack operation (Push instruction).
Reprinted courtesy of Hitachi America

of the machine having the low addresses on top and the high addresses on the bottom.)

Previously stacked data went into m+1 ($101) and m+2 ($102), etc. Now, when the PSHA is executed, the F3 from ACC A is put on the stack at m (e.g.,$100), and the stack pointer now points to the next free slot m−1 (e.g., $FF). The program counter points to the next instruction after the PSHA. Everything that was on the stack from before remains on the stack.

If the instruction had been a PSHX, and the index register X had contained $8001, then the stack would have ended up with $80 at location m ($100) and $01 at location m−1 ($FF). The stack pointer would have been decremented 2 bytes and would now point to m−2 ($FE).

Pull instructions work like push instructions in reverse. See Figure 5.6. A PULA instruction would increment the stack pointer by one, taking the top byte ($1A) off the stack and placing it into the A register. (Again, keep in mind that the 'top' of the stack is towards the low-address part of the machine.) The space occupied by $1A would now be 'free' and would be re-used when the next push was done.

When you call a subroutine, you'll be using a stack — the system will put your return address on the stack and pull it off when your subroutine issues a return instruction. More on the stack later.

Addressing modes

As we know, each op code tells the mpu to execute a specific function. An 86, for instance, tells the mpu to load accumulator A with the byte immediately following the op code. On the other hand, a B6, which is also a load accumulator A instruction, tells the mpu to load A with the contents of the 2-byte memory location specified after the op code.

Example Before instruction 8605
contents of A: xx (anthing)
After instruction contents of A: 05
Before instruction B60B00
contents of A: xx
contents of memory location $0B00: 31
After instruction
contents of A: 31

The 05 in the first example and the OB00 in the second are called the operands of the instruction. The different ways by which the mpu carries out the same function are called

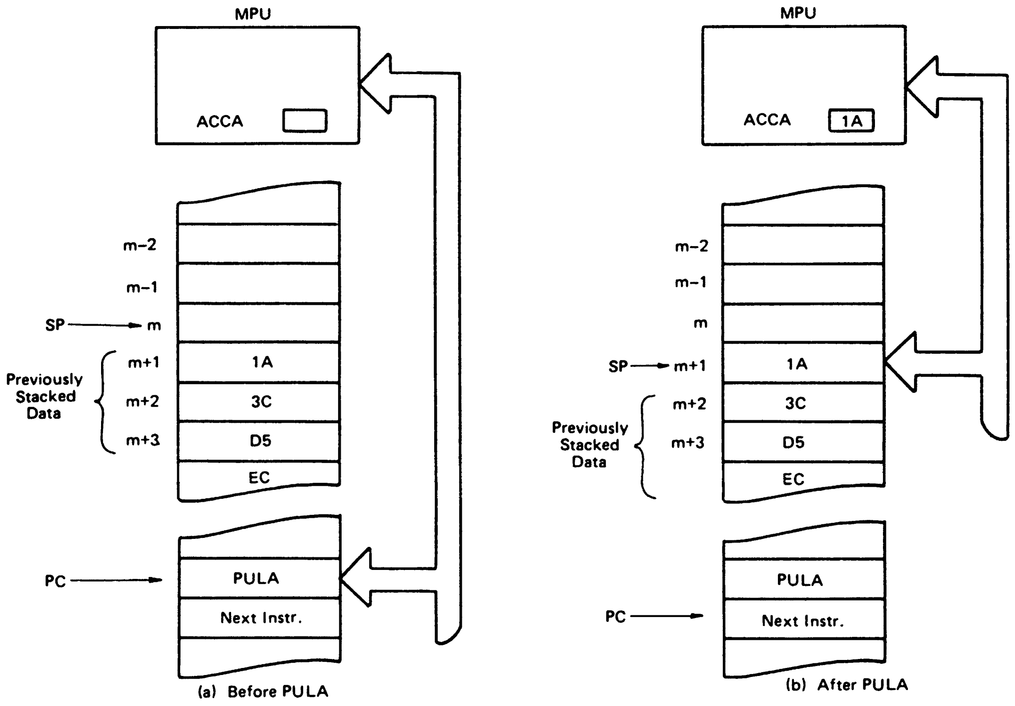


Fig. 5.6 Stack operation (Pull instruction).
Reprinted courtesy of Hitachi America

addressing modes. The various addressing modes defined for the 6301 mpu allow the programmer to access and manipulate the mpu's internal registers as well as main memory.

If the programmer uses an assembler, then an op code will be selected by the assembler based on the mnemonic and the operands coded. For instance:

LDAA #$05 will assemble as 8605 (2 bytes)
LDAA $0B00 will assemble as B60B00 (3 bytes)

If you code the operands directly or 'hand-assemble' code, then you are responsible for using the right op code/operand combination.

There are six addressing modes on the 6301: implied, immediate, direct, extended, indexed, and relative.

Implied (or Inherent) Addressing Mode

The op code of the instruction provides the complete specifications as to what is to be acted upon. For instance, PSHX, the instruction to push the contents of the index register onto the stack, tells everything that is to be done. As no operands are necessary in this mode, the instructions are 1-byte long.

A related mode, called accumulator addressing, which works on just accumulator instructions, is often listed separately in technical documentation. In fact, the original Motorola manuals had the accumulator as a separate instruction operand, e.g., LDA B. But it is really just a subject of implied addressing and is being treated as such in this book.

Immediate mode

In this mode, the value to be processed is specified in the second byte of the assembled instruction, or the first (and only) operand of the source code. For instance:

LDAA #$05 will assemble as 8605 and will load the value 05 into accumulator A.

(**Exceptions** Because CPX, LDX, and LDS operate on 16-bit quantities, the value to be processed is specified in the second and third bytes.)

Immediate mode instructions are relatively fast because no further memory accesses are required — once both bytes of the instruction have been fetched, the mpu can process to completion, update the program counter and fetch the next instruction.

Direct addressing

Like immediate addressing, the mpu will fetch the second byte of this instruction after decoding the op code. But unlike the immediate mode, this second byte is not a value to be acted upon directly but is instead a 1-byte address. (**Remember** The mup can tell what addressing mode is used from the op code.) If you're quick, you've realized that a 1-byte address only allows you to access memory locations0–FF. (FF is the highest hex number that can be stored in 1 byte.)

Example Before instruction LDAA $BC (96BC)
  contents of A: xx
  contents of memory location $00BC: 05
After instruction
  contents of A: 05

AIM, OIM, EIM, TIM are 3-byte instructions, the remainder are 2-byte instructions.

Extended addressing

Like direct addressing, what follows the op code is the address whose contents are to be operated on. The difference is that instructions in extended addressing mode use a 2-byte address.

Example Before instruction LDAA $0A40 (B60A40)
  contents of A: xx
  contents of memory location $0A40: 05
After instruction
  contents of A: 05

With extended addressing, any place in memory can be accessed. (**Remember** The highest address the HX-20's 6301 processor can access is $FFFF, which is the highest possible binary number that will fit into 2 bytes.)

These are all 3-byte instructions.

Note We've used hex values for all memory locations, but most assemblers will accept a label instead. An LDAA $0B00, where $0B00 contained $05 could also be an LDAA ADR1 if ADR1 was a label for a memory location that had a $05 in it.

Another note Programs that use extended addressing normally will not be relocatable, i.e., cannot run at an address other than that at which they were assembled, because they will contain actual addresses.

Indexed addressing

Indexed addressing looks like direct addressing with an added operand — 'X' — the index register. But it doesn't have anything in common with the other addressing methods. The number specified in the first operand is simply a number that is added to the contents of the index register to form a 16-bit memory address. The contents of this location are then processed.

Example: Before instruction LDAA $01,X (A601)

contents of A:	xx
contents of X:	0A40
contents of memory starting at $0A40:	040506. . .
After instruction	
contents of A:	05

Instructions in indexed mode are 2-bytes long, except for AIM, EIM, OIM, and TIM which are 3 bytes. It's the last byte of the instruction that is added into the index register. The index register is not actually changed by this process, as all manipulations are carried out by the mpu 'behind the scenes'.

The value of indexing is that it allows memory accesses to be changed while a program is running. This is very useful for table handling, for instance, when you want to use the same code to retrieve the 2nd–nth element of a table. (Just increment the index register between accesses.)

Relative addressing

Like direct addressing, instructions in relative addressing format have, as their second byte, a memory location that the mpu will access. The difference is that, with relative addressing, this address is not an absolute value but is instead a memory location *relative* to the current location pointed to by the program counter. In other words, this second byte is added to the program counter to obtain the address.

A range of addresses 127 bytes forwards or backwards counting from the byte after the instruction can thus be accessed. (0–127 is the maximum range that can be specified in 7 bits; the high-order bit of the address byte is the sign bit, i.e., it indicates whether the branch is forwards or backwards.) With respect to the start of the branch instruction (or the PC) — only branch instructions use relative addressing — the range is −126 to +129, because the branch instruction is 2 bytes long.

Keeping branches within this range means not violating any relocatability rules. Branches outside of this range require JMP (jump) instructions, which have the absolute address in the instruction.

Note If a branch test is successful, the relative displacement is added to the PC, plus two additional bytes for the instruction length, and then the PC will point towards the next instruction to be executed.

THE HX-20 INSTRUCTION SET

The following is based largely on information supplied by Hitachi America Ltd. All of the diagrams (Tables 5.1-5.5) have been reproduced, with permission, from Hitachi's *Microcomputer Data Book* (HLN062). For a fuller explanation of the instruction set, with examples, use a book like *Basic Microprocessors and the 6800* by Bishop (Hayden, 1979). Chapter 6 explains how to use the machine instructions from an assembly language program.

Explanation of the Tables

Operations: type of instruction

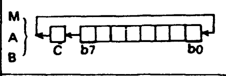
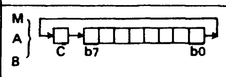
Mnemonic — this is the name of the instruction as used by the Hitachi assembler and by the 6301 mini-assembler in Chapter 6. **Note** Assemblers based on the original Motorola 6800 documentation use a space between the common part of the mnemonic and the register (if any) that it acts upon. For example, ADDA nnnn — the instruction to add the contents of memory location nnnn to the contents of accumulator A — could be written as ADD A nnnn.

Addressing modes — the legal addressing mode for each instruction. Note that relative mode is not legal for the instructions in the first part of the chart. If a box for an instruction is empty, then that mode is not legal for that particular instruction. For instance, ADDA does not have an implied mode, because implied mode specifies that no operands are necessary — which, of course, is not true for ADDA. (You can't tell from looking at just the mnemonic 'ADDA' what is supposed to be added to accumulator A.)

Op — the operation code in hex of the instruction. As we've noted previously, this number is what makes each instruction unique, i.e., the processor knows what instruction you want executed by examining this number.

~ — this is the time it takes to execute this

Table 5.1 Accumulator and memory manipulation instructions
Reprinted courtesy of Hitachi America

Operations	Mnemonic	Addressing Modes															Boolean/ Arithmetic Operation	Condition Code Register					
		IMMED.			DIRECT			INDEX			EXTEND			IMPLIED				5	4	3	2	1	0
		OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#		H	I	N	Z	V	C
Add	ADDA	8B	2	2	9B	3	2	AB	4	2	BB	4	3				A + B → A	↕	●	↕	↕	↕	↕
	ADDB	CB	2	2	DB	3	2	EB	4	2	FB	4	3				B + M → B	↕	●	↕	↕	↕	↕
Add Double	ADDD	C3	4	3	D3	5	2	E3	6	2	F3	6	3				A : B + M : M + 1 → A : B	●	●	↕	↕	↕	↕
Add Accumulators	ABA													1B	2	1	A + B → A	↕	●	↕	↕	↕	↕
Add With Carry	ADCA	89	2	2	99	3	2	A9	4	2	B9	4	3				A + M + C → A	↕	●	↕	↕	↕	↕
	ADCB	C9	2	2	D9	3	2	E9	4	2	F9	4	3				B + M + C → B	↕	●	↕	↕	↕	↕
AND	ANDA	84	2	2	94	3	2	A4	4	2	B4	4	3				A·M → A	●	●	↕	↕	R	●
	ANDB	C4	2	2	D4	3	2	E4	4	2	F4	4	3				B·M → B	●	●	↕	↕	R	●
Bit Test	BIT A	85	2	2	95	3	2	A5	4	2	B5	4	3				A·M	●	●	↕	↕	R	●
	BIT B	C5	2	2	D5	3	2	E5	4	2	F5	4	3				B·M	●	●	↕	↕	R	●
Clear	CLR							6F	6	2	7F	6	3				00 → M	●	●	R	S	R	R
	CLRA													4F	2	1	00 → A	●	●	R	S	R	R
	CLRB													5F	2	1	00 → B	●	●	R	S	R	R
Compare	CMPA	81	2	2	91	3	2	A1	4	2	B1	4	3				A − M	●	●	↕	↕	↕	↕
	CMPB	C1	2	2	D1	3	2	E1	4	2	F1	4	3				B − M	●	●	↕	↕	↕	↕
Compare Accumulators	CBA													11	2	1	A − B	●	●	↕	↕	↕	↕
Complement, 1's	COM							63	6	2	73	6	3				$\overline{M}$ → M	●	●	↕	↕	R	S
	COMA													43	2	1	$\overline{A}$ → A	●	●	↕	↕	R	S
	COMB													53	2	1	$\overline{B}$ → B	●	●	↕	↕	R	S
Complement, 2's	NEG							60	6	2	70	6	3				00 − M → M	●	●	↕	↕	①	②
(Negate)	NEGA													40	2	1	00 − A → A	●	●	↕	↕	①	②
	NEGB													50	2	1	00 − B → B	●	●	↕	↕	①	②
Decimal Adjust, A	DAA													19	2	1	Converts binary add of BCD characters into BCD format	●	●	↕	↕	↕	③
Decrement	DEC							6A	6	2	7A	6	3				M − 1 → M	●	●	↕	↕	④	●
	DECA													4A	2	1	A − 1 → A	●	●	↕	↕	④	●
	DECB													5A	2	1	B − 1 → B	●	●	↕	↕	④	●
Exclusive OR	EORA	88	2	2	98	3	2	A8	4	2	B8	4	3				A ⊕ M → A	●	●	↕	↕	R	●
	EORB	C8	2	2	D8	3	2	E8	4	2	F8	4	3				B ⊕ M → B	●	●	↕	↕	R	●
Increment	INC							6C	6	2	7C	6	3				M + 1 → M	●	●	↕	↕	⑤	●
	INCA													4C	2	1	A + 1 → A	●	●	↕	↕	⑤	●
	INCB													5C	2	1	B + 1 → B	●	●	↕	↕	⑤	●
Load Accumulator	LDAA	86	2	2	96	3	2	A6	4	2	B6	4	3				M → A	●	●	↕	↕	R	●
	LDAB	C6	2	2	D6	3	2	E6	4	2	F6	4	3				M → B	●	●	↕	↕	R	●
Load Double Accumulator	LDD	CC	3	3	DC	4	2	EC	5	2	FC	5	3				M + 1 → B, M → A	●	●	↕	↕	R	●
Multiply Unsigned	MUL													3D	10	1	A x B → A : B	●	●	●	●	●	⑪
OR, Inclusive	ORAA	8A	2	2	9A	3	2	AA	4	2	BA	4	3				A + M → A	●	●	↕	↕	R	●
	ORAB	CA	2	2	DA	3	2	EA	4	2	FA	4	3				B + M → B	●	●	↕	↕	R	●
Push Data	PSHA													36	3	1	A → Msp, SP − 1 → SP	●	●	●	●	●	●
	PSHB													37	3	1	B → Msp, SP − 1 → SP	●	●	●	●	●	●
Pull Data	PULA													32	4	1	SP + 1 → SP, Msp → A	●	●	●	●	●	●
	PULB													33	4	1	SP + 1 → SP, Msp → B	●	●	●	●	●	●
Rotate Left	ROL							69	6	2	79	6	3					●	●	↕	↕	⑥	↕
	ROLA													49	2	1		●	●	↕	↕	⑥	↕
	ROLB													59	2	1		●	●	↕	↕	⑥	↕
Rotate Right	ROR							66	6	2	76	6	3					●	●	↕	↕	⑥	↕
	RORA													46	2	1		●	●	↕	↕	⑥	↕
	RORB													56	2	1		●	●	↕	↕	⑥	↕

(Continued)

Table 5.2 Continuation of accumulator and memory manipulation instructions
Reprinted courtesy of Hitachi America

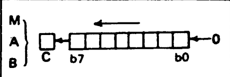
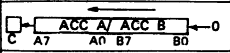
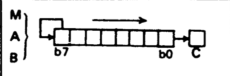
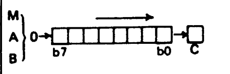
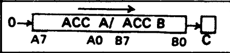
Operations	Mnemonic	Addressing Modes															Boolean/ Arithmetic Operation	Condition Code Register					
		IMMED.			DIRECT			INDEX			EXTEND			IMPLIED				5	4	3	2	1	0
		OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#		H	I	N	Z	V	C
Shift Left Arithmetic	ASL							68	6	2	78	6	3					●	●	↕	↕	⑥	↕
	ASLA													48	2	1		●	●	↕	↕	⑥	↕
	ASLB													58	2	1		●	●	↕	↕	⑥	↕
Double Shift Left, Arithmetic	ASLD													05	3	1		●	●	↕	↕	⑥	↕
Shift Right Arithmetic	ASR							67	6	2	77	6	3					●	●	↕	↕	⑥	↕
	ASRA													47	2	1		●	●	↕	↕	⑥	↕
	ASRB													57	2	1		●	●	↕	↕	⑥	↕
Shift Right Logical	LSR							64	6	2	74	6	3					●	●	↕	↕	⑥	↕
	LSRA													44	2	1		●	●	↕	↕	⑥	↕
	LSRB													54	2	1		●	●	↕	↕	⑥	↕
Double Shift Right Logical	LSRD													04	3	1		●	●	R	↕	⑥	↕
Store Accumulator	STAA				97	3	2	A7	4	2	B7	4	3				$A \rightarrow M$	●	●	↕	↕	R	●
	STAB				D7	3	2	E7	4	2	F7	4	3				$B \rightarrow M$	●	●	↕	↕	R	●
Store Double Accumulator	STD				DD	4	2	ED	5	2	FD	5	3				$A \rightarrow M$ $B \rightarrow M+1$	●	●	↕	↕	R	●
Subtract	SUBA	80	2	2	90	3	2	A0	4	2	B0	4	3				$A - M \rightarrow A$	●	●	↕	↕	↕	↕
	SUBB	C0	2	2	D0	3	2	E0	4	2	F0	4	3				$B - M \rightarrow B$	●	●	↕	↕	↕	↕
Double Subtract	SUBD	83	4	3	93	5	2	A3	6	2	B3	6	3				$A : B - M : M+1 \rightarrow A : B$	●	●	↕	↕	↕	↕
Subtract Accumulators	SBA													10	2	1	$A - B \rightarrow A$	●	●	↕	↕	↕	↕
Subtract With Carry	SBCA	82	2	2	92	3	2	A2	4	2	B2	4	3				$A - M - C \rightarrow A$	●	●	↕	↕	↕	↕
	SBCB	C2	2	2	D2	3	2	E2	4	2	F2	4	3				$B - M - C \rightarrow B$	●	●	↕	↕	↕	↕
Transfer Accumulators	TAB													16	2	1	$A \rightarrow B$	●	●	↕	↕	R	●
	TBA													17	2	1	$B \rightarrow A$	●	●	↕	↕	R	●
Test Zero or Minus	TST							6D	6	2	7D	6	3				$M - 00$	●	●	↕	↕	R	R
	TSTA													4D	2	1	$A - 00$	●	●	↕	↕	R	R
	TSTB													5D	2	1	$B - 00$	●	●	↕	↕	R	R

Table 5.3 Index register, stack manipulation instructions
Reprinted courtesy of Hitachi America

Pointer Operations	Mnemonic	Addressing Modes															Boolean/ Arithmetic Operation	Condition Code Register					
		IMMED			DIRECT			INDEX			EXTEND			IMPLIED				5	4	3	2	1	0
		OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#		H	I	N	Z	V	C
Compare Index Reg	CPX	8C	3	3	9C	4	2	AC	5	2	BC	5	3				$X - M:M+1$	●	●	↕	↕	↕	↕
Decrement Index Reg	DEX													09	1	1	$X - 1 \rightarrow X$	●	●	●	↕	●	●
Decrement Stack Pntr	DES													34	1	1	$SP - 1 \rightarrow SP$	●	●	●	●	●	●
Increment Index Reg	INX													08	1	1	$X + 1 \rightarrow X$	●	●	●	↕	●	●
Increment Stack Pntr	INS													31	1	1	$SP + 1 \rightarrow SP$	●	●	●	●	●	●
Load Index Reg	LDX	CE	3	3	DE	4	2	EE	5	2	FE	5	3				$M \rightarrow X_H, (M+1) \rightarrow X_L$	●	●	⑦	↕	R	●
Load Stack Pntr	LDS	8E	3	3	9E	4	2	AE	5	2	BE	5	3				$M \rightarrow SP_H, (M+1) \rightarrow SP_L$	●	●	⑦	↕	R	●
Store Index Reg	STX				DF	4	2	EF	5	2	FF	5	3				$X_H \rightarrow M, X_L \rightarrow (M+1)$	●	●	⑦	↕	R	●
Store Stack Pntr	STS				9F	4	2	AF	5	2	BF	5	3				$SP_H \rightarrow M, SP_L \rightarrow (M+1)$	●	●	⑦	↕	R	●
Index Reg → Stack Pntr	TXS													35	1	1	$X - 1 \rightarrow SP$	●	●	●	●	●	●
Stack Pntr → Index Reg	TSX													30	1	1	$SP + 1 \rightarrow X$	●	●	●	●	●	●
Add	ABX													3A	1	1	$B + X \rightarrow X$	●	●	●	●	●	●
Push Data	PSHX													3C	5	1	$X_L \rightarrow M_{sp}, SP - 1 \rightarrow SP$ $X_H \rightarrow M_{sp}, SP - 1 \rightarrow SP$	●	●	●	●	●	●
Pull Data	PULX													38	4	1	$SP + 1 \rightarrow SP, M_{sp} \rightarrow X_H$ $SP + 1 \rightarrow SP, M_{sp} \rightarrow X_L$	●	●	●	●	●	●
Exchange	XGDX													18	2	1	ACCD↔IX	●	●	●	●	●	●

Table 5.4 Jump, branch instructions
Reprinted courtesy of Hitachi America

Operations	Mnemonic	Addressing Modes															Branch Test	Condition Code Register					
		RELATIVE			DIRECT			INDEX			EXTEND			IMPLIED				5	4	3	2	1	0
		OP	~	#	OP	~	#	OP	~	#	OP	~	#	OP	~	#		H	I	N	Z	V	C
Branch Always	BRA	20	3	2													None	●	●	●	●	●	●
Branch Never	BRN	21	3	2													None	●	●	●	●	●	●
Branch If Carry Clear	BCC	24	3	2													C = 0	●	●	●	●	●	●
Branch If Carry Set	BCS	25	3	2													C = 1	●	●	●	●	●	●
Branch If = Zero	BEQ	27	3	2													Z = 1	●	●	●	●	●	●
Branch If ≥ Zero	BGE	2C	3	2													N ⊕ V = 0	●	●	●	●	●	●
Branch If > Zero	BGT	2E	3	2													Z + (N ⊕ V) = 0	●	●	●	●	●	●
Branch If Higher	BHI	22	3	2													C + Z = 0	●	●	●	●	●	●
Branch If ≤ Zero	BLE	2F	3	2													Z + (N ⊕ V) = 1	●	●	●	●	●	●
Branch If Lower Or Same	BLS	23	3	2													C + Z = 1	●	●	●	●	●	●
Branch If < Zero	BLT	2D	3	2													N ⊕ V = 1	●	●	●	●	●	●
Branch If Minus	BMI	2B	3	2													N = 1	●	●	●	●	●	●
Branch If Not Equal Zero	BNE	26	3	2													Z = 0	●	●	●	●	●	●
Branch If Overflow Clear	BVC	28	3	2													V = 0	●	●	●	●	●	●
Branch If Overflow Set	BVS	29	3	2													V = 1	●	●	●	●	●	●
Branch If Plus	BPL	2A	3	2													N = 0	●	●	●	●	●	●
Branch To Subroutine	BSR	8D	5	2													See Special Operations	●	●	●	●	●	●
Jump	JMP							6E	3	2	7E	3	3					●	●	●	●	●	●
Jump To Subroutine	JSR				9D	5	2	AD	5	2	BD	6	3					●	●	●	●	●	●
No Operation	NOP													01	1	1	Advances Prog. Cntr. Only	●	●	●	●	●	●
Return From Interrupt	RTI													3B	10	1	See Special Operations	⑧					
Return From Subroutine	RTS													39	5	1		●	●	●	●	●	●
Software Interrupt	SWI													3F	12	1		●	S	●	●	●	●
Wait for Interrupt*	WAI													3E	9	1		●	⑨	●	●	●	●
Sleep	SLP													1A	4	1		●	●	●	●	●	●

*WAI puts R/$\overline{W}$ high; Address Bus goes to FFFF; Data Bus goes to the three state level.

Table 5.5 Condition code register manipulation instructions
Reprinted courtesy of Hitachi America

Operations	Mnemonic	AddressingModes			Boolean Operation	Condition Code Register					
		IMPLIED				5	4	3	2	1	0
		OP	~	#		H	I	N	Z	V	C
Clear Carry	CLC	0C	1	1	0 → C	●	●	●	●	●	R
Clear Interrupt Mask	CLI	0E	1	1	0 → I	●	R	●	●	●	●
Clear Overflow	CLV	0A	1	1	0 → V	●	●	●	●	R	●
Set Carry	SEC	0D	1	1	1 → C	●	●	●	●	●	S
Set Interrupt Mask	SEI	0F	1	1	1 → I	●	S	●	●	●	●
Set Overflow	SEV	0B	1	1	1 → V	●	●	●	●	S	●
Accumulator A → CCR	TAP	06	1	1	A → CCR	⑩					
CCR → Accumulator A	TPA	07	1	1	CCR → A	●	●	●	●	●	●

instruction, specified in machine cycles. If you have an interest in writing a program that runs as quickly as possible, use instructions that take the smallest number of cycles. (Note that indexed and extended addressing instructions are slower than other modes.) Or, you may write a program that is timing dependent, such as I/O drivers. In this case, you can time events fairly accurately based on program loops. (The actual instruction cycle time in the HX-20 is 1.6 microseconds.) If you don't have any interest in either of the above situations, then you can ignore this column completely.

# — the number of bytes taken up by this instruction in this addressing mode. This is handy information for debugging, or for reducing program size. Normally, in writing a program, you'd ignore this column.

Boolean/arithmetic operation — what the instruction actually does.

+ arithmetic plus (add) or Boolean Inclusive

OR (if either operand is on, turn on result bit)
– arithmetic minus (subtract)
● Boolean AND (if both bits are on, turn on result bit)
⊕ Boolean Exclusive OR (if either bit is on, but not both, turn result bit on)
M memory location
Msp contents of memory location pointed to by the stack pointer
$\overline{M}$ complement of M
→ transfer into
0 bit = zero
00 byte = zero
b bit
IMM = immediate data
A or ACCA= accumulator A
B or ACCB = accumulator B
ACCD = accumulator D (A & B operated on together)
X or IX = index register
SP = stack pointer
H = high order byte (most significant digit), e.g., the 0A in $0A40
L = low order byte (least significant digit), e.g., the 40 in $0A40

Examples ADDA arithmetically adds accumulator A plus the contents of a memory location and stores the result in A. This memory location can be the instruction itself + 1 (immediate mode), in the first page of memory (direct mode), found by adding a displacement to the contents of the index register (indexed mode) or specified by a 2-byte address (extended mode).

Another example: ROL (rotate left) moves all bits 1 to the left. Bit 7 (b7) gets moved to the carry flag (C), which itself is moved to bit 0 of the byte being processed. (The byte operated upon can be the A or B accumulator or a memory location.)

Condition code register

The six flags in the condition code register (CCR) may be changed by the execution of the instruction. The resulting condition of each flag is given in these columns.
H = half carry flag (from bit 3)
I = interrupt mask
N = negative flag (sign bit)
Z = zero flag
V = overflow, 2's complement
C = carry bit (from bit 7)
R — reset always
S — set always
↕ — test and set if true, cleared otherwise
● — not affected

Examples All of this may be made clearer with some examples.

For instance:
ADDA #$0F where accumulator A contains $03.
Result is $12.
What happens:

1. The half-carry flag is tested. In this case, it will be set on because the operation caused bit 3 (counting from right and starting with zero) to overflow and thus cause a bit to be carried over to the left nibble. (You can try this yourself using the Monitor, as we'll show you later on.)
2. The interrupt mask is unaffected.
3. The sign bit is not set negative in this case. (It would be if we had added a –$10 to our original $0F.)
4. The zero flag is not set. (It would have been if we had added –$0F to our original $0F.)
5. The two's complement overflow flag is not set. (It would have been if we had added two negative numbers and the result was more minus than –$7F.)
6. The carry bit is not set. (It would have been if we had added $F1 to our original $0F.)

More examples PSHA — No condition code flags are affected by this operation.
CLRA — This operation to zero-out accumulator A will turn off (if on) the negative, overflow, and carry flags and turn on the zero flag.

BCD

The explanation of the Decimal Adjust A (DAA) instruction on the chart notes that it 'converts binary addition of BCD characters into BCD format'. BCD is a handy way of storing decimal digits. Each nibble is assigned one digit. For instance,
24 (decimal) would be: 0010 0100
If this were straight binary, 00100100 would be $24 (hex) or 36 (decimal).
But if we try to do arithmetic with BCD numbers, we don't get the right results because the computer thinks the numbers are ordinary binary. For instance,

```
  0000 0110   6
+ 0001 0100  14
-----------
  0001 1010     which is not a valid BCD
```

number. We tell the processor to transform this quantity into a valid BCD number with the DAA instruction. The A accumulator would then have:

0010 0000 20

Additional notes

Flag is set if the test is true, reset (cleared) otherwise.

① (Flag V) Test: Result = 1000 0000 ?
② (Flag C) Test: Result <> 0000 0000 ?
③ (Flag C) Test: BCD character of high-order byte greater than 10? (Not cleared if previously set.)
④ (Flag V) Test: Operand = 1000 0000 prior to execution?
⑤ (Flag V) Test: Operand = 0111 1111 prior to execution?
⑥ (Flag V) Test: Set equal to result of N ⊕ C after execution
⑦ (Flag N) Test: Result < 0 ? (Bit 15 = 1)
⑧ (All): Load condition code register from stack
⑨ (Flag I): Set when interrupt occurs. If previously set, a non-maskable interrupt is required to exit the wait state.
⑩ (All): Set according to the contents of accumulator A
⑪ (Flag C) Test: Result of multiplication, is bit 7 of accumulator B = ?

Special operations

Note 1 The RTI, RTS, SWI, and WAI instructions affect the stack and are explained in the interrupt section.

Note 2 BSR and JSR will increment the program counter so that it will point to the next sequential address, *then* push the PC on to the stack, *then* complete a new value for the PC (the address to be branched to). Then, when the RTS pops its return address off the stack into the PC, it will point to the correct place.

Note 3 JMP in the indexed mode will add the displacement (offset) to the value in the index register in computing the number to put into the PC.

For old 6800 hands:

The 6801 adds the following instructions to the basic 6800:

ABX — Adds the 8-bit unsigned accumulator B to the 16-bit X-register taking into account the possible carry out of the low-order byte of the X-register.

ADDD — Adds the 2-byte value found at the specified memory location to the contents of ACCD (accumulator A + accumulator B).

ASLD — Shifts all bits of ACCD one place to the left. Bit 0 is loaded with 0. The C bit is

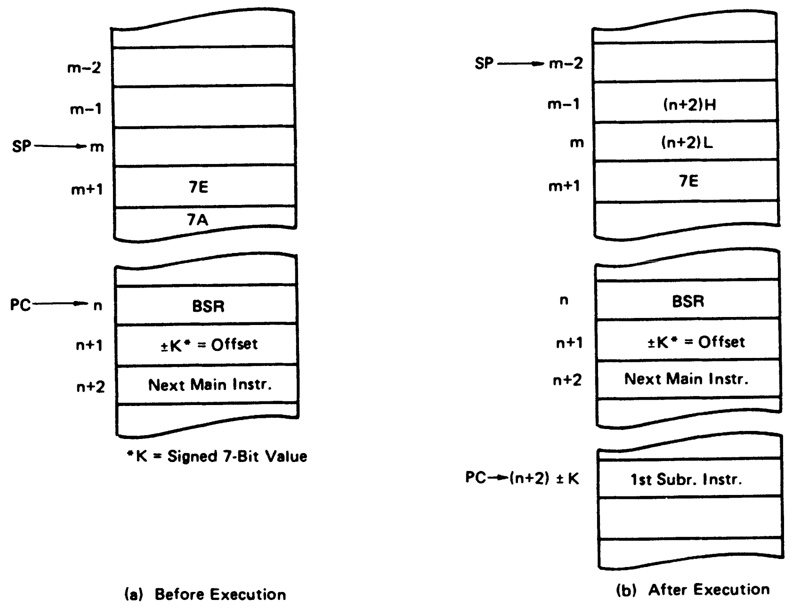


Fig. 5.7 Program flow for BSR

loaded from the most significant (leftmost) bit of ACCD.

LDD — Loads 2 bytes of memory into ACCD.

LSRD — Shifts all bits of ACCD one place to the right. The high order (furthest left) bit of accumulator A is loaded with 0. The C bit is loaded from the least significant bit of accumulator B.

MUL — Multiplies the 8 bits in accumulator A with the 8 bits in accumulator B to obtain a 16-bit unsigned number. The most significant byte is stored in A, the least significant in B.

PSHX — The contents of the index register are pushed onto the stack at the address contained in the stack pointer. The stack pointer is decremented by 2.

PULS — The index register is pulled from the stack beginning at the curent address contained in the stack pointer + 1. The stack pointer is incremented by 2 in total.

STD — Stores the contents of ACCD in memory.

SUBD — Subtracts the contents of a 2-byte memory location from ACCD.

BRN — Never branch. A 2-byte NOP.

Also, the CPX instruction has been changed to permit its use with any conditional branch instruction.

On top of that, the 6301 adds the following instructions to the 6801:

AIM — AND Immediate with Memory; evaluates the AND of the immediate data and the contents of a memory location and stores the result at the memory location.

OIM — OR Immediate with Memory; evaluates the OR of the immediate data and the contents of a memory location and stores the result at the memory location.

EIM — EOR Immediate with Memory; evaluates the exclusive OR (EOR) of the immediate data and the contents of a memory location and stores the result at the memory location.

TIM — Test Immediate with Memory; evaluates the AND of the immediate data and the contents of a memory location and sets the CCR flag.

XGDX — Exchange D with X; exchanges the contents of accumulator D with the index register.

SLP — Sleep; put the microprocessor into the sleep mode.

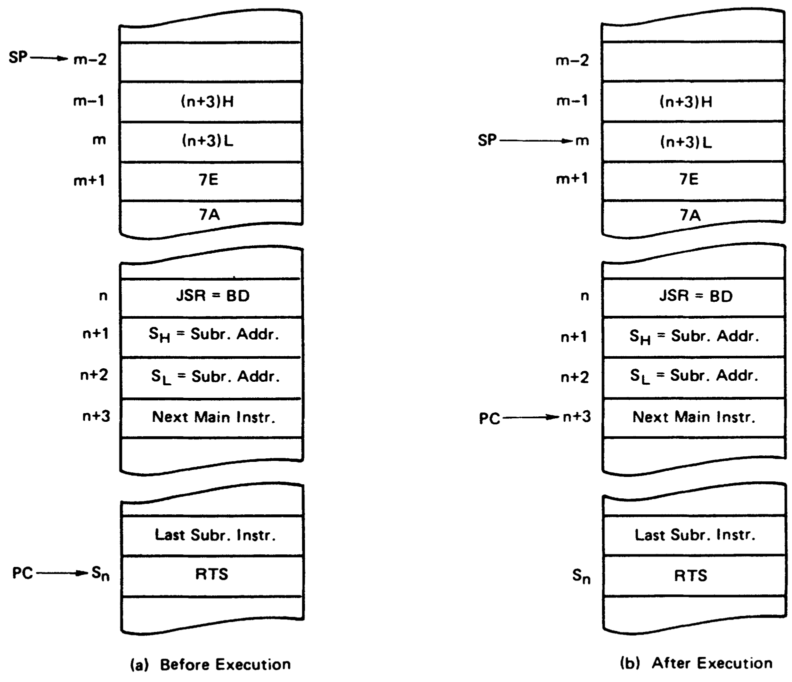


Fig. 5.8 Program flow for RTS

Table 5.6 OP Code map
Reprinted courtesy of Hitachi America

OP CODE						ACC A	ACC B	IND	EXT/DIR*	ACCA or SP				ACCB or X				
										IMM	DIR	IND	EXT	IMM	DIR	IND	EXT	
HI / LO		0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111	
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
1000	0	▨	SBA	BRA	TSX	NEG				SUB								0
0001	1	NOP	CBA	BRN	INS	▨	▨	AIM		CMP								1
0010	2	▨	▨	BHI	PULA	▨	▨	OIM		SBC								2
0011	3	▨	▨	BLS	PULB	COM				SUBD				ADDD				3
0100	4	LSRD	▨	BCC	DES	LSR				AND								4
0101	5	ASLD	▨	BCS	TXS	▨	▨	EIM		BIT								5
0110	6	TAP	TAB	BNE	PSHA	ROR				LDA								6
0111	7	TPA	TBA	BEQ	PSHB	ASR				▨	STA			▨	STA			7
1000	8	INX	XGDX	BVC	PULX	ASL				EOR								8
1001	9	DEX	DAA	BVS	RTS	ROL				ADC								9
1010	A	CLV	SLP	BPL	ABX	DEC				ORA								A
1011	B	SEV	ABA	BMI	RTI	▨	▨	TIM		ADD								B
1100	C	CLC	▨	BGE	PSHX	INC				CPX				LDD				C
1101	D	SEC	▨	BLT	MUL	TST				BSR	JSR			▨	STD			D
1110	E	CLI	▨	BGT	WAI	▨	▨	JMP		LDS				LDX				E
1111	F	SEI	▨	BLE	SWI	CLR				▨	STS			▨	STX			F
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	

UNDEFINED OP CODE ▨
* Only each instructions of AIM, OIM, EIM, TIM

INTERRUPTS

In normal circumstances, the processor is controlled by a program. This program can be the operating system or a machine language program that you've loaded, or the BASIC interpreter executing statements that you've typed in, etc. The processor fetches an instruction in memory, does whatever that instruction calls for, then fetches another instruction. But there are times when we want to interrupt the normal program flow to process special situations.

While your program, for instance, is awaiting keyboard input, the HX-20 goes into a reduced power mode called the sleep mode. But then how do we wake it up? Or, when the power switch on the HX-20 has been turned off, how do we tell the processor that you've turned it on again? The way it's done is through a mechanism common to nearly all computers, a hardware/software combination called an interrupt.

Most microcomputers, including the HX-20, are designed so that certain activities outside of the processor will cause a signal to be sent to a certain pin on the processor chip. When the 6301 gets this IRQ signal, it finishes the instruction it has been processing and before starting another it looks at the interrupt mask in the condition code register. If this bit is on, then interrupts are 'masked off', i.e., they are ignored. If the bit is off, then the processor jumps to a particular address in memory.

There are two other interrupt signals that work similarly. A signal on the NMI pin of the processor is a non-maskable interrupt. When the 6301 gets an NMI, it ignores the setting of the interrupt mask in the condition code register. Similarly, if a signal is present on the RES pin, then the 6301 does a complete reset of the system.

The address that is jumped to depends on the type of interrupt. So far, all of this is handled in hardware, outside the control of the programmer.

The addresses that the 6301 will jump to are specified in a ROM area on the chip itself. These are shown in Table 5.7

Table 5.7

Priority	Vector	Type of Interrupt	Jump 1	Jump 2
Highest	$FFFE – FFFF	Reset (RES)	$E000	$0F
	FFEE – FFEF	Trap	0106	7EDFFA
	FFFC – FFFD	Non-maskable (NMI)	011B	—
	FFFA – FFFB	Software (SWI)	0118	—
	FFF8 – FFF9	IRQ	0115	7EEF49
	FFF6 – FFF7	Timer input (ICF)	0112	7EEF97
	FFF4 – FFF5	Timer output (OCF)	010F	7EF590
	FFF2 – FFF3	Timer overflow (TOF)	010C	7EEF9F
Lowest	FFF0 – FFF1	Serial comm	0109	7EEE4A

Note RES, NMI, and IRQ are normally shown with a bar over them. This means that they are signals which are active when they're logically 0. In other words, if you put a logic probe on the RES pin of the 6301 processor it would show +5 volts (logic 1) all the time *except* when a reset interrupt occurred. But programmers, as opposed to design engineers, don't need to worry about this, so we've dispensed with the bar in our discussions.

Explanation of the table

Priority — If more than one interrupt occurs at one time, then the one with the highest priority is handled first.

Vector — The processor will look at the contents of this location and jump to whatever address is specified there.

Type of interrupt — In addition to the NMI, RES, and IRQ interrupts previously mentioned, there are several others.

Trap — Occurs when the 6301 tries to execute an invalid op code.

SWI — This is a pseudo-interrupt that can be used by the programmer. The SWI (3F) instruction in a machine language program will cause this processing to begin.

SCI — Generated by an event occurring on the high-speed serial path between the two 6301s.

ICF, OCF, TOF timer signals — Not to be confused with real-time clock interrupts, their explanation is highly technical and best left to the manufacturer.

Jump 1 — This is the contents of the vector address. You can verify this by using the HX-20 Monitor. The factory sets this address in the 6301 on board ROM and it is jumped to when at interrupt occurs.

Jump 2 — After the processor jumps to the address that had been specified in high ROM, what we called Jump 1, it tries to execute what it finds there. What it will find there is a jump instruction, that is a 7E followed by a 2-byte address. In the table above, we called this 2-byte address Jump 2 and we can note that it is the address of a subroutine in the HX-20's operating system. RESET is an exception; the first instruction at $E000 is an SEI (0F) to prevent any other interrupts from taking place.

What does all this jumping around mean? We've gone from on board ROM to RAM and then to the operating system in ROM. The reason this is done is to give the programmer the capability of handling the interrupts in his own fashion. Note that if we were to put a different address in Jump 1, then the processor would execute *our* subroutine rather than the one in the operating system.

More usefully, we could 'front-end' an interrupt. That is, we could have our code executed and then as the last instruction in our subroutine, we could have a jump to the operating system's normal interrupt processing routine.

For instance, we could intercept TRAP interrupts by putting an address in $106. When we finished processing, we could issue an RTI (3B) instruction or a JMP (7EDFFA) to the normal Trap routine.

Note, also, that the operating system does not do anything on either NMI or SWI interrupts. It simply doesn't expect to see any. Were it to see any, the processor would try executing the instruction at $11B or $10B. Only there is no instruction there, only zeros, and so a trap interrupt would occur. But, of course, we could put the address of our own routine there.

Figure 5.9 shows interrupt processing. Note the stack activity. Certain operations, including hardware interrupts plus the software interrupt (SWI) and the wait for interrupt (WAI) instructions, cause stack activity — they automatically save the system's registers on the stack.

Say the stack pointer has $FF and the program counter contains $0A60, which is the address of an SWI instruction. $0A goes into location $00FF (where the stack pointer is pointing), $61 goes into location $00FE. (What is saved is the program counter + 1, so that when control returns to the program, it'll pick up at the next instruction.) The index register, which we'll say contains $1234, has the most significant byte ($12) saved at $00FC and the least significant byte ($34) saved at $00FD. Then accumulator A (contents: $22) is saved at $00FB. Accumulator B follows (contents: $33), being saved at $FA. The last thing to be pushed onto the stack is the condition code register, it goes in at $00F9. The stack pointer now points to the next available slot — location $F8. When a RTI (return from interrupt) instruction is issued, all of this data will be pulled off the stack and loaded into the appropriate registers.

THE REAL-TIME CLOCK

The HX-20 contains a time-of-day clock with an

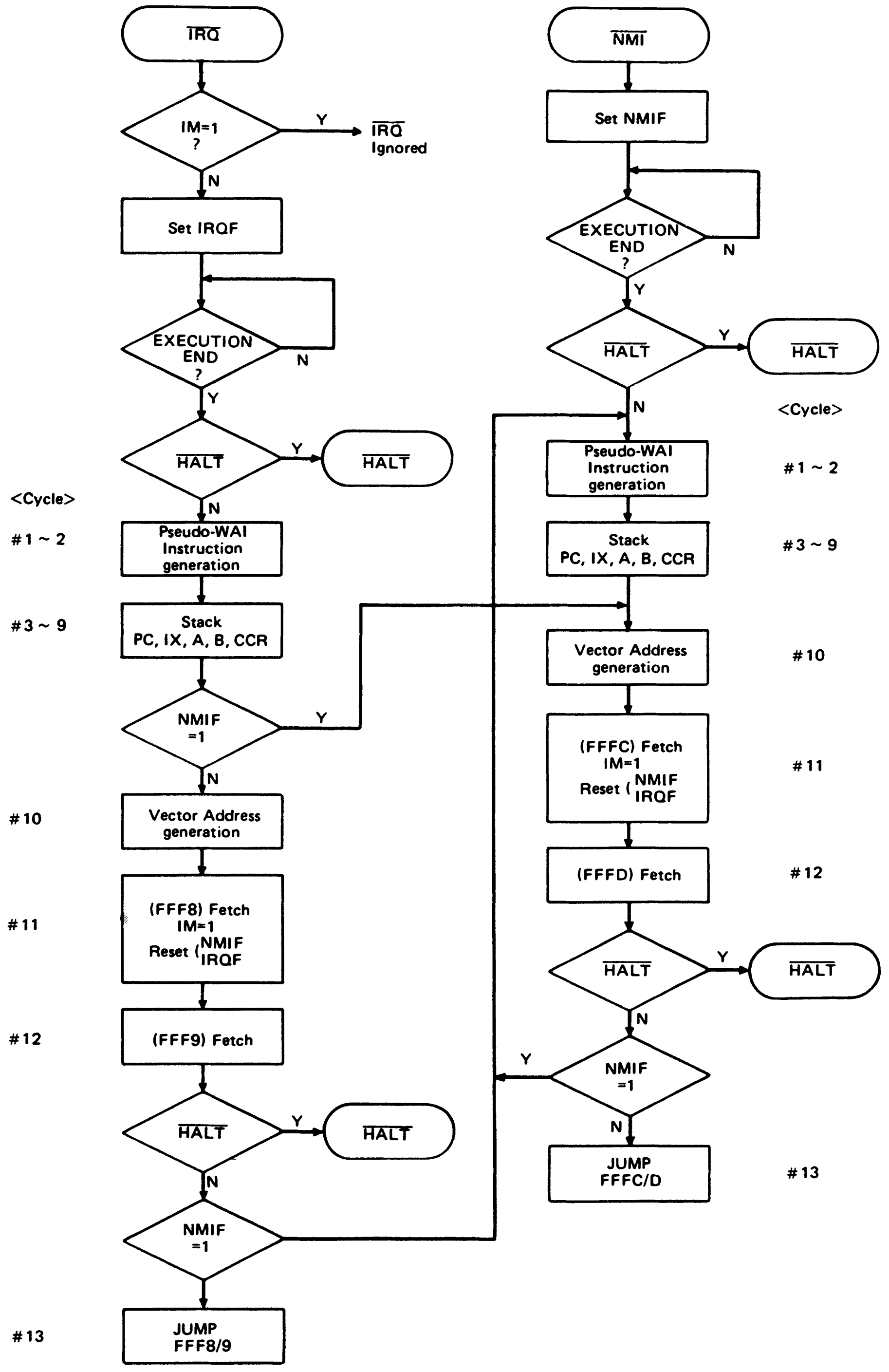


Fig. 5.9 MPU interrupt flow chart.
Reprinted courtesy of Hitachi America

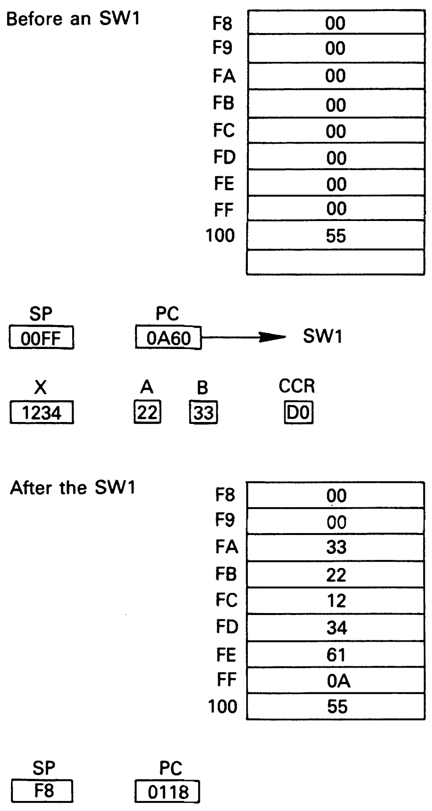


Fig. 5.10 Stack processing on interrupts

alarm and a 100-year calendar. Some of the clock's functions are available through BASIC, some aren't. Using BASIC, the programmer can put a time–date stamp on any event that occurs while his program is in control. This capability has been used in a number of commercial programs that we've seen. But there is more to the clock than that and we hope that future programmers will find other uses for it.

The clock is a CMOS chip, specifically the HD146818 from Hitachi. The following has been largely drawn from Hitachi America's *Microcomputer Data Book.*

Some of the clock's features are:
- time-of-day and calendar
- counts seconds, minutes, and hours of the day
- counts days of the week, date, month, and year
- binary or BCD representation of time, calendar, and alarm; 12 or 24 hour clock with AM and PM in 12 hour mode
- automatic end of month recognition
- automatic leap year recognition
- interfaced as 64 RAM locations to be accessible to software
  - 14 bytes of clock and control registers
  - 50 bytes of general purpose RAM
- three interrupts are separately software maskable and testable

Table 5.8 Clock modes
Reprinted courtesy of Hitachi America

Address Location	Function	Decimal Range	Range		Example*	
			Binary Data Mode	BCD Data Mode	Binary Data Mode	BCD Data Mode
0	Seconds	0~59	\$00~\$3B	\$00~\$59	15	21
1	Seconds Alarm	0~59	\$00~\$3B	\$00~\$59	15	21
2	Minutes	0~59	\$00~\$3B	\$00~\$59	3A	58
3	Minutes Alarm	0~59	\$00~\$3B	\$00~\$59	3A	58
4	Hours (12 Hour Mode)	1~12	\$01~\$0C (AM) and \$81~\$8C (PM)	\$01~\$12 (AM) and \$81~\$92 (PM)	05	05
	Hours (24 Hour Mode)	0~23	\$00~\$17	\$00~\$23	05	05
5	Hours Alarm (12 Hour Mode)	1~12	\$01~\$0C (AM) and \$81~\$8C (PM)	\$01~\$12 (AM) and \$81~\$92 (PM)	05	05
	Hours Alarm (24 Hour Mode)	0~23	\$00~\$17	\$00~\$23	05	05
6	Day of the Week Sunday = 1	1~7	\$01~\$07	\$01~\$07	05	05
7	Day of the Month	1~31	\$01~\$1F	\$01~\$31	0F	15
8	Month	1~12	\$01~\$0C	\$01~\$12	02	02
9	Year	0~99	\$00~\$63	\$00~\$99	4F	79

* Example: 5:58:21 Thursday 15th February 1979

- time-of-day alarm, once-per-second to once-per-day
- periodic rates from 30.5 microseconds to 500 milliseconds
- end-of-clock update cycle

Ten RAM locations are used for time, calendar, and alarm. Another four are for control registers. In the HX-20, this RAM starts at location $40. (Remember to set $7E to $80 if you want to look at this area with the Monitor.)

Table 5.8 shows the contents of the first 10 bytes, starting at memory location $40. Shown in Table 5.9 are the four control registers: (b7 is the high-order, $80 bit, b0 is the low-order, $01 bit).

Explanation of the clock control registers

UIP — The update in progress flag is a status flag that may be monitored by the program. When UIP is a '1' the update cycle is in progress or will soon begin. When UIP is a '0' the update cycle is not in progress and will not start for at least 244 microseconds. The time, calendar, and alarm information in RAM is fully available to the program when the UIP bit is zero, that is, when the information is not in transition.

The update cycle is normally done once per second. Its primary function is to increment the seconds byte, check for overflow, increment the minutes byte when appropriate, and so forth through to the year byte. The update cycle also compares each alarm byte with the corresponding time byte and issues an alarm if a match or if a 'don't care' code (11xx xxxx) is present in all three positions. During the update cycle, the time, calendar, and alarm bytes are not accessible by the program.

DV2, DV1, DV0 — The divider bits set the time base.

000 is a 4.19 MHz clock
001 is a 1.05 MHz clock
010 is a 32.7 kHz clock

RS3, RS2, RS1, RS0 — These four bits select a rate at which a periodic interrupt can be generated. Some permissible values are:

Register A	Interrupt period
x00x 0000	none
x00x 0001	30.5 us
x00x 0010	61.0 us
x00x 0011	122.1 us
x00x 1000	3.9 ms
x00x 1111	500 ms

Each step increase in register A doubles the length of the interrupt period.

SET — When the SET bit is a '0', the update cycle functions normally by advancing the counts once-per-second. When the SET bit is a '1', then any update cycle in progress is aborted and the program may initialize the time and calendar bytes without an update occurring in the midst of intializing.

PIE — The periodic interrupt enable bit is a read/write bit which causes periodic interrupts to be effective. Putting a '1' in PIE sets the interrupts at the rate specified by the RS3, RS2, RS1, and RS0 bits in register A. A '0' means that the periods are still being ticked off, but no interrupts will take place.

AIE — The alarm interrupt enable bit is a read/write bit which causes alarm interrupts to be effective. A '1' in AIE will produce an alarm interrupt for each second that the three time bytes equal the three alarm bytes. A '0' means no interrupts will occur.

UIE — The update-ended interrupt enable bit will set this interrupt active on a '1', not active on a '0'.

DM — The data mode bit indicates whether

Table 5.9
Clock control registers

	MSB							LSB	Memory Location
	b7	b6	b5	b4	b3	b2	b1	b0	
Register A	UIP	DV2	DV1	DV0	RS3	RS2	RS1	RS0	4A
Register B	SET	PIE	AIE	UIE	SQWE	DM	24/12	DSE	4B
Register C	IRQF	PF	AF	UF	0	0	0	0	4C
Register D	VRT	0	0	0	0	0	0	0	4D

time and calendar updates are to use binary or BCD formats. A '1' means binary, '0' means BCD.

24/12 — A '1' means use the 24 hour clock, a '0' means use the 12 hour clock.

DSE — A '1' means that the clock should switch to daylight saving time on the last Sunday in April and be switched back again on the last Sunday in October. A '0' means no daylight saving time.

IRQF — If '1', then at interrupt has occurred. That is, PF and PIE were each equal to 1 OR AF and AIE were each equal to 1 OR UF and UIE were each equal to 1.

PF — The periodic interrupt flag is a read-only bit that will go to 1 when a periodic interrupt is capable of taking place. (Whether one will or not depends on how PIE is set.)

AF — The alarm interrupt flag will be '1' when the current time matches the alarm time. If AIE is also a '1', then an interrupt will take place.

UF — The update-ended interrupt flag is set after each update cycle. If UIE is also a '1', then an interrupt will occur.

As can be seen from the above, there are three distinct sources of interrupts that can be handed to the 6301. The alarm interrupt may be programmed to occur at rates from once per second to one a day. The periodic interrupt may be selected for rates from half-a-second to 30.517 microseconds. The update-ended interrupt may be used to indicate to the program that an update cycle has been completed.

The processor program selects which interrupts, if any, it wishes to receive by setting the bits in register B. If more than one type of interrupt can occur, the program can check the contents of register C to see which has occurred. If the programmer decides not to have the program interrupt-driven, then he can still tell whether an interrupt would have occurred by checking the contents of register C. Every time register C is read, the contents are set back to 0. Also location $7D, bit $08, is set by the operating system when a clock alarm has occurred.

Further information on the clock is available from Hitachi America and from Epson.

6

ASSEMBLY LANGUAGE

'Profanity is the one language understood by all programmers.'

This chapter covers:
- Why use machine/assembly language?
- An assembler
- Other assemblers
- Storing machine code into memory
- Using the Monitor as a learning tool
- Assembly language coding hints, sample program

WHAT IS ASSEMBLY LANGUAGE AND WHY USE IT?

Using the language of the machine — the hex, really binary, digits that the processor understands — opens up more of the computer's power to us. Coding directly in machine language as opposed to BASIC gives us more speed and more functions. You'll probably find that it is most practical to use BASIC for most of a program, and those parts of the program that would be too difficult to write in BASIC or are time-sensitive can be done in machine language.

Assembly language may seem forbidding if you haven't been exposed to it before. It's not like BASIC at all. In fact, you should forget everything you know that's specific to BASIC before trying to learn assembly language. Just remember your fundamental programming techniques that are applicable to any language: what a branch is, what a loop is, etc. Read the previous chapter on machine architecture and go on from there.

But assembly language isn't hard, just different. People who learned assembly language as their first language — as this writer did (RPG doesn't count) — find it comfortable to use and easy to debug.

When you looked at the previous chapter, you noticed that the HX-20 has a lot of different machine instructions. But you really don't need to know very many of them to write an assembly language program. It's been estimated, for instance, that 70% of a typical assembly language program consists of only three groups of instructions: load registers, store into memory, and branch. Add a couple of other instructions — add 1 to a register, subtract 1 from a register — and you're practically all set.

Coding directly in machine language can become difficult when actual memory locations are used. Any time the program is changed, for instance, those locations may change. So assemblers were developed. With an assembler, you write a program using a special set of reserved words, in a specified format, and the output is converted into machine language. You can write your program in assembly language first and then use a number of methods, POKE for instance, to get the resulting machine code into memory.

The following is a typical assembly language program and one that you may want to use. What it does is allow you to call any machine language routine from BASIC, including the routines in the HX-20's ROM. Basically, what the program does is look in certain locations for data that has been saved there by a BASIC program, take that data and load it into the registers, and then call the routine whose address has also been stored in memory by the BASIC program. After the return from the called routine, this program will store the contents of the registers back into the same locations that were set on entry.

```
0    000                      ORG    $0AE0
1    AE0    FE0AFB            LDX    RTRN
2    AE3    3C                PSHX
3    AE4    FE0B01            LDX    ROM
4    AE7    3C                PSHX
5    AE8    FE0AFD            LDX    REGX
6    AEB    B60AFF            LDAA   REGA
7    AEE    F60B00            LDAB   REGB
8    AF1    39                RTS
9    AF2    FF0AFD            STX    REGX
10   AF5    B70AFF            STAA   REGA
11   AF8    F70B00            STAB   REGB
12   AFB    01        RTRN    NOP
13   AFC    39                RTS
14   AFD    0000      REGX    $0000
15   AFF    00        REGA    $00
16   B00    00        REGB    $00
17   B01    0000      ROM     $0000
```

To call this from BASIC, you could do:

```
MEMSET &HB02          'Provide room for the
                       program
POKE &HAFF,&H30       'Load the A accumulator
POKE &HB00,&H46       'Load the B accumulator
POKE &HB01,&HFF       'First byte of a ROM routine
                       address
POKE &HB02,&H2B       'Second byte of a ROM
                       routine address
EXEC &HAE0            'Call the routine
PRINT PEEK (&HAFF)    'Check for a returned value
```

If the index register needs to be loaded, then:

If the index register needs to be loaded, then:

```
POKE &HAFD, first byte
POKE &HAFE, second byte
```

The listing above was transcribed from the output of the assembler which appears later in this chapter. We want you to note several things:

- Each assembler instruction takes up one line.
- Each assembler instruction corresponds to one machine operation, except for special instructions like ORG that tell the assembler what to do.
- The listing shows the locations in memory into which the program is to be stored: AE0-B01.
- The listing shows the machine code into which the instruction is assembled.

All assemblers follow those conventions. Different assemblers may assign different names, called mnemonics, to the machine's instruction côdes and may require that the operands be entered differently but they all adhere to the above conventions. More sophisticated assemblers, called macro assemblers, allow you to call several assembly language instructions with one macro instruction. But there is still one assembly language instruction to one machine instruction.

Another thing common to all assemblers are labels. (If you want to branch somewhere, you need to know where to branch.) Our assembler uses 4-character labels. Other assemblers may allow longer labels.

You may find it possible to learn assembly language just from this book. If it still seems too confusing, try one of the following books:

Basic Microprocessors and the 6800 — Bishop (Hayden, 1979);
6800 Assembly Language Programming — Leventhal (Osborne, 1978);
Using Microprocessors and Microcomputers: The 6800 Family — Greenfield and Wray (John Wiley, 1981).

If you have trouble locating these books locally or from the publishers, try the YES! bookshop (address in Appendix B).

AN ASSEMBLER

What follows is an assembler that you can use to develop your own assembly/machine language programs. It's written in BASIC, it's slow, and it can't hold large programs without the expansion unit. But it's considerably more useful than trying to hand-assemble machine code. Other alternatives are the assemblers available from software publishers, mentioned later, and cross-assemblers that run on other systems. (The idea behind a cross-assembler is that you develop the code on a system with good development tools such as a good text editor, and then download or otherwise transfer the machine code into the target machine.)

This program was originally written by Robert Labenski to run on a TRS-80 Model I, producing cross-assembled code for a 6800 system. It was originally published in the magazine *Byte* (Dec. 1981). We took that listing, added the 6301 instructions and made many further enhancements. We are considering making this program available on microcassette, so if interested, contact us at the address in the Preface.

The assembler is easy to use. A list of commands is displayed upon start-up and any time an H (or Help) is typed after a Ready* prompt. Use the SCRN key to review the part of the list that scrolls off the screen.

The available commands are:

I(nsert) — insert a new line of source code before a specified line. If no line is specified, insert after the last line of code in the program.
D(elete) — remove a line of source code (or multiple lines).
C(clear) — clear out any previous source code.
L(ist) — display all source code or any single line or any range of lines.
P(rint) — same as List, but send the output to the printer.
A(ssemble) — assemble the source code that has been entered.
S(ymbol) — display the symbol table created after the assembly process.
M(emory) — store the created object into memory using the assembled address locations.
E(xamples) — display examples of each type of instruction.

The assembler is column oriented and depends on the use of the TAB key to line up labels in one column, operations in another, operands in another.

```
10  ' Mini 6301 Assembler
20  ' Original code Copyright 1981 Robert Labenski.
30  ' Enhancements Copyright 1983 Eric Balkan.
40  ' Original 6800 version first published in
    Byte 12/81
130 CLEAR 1000: DEFINT A-Z
135 WIDTH 46,16
137 SCROLL 9,0,8,4
140 DIM S$(160) 'Source Data
150 DIM NO$(156) 'Implied Operands
160 DIM OP$(41) 'Full Opcodes
170 DIM BR$(17) 'Br inst
180 DIM OB$(160) 'Object
190 DIM AD(160) 'Address
200 DIM LA$(70) 'Source Labels LC=Index
210 DIM LN(70) 'Line # of Labels
220 DIM AR(80) 'Addr Resolution AC=Index
225 DIM EQ$(30) 'Equates
230 GOSUB 1550:GOTO 1200
240 RESTORE 'Assemble
250 LC=0: AC=0: CD=0:EQ=0
260 IF OT THEN 340 ELSE OT=1:GOTO 310
270 CD=0:FOR X=1 TO LEN(A$):Y=ASC
    (MID$(A$,X,1))
280 IF Y<=57 AND Y>=48 THEN Y=Y-48
290 IF Y>64 THEN Y=Y-55
300 CD=16*CD+Y:NEXT:RETURN
305 'Build instr table
310 FOR A=O TO 55:READ NO$(A):NEXT
320 FOR A=O TO 40:READ OP$(A):NEXT
330 FOR A=O TO 16:READ BR$(A):NEXT
340 OK=1 'Main assembly loop
350 FOR A=O TO N-1
360 IF LEFT$(S$(A),1)="*" THEN OB$(A)=" ":
    AD(A)=CD:GOTO 450
370 IF MID$(S$(A),9,1)<>" ' " THEN 400
380 AD(A)=CD
390 OB$(A)=" ":FOR B=10 TO 40:A$=MID$(S$(A),
    B,1):IF A$=" ' " THEN 450 ELSE Y=ASC(A$):
    X=0:A$=" ":GOSUB 950:OB$(A)=OB$(A)+A$:
    CD=CD+1:NEXT
400 A$=MID$(S$(A),9,4):IF LEN(A$)=3 THEN
    A$=A$+"  "
410 IF A$="ORG" THEN A$=MID$(S$(A),18,4):
    OB$(A)=" ":GOSUB270:GOTO 450
420 IF LEFT$(S$(A),4)<> "  " THEN LA$(LC)=LEFT$
    (S$(A),4):LN(LC)=A:LC=LC+1
425 IF A$="EQU " AND MID$(S$(A),17,1)="$"
    THEN A$=MID$(S$(A),18,4):OB$(A)=" ":
    EQ$(EQ)=LEFT$(S$(A),4)+A$+"$":
    EQ=EQ+1:GOTO450 ELSE IF A$="EQU "
    THEN A$=MID$(S$(A),17,4):OB$(A)=" ":
    EQ$(EQ)=LEFT$(S$(A),4)+A$:EQ=EQ+1:
    GOTO 450
429 IF LEFT$(A$,3)="BIT" THEN 440
430 IF LEFT$(A$,1)="B" THENGOSUB 710:GOTO
    450
440 IF LEN(S$(A))<17 THENGOSUB 530 ELSE
    GOSUB 600
450 NEXT A
460 IF SW=0 THEN 520
470 FOR A=0 TO AC-1
475 FOR B=0 TO EQ-1:IF RIGHT$(OB$(AR(A)),4)<>
    LEFT$(EQ$(B),4) THEN NEXT ELSE
    OB$(AR(A)=LEFT$(OB$(AR(A)),
    LEN(OB$(AR(A)))-4 + MID$(EQ$(B),5,4):IF
    RIGHT$(EQ$(B),1)="$" THEN 510
480 FOR B=0 TO LC-1:IF RIGHT $(OB$(AR(A)),
    4)<>LA$(B) THEN NEXT ELSE 490
485 OB$(AR(A))="?Labl":GOTO510
490 IF MID$(S$(AR(A)),9,1)="B" THEN X=
    AD(AR(A)):Y=AD(LN(B)):AD(100)=Y-(X+2):
    C=100: GOSUB 940: OB$(AR(A))=
    LEFT$(OB$(AR(A)),2) + RIGHT$(A$,2):GOTO
    510
500 C=LN(B):GOSUB 940:OB$(AR(A))=LEFT$(OB$
    (AR(A)),2)+"0"+A$
510 NEXT A
520 RETURN
530 ' Implied Operands
540 IF MID$(S$(A),9,1)="$" THEN OB$(A)=RIGHT
    $(S$(A),LEN(S$(A))-9:AD(A)=CD:CD=CD+
    LEN(S$(A))-9)/2:RETURN
550 FOR B=0 TO 55
560 IF LEFT$(NO$(B),4)<A$ THEN NEXT
565 IF LEFT$(NO$(B),4)>A$ THEN OB$(A)=
    "?INST": RETURN
570 OB$(A)=RIGHT$(NO$(B),2):AD(A)=CD:CD=
    CD+1:RETURN
580 NEXT
585 OB$(A)="?Inst":RETURN
600 ' Other OPs
```

```
610 AD(A)=CD
620 FOR B=0 to 40
625 IF LEFT$(OP$(B),4)<A$ THEN NEXT
630 IF LEFT$(OP$(B),4)>A$ THEN OB$(A)="?Inst":
    RETURN
645 IF MID$(OP$(B),2,2)="IM" THEN GOSUB 1800:
    RETURN
650 IF MID$(S$(A),20,2) = ",X" THEN OB$(A) =
    MID$(OP$(B),10,2)+MID$(S$(A),18,2):CD=
    CD+2:RETURN
660 IF MID$(S$(A),17,1) = "#" THEN OB$(A) =
    MID$(OP$(B),6,2):OB$(A) = OB$(A)
    +MID$(S$(A),19,2): CD=CD+2: B$=LEFT$
    (OB$(A),2): IF B$<>"8C" AND B$<>"CE" AND
    B$<> "8E" THEN RETURN ELSE CD=CD+1:
    OB$(A)=OB$(A)+RIGHT$(S$(A),2):RETURN
670 IF MID$(S$(A),17,1)=" " THEN OB$(A)=
    "?????":RETURN
680 IF MID$(S$(A),17,1)="$" THEN A$=
    MID$(S$(A),18,4) ELSE A$=MID$(S$(A),17,4):
    AR(AC)=A: AC=AC+1: SW=1:A$=
    A$+STRING$(4-(LEN(A$))," ")
690 IF LEN(A$)=4 THEN OB$(A)=MID$(OP$(B),12,
    2):OB$(A)=OB$(A)+A$:CD=CD+3:RETURN
700 OB$(A)=MID$(OP$(B),8,2):OB$(A)=OB$(A)
    +A$:CD=CD+2:RETURN
710
720 FOR B=0 TO 16
722 IF LEFT$(BR$(B),2)<MID$(A$,2,2) THEN NEXT
725 IF MID$(A$,2,2)<LEFT$(BR$(B),2) THEN
    OB$(A)="??BR ":RETURN
740 OB$(A)=RIGHT$(BR$(B),2):AD(A)=CD:CD=
    CD+2:AR(AC)=A:AC=AC+1
750 A$=MID$(S$(A),17,4):OB$(A)=OB$(A)+A$+
    STRING$(4-LEN(A$)," "):SW=1:RETURN
760 OK=0:LC=0:AC=0
770 IF LEN(A$>1 THEN 810
775 IF N<0 THEN N=0
780 LINE INPUT S$(N)
790 IF S$(N)=" " THEN RETURN
800 N=N+1:GOTO 780
810 A=VAL (RIGHT$(A$,LEN(A$)-1)): IF A>N
    THEN 780
820 LINEINPUT A$
830 IF A$=" " THEN RETURN
840 FOR B=N+1 TO A STEP -1:IF B=0 THEN 850
    ELSE S$(B)=S$(B-1):NEXT
850 S$(A)=A$:A=A+1:N=N+1:GOTO 820
860 IF LEN(A$)=1 THEN A=0:B=N-1:GOTTO 900
880 A=VAL(RIGHT$(A$,LEN(A$)-1)):B=A
890 IF MID$(A$,3,1)="-" THEN B=VAL
    (MID$(A$,4,2))
895 IF MID$(A$,4,1)="-" THEN B=VAL (MID$
    (A$,5,2))
900 IF B>N THEN B=N-1
910 IF A>N THEN A=N-1
915 IF A<0 THEN A=0
920 IF OK THEN FOR C=A TO B: GOSUB 940:
    PRINT C;TAB(4)A$;"  ";OB$(C);TAB(18)S$(C)
    ELSE GOTO 930
926 IF B$="P" THEN LPRINT C;TAB(4)A$;"  ";
    OB$(C);TAB(18)S$(C)
927 NEXT:RETURN
930 FOR C=A TO B:PRINT C;"  ";S$(C):IF B$="P"
    THEN LPRINT C;"  ";S$(C)
935 NEXT:RETURN
940 A$=" ":Y=AD(C):X=INT(Y/256):GOSUB 970
950 X=INT((Y-(X*256))/16):GOSUB 970
960 X=INT(Y-(INT(Y/16)*16))
970 IF X>9 AND X<16 THEN A$=A$+CHR$(X+55)
    ELSE A$=A$+RIGHT$(STR$(X),1)
980 RETURN
990 OK=0:LC=0:AC=0
1000 B=VAL(RIGHT$(A$,LEN(A$)-1))
1010 IF B>N THEN RETURN
1020 FOR C=B TO N-1:S$(C)=S$(C+1):NEXT
1030 N=N-1:RETURN
1040 'Symbol Print
1060 FOR A=0 TO LC-1:C=LN(A):GOSUB 940:
     PRINT LA$(A);"  ";LN(A);"  ";A$:
1070 NEXT:RETURN
1080 INPUT "L=Load S=Save ";B$
1100 IF(B$<>"S")*(B$<>"L") THEN RETURN
1110 INPUT " File  Specs ";A$
1120 IF B$="S" THEN 1170
1130 OPEN "I",1,A$:INPUT#1,OK,N
1140 ' FOR A=0 TO N-1:INPUT #1,S$(A),OB$(A),
     AD(A):NEXT
1145 FOR A=0 TO N-1:INPUT#1,ZZ$,S$(A),ZZ$,
     ZZ$,OB$(A),ZZ$AD(A):NEXT
1150 CLOSE:RETURN
1160 PRINT "No Source":RETURN
1170 IF N=0 THEN 1160 ELSE OPEN "O",1,A$:
     PRINT#1,OK;N;
1180 FOR A=0 TO N-1:PRINT#1,CHR$(34);S$(A);
     CHR$(34);CHR$(34); OB$(A);CHR$(34);AD(A);:
     NEXT
1190 B$=" ":CLOSE:RETURN
1200 LINEINPUT "Ready* ";A$:B$=LEFT$(A$,1)
1220 IF B$="L" OR B$="P" THEN GOSUB 860
1230 IF B$="I" THEN GOSUB 760
1240 IF B$="D" THEN GOSUB 990
1250 IF B$="C" THEN INPUT "Sure (Y/N)";Z$:IF Z$
     ="Y" THEN 130
1260 IF B$="A" THEN GOSUB 240
1270 IF B$="F" THEN GOSUB 1080
1280 IF B$="S" THEN GOSUB 1040
1290 IF B$="H" THEN GOSUB 1550
1293 IF B$="?" THEN GOSUB 1550
1295 IF B$="E" THEN GOSUB 1610
1297 IF B$="M" THEN GOSUB 1900
1300 GOTO 1200
1301 'Implied Operands
1315 DATA ABA 1B,ABX 3A,ASLA 48,ASLB 58,
     ASLD 05,ASRA 47,ASRB 57
1320 DATA CLC 0C,CLI 0E,CLRA 4F,CLRB 5F,CLV
     0A,COMA 43,COMB 53
1322 DATA DAA 19,DECA 4A,DECB 5A,DES
     34,DEX 09,INCA 4C,INCB 5C,INS 31,INX 08
1325 DATA LSRA 44,LSRB 54,LSRD 04,MUL
```

```
     3D,NOP 01
1340 DATA PSHA 36,PSHB 37,PSHX 3C,PULA
     32,PULB 33,PULX 38,ROLA, 49,ROLB 59,RORA
     46,RORB 56,RTI 3B,RTS 39
1360 DATA SBA 10,SEC 0D,SEI 0F,SEV 0B,SLP
     1A,SWI 3F, TAB 16,TAP 06,TBA 17,TPA
     07,TSTA 4D,TSTB 5D,TSX 30,TXS 35,WAI
     3E,XGDX 18
1420 ' Other Operands
1430 DATA ADDA 8B9BABBB,ADDB CBDBEBFB,
     ADCA 8999A9B9, ADCB C9D9E9F9,ADDD
     C3D3E3F3
1440 DATA AIM ??7161??,ANDA 8494A4B4,ANDB
     C4D4E4F4, BITA 8595A5B5,BITB C5D5E5F5
1450 DATA CLR ????6F7F,CMPA 8191A1B1,CMPB
     C1D1E1F1,CPX 8C9CACBC, DEC ????6A7A,EIM
     ??7565??,EORA 8898A8B8,EORB C8D8E8F8,INC
     ????6C7C
1455 DATA JMP ????6E7E,JSR ??9DADBD
1460 DATA LDAA 8696A6B6,LDAB C6D6E6F6,LDD
     CCDCECFC,LDS 8E9EAEBE, LDX CEDEEEFE,
     OIM ??7262??,ORAA 8A9AAABA,ORAB
     CADAEAFA
1470 DATA SBCA 8292A2B2,SBCB C2D2E2F2
1505 DATA STAA ??97A7B7,STAB ??D7E7F7,STD
     ??DDEDFD,STS ??9FAFBF, STX ??DFEFFF,
     SUBA 8090A0B0,SUBB C0D0E0F0,SUBD
     8393A3B3, TIM ??7B6B??,TST ????6D7D
1530 ' Branch Instructions
1540 DATA CC24,CS25,EQ27,GE2C,GT2E,HI22,
     LE2F,LS23,LT2D,M12B,NE26,PL2A,RA20,RN21,
     SR80,VC28,VS29
1550 ' Operator's Guide
1560 CLS:PRINT "H=Help (This Page)"
1562 PRINT "F=File (Save/Load)"
1570 PRINT "I=Insert"
1572 PRINT "Ixx=Insert before L#xx"
1580 PRINT "Dxx=Delete Line#xx"
1582 PRINT "C=Clear"
1590 PRINT "L=List all text"
1592 PRINT "Lxx=List Line#xx"
1594 PRINT "Lxx-xx=List Range"
1596 PRINT "P=Print Text"
1600 PRINT "A=Assemble"
1605 PRINT "S=Symbol Display"
1606 PRINT "M=Put Obj in Memory"
1608 PRINT "E=Examples"
1609 RETURN
1610 PRINT "Instruction Examples"
1620 PRINT "Immed (ADDA #$1A)"
1630 PRINT "Direct (ADDA $1A)"
1640 PRINT "Indxd (ADDA $1A,X)"
1650 PRINT "Extnd (ADDA $0A40)"
1660 PRINT "Implied — No Operand"
1665 PRINT "Other (OIM $10,#$01)"
1666 PRINT " (OIM $10,X,#$01)"
1668 PRINT "LBL1 EQU LBL2"
1669 PRINT "LBL1 EQU $XXXX"
1670 PRINT "ORG $XXXX"
1672 PRINT "Literals $XX For Hex"
1674 PRINT "   'XX' For ASCII"
1680 PRINT "Source is Positional"
1682 PRINT "(Use Tabs)"
1690 PRINT "LABEL t OP t OPERAND"
1695 RETURN
1800 ' 3-Operand Instructions
1810 IF MID$(S$(A),20,2)=",X" THEN OB$(A)=
     MID$(OP$(B),10,2) + MID$(S$(A),25,2)+MID$
     (S$(A),18,2):CD=CD+3:GOTO 1830
1820 OB$(A)=MID$(OP$(B),8,2)+MID$(S$(A),23,2)
     + MID$(S$(A),18,2):CD=CD+3
1830 RETURN
1900 FOR C=0 TO N-1
1910 GOSUB 940
1920 IF OB$(C)=" " THEN 1950
1925 FOR LL=0 TO LEN (OB$(C))/2-1
1927 GOSUB 2000
1930 POKE AD(C)+LL,B
1940 NEXT LL
1950 NEXT C
1960 RETURN
2000 CB$="0123456789ABCDEF"
2005 B=0
2030 FOR CX=1 TO 16
2040 IF MID$(OB$(C),2*LL+1,1)=MID$(CB$,CX,1)
     THEN B=(CX-1)*16:GOTO 2055
2050 NEXT CX
2055 FOR CX=1 TO 16
2060 IF MID$(OB$(C),2*LL+2,1)=MID$(CB$,CX,1)
     THEN B=(CX-1)+B:GOTO 2080
2070 NEXT CX
2080 RETURN
```

How the assembler program works

100–225 — Initialization: If the assembler is not to be run on an HX-20, then comment out (i.e., put a quote in front of) lines 135 and 137. The array sizes selected have been tested, but feel free to change them to suit your needs.
230 — Print out the list of operator's commands and ask for one to be selected. If you have hit BREAK while running the program and don't want to (or can't) continue from where you were, you can issue a GOTO 230 to restart without losing any of your source code.
240–300 — Assemble the source code. If this is the first assembly, go build the instruction table first (310).
310–330 — Read the set of 6301 instructions (DATA statements) into three arrays, one for instructions with implied operands, one for non-branch instructions with operands, and one for branch instructions.
340–520 — Assemble the program. ORGs and EQUs are handled separately from 6301 instructions. Possible error messages:

?Inst — the instruction op code is not valid
?Labl — the label is undefined
?Br — invalid branch instruction
????? — indeterminate error

540–585 — Subroutine to assemble instructions with implied operands.
600–700 — Subroutine to assemble instructions with operands.
710–750 — Subroutine to assemble branch instructions.
760–850 — Insert a line of source code.
860-980 — List program on screen or printer.
990–1030 — Delete a line of source code.
1040–1070 — Display the symbol table.
1080–1190 — Load or save the source program to/from tape. If not running on the HX-20, line 1140 may need to be used instead of 1145.
1200–1300 — Ask for a command from the user and execute the appropriate subroutine. Then return to ask for another command.
1301–1540 — The lists of valid 6301 instruction op codes.
1550–1609 — The list of permissible operator commands.
1610–1695 — Some information on the format of 6301 instructions.
1800–1830 — This routine handles those 6301 instructions that have three operands.
1900–1960 — Places object code in memory.
2000–2080 — Compresses a printable (zoned) hex character into its binary equivalent. That is, the object code displayed by the mini-assembler is not stored as hex numbers. For instance, an LDX $01,X instruction assembles as A601, but what we store in memory is: 41 36 30 31. (So it's readable.) Called from line 1920, this subroutine would take the 41 36 30 31 and store it into memory as A6 01 so that it could be executed.

There have been a great number of magazine articles written about the 6800 since its creation, especially at the height of its popularity in 1977–78. You may find some of these useful. Here are a few to look up:

Explained: String Interpretations. Parsing Techniques for the 6800, by Gary Gaugler, *Kilobaud Microcomputing*, 4/78. Well-written article on how to parse (interpret) character strings, with examples and diagrams.

6800 Trace and Disassemble Program, by Richard Carickhoff, *Kilobaud Microcomputing*, 5/80. Uses some Mikbug I/O routines, so will not run on the HX-20, but could give you some ideas. No source code provided, just hex object code.

GPM for the M6800, by Fritz van der Wateren, *Dr Dobb's Journal*, 11–12/77. A macro generator. Assembly language source code provided.

M6800 Disassembler, by Gordon Stallings, *Dr Dobb's Journal*, 3/77. Written in assembler language.

Address List Program, by C.H. Looney, *Kilobaud Microcomputing*, 10/80. In 6800 machine language.

Build a SISTER for your 6800, by Ray Boaz, *Kilobaud Microcomputing*, 12/79. Hardware: single stepper.

Condensed Reference Chart for the 6800, by Robert J. Bormann, *Byte*,7/77. Information on instruction groups; looking for 'holes' in the instruction set.

Undocumented M6800 Instructions, by Gerry Wheeler, *Byte* 12/77. Experiences testing some of the holes mentioned above (but will be different on the Hitachi).

The Motorola 6800 Instruction Set, by Paul M. Jessop, *Byte*, 1/78. Another way to look at the instruction set.

Hand Assembling M6800 Relative Addresses, by Ray Boaz, *Byte*, 4/78. A handy table for calculating branch addresses.

Diagnostic and utility software, by Robert Harwood, *Personal Computing*, 10/81. Two memory tests for 6800-based machines, unfortunately using operating system calls for I/O. Would need some work to be able to use on the HX-20.

How to program and interface the 6800, by Dennis Doonan, *Interface Age*, 9/81.

Fast Fourier for the M6800 by Richard H. Lord, *Byte*, 2/79. Assembly language program to analyse music and speech. Correction of a bug in the above by Alastair Roxburgh, *Byte*, 5/81, p.458.

DEMONS: a symbolic debugging monitor (6800), A.I. Halsema, *Byte*, 5/81. Includes a source code listing (6800) of a disassembler.

6801: a one-chip system, by H.W. Neff, *Kilobaud Microcomputing*, 1/81. Describes the capabilities of the 6801, with some background on the 68xx family.

Write Your Own Assembler, by Dan Fylstra, *Best of Byte*, Vol. 1. Explains what an assembler does, how to write your own. Good diagrams.

OTHER ASSEMBLERS

As we noted earlier, our assembler is not the greatest. You may want to spend a little money and buy something better.

Epson America has an HX-20 assembler on ROM that they've been making available to their software developers. It may be available to the public by the time you read this.

Warburton Franki has a BASIC program that will assemble Hitachi mnemonics from microcassette tape to object code and store it in the machine language area. Mnemonics may be input and edited under BASIC and saved to tape. An assembly listing and symbol table is available on the HX-20 screen, microprinter or external printer. $20 on microcassette.

WF also has a disassembler. This is a reverse assembler — a program that will take object code and convert it back into source code. It's useful both in figuring out the functions of other codes as well as helping in your own debugging. The disassembler operates on machine code within the HX-20's address space with optional output to either the microprinter or an external printer. A memory dump in either ASCII or hex format can also be obtained.

Kuma Computers markets a 6301 editor/assembler developed by Appollo. It's £14 and looks much like the Labenski assembler we've reprinted in the book. It's written in BASIC, uses BASIC string space to hold the code, has the same editing commands — insert, delete, list. It's a little more sophisticated in its error handling, it accepts decimal and octal numbers as well as hex, it differentiates between 1 and 2 byte literals, it doesn't require the use of tabs for source code entry and it is probably faster. (The latter is said, without personal experience, because we can't imagine anything slower than ours.) It also takes a little less memory than ours, which could be important when working on larger programs. The savings in memory probably come from the use of non-standard addressing modes, which means that a little more work is involved in converting programs written for a more standard assembler.

Cross-assemblers

If you're really serious about assembly language programming, if you intend to develop sophisticated, complex programs, then you could use more powerful development tools than those we've discussed.

The physical limitations of the HX-20 preclude running the kind of programs we're talking about. You need 'cross-software'. The assembler and the other development tools are run on another, larger computer with the resulting object code downloaded into the HX-20.

One company with such a set is Microtec. This company has a macro assembler ($1800), a linking loader (another $300), and an interactive simulator, including simulated I/O ($1250).

Microtec's macro assembler allows nesting, recursive calls, global and local symbols, and conditional assemblies. Multiply, divide, Boolean, relational operators, and parentheses are allowed in expressions. A common section, a unique feature for an assembler, allows separate programs to share variables. Output is relocatable and the linking loader allows several object modules to be joined together.

According to Microtec, the interactive simulator simulates all aspects of the 6301 by implementing, in software, the registers and logic control functions of the 6301. The timer and serial communications interface are also simulated. A command language allows registers to be set, simulated memory to be displayed/altered, breakpoints to be set, trace records to be displayed, interrupts to be initiated, I/O to be done for any port address, etc. Symbolic debugging is also provided, using the assembly language source code labels.

Microtec's software is intended for minicomputer and mainframe use. Any 16 (or more) bit computer with Fortran IV can run the package. In fact, CDC's Cybernet time-sharing system has it up and programmers can use it via dial-up terminals. The assembler/loader manual and the simulator manual are available separately for $15 each.

Warburton Franki, the Epson distributor for Australia, markets a set of cross-software. The 6301 assembler, $150, runs on any CP/M disk system with at least 32K of memory. It produces a symbol cross-reference table, a list file, and an object file which may be downloaded into the HX-20.

A machine language program running on the HX-20 reads the object file coming across the RS-232 port and loads it into memory. This loader is $20, takes just 256 bytes of memory, and can be located anywhere within HX-20 RAM.

Epson America and Hitachi America also have 6301 cross-assemblers. Epson's runs on CP/M systems, Hitachi's runs on the IBM 370 and the Intel MDS.

A company with a cross-assembler for the 6801 is Avocet Systems Inc. This program runs on any CP/M microcomputer. The company also has an EPRON programmer so that you can 'burn' your object code into a PROM.

Companies with cross-assemblers for the 6800 are Transam Microsystems (£145) and Sorcim.

CompuServe, about which more is mentioned in Chapter 8, Communications, offers a 6800 assembler on its time-sharing service. This assembler can produce either a listing or object code. Usual CompuServe rates apply.

STORING MACHINE CODE INTO MEMORY

After we've assembled our program and produced machine language code, we can store it into the HX-20 in five ways:

- downloading over the RS-232 port from another computer;
- POKE'ing it into memory from BASIC;
- using a combination of DATA/READ/VARPTR statements from BASIC;
- using the Monitor;
- using an assembler.

As an example, if you wanted to store an LDAA #$10 at location 2640, you could do:

```
MEMSET 2645
POKE 2640,&H86
POKE 2641,&H10
```

If you wanted to execute, from BASIC, any code that you stored there, you could do:

```
DEFUSRO=2640
X=USRO(0)
```

Remember to end all machine code programs with an RTS ($39) if you want control to return to BASIC. Don't forget, also, to set aside space for your program by using the MEMSET instruction. If you don't, you may write over the information that the BASIC interpreter needs, such as the memory map used to keep track of what is in the five program areas.

For long programs, this will take less memory:

```
Ex:
MEMSET 2660
DEFUSRO=2641
DEFINT A
DIM A(20)
DATA &H86,&H10
FOR I=0 TO 2
READ A(I):POKE 2641, A(I)
NEXT I
X=USRO
```

Here's another way:

```
DEFINT A
DIM A(2)
FOR X=1 TO 2
DATA &H17,&H39
READ A1,A2
B=A1*256+A2
IF B>32767 THEN A(X)=B-65536 ELSE A(X)=B
NEXT X
U=VARPTR(A(1))
DEFUSR=U
X=USR(0)
```

Note No POKEs are used in this last example. But an integer variable in BASIC takes up 2 bytes; therefore we have to combine the 2 bytes to form 1 hex number. Also, BASIC will store a number larger than 32,767 as its complement (i.e., negatively, so we have to fix that too. This type of program must be *relocatable*, as you can't have any instructions that use fixed addresses. (A machine language program to relocate any 6800 program can be found in the 3/78 issue of *Interface Age*, written by Neal Chapman).

Further information on this topic (though written for Z80 computers) can be found in *80-US Magazine*, 5/81 and 9/81.

Once you have the machine code in memory, you can save it to tape via the SAVEM command from BASIC or the WM command from the Monitor. Use LOADM or RM to get it back in again.

USING THE MONITOR

The Monitor provides an easy way to store machine code into memory. The monitor is also a good way to learn the machine and we're going to give you some instructions for using the monitor as a learning tool.

Note It's possible, using the monitor, to get into difficulties. Setting the system stack pointer to zero, for instance, is definitely out. You may have to hit RESET (the little button on the side of the machine) to continue processing. If you really mess up memory, you may have to reinitialize the machine (hitting CTRL/@ on the main menu), so save any programs and data in memory to tape before starting.

The monitor is entered directly off the menu or with the MON command from BASIC. The results are the same either way. The differences are in the contents of the registers upon entry, including the stack pointer, and a change in some memory locations that the BASIC interpreter has set. If you hit the BREAK key, for instance, you'll return to BASIC if you came from BASIC.

The commands to use the Monitor are explained starting on p.399 of Epson America's *BASIC Reference Book.* If you don't have a copy handy, here's a quick review:

Generally, a period (.) and a RETURN will terminate any action.

B = go back, same as hitting the BREAK key

K = sets up a power-up sequence of simulated keystrokes.
Examples: K2 will automatically put you into BASIC whenever you turn the machine on

D = display memory.
Example: D0A40 will display the 15 bytes of memory starting at location $A40. Hit <RETURN> for each succeeding screenful.

G = go to the specified address
Example: G0A40,A4E will go to a routine starting at location $A40 and return when (if) the program counter contains $A4E.

S = store memory
Example: S0A40 will cause the Monitor to prompt for the byte to be entered in that location.

X = examine/change registers
Example: X will bring up each register one at a time. Type <RETURN> to look at the next; value <RETURN> to change the contents. A is the A accumulator, B the B accumulator, X the index register, C the condition code register, S the stack pointer, and P the program counter.

R = read an object file. This is similar to the LOADM command in BASIC.
Example: RM,TEST.OBJ,R will read in a file from microcassette called TEST.OBJ and start running it. A filename suffix (like .OBJ) is always required. The other device names are C for cassette, R for PROM cartridge, 0–7 for the disk. As with the W command, you must be sure to set the address parameters (with the A command) beforehand.

V = verify an object file

W = write an object file. This is similar to the SAVEM command in BASIC.

A = set addresses for reading/writing files. T is the top address, L the last address, O an offset into memory producing a new load address, E the entry point.

The monitor can be used to learn machine language by allowing you to try out each instruction without writing a program. First, go into BASIC and set MEMSIZE equal to at least 2660. Then get into the monitor — from 1 off the main menu or by typing MON from BASIC. If you'd like your registers to look the same as ours, hit MENU and 1 after each test.

While in the monitor, the cursor control keys don't work and you must use CTRL/H or DEL to backspace. Typing in period (.) will also take you back to the general command level.

More information can be found in the Epson *BASIC Reference Book* starting on p.399.

Examples Clear the B accumulator:

```
5F     CLRB
A=00  B=00    X=D310
C=CB  S=04AF  P=D23B
```

Type:

```
SA40 <return> 5F <return> . <return>
```

This puts the CLRB in location $A40.

Then type:

```
X  <return> <return> FF <return> . <return>
```

You now have accumulator B loaded with FF

Now type:

```
GA40,A41 <return>
```

This executes the 1-byte instruction at location A40 and stops the machine at location A41

The registers now read:

B=00 it got cleared
P=0A41 this is the next instruction to be executed.

The monitor doesn't save the contents of the condition code register, but if it did, it would be a C4, D4, E4, or F4

```
1 7 always one
1 6 always one
x 5 (H) not affected
x 4 (I) not affected
0 3 (N) cleared
1 2 (Z) set
0 1 (V) cleared
0 0 (C) cleared
```

The fact that the monitor doesn't save the contents of the CCR before returning to command level makes it less valuable as a debugging tool. But you can check what happens to the condition code register by adding an 07— Transfer Condition Code Register into accumulator A — right after the CLRB, e.g., 5F 07. That is, type:

SA41 <return> 07 <return> . <return>

Then put something back in B (like FF) and then execute the two instructions with: GA40,A42

Some instructions don't lend themselves to this kind of treatment, but many do. Here's a few more to help you get started.

Increment B accumulator
```
5C     INCB
A=00  B=00     X=D310
C=C8  S04AF    P=D23B
```

Type:
SA40 <return> 5C <return> 07 <return> . <return>

This puts an INCB in location $A40 and a TPA in $A41.

Then type:
GA40,A42 <return> to execute it.

The registers now read:
```
B=01  we added 1
A=C0 (or D0 or E0 or F0)
1 7 always one
1 6 always one
x 5 (H) not affected
x 4 (I)  not affected
0 3 (N) cleared
0 2 (Z) cleared
 0 1 (V)cleared
 0 0 (C)not affected
```

How about subtracting 1 from the B accumulator:
Decrement B accumulator

```
5A     DECB
A=00  B=00     X=D310
C=C8  S=04AF  P=D23B
```

Type:
SA40 <return> 5A <return> 07 <return> . <return>

This puts a DECB in location $A40 and a TPA in $A41.

Then type:
GA40,A42 <return> to execute it.

The registers now read:
```
B=FF (we subtracted 1 from 00).
A=C8
1 7 always one
1 6 always one
x 5 (H) not affected
x 4 (I)  not affected
1 3 (N) set; the register went minus
0 2 (Z) cleared
0 1 (V) cleared
0 0 (C) not affected
```

The above examples were instructions that didn't use any operands, i.e., implied mode instructions. Now let's try some instructions that are a little more complex:

Load B accumulator with a 'constant' value of hex 45 ($45).
```
C645    INCB #$45
A=00  B=00    X=D310
C=C8  S=04AF  P=D23B
```

Type:
SA40 <return> C6 <return> 45 <return> 07 <return> . <return>

This puts an LDAB in location $A40, the operand ($45) in $A41, and a TPA in $A42.

Then type:
GA40,A43 <return> to execute it.

The registers now read:
```
B=45 it got loaded
A=C0
1 7 always one
1 6 always one
x 5 (H) not affected
x 4 (I)  not affected
0 3 (N) cleared
0 2 (Z) cleared
0 1 (V) cleared
0 0 (C) not affected
```

How about loading accumulator B with a value in memory? Just arbitrarily, let's use location $F000 which is in ROM and always has a value of $BD.

```
F6F000   LDAB  $F000 (extended mode addressing)
A=00  B=00     X=D310
C=C8  S=04AF  P=D23B
```

Type:
SA40 <return> F6 <return> F0 <return> 00 <return> 07 <return> . <return>

This puts an LDAB in location $A40, the first byte of the memory location in $A41, the second byte of the location in $A42, and a TPA in $A43.

Then type:
GA40,A44 <return> to execute it.

The registers now read:
```
B=BD it got loaded
A=C8
1 7 always one
1 6 always one
x 5 (H) not affected
```

x 4 (I) not affected
0 3 (N) got set ($BD looks like a negative number to the processor)
0 2 (Z) got cleared
0 1 (V) got cleared
0 0 (C) not affected

How about reversing the above action and storing that $BD into a location in memory?

```
F6F000 LDAB $F000
F70A48 STAB $0A48
```

Type:
SA40 <return> F6 <return> F0 <return> 00 <return> F7 <return> 0A <return> 48 <return> 07 <return> . <return>

As before, this puts an LDAB in locations $A40–$A42, then a STAB instruction in locations $A43–$A45. The operand on the STAB, $0A48, tells the machine where to store our $BD.

Then type:
GA40,A47 <return> to execute it.

Now do a D0A40 <return>
You'll see your two-instruction program in memory, along with two $BD you stored in location $0A48.

SOME CODING HINTS

The following is offered as general assembly language programming tips, useful for any machine.

1. Put NOP instructions into your program frequently. This leaves space for 'zapping' memory with your changes, without having to re-assemble. Even large changes can be made with just 2 bytes worth of free space — putting in an instruction to branch to a patch area.
2. No instructions should have labels except those that are being branched to. This shows at a glance how a particular section of code can be entered.
3. Use NOP instructions for branch labels. This lets you insert source code at the beginning of the routine, without having to retype the labelled statement.
4. If you can, use a comment on each line of code. If you use our assembler, you'll run out of space, which is why we didn't do it. Also, without assembler, only instructions that have operands can accept a separate comment. (In most assemblers, anything separated by a space from the operands is taken as a comment.)

Tables

Tables are often of use in programming. There are several ways of organizing data into tables. For instance,

```
$0000
$F621
$3829
$FFFF
```

or

```
'HELLO    '
'GOOD-BYE'
'END      '
'         '
```

In this case, each table entry is of equal length. This makes it easy for subroutines to handle. Just use a loop to inspect each element sequentially. The end of the table can be a marker such as all FFs or blanks or anything else that would not normally come up in the data. Or, you can load the number of table elements into a counter and execute your loop that many times, decrementing the counter each time.

If each element in the table has a variable length, e.g.,

```
'HELLO'
'GOOD-BYE'
'END'
```

you have a problem in that you don't know how long each entry is. Here's three ways to keep track:

```
$05
'HELLO'
$08
'GOOD-BYE'
$03
'END'
```

Each element is preceded by its length.

Or: turn on the high-order bit of the last byte in each element. (This assumes you're working with ASCII text data, rather than binary or graphics data.)

Or:

```
'HELLO'
$00
'GOOD-BYE'
$00
```

With this arrangement, your program would examine each character of the table entry and stop when it hit a zero.

SOME HANDY ASSEMBLY LANGUAGE SUBROUTINES

The BIT instruction will allow you to test a bit to see if it's on or off. Here's how to:

Set a bit flag on, e.g., bit 0 in location $0A48

```
LDAA  $0A48 put flag byte in register
ORAA  $80 turn on the appropriate bit
STAA  $0A48 save it back
```

(If *either* of the two bits under examination is on, turn on the corresponding bit in the accumulator.)

Reverse a flag, e.g., bit 0 in location $0A48

```
LDAA $0A48
EORA $80
STAA  $0A48
```

(If the bit under examination is on, an Exclusive OR will turn it off. If it's off, an EOR will turn it on. Other bits are unaffected.)

Set a bit flag off, e.g., bit 0 in location $0A48

```
LDAA  $0A48
ANDA $7F
STAA  $0A48
```

What we did here was to AND all the bits in the flag byte *except* the one we wanted off. This leaves all those other bits the way they were, but turns the relevant one off. (An AND will turn on the result bit if *both* the corresponding operand bits are on.)

The same thing, in the more general case:

```
LDAA  #$FF
SUBA  #$nn
ANDA $nnnn
STAA  $nnnn
```

This just subtracts the wanted bit from $FF.

The 6301, unlike the 6800, has an 8-bit multiply instruction. But there's still no one instruction to multiply the 16-bit value in the D (A+B) accumulator by an 8-bit number. The following routines will help.

Multiply a 16-bit number by 2

```
ASLD
BCS   to a routine to handle overflows
```

Multiply a 16-bit number by 3

```
STD    $nnnn save original value somewhere
ASLD  multiply by 2
BCS    overflow routine
ADDD $nnnn add value back in
BCS    overflow routine
```

Multiply a 16-bit number by 4

```
ASLD
BCS
ASLD
BCS
```

Multiply a 16-bit number by 5

```
STD     $nnnn
ASLD
BCS
ASLD
BCS
ADDD
BCS
```

Multiply a 16-bit number by 6

```
STD     $nnnn
ASLD
BCS
ASLD
BCS
ADDD
BCS
ADDD
BCS
```

There are no division instructions, either. But to divide by two, use an LSR instruction:

```
   LSRA for 8-bit values in accumulator A
or LSRB for 8-bit values in accumulator B
or LSR $nnnn for an 8-bit value at memory location $nnnn
or LSR $nn,X for an 8-bit value at memory location $nn+X
or LSRD for a 16-bit value in accumulator D
```

To divide by 4, 8, etc., just keep repeating the instruction.

A specific HX-20 coding tip

An HX-20 machine code program will not automatically check for the BREAK key being hit or the power switch being 'thrown' or even a low battery condition occurring. If your program is at a critical point, perhaps when it's doing tape I/O, then you may not want the operator to interrupt it by accident or confusion over what to do next. Talbot Computers, in their Intext program, took this approach. The operator is forced to go back to the main menu to terminate the program — turning power off won't do it. (Remember: the HX-20 has continuous battery power. All the 'power on/off' switch really does is cause an interrupt that results in the setting of a flag in memory.)

If you want to know when battery power is low or when other specific conditions occur, you'll need to test for them.

To test for a battery low condition, do:

```
TIM   $02,#$10 look at bit 4 of I/0 port 1
BE    $FF1F    do the warning message (no
               return)
```

(Note that bits are numbered right to left, i.e., bit 7 is the high-order bit, bit 0 is the low-order bit.)

To test for the power switch having been turned to off:

```
TIM   $28,#$40 look at bit 6 at location hex 28
BE    $FFAC    turn off the power (no return)
```

To test for the BREAK key being hit:

```
JSR   $FF6A    scan the keyboard
LDAA  $14C     get contents of byte $14C
ANDA  #$04     only look at bit 2
BE             if pressed, go. . .
```

To test for the microprinter being turned on:

```
JSR   $FF6A    scan the keyboard
LDAA  $14E     get contents of byte $14E
ANDA  #$80     only look at bit 7
BE             if on, go. . .
```

To test for the MENU key being hit:

```
JRS   $FF6A    scan the keyboard
LDAA  $14C     get contents of byte $14C
ANDA  #$20     only look at bit 5
BE             if pressed, go. . .
```

The reason we didn't do a TIM in these last three examples is that TIM has no extended addressing mode. It can access the first 'page' of memory using direct mode, or it can access the remainder of memory using indexed mode (which we could have used by loading the index register with $14C or $14E).

Additional programming tips may be found in Chapter 4, Using and Writing BASIC Programs.

A GENERAL MOVE PROGRAM

The following program will move a block of data around in memory. The options are:

- move data in the virtual screen to anywhere else in memory;
- move data from elsewhere in memory to the virtual screen

The program is callable from BASIC. To call it:
DEFUSR0=&HA40
X$=USR0(A$)
where A$ is a character string of the form:
FnnnnLnnnnTnnnn
F is the 'from' field and can be V or R
T is the 'to' field and can be V or R
Vnnnn is an offset from the start of the virtual screen
Lnnnn is the length of the data to move
Rnnnn is an address in memory

For instance, to take $80 bytes of data from address $378 and put it onto the beginning of the virtual screen, A$ would equal 'R0378L080V0000'. (For simplicity, this program expects all passed numbers to be in hex.)

```
0  000              * A0000L0000A0000
1  000              * A=Virtual Or
2  000              *   Ram
3  000              * L=Length
4  000              * e.g. V0040L0020RB000
5  000              * would move 20 (hex)
6  000              * bytes from the start
7  000              * of the virtual
8  000              * screen + $40 to
9  000              * address $B000
```

Table 6.1
Keyboard matrix updated by the BASIC interpreter or by calling $FF6A)

Memory Location	$80	$40	$20	$10	$08	$04	$02	$01
$145	7	6	5	4	3	2	1	0
146	/		–	,	;	:	9	8
147	G	F	E	D	C	B	A	@
148	O	N	M	L	K	J	I	H
149	W	V	U	T	S	R	Q	P
14A	←	→		]	[	Z	Y	X
14B	CAPS	GRPH	NUM			TAB	sp	CR
14C			MENU	DEL	PAUSE	BRK	SCR	CLR
14D			PapF	PF5	PF4	PF3	PF2	PF1
14E	PrON	CTRL	SHIFT		SW6-4	SW6-3	SW6-2	SW6-1

```
10  000                   ORG   $0A40
11  A40   EE01            LDX   $01,X
12  A42   A601            LDAA  $01,X
13  A44   E602            LDAB  $02,X
14  A46   BDFF2B          JSR   $FF2B
15  A49   B70ACF          STAA  W1
16  A4C   A603            LDAA  $03,X
17  A4E   E604            LDAB  $04,X
18  A50   BDFF2B          JSR   $FF2B
19  A53   B70AD0          STAA  W2
20  A56   A606            LDAA  $06,X
21  A58   E607            LDAB  $07,X
22  A5A   BDFF2B          JSR   $FF2B
23  A5D   B70AD1          STAA  W3
24  A60   A608            LDAA  $08,X
25  A62   E609            LDAB  $09,X
26  A64   BDFF2B          JSR   $FF2B
27  A67   B70AD2          STAA  W4
28  A6A   A60B            LDAA  $0B,X
29  A6C   E60C            LDAB  $0C,X
30  A6E   BDFF2B          JSR   $FF2B
31  A71   B70AD3          STAA  W5
32  A74   A60D            LDAA  $0D,X
33  A76   E60E            LDAB  $0E,X
34  A78   BDFF2B          JSR   $FF2B
35  A7B   B70AD4          STAA  W6
36  A7E   8656            LDAA  #$56
37  A80   A100            CMPA  $00,X
38  A82   2702            BEQ   FRMV
39  A84   2011            BRA   FRMR
40  A86   01      FRMV    NOP
41  A87   CE0270          LDX   #$0270
42  A8A   B60ACF          LDAA  W1
43  A8D   F60AD0          LDAB  W2
44  A90   E300            ADDD  $00,X
45  A92   FD0ACF          STD   W1
46  A95   01              NOP
47  A96   01              NOP
48  A97   01      FRMR    NOP
49  A98   8656            LDAA  #$56
50  A9A   A100            CMPA  $00,X
51  A9C   2702            BEQ   TOV
52  A9E   200F            BRA   LOOP
53  AA0   01      TOV     NOP
54  AA1   CE0270          LDX   #$0270
55  AA4   B60AD3          LDAA  W5
56  AA7   F60AD4          LDAB  W6
57  AAA   E300            ADDD  $00,X
58  AAC   FD0AD3          STD   W5
59  AAF   01      LOOP    NOP
60  AB0   FE0AD1          LDX   W3
61  AB3   2718            BEQ   END
62  AB5   09              DEX
63  AB6   FF0AD1          STX   W3
64  AB9   FE0ACF          LDX   W1
65  ABC   A600            LDAA  $00,X
66  ABE   08              INX
67  ABF   FF0ACF          STX   W1
68  AC2   FE0AD#          LDX   W5
69  AC5   A700            STAA  $00,X
70  AC7   08              INX
71  AC8   FF0AD3          STX   W5
72  ACB   20E2            BRA   LOOP
73  ACD   01      END     NOP
74  ACE   39              RTS
75  ACF   00      W1      $00
76  AD0   00      W2      $00
77  AD1   00      W3      $00
78  AD2   00      W4      $00
79  AD3   00      W5      $00
80  AD4   00      W6      $00
```

This is about the largest program that can be assembled with our mini-assembler in an unexpanded 16K HX-20. Additional comments in the program would have exceeded the space available, so we decided to put them here separately.

10 Assemble the program to begin at location $A40. (Remember to set MEMSET before loading/using the program.)

11–15 Get the first two digits that we've been past and convert them from ASCII hex into binary, storing the new value in the byte we labelled W1.

16–19 Ditto for the second byte of the 'from' field.

20–23 Ditto for the first byte of the length field.

24–27 Ditto for the second byte of the length field.

28–31 Ditto for the first byte of the 'to' field.

32–35 Ditto for the second byte of the 'to' field.

36–39 If the first field starts with 'V', branch to FRMV, else branch to FRMR.

40–45 Find the start of the virtual screen and add the specified displacement. Store back into W1–W2.

48–52 Figure out if we have to find the start of the virtual screen for the 'to' field.

53–58 If so, compute the new address and store back into W5–W6.

59 At this point, we have real memory addresses in W1–W2 and W5–W6.

60–61 Get the length bytes W3–W4. If they're zero, then we're done.

62–63 Otherwise, subtract 1 from the length and store back into W3–W4.

64–65 Get the next (or first) 'from' byte by putting W1–W2 into the index register and using that as an address to load accumulator A.

66–67 Increment this address and store it back into W1–W2 for the next time.

68–69 Get the next (or first) 'to' byte and store the contents of accumulator A into it.

70–71 Bump the index register and save it back into W5–W6 for the next time.

72 Continue looping until all bytes have been loaded/stored.

74 Return to the caller.

As an exercise, you may want to tighten up the code and save assembler space by creating a loop to process the values passed from BASIC.

7 FORTH

'If you don't have time to do it right, when will you have time to do it over?'

This chapter covers:
An introduction to Forth
HCCS Forth

Sometimes tools — computer and otherwise — can be judged to be:

Easy to learn/easy to use.
This is the ideal state, but usually too simple. Most people will get used to what the product does and then wish it could do more. But adding these additional features often makes the product:

Easy to learn/hard to use.
This is what happens when too many functions are added that are not well integrated into the original design. On the other hand, if you redesign the product from scratch with all possible features in place, then you run the risk of producing something that's:

Hard to learn/easy to use.
The product features are consistent, but there are so many such features that the novice user is overwhelmed. Still, if you plan to use the product a lot, riding the learning curve would often be a good investment of time. However, if you never get the hang of it, then the product is just:

Hard to learn/hard to use.
After you've spent some time with Forth, you'll probably want to put it in one of the latter two categories. Forth is not easy to learn, but it can (we've been told) be very worth while to use. Your reaction to it probably depends more on your personality type than on anything else.

People with neat, orderly minds like to write computer programs in Pascal. They view assembler as painful, and BASIC as a disaster. They won't like Forth. Forth is a hacker's language. Like BASIC (but even more so) it lets you do what you want when you want it. In Forth, there are many fewer rules than in Pascal, where the syntax of the language practically prohibits logic errors.

WHAT IS FORTH?

Forth is a language, as you may have gathered. But in at least one sense, it's more than that — it's an operating environment. It has its own built-in text editor and its own disk/tape handler.

Forth started as a way of doing process control. But since those early beginnings in the 1960s, it has found its way into all sorts of applications on all sorts of computers: word processing, data management, arcade games, etc.

WHY FORTH?

Like BASIC, Forth is an interpreter that can execute your source statements directly. This speeds program development. But Forth is also a compiler. Once a routine has been debugged, it can be stored in executable form. This makes run-time speed much faster than the equivalent BASIC program.

According to Dr Phillip Good, a respected software reviewer, Forth is a good language for one or two programmers trying to get a job done as quickly as possible. But it's not a good language for team projects where several programmers have to be able to read each other's code. Forth is 'ideal for real-time data acquisition', according to Dr Good, because there is a smooth transition between the language and machine code. But, Dr Good notes, while Forth programs are good for accessing the I/0 ports and timing parameters of an individual machine, this makes them less portable, i.e., less transferable, to another machine.

FORTH ON THE HX-20

As we write this, there are two versions of Forth available for the HX-20. One is from Talbot Microsystems which has long been producing well-regarded Forth systems for other (mostly 6800/6809) computers. Some of the new software developed for the HX-20 has been written in Talbot's Forth.

The other Forth which is available is from HCCS in England. HCCS has been kind enough to send us a copy of their Forth and the remainder of this chapter is a discussion of that system. Most implementations of Forth are similar, adhering to the Forth Interest Group's Forth (fig-Forth). So, much of what we describe about HCCS Forth will also apply to any other Forth released for the HX-20.

LEARNING HCCS FORTH

HCCS Forth comes with a 100-page manual. This manual assumes that you know nothing about Forth, which was a valid assumption in this writer's case.

But we wouldn't want to tell you that you can learn Forth just from the manual. Especially this manual, where the examples have not been converted to the HX-20's 20 column screen. The book everyone seems to recommend for beginners is *Starting Forth*, by Leo Brodie (Prentice-Hall), 1981). This is not the best learning book in the world — it's missing an index, for one thing — but it is reasonably well done and easy to read. *Starting Forth* explains a Forth (Forth, 79) that is a little bit different from HCCS Forth (fig-Forth), so you'll need to keep both books in front of you (as well as the machine) while learning. Both books throw you right into Forth arithmetic shortly after the beginning — and if you don't have a mathematical bent you'll just have to suffer through it. Many BASIC books do the same thing, though, even though most people use their computers for other than computations.

Being an interpreter, Forth is easier to learn than compiled languages. You can just type in a command and see what happens. In fact, both books use a tutorial approach in just such a manner. The HCCS Forth manual, for instance, advises you to type in: 54 EMIT <CR> and then watch what happens. (A 'T' is displayed; 54 being the decimal value for T and EMIT being the statement that writes a single character to the screen.)

If you're sharp, you've noticed in the above example that the data preceded the operator, unlike BASIC, i.e., PRINT CHR$(54). Forth has often been called the language for those who think backwards. Forth uses Reverse Polish Notation, the same as some calculators. For instance, two numbers added together would be entered as: 3 5 + <return>. What is actually happening here is that any numeric data you enter goes onto a stack. So, in our example, the 3 would go onto the stack, then the 5. The definition of '+' is: a function that takes the top two values off the stack, combines them and puts the result back on the stack. That is, if you did: 6 13 22 + +. What would happen is that the number 41 would be displayed. (A '.' displays the top value on the stack.)

We referred to definitions in the preceding paragraph. Every word in Forth is a definition. Some definitions are supplied, others are created by the user. As Brodie points out, Forth works in reverse compared to other languages. Instead of having all possible operations pre-defined, Forth provides a set of basic tools that let the user build his own operational set.

The HCCS manual offers a number of examples of this. Here's one:

```
:KTEST BEGIN KEY DUP . 08 = UNTIL;
```

Executing this routine will cause any key the user types to be displayed as its decimal ASCII value. The routine is terminated when the DEL key (an 08) is pressed. Note that the order of the statements is not what you may be used to, but it is simple to interpret given a description of each keyword and the knowledge that interpretation is done in a strictly left-to-right fashion.

Like many Forths, HCCS Forth uses a line editor to enter source code. The HX-20's cursor keys are not deactivated, they just don't function correctly.

Other points of note about HCCS Forth:

- message numbers are used instead of text, probably to save memory;
- it doesn't like having BASIC in the same machine. We found that doing a cold start was necessary to go back and forth between HCCS Forth and BASIC.

HCCS FORTH DEFINITIONS

!	!CSP	#
#>	#S	'
(	(.")	(;CODE)
+LOOP	ABORT	DO
FIND	LINE	LOOP
NUMBER	*	*/
*/MOD	+	+1
+-	+BUF	+LOOP
+ORIGIN	,	-
→	DUP	-FIND
—TRAILING	"	."
.LINE	.R	/
/MOD	0	1
2	3	0<
0=	OBRANCH	1+
2+	:	;
;CODE	;S	<
<#	<BUILDS	=
>	>R	?
?COMP	?CSP	?ERROR
?EXEC	?LOADING	?PAIRS
?STACK	?TERM	@
ABORT	ABS	AGAIN
ALLOT	AND	BACK
BASE	BEGIN	BL
BLANKS	BLK	BRANCH
C1	C,	C@
CFA	CMOVE	COLD
COMPILE	CONSTANT	CONTEXT
COUNT	CR	CREATE
CSP	D+	D+-
D.	D.R	DABS
DECIMAL	DEFINITIONS	DIGIT
DLITERAL	DMINUS	DO
DOES>	DP	DPL
DROP	DUP	ELSE
EMIT	EMPTY-BUFFERS	ENCLOSE
ERASE	ERROR	EXECUTE
EXPECT	FENCE	FILL
FIRST	FLD	FORGET
FORTH	HERE	HEX
HLD	HOLD	I
1D.	IF	IMMEDIATE
IN	INTERPRET	KEY
LATEST	LEAVE	LFA
LIMIT	LIST	LIT
LITERAL	LOAD	LOOP
M*	M/	M/MOD
MAX	MESSAGE	MIN
MINUS	MOD	NEXT
NFA	NUMBER	OFFSET
OR	OUT	OVER
PAD	PFA	QUERY
QUIT	R	R#
R>	RO	REPEAT
ROT	RP!	S->D
SO	SCR	SIGN
SMUDGE	SP!	SP@
SPACE	SPACES	STATE
SWAP	TASK	THEN
TIB	TOGGLE	TRAVERSE
TYPE	U*	U/
UNTIL	USER	VARIABLE
VOC-LINK	VOCABULARY	VLIST
WARNING	WHILE	WIDTH
WORD	X	XOR
[	[COMPILE]	]
2@	2!	(CLK)
TRAM	BEEP	CASS
FILE	XFILE	CONV
COPY	D/L	HALT
DATE	YR	MTH
DAY	DCHAR	DRAW
FEED	HOME	LAST
MASK	MCASS	MON
?TERM	MIC	PICK
CLS	PGET	PLOT
PRINT	PSET	SEEK
TAPCNT	TIME@	TIME
UDP	WIND	

INSTALLATION

Installing HCCS Forth is a snap. . . literally. It comes on a ROM which plugs into the optional ROM slot of the main unit (instead of Ski-Writer, for instance). Alternatively, it may be placed in the expansion unit.

8

COMMUNICATIONS

'Magic is the inability to perceive the relationship between cause and effect' — *Orr*

This chapter covers:

WHY DATA COMMUNICATIONS

Probably anyone who has ever used a computer has at some time or another wanted to transfer data and programs from his computer or terminal to another computer or terminal.

Most large organizations depend on it. Every day, tens of thousands of people sit at terminals sending and receiving information from large computers. With the advent of microelectronics, of course, computers no longer need to be large and terminals no longer need to be 'dumb'. Microcomputers like the HX-20 can, with the right software, now serve as terminals.

As a terminal, the HX-20 can access another computer system and retrieve data or run programs on that system. The benefits of this are basically twofold:

- enabling company staff to do work outside the office, away from the corporate computer;
- buying time from a commercial time-sharing service so as to run programs that you could not run in-house.

As an intelligent terminal, the HX-20 can transfer data back and forth to another computer. Unlike ordinary terminals that are restricted to sending only what can be typed on the keyboard and showing only what can fit on the screen, the HX-20 can send data stored in memory or on tape. Conversely, received data can be saved in memory or on tape.

An intelligent terminal makes data retrieval much more efficient. By storing what comes across the communications link, further computer processing can be done on that data. Stock market quotations, for instance, can go into a database that would eventually show historical price activity for the stock.

Intelligent terminal software also facilitates data collection. Data entered into the HX-20 — from the keyboard, from data acquisition devices, from bar code readers, etc. — can be stored and then transmitted to another computer at the operator's convenience. With the real-time clock in the HX-20, the data can even be transmitted while the computer is unattended.

One of the most widely used features of intelligent terminal software is electronic mail. Messages can be created on the HX-20 and, when complete, transmitted to the 'mailbox' of another electronic mail user.

Programs as well as data can be sent across a communications link. In fact, much of the software commercially available for the HX-20 was developed in this fashion: written on a larger machine with better software development tools and then 'downloaded' into the HX-20.

The following sections go into more detail on communications uses. Some of this material has previously appeared in *In Business* magazine.

Outside services

For most of the past 15 years, remote computing was practically the only way to do data processing. The central computer was locked in a back room somewhere, probably at a service bureau, and you accessed it with a terminal. The advent of microcomputers has moved many of these functions into your office. But there are some functions which are better done on large computer systems: 'number crunching' for instance, or typesetting.

The old way of doing typesetting — and still the most common way — was for the typesetter to take a printed draft of the text and key it into the typesetting machine. When text started being produced on word processors and computers rather than typewriters, it made less sense to have the typesetter re-key it back into another machine. Daily newspapers were the first to go this route. But commercial firms are starting to offer this service to business firms and the general public. Now you can not only write an operations manual on the HX-20 but also add special typesetting codes to the text, transmit it electronically to a typesetting service, and have finished proofs delivered back to you.

Electronic mail

Playing telephone tag is time consuming. And expensive in terms of lost time. It is even more expensive if you play this game long distance. One solution is the postal service. Another is electronic mail.

With electronic mail you key your message into the system and your task is done. The message will be stored in the recipient's mailbox, waiting for him/her to sign onto the system. Of course, you don't get the instant feedback you'd expect from someone on the other end of a phone call, but it's cheaper than long-distance calls and faster than the mails.

Telemail from GTE is a typical electronic mail system. You get unlimited message storage for any period of time, an electronic version of the telephone directory, security/privacy features, mailgram and telex capability — at a cost of 12 cents per message unit. There's a minimum monthly charge.

Similar services are available from at least two other national services — CompuServe and The Source — but without high minimum monthly charges. Each requires dialling into a central computer and following some simple formatting instructions. (Most users regard SourceMail as being much easier to use than CompuServe's EMail.) British users should check into Comet.

Conceivably, an HX-20 with an auto-dial modem can be set up to dial the remote computer automatically and even format the message for you properly, including looking up the recipient's mailbox number.

The drawback to all this, of course, is that the recipient must be a subscriber to the same system you're sending the message on — and he must dial into the system frequently to check his mailbox, though some desktop computers can be programmed to do this automatically without operator intervention. At this time, electronic mail works best among several scattered offices within a single company.

But MCI Communications has a good idea. This new service from the 'long-distance phone company' will take your machine-readable message and transmit it to an MCI subscriber with a terminal *or*, for non-subscribers, transmit it to an MCI regional centre which will print it off and send it

- by regular US mail (for $2); or
- by Purolator Corp for next day delivery ($6); or
- (if local) by same-day delivery ($25).

There is one other major benefit to electronic mail, i.e., the ability to send easily the same message to multiple recipients. This service is available from The Source and the US Postal Service, among others.

With an auto-answer modem and appropriate software, the HX-20 can even be used to collect electronic mail while unattended.

A form of communications that is a sort of half-way electronic mail is the Mailgram service. If your company sends enough mailgrams to justify the $75 minimum monthly charge, you can use your HX-20 as a mailgram terminal. Contact Western Union for more information.

Information retrieval

One of the 'hot buttons' that has propelled the sale of communications gear is the vast store of computerized information available to the user of a desktop computer or terminal.

There are so many of these information sources, called 'databases', that just keeping track of them requires a computer, which the Information Retrieval Research Laboratory at the University of Illinois has. A 1200+ page print version of their database on databases, called *Computer-Readable Data Bases: A Directory and Source Book* is available periodically from Knowledge Industry Publications. Cuadra Associates is another organization that has taken pains to provide a complete directory of databases.

Many of these databases are not for casual users. They require the services of a skilled searcher. Not only must you learn the basics of doing a search on a particular system, but time equals money here. With access charges that range from $40/hour to over $100/hour, any time spent searching the database is money spent regardless of whether or not you find the information you seek. It is not unusual for a novice searcher to run up a bill more than twice as high as that of a professional.

Some information vendors have taken a different approach to meeting users' needs. Sofsearch International, for instance, maintains a database of computer software packages. Rather than have you meander your own way through the database, Sofsearch will take your telephone request and do the search for you.

The new alternatives

Because traditional database access is hard to learn and expensive, a number of cheaper, easier-to-use services have sprouted up. The best overview of these services, and of data communications in general, is probably *Personal Computer Communications* by Alfred Glossbrenner (St Martin's Press, 1983).

Here's a run-down of some of the most popular.

The Source

The award for variety has to go to The Source, which coined the term 'information utility' several years ago to describe a revolution that, unfortunately, it's still waiting for. The Source has over 1000 features and databases — but it has also been described as 'a mile wide and an inch deep'. It reflects the current confusion among database services as to whether their market is business or consumers.

Services available on The Source include:

- stock and commodity market quotations
- a public bulletin board, divided into categories;
- mailgrams;
- CHAT (a one-to-one communication);
- UPI newswire;
- historical stock market information;
- programs for business and entertainment;
- programming languages;
- provisions for mass mailings from users;
- text editor;
- Comp-U-Store, for teleshopping;
- teleconferencing;
- airline schedules;
- restaurant guides;
- movie reviews;
- career assistance.

One nice aspect is that of user publishing. You can put out your own database and collect royalties whenever another user accesses it.

The Source is both command and menu driven, meaning that you can readily access the information you want *or* wander around the service looking for interesting items.

Current costs are $100 to sign up, $20.75 for prime time (before 6 p.m.) use, $7.75/hr for non-prime time.

CompuServe

The CompuServe Information Service was apparently modelled on The Source. Many of the features are similar, though there may be a more programmer-oriented atmosphere. The most popular features on CIS are:

- multi-player games, which won't work too well on the HX-20's small screen;
- a CB simulator, which can also be used for conferencing;
- Special Interest Groups (SIGs) that serve as combination bulletin boards/databases for various subjects;
- stock market quotations.

Other features include:

- electronic banking;
- programming, including cross-assemblers;
- text editor;
- encyclopedia;
- single-player games;
- Comp-U-Store;
- program libraries which can be downloaded. (But most BASIC programs are in a compressed format, which the HX-20 can't read.)

CIS is both menu and page driven. You can access information through a succession of menus or proceed directly to a specific page — if you know what page you want and the number hasn't changed since the last time you accessed it. Typically, only 'entry-level' pages in a particular database keep the same page numbers, making this capability less useful than the command capability of The Source.

Current rates are: $29 to sign up, $22.50/hr prime time, $5/hr non-prime time.

Knowledge Index

Dialog Information Services is probably the largest database vendor in the US. But Dialog is not geared towards the casual user. Rather than change Dialog, a new service was set up, strictly for use during non-business hours. This Knowledge Index contains a subset of the regular Dialog databases, with a lower access charge ($24/hr), but still with a keyboard search capability.

Keyword searching allows you to retrieve the information you want by specifying a word or combination of words, rather than a subject heading. For instance, if you were in the International Software Database, you could specify: FIND INVENTORY to obtain a descriptive abstract of any program that was indexed with the word Inventory. More powerful combinations are allowed, such as: this AND that, this OR that, and wildcard searches where spelling or word endings may not always be the same. (FIND MEDIC? for programs relating to medicine/medical applications.)

Like the traditional database services, if you were looking for information on a particular topic, KI would generally give you an abstract of a magazine article rather than the full text of that article. The option to order the full article is there, though, at $4.50+ per article.

Databases currently on KI — and more are constantly being added — include:

Agricola for agricultural information
Books in Print
ABI/Inform for business information
INSPEC for scientific journals
International Software Database
Microcomputer Index for micro magazine articles
Standard & Poor's News
ERIC for educational materials
Engineering Literature Index
GPO Publications Reference File for an index of government publications
Magazine Index for articles in popular magazines
Medline for medical information
International Pharmaceutical Abstracts
Newsearch for current newspaper and magazine articles
National Newspaper Index for getting background information
NTIS for Dept of Commerce technical reports
Psycinfo for psychological abstracts

Unlike The Source or CompuServe, KI runs on IBM mainframes. IBM mainframes typically look for a 'break' signal — not to be confused with the BREAK key — to stop sending. (ASCII systems usually look for a CTL-C to kill transmission, or a CTL-S to stop it temporarily.) Your HX-20 terminal program should be capable of providing this break signal in order to make the best use of KI or other IBM mainframe-based system.

Current prices are: $35 sign-up charge which includes 2 free hours, $24/hr during non-prime time, i.e., weekends and after 6 p.m. on weekdays.

BRS/After Dark

Bibliographic Retrieval Services, which has over 60 databases online, is also trying to make things easier for non-professional searchers. A new service called BRS/After Dark reportedly provides a more 'user-friendly' — and cheaper — means of doing a search. However, it also applies only to users signing on after 6 p.m.

Current prices are $50 one-time and $6-12/hr non-prime time, depending on database accessed.

Dow Jones news retrieval

DJNS started with stock market quotations and business news reports — which are probably still its bread and butter. But now it includes:

Compu-U-Store;
movie reviews;
encyclopedia.

Both current and historical stock quotes are offered, as well as other financial information. DJNS has the largest subscriber base of any of the services mentioned in this chapter, probably because of its financial information. One future possibility for the HX-20 is a stock market package that will log in, retrieve specified stock quotations, and log off again — all without operator intervention. This capability now exists for other micros and could be done on the HX-20 with the appropriate software.

Rates vary with arrangement.

Newsnet

Then there's Newsnet, a service that limits itself to electronic versions of business newsletters. The industry newsletter that your company subscribes to might be among the ones online at Newsnet — the list includes newsletters like *Construction Claims Monthly, Satellite News, Fiber Optics Patents Newsletter*, etc. The full text of these and about 100 other newsletters in varied fields is provided. Headline scanning and keyword searching are supported — even across newsletters. Rates are set by the individual publisher and can range from $24/hr to over $100/hr, with higher rates for non-subscribers.

Newsnet has also initiated a Newsflash system which returns information to you automatically depending on a user profile you previously set.

Rates are set by the individual newsletter publisher, with a $15/month minimum.

Mead Data Central

LEXIS and NEXIS, which includes what used to be the *New York Times* Information Service, has formerly been available only to those who rented Mead terminals. But Mead has now loosened up to allow access from other hardware. Conceivably, if the HX-20 were made to emulate an IBM 3101 or Televideo 950 terminal, it could receive these services.

Micronet 800

This British service brings Prestel to microcomputer users: electronic mail, news, travel, entertainment, and much more. Of particular note are the computer product comparisons, reviews, and prices: 24 hours/day, 7 days/week. Initial rate is $99.95 including VAT, but there are occasional discounts.

Service summary

We've noted previously that many of the new services provide full text. But with full text come new problems. One is speed. Reading a bibliographic reference at 30 characters a second is no problem. Having to sit through pages and pages of a magazine article at 30 characters a second is another matter. While CompuServe and Dow Jones have even put up entire encyclopedias online, this is probably something less than useful for anyone who doesn't have time to burn.

Many of the new databases are full text, i.e., the entire printed form of the original (if there was an original) is now machine-readable. This contrasts to many of the older databases on services like BRS which provide only a library reference to a printed article.

Computerized bulletin boards

Individuals, clubs, stores and other groups have often wanted to get into networking. And they have — starting hundreds of one-user-at-a-time dial-in computerized bulletin boards. And often they get out again, once they see how much work is involved. But some, particularly those founded by special interest groups, have survived. They've become centres for people with common interests in things like genealogy or amateur radio or CP/M. Some of the commercial services, such as CompuServe, are also home to these special interest groups, called SIGs.

If you have an interest (and time) to spend networking in this manner, contact organizations that have printed lists of CBBSs — ComputerFood, The Other Networks Newsletter and The On-Line Computer Telephone Directory. (Any list we printed here would be out of date by the time the book was printed.)

For British/European dial-up services, try the HX-20 Users' Group or *Personal Computer World* magazine for up-to-date information. At the risk of becoming obsolete, and because there are many fewer of them, we'll list the ones we're aware of in Table 8.1.

THE HOW OF DATA COMMUNICATIONS

OK, you know why you want to communicate. What do you actually need to do it?

Modems

Digital signals, such as those produced by microcomputers, can't be transmitted directly over most telephone lines. A conversion process, called modulation/demodulation, is first required. The device that performs this conversion is called a modem or data set and one is required at each end of the phone line.

There are different types of modems. The most widely seen in the US is the Bell 103 type.

Table 8.1

Name	Country	Number	Times/Days	Type
Distel	UK	01-683-1133	24 hours	supplier
Maplin	UK	0702-552941	24 hours	supplier
Forum-80 London	UK	01-747-3191	T/F/Su 1900–2300	group
Forum-80 Milton	UK	0908-566660	1900–2200	group
Forum-80 Holland	UK	01–313–512533	most eves	group
Forum-80 Hull	UK	0482-859169	–	group
CBBS Northeast	UK	0207-43555	1430–2100	group
CBBS London	UK	01-399-2136	W/F/Su eves	group
Mailbox-80	UK	051-220-9733	eves	group
ACC	UK	0908-44262	?	group
ABC Stockholm	Sweden	010-468-190522	?	?
Univ. Research	Sweden	010-468-23660	?	univ.
Elfa	Sweden	010-468-7300706	?	?
Tree Tradet	Sweden	010-468-190522	?	?
Micom CBBS	Melbourne Australia	762-5088	24 hours	club
Durban BBS	Durban SA	031-66356		
Capetown BBS	Capetown SA	021-457750		
Johannesburg BBS	Johannesburg, SA	011-834-5135		

(AT&T, by and large, has set the standards to which modems made by other manufacturers comply.) The Bell 103 types can talk to other Bell 103s at anywhere up to 300 baud. They can also talk to Bell 212As operating in 103 mode and to Bell 113s. Bell 103 types cost from $70 up. If you haven't realized it by now, we'll make it clear — you don't need to buy the modem marketed by Epson — any 103/113/212 will work.

Bell 113 types are not often seen, because they are either originate only or answer only. We'll dispense with what that means. Just avoid them.

Another modem you'll see around but which should generally be avoided is the Bell 202 type. By and large, this modem can transit in only one direction at a time. Often you'll see these built into data capture terminals where information is always travelling in one direction — up to another computer. Also, the 202 can talk only to another 202. Transmission speed is 0–1200; cost is $200 and up.

The Bell 212A can talk at 0–300 baud (it's compatible with 103s and 113s) and also at 1200 baud. This is the modem of choice if you plan on doing a lot of communications, but be sure the communications program you'll be using can operate that fast. (The one we provide in this book can't, for instance.) Prices for these modems start at $449.

There are other distinctions among modems:

1. Acoustic coupler versus direct connect — Acoustic couplers require that an actual telephone handset be placed into them; direct connect modems plug into the phone line using the standard RJ11 modular jack. Direct connect modems make for more reliable, less noisy communications. But if you plan to use your HX-20 where there are no modular jacks, such as in telephone booths or in offices with multi-line phones, the coupler is the better bet.
2. Auto-dial — A direct connect modem with auto-dial dispenses with the need for having an actual telephone set present. The modem does the dialling by generating tones or pulses directly over the phone wires.
3. Auto-answer — This is a common feature that allows your modem to 'answer the phone', i.e., to accept a call from another computer. Some sort of software on the HX-20 would be required to make good use of this.

All of the above refers to asynchronous modems, because that is the only type the HX-20 can directly support. Synchronous modems will only work with a protocol converter, mentioned below.

Some available modems

Epson manufactures the 300 bps CX-20 acoustic coupler which runs off its own NiCad batteries. As far as power goes, this is a good arrangement. You don't need a source of AC voltage and you don't drain the HX-20 batteries.

Universal Data Systems manufactures a line of data communications equipment that includes two modems that can be classified as portable. The 103LP and the 212LP operate at 300 bps and 1200 bps respectively. (The 212LP is not a true 212-compatible modem, because it operates *only* at 1200.) Both modems are inexpensive, no-frills units that draw power only from the telephone line, i.e., no drain on the HX-20's batteries.

Inmac is marketing a $130 300 bps direct-connect modem, which like the UDS 103LP, requires a separate telephone (to dial) and an RJ-11 jack. (RJ-11 is a modular plug that is standard for single-line phone use.) Inmac's modem runs off an internal 9V battery, claimed to be good for 50 hours of use.

Yet another 9V battery powered, direct-connect modem with an RJ-11 jack and a voice/data switch is the Volksmodem. Triad Distributing has these for $69.95.

Modems are generally built according to national standards. As we noted previously, in the US AT&T has almost always been the one to set those standards. In other countries there are other standards. For UK use, the Sendata coupler at £220, manufactured in Australia, is recommended by Onestop. For internal use, Onestop uses INCA couplers with an error-checking feature, but these cost £750. Check with your dealer for a modem to meet your needs. Keep in mind that practically any asynchronous modem can be connected to the HX-20's RS-232 port.

RS-232C

In order for a computer to talk to a standard modem, it must deliver a data signal and various control signals in a manner that both sides have agreed upon. The voltage definition most often used to interface computers and modems is technically known as RS-232C.

Few things about computers are standardized — most companies are unwilling to give up their independence by agreeing to some common denominator. But there are *some* standards that pre-exist the microcomputer boom. One of these is the connection between a computer and a modem: the RS-232C interface.

RS-232 (the 'C' is usually dropped) has become a semi-standard because people are using it for purposes other than that for which it was defined, such as for computer-to-printer hook-ups. In discussing RS-232, we'll start with the standard and then cover the exceptions. (Some of this material has previously appeared in *Link-Up* magazine.)

What, when, where, and why

The RS-232 standard defines 25 signals, listed in Table 8.2. The average HX-20 user will never see most of these signals, but we've listed them for convenience.

Since the HX-20 uses only asynchronous transmission, you really need no more than the leads that correspond to DB-25 pins #1–7 and 20, which is what the HX-20 has. What happens on these pins when you start your communications is described below. (The numbers in Table 8.3 refer to the usual DB25 connector, as these are the numbers you'll most often see.)

An explanation of what is happening is:

1. DSR (Data Set Ready) going on means the modem is ready to receive input from the computer. (The official Bell term for a modem is a 'data set'.) With an auto-dial modem, it would probably go on when the modem is turned on. With other modems, it might only go on if the modem were attached to the phone line.
2. When you enter a communications command or program, DTR (Data Terminal Ready) goes on. Now each device knows that the other is operative.
3. When you want to transmit over the line (e.g., a log on sequence), the HX-20 sets on RTS (Request To Send), but does not send any data to the modem just yet.
4. If the modem is ready to receive data, it will signal your terminal by setting CTS (Clear To Send).
5. Now your HX-20 can transmit data to the modem and over the phone line, using pin #2.
6. The modem accepts the data on pin #2, converts it into an analogue signal, and sends it out over the phone line.

7–8. Any data sent back over the phone line from the remote computer site will be sent to your HX-20 by the modem over pin 3. (The RS-232 pinout chart looks at signals from the terminal's point of view.)

We've been talking about various pins on an RS-232 connector. But, strictly speaking, the RS-232 standard defines voltage levels for various signals, rather than a particular configuration of pins.

All signals and data on the interface are

Table 8.2
The RS-232 definition

Signal Type (abbrev.)	DB-25/Connector /Pin #	HX-20 DIN /Connector/Pin #
Frame Ground (FG)	1	8
Transmitted Data (TxD)	2	2
Received Data (RxD)	3	3
Request To Send (RTS)	4	4
Clear To Send (CTS)	5	5
Data Set Ready (DSR)	6	6
Signal Ground (SG)	7	1
Data Carrier Detect (DCD or CD)	8	
Positive DC Test Voltage	9	
Negative DC Test Voltage	10	
Equalizer Mode (QM)	11	
Secondary Data Carrier Detect (SDCD)	12	
Secondary Clear To Send (SCTS)	13	
Secondary Transmitted Data (STD) or New Sync (NS)	14	
Transmitter Clock (TC)	15	
Secondary Received Data (SRD) or Divided Clock, Transmitter (DCT)	16	
Receiver Clock (RC)	17	
Divided Clock Receiver (DCR)	18	
Secondary Request To Send (SRTS)	19	
Data Terminal ready (DTR)	20	7
Signal Quality Detect (SQ)	21	
Ring Indicator (RI)	22	
Data Rate Selector	23	
External Transmitter Clock (TC)	24	
Busy	25	

Note: Positive voltage = binary 0, signal space, control on.
Negative voltage = binary 1, signal mrk, control off.

Table 8.3

Sequence	HX-20	Modem
1. Modem turns on		DSR goes on (6)
2. RS-232 port initializes	DTR goes on (20)	
3. HX-20 wants to transmit	turns on RTS (4)	
4. Modem →		turns on CTS (5)
5. HX-20 →	transmits (2)	
6. Modem →		receives (2)
7. Modem →		transmits (3)
8. HX-20 →	receives (3)	

binary. A '1' data bit is called a mark and is the normal condition of a line when there is nothing happening, i.e., when there is no data. This seems backwards, but just takes a little getting used to. This 'mark' condition is set by a negative voltage (−3 volts or less). A '0' data bit, on the other hand, is called a space and refers to a positive voltage (+3 volts or more). Therefore, when a pin on the connector goes from a negative voltage to a positive voltage, from mark to space, that tells your HX-20 that a control or data signal is present. (The first 'space' data bit is called a start bit and tells your HX-20 that more data bits follow.)

HX-20 hardware

The HX-20 contains two serial ports. One is intended for high-speed communications to special peripherals, such as a disk drive or TV monitor adaptor, and to other HX-20s. The other, an RS-232 port, uses the standard interface to connect to modems, printers, plotters, different types of computers — any device with an RS-232 port, including other HX-20s. This chapter concerns itself with the RS-232 port.

The RS-232 port is software-configurable. That is, no physical switches are required to set its options. Any of the options can be set through a BASIC program or direct statement. **Note** Both sides of the connection must be set the same way, i.e., the same baud rate, the same number of stop bits, etc.

These options are:

Baud rate:
110 = 0
150 = 1
300 = 2
600 = 3
1200 = 4
2400 = 5
4800 = 6

Word length:
7 or 8

Parity:
None = N
Even = E
Odd = O

Stop bits:
1 or 2

RS-232 control lines:
Carrier Detect (CD): normal or ignore
Ready To Send (RTS): low or high
Data Set Ready (DSR): normal or ignore
Clear To Send (CTS): normal or ignore

In common usage, baud rate refers to the amount of data that can travel over a line. Technically speaking, bits per second (bps) is a more accurate way of phrasing this measurement. Seven bits of data, plus a start bit, a stop bit, and a parity bit, give you 10 bits for each character to be transmitted. Therefore, 300 bps (or 300 baud) is 30 characters per second (30 cps).

```
   start b0 b1  b2  b3 b4 b5  b6  P stop
1  _____!__!_________!__!_________!_____
0  _____!__!_________!__!_________!
```

Fig. 8.1 Transmitted data

Note The line is normally high or in a 'mark' state, until a low or 'space' is received. The above character is hex 11, 7 data bits, 1 stop bit, even parity. Note, also, that the low-order bit is transmitted first and will be reversed by the receiver.

Word length in this usage refers to the number of data bits that make up a character. The ASCII character set that is universally used among microcomputers uses 7 bits for each character (e.g., 1000001 is hex 41 or the letter A). But you may occasionally want to transmit something other than ASCII — Epson graphics characters, for instance — and then you'll need that high-order eighth bit. (**Note** The parity bit is not counted as part of the word length.)

Parity is an error-checking concept that is used on some transmissions. Here's how it works: Count the number of 'on' bits in the character to be sent. If it's an odd number, turn 'on' the parity bit to get an even number of bits, for even parity. If you want odd parity, set the parity bit on or off to give you an odd number of bits. (Don't count the start and stop bits as part of this.) No parity simply means that this eighth (high-order) bit is always turned off for 7-bit transfer or forms part of the data on 8-bit transfer. Many time-sharing utilities use even parity.

At least 1 stop bit is attached to the end of every character transmitted. Some machines will attach 1.5 or 2 bits to the end. (A half-bit is transmitted by sending a signal for only half the time it takes to send a full bit. A break signal is the reverse: by convention, it's a space signal that is 8 times as long as a single bit.) The HX-20 hardware gives you the choice of either 1 (the usual) or 2 stop bits.

Many computers give you control over the RS-232 signals. Because all incoming signals on the HX-20's RS-232 port are handled by the slave processor rather than by the master processor upon which your programs run, there is no such direct control. The only exception to this is the Request To Send (RTS) signal. You can set this to turn on or off when the RS-232 port is initialized. (The Epson BASIC manual uses the terms high and low, but it really means on and off, respectively. Actually, they're referring to the signals inside the HX-20, rather than the signals at the RS-232 connector.)

You also get the opportunity to specify whether CD, CTS, and DSR are to be ignored. If you are 'talking' to a modem and want to be notified if the line 'goes down' while you are transmitting, specify CD, DSR and CTS as 'normal'. If you're talking to another computer or to a printer, you can safely specify 'ignore' on all three. One exception to this last is when you need to have hand-shaking. More on hand-shaking later.

All of these options are specified in a five-character 'word' that is used as a parameter with BASIC statements. The last character of the word controls the RS-232 lines. To compute it:

Add 8 if CTS ignored
or 0 if wait on CTS

Add 4 if DSR ignored
or 0 if DSR required
Add 2 to turn on RTS (normal)
or 0 to turn off RTS
Add 1 if CD ignored
or 0 if CD required

If you wanted to access CompuServe using a 300 bps modem, for instance, you would select: 300 bps, 7 data bits, even parity, 1 stop bit, CD normal, RTS on, DSR normal, CTS normal — 27E12. What this means is:

- You are transferring data at a rate of 300 bits per second (roughly 30 characters per second).
- There are 7 data bits (enough for the entire ASCII character set) in each data character.
- There is a parity bit that will turn on or off depending on the status of the data bits.
- The data is terminated by 1 'mark' bit.
- The HX-20 program will return an error if you lose your line (CD normal, DSR normal), the RTS signal will go on when the HX-20 desires to transmit, and the HX-20 will only transmit if it sees CTS go on.

A 1200 baud modem — 47E12 — would work equally as well. That's as fast as you can generally go if you're talking to another computer using asynchronous protocol on a dial-up phone line.

If the other computer you want to communicate with is nearby, you can string a direct wire between the two machines. Any modem, computer or terminal manufacturer specifying an RS-232 interface guarantees that signals will be OK up to 50 feet. In actual practice, 200 feet is more likely to be the maximum permissible distance between a terminal and a modem, though special low-capacitance cable can extend this. If the two machines are further apart than that, you can get a device to amplify the data signal to enable it to cross the distance without fading. More than a few hundred feet, but less than 5 miles or so, you can get your phone company to lease you a private line on which you can use a pair of 'short-haul' modems, also called limited distance modems. Any of these options will give you data communications at up to 4800 bps.

The usual — as opposed to standard — cable has a male DB-25 connector on each end. This is a connector with 25 pins that would normally plug into female DB-25 connectors on the modem and on the computer. For space-saving reasons, the HX-20 does not use this plug but instead uses a DIN connector. But most devices that you'll want to plug your HX-20 into will use a DB-25.

Some devices, like an IBM PC, may require a cable with a female connector. (A 'gender changer' can be tacked onto the end of a male-terminated cable to make this hook-up work. But, of course, it pays to look at your connectors before ordering any cables.)

In the course of hooking the HX-20 up to different devices, you'll come across the term 'null modem'. This is not a communications device, but rather a cable that allows two pieces of 'data terminal equipment' (DTE) to communicate with each other.

That is, two of the same kind of communications device could not be hooked directly together. The reason is that pin #2 of the RS-232 interface is assigned for transmissions (to a modem) and pin #3 is assigned for receiving (from a modem). If two computers were hooked together and both computers 'listened' on pin 3, they would never 'hear' each other transmit — because each would be trying to transmit on pin 2. Therefore a special cable that reverses pins 2 and 3 is required.

Often, this cable also reverses 4 and 5 and 6 and 20, or at least ties them to a permanently high signal. The reasoning here is that pin 6, Data Set Ready, is sent by the modem when it is turned on. If there's no modem, then the HX-20 will never know the line is ready. Data Terminal Ready (DTR), pin 20, is sent by the HX-20 when its RS-232 port is initialized, so this pin can be tied to pin 6 to simulate DSR.

The same problem comes up with pin 4, Request To Send, which is raised when the computer wants to transmit. The computer will not initiate the transmission until it sees the Clear To Send (CTS) signal raised by the modem. The easiest way to solve this problem is to connect CTS right to RTS.

So, if there's no modem, we have to reverse or get around these three pairs of signals, and this is what a null modem does.

If you open the cover of the Epson 715 null modem printer cable, you'll see that 4 and 5 are tied together. The HX-20 technical reference manual also explains that 6 and 20 and 2 and 3 will be reversed if you're connected to an MX-80 printer with the 8145 interface.

The way that the printer tells the HX-20 it's busy and that data flow should pause, is to drop its DTR, which the HX-20 interprets as a drop in

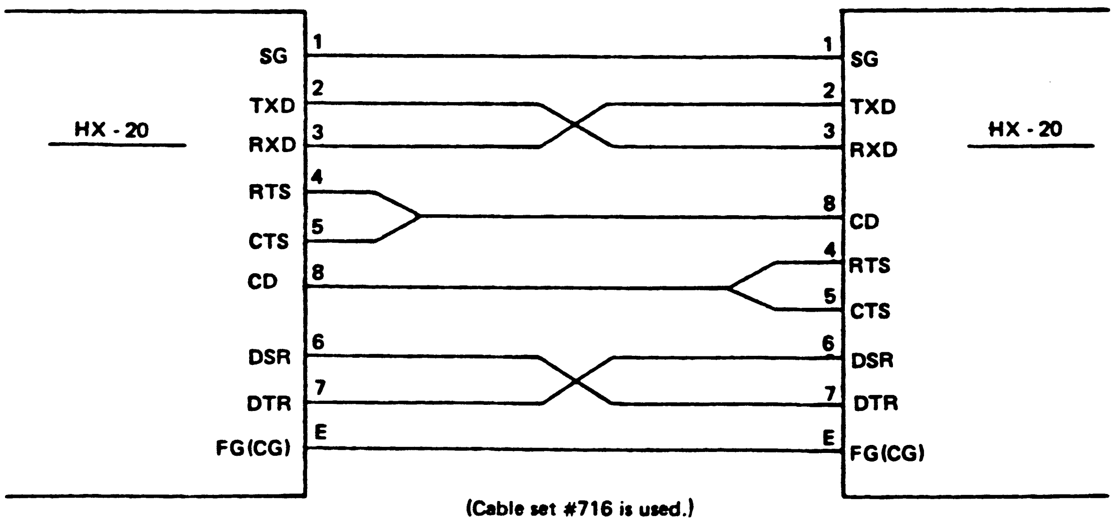


Fig. 8.2 Data transfer between the RS-232 ports of two HX-20s.
Reprinted courtesy of Epson Corp

Sequence	HX-20	printer or other HX-20
HX-20 RS-232 port initializes	DTR goes on (20)	
printer RS-23 port initializes		DTR goes on, switched to DSR (6)
HX-20 wants to transmit	turns on RTS (4)	CTS (5) turns on (wires connected)
HX-20 ———→	transmits (2)	
printer ———→		receives (3) (wires switched)

DSR, i.e., the connection has been broken. This is OK for this particular printer, but other printers use other hand-shaking schemes and you'll need a cable tailored for your configuration. Hand-shaking, by the way, refers to how two devices decide among each other how to interpret the other's signals.

Besides talking to a printer, you'll use a null modem if you want to hook two HX-20s up to each other. Like the 715, Epson cable 716 reverses 2 and 3 and 6 and 20, and connects 4 and 5 to each other.

The HX-20 RS-232 port is asynchronous. Each byte of data transmitted is separated by an indeterminate length of time from every other byte. Only the start and stop bits that precede and terminate the data bits tell the HX-20 (or the remote computer) what is data and what is not. As we noted earlier, the data signal is always in a high, logical one state, called a mark. A start bit is low, logical zero, a space. When the line stops 'marking time' and a space is received, the HX-20 knows that the next 7 bits (or 8, depending on how that option was set) are data.

Perhaps the computer you want to talk to uses synchronous communications: BSC 2780 protocol, for instance. Is it possible with the HX-20? As configured, no. But you can get a protocol converter to make the necessary translation. This is a $2000 device, available from sources like the Black Bow catalogue, that will take your HX-20 out of the portable arena. Then you'll need a synchronous modem. But we mention it to point out that virtually anything can be hooked up to anything else, given that you want to spend the money to do it.

Another conversion you might be interested in is serial to parallel. If you already own a printer with a parallel port, you can buy a converter for it so that it can accept serial, RS-232 input. Many of the printer buffering devices (spoolers) on the market offer this as an optional feature.

Software

For transfer of BASIC programs, HX-20 extended BASIC will most likely suffice.

For data transfer, the program later this chapter will meet most of your needs. A simpler terminal program is contained on p. 390 of the Epson *BASIC Reference Book.*

Data communications from HX-20 basic

As we noted in Chapter 3, HX-20 BASIC, the HX-20 has some built-in capabilities not usually found in microcomputers. Owing, perhaps, to its portable nature, the HX-20 has been designed with communications in mind.

HX-20 BASIC extends the usual function of several ordinary statements and functions to access the RS-232 port. These are:

LIST
SAVE
LOAD
PRINT#/PRINT# USING
OPEN/CLOSE/EOF
INPUT#/LINE INPUT#
INPUT$
LOF
MERGE
RUN

You'll use some from the command level; others are best executed from programs.

The value of these statements is that simple RS-232 operations can be performed without the necessity for writing communications software. For instance, you might use a LIST 'COMO:-(48N2F)' to list a BASIC program on an external printer.

A communications program

What follows is an 'intelligent' terminal program. Programs of this type get that nickname by virtue of doing something with data they receive, other than just displaying it. This program performs the following functions:

- Take whatever is on the virtual screen and send it out the RS-232 port or save it on cassette.
- Read a cassette file into the virtual screen area and/or send it out the RS-232 port at the same time.
- Save anything coming in from the RS-232 port into the virtual screen area and directly into a cassette file.
- Display normally unprintable control characters as an aid to trouble-shooting problems. (We add (hex) $80 to the ASCII value of these characters so they'll display as graphics characters.)

Some of the things this program doesn't handle and which you may consider adding:

- Show how much of the virtual screen memory area is left for data.
- Translate tables to convert automatically characters received or transmitted, though you can do this manually with the screen editor.
- Display status of RS-232 leads: CD, CTS, DSR.
- More sophisticated error processing.
- Keep track of how long you're 'logged on' and display this time upon request.
- Send a BREAK signal (needed by IBM mainframes).
- Distinguish between manual and automatic CTL/S (X-OFF).
- With auto-dial modems, send the phone number to the modem.

If there's a major defect with this program, it's that it's slow. It's slow for three reasons:

- It's written in Epson BASIC.
- It's written so that users can understand what it does, without any attempt to tighten the code for efficiency reasons.
- It has multiple functions, all of which would reduce the speed compared with a single-purpose program.

If you modify the program to improve it in some respect, why not share it with everyone? Send us a copy and, if the change looks good, we'll try and disseminate the information to others. We may also do further work on this program ourselves, perhaps even convert it into machine language for speed. Write to us at the address in the Preface and we'll let you know if we have an improved version available.

We may be biased, but we think the program is fairly easy to understand. If you take the time to look at it and at the following comments, we think you should come away with a better understanding of what communication programs can do and how they do it.

Initial housekeeping

100 clear the screen
110 set aside enough string space for normal operation

120	set the virtual screen up as 252 lines of 20 columns each; this can be changed to fit your requirements
130	tell BASIC that numeric variables are integer-type unless specifically defined otherwise
140	set up the RS-232 port and do other housekeeping
150	main loop

Keyboard processing

1000	check the keyboard for input; if we have some, go to 1040 to see what it is
1010-1030	see if any special features are turned on, if so, process accordingly
1035	process any RS-232 input, then exit back to the main loop
1040-1050	check for two '@'s (a control function)
1100	allow vertical scrolling
1105	send the character out of the RS-232 port
1110	a full-duplex system will return the character we entered, so no need for us to display it
1115	we need a linefeed after a carriage return
1120	if they need a linefeed, send one
1190	back to 150 to start again

Read RS-232 port

2000	if nothing on the port, terminate any RC operation and just return
2010	if XON/XOFF is legal and we had stopped the remote computer by sending an XOFF (CTL/S and there are less than 30 characters in the RS-232 input buffer, then tell them to start transmitting again (by sending a CTL/Q)
2020	if more than 80 characters build up in the RS-232 buffer, send the remote site an XOFF.
2030	get a character from the RS-232 input buffer
2060	don't display unprintable characters (unless in test model; if it's an XON, remove the hold on output
2070	display the received character
2080	if the received character is the prompt character we've been waiting for (on VR and CR operations), then stop waiting
2090	return to caller (line 1035 or 6000)

CR — cassette to RS-232

3000	if we're waiting for a prompt character, don't do anything
3005	if we're not allowed to send (because we received an XOFF) then just return
3010	if at the end of the cassette file, then close it and return
3020	read an input record from the cassette
3025	if we got a null record, just return
3030	write the record out the RS-232 port
3040	display it on the screen
3050	if they want a linefeed after a carriage return, give it to them; set to wait (if remote computer wants to prompt)
3190	return to caller (line 1030)

VR — virtual screen to RS-232

4000	if we're waiting for a prompt character, don't do anything
4005	if stopped (by an XOFF), just return
4100	send the next character in the virtual screen area out the RS-232 port
4120	if they want a linefeed after a carriage return, give it to them; wait for a prompt from remote computer if that option specified
4130	bump pointer into virtual screen
4140	if the virtual screen pointer is up to where the cursor is, then we're finished
4190	return to caller (line 1020)

VC — virtual screen to cassette

5000	take the next character in the virtual screen area and write it into the cassette output buffer. (HX-20 OS will automatically write to cassette when buffer fills or on a CLOSE.)
5010	bump virtual screen pointer to next character
5020	if the virtual screen pointer is up to where the cursor is, then we're done
5190	return to caller (line 1027)

RC — from RS-232 to cassette

6000	get a character from the RS-232 input buffer
6010	put the received character in the cassette output buffer
6190	return to caller (line 1028)

CV — cassette to virtual screen

7000	if at the end of the cassette file, then we're finished
7010	get the neat cassette record
7020	display it
7190	return to caller (line 1010)

PF keys to change data directions

8000-8010	look at the two characters after the @@
8020-8060	turn on the flag for the selected action; turn off flags for any conflicting actions
8070	see if user wants help information printed
8080	see if reset is to be done
8085	see if test mode wanted (prints out all characters)
8087	see if XON (CTL/Q) is to be sent. (Can't type it from the keyboard because it is the same code as a screen scroll.)
8090	turn off the flag signifying PF key input
8095	return to caller (line 1050)

Initialization for VR and VC — Find the start of the virtual screen, and the cursor location

8110	Memory locations $270-271 contain the pointer to the first character in the virtual screen
8120	Memory locations $274-275 contain the pointer to the start of the physical screen; locations $278-279 have the cursor position on the screen. Added together, we have

the memory location of the cursor — which will be the point at which virtual screen transfer operations stop. (After we wrote this routine, we realized we could have done the same thing in BASIC.)
8130 Ask if we need a prompt character
8190 return to caller (line 8020 or 8400)

Initialize CR — open the microcassette for input
8200 Tell the user we're going to read the cassette
8210 Ask for a tape location.
8220 9999 means kill the operation
8230 Wind down to the requested location
8240 If we didn't actually get there, then we have an error so turn off the CR flag and go to the return point
8250 Ask for a file name
8270 If we're doing a Cassette-to-RS232 operation, then ask if we need a prompt character
8290 return to caller (line 8030 or 8510)

Initialize RC — open the microcassette for output
8300 Tell the user we're going to write to the cassette
8310 Ask for a tape position or 9999 to kill the operation
8320 If we got a 9999, then turn off flags and go to return point
8330 Wind down to where the file will be put
8340 If we have a tape error, turn off directional indicators and go to return point
8350 Ask for a file name
8360 Open the file for output
8390 Return to the caller (line 8040 or 8410)

Initialize VC — find the start of the virtual screen and open the microcassette for output
8400 Get the screen beginning and end
8410 Open the microcassette for output
8490 Return to the caller (line 8050)

Initialize CV — open the microcassette for input to the screen
8500 Clear the screen
8510 Open the microcassette for input
8590 Return to the caller (line 8060)

?? — Print the PF key assignments on request
8600- 8685 On the microprinter, print each PF key and its function
8690 Return to the caller (line 8070)

Process a user requested Reset
8700 Close the cassette file
8710 Turn off all directional indicators
8720- 8730 Throw away the last 3 characters from the PF key ("SET")
8790 Return to caller (line 8080)

Prompt character check
8800- 8820 Ask if a prompt character is necessary when uploading files. Note: the prompt character handling has not been tested.
8890 Return to caller (line 8130 or 8270)

Initialization
9000 Set the phone numbers at two for now
9010 Build an array for as many phone numbers as we need
9020- 9030 Put your commonly used phone numbers and other info here
9100- 9110 Need to know if half or full duplex
9120 If half duplex, set flag. If full duplex, then assume XON/XOFF flow-control
9200- 9210 Need to know if the other system wants a separate line feed after a carriage return
9220 If so, set flag
9240- 9260 Print each array element
9300 Ask for RS-232 parameters
9310- 9320 Open the port for input and output based on parameters entered
9510- 9585 Assign functions to the PF keys. Key 10 is left for your use. Note that a CTL/Q is assigned to a PF key because the actual key combination of CTL & Q is reserved for scrolling the screen
9590 Make the cursor visible
9600 Set a place to go when have errors
9990 Return to caller (line 140)

Error handling
10000 Display a box when there is an error. If you see a lot of boxes, you probably have the parity or the number of data bits set wrong.
10010 Continue processing with the next instruction after the error

```
10     'COMM20 COMMUNICATIONS PROGRAM
20     'AUTHOR: ERIC BALKAN
40     'IF PROGRAM HANGS AT ANY TIME,
50     'HIT BREAK AND RE-RUN OR GOTO 8700
100    CLS
110    CLEAR 1000
120    WIDTH 20,252
130    DEFINT A-Z
140    GOSUB 9000 'INITIALIZATION
150    GOSUB 1000:GOTO 150
1000   KB$=INKEY$:IF KB$<>" " THEN 1040
1010   IF CV THEN GOSUB 7000:GOTO 1190
1020   IF VR THEN GOSUB 4000:GOTO 1190
1027   IF VC THEN GOSUB 5000:GOTO 1190
1028   IF RC THEN GOSUB 6000:GOTO 1190
1030   IF CR THEN GOSUB 3000:GOTO 1190
1035   GOSUB 2000:GOTO 1190
1040   IF KB$="@" THEN AT=AT+1 ELSE IF AT=1
       THEN GOSUB
1100:  GOTO 100 ELSE GOTO 1100
1050   IF AT=2 THEN GOSUB 8000:GOTO 1190
       ELSE 1000
1100   IF ASC(KB$)= 16 OR ASC(KB$)=17 OR ASC)
       (KB$)=31 OR ASC(KB$)=30 THEN PRINT KB$;
       :GOTO 1190
```

```
1105 PRINT#2,KB$;
1110 IF HALFDUPLEX THEN PRINT KB$;:LPRINT
     KB$;
1115 IF HALFDUPLEX AND ASC(KB$)=13 THEN
     PRINT CHR$(10);:LPRINT CHR$(10);
1120 IF LFREQUIRED AND KB$=CHR$(13) THEN
     PRINT #2,CHR$(10)
1190 RETURN
1999 'READ RS-232 PORT
2000 IF EOF(1) THEN RC=0:GOTO 2090
2010 IF FLOWCTRL AND STPPED AND LOF(1) < 30
     THEN STPPED=0:PRINT #2,CHR$(17);
     'CTL/Q
2020 IF NOT STPPED THEN IF FLOWCTRL AND
     LOF(1) > 80 THEN STPPED=-1:PRINT
     #2,CHR$(19); 'CTL/S
2030 C$=INPUT$(1,#1)
2060 IF ASC(C$)<32 THEN IF TEST THEN C$=CHR$
     (ASC(C$)+&H80) ELSE IF NOT RC THEN IF
     ASC(C$)=19 THEN NOINPUT=-1: GOTO
     2030 ELSE IF ASC (C$)=17 THEN NOINPUT=
     0:GOTO 2090 ELSE IF ASC(C$)= 10 OR
     ASC(C$)= 13 THEN 2070 ELSE 2080
2070 PRINT C$;:LPRINT C$;
2080 IF C$=PCHAR$ THEN WAIT=0
2090 RETURN
2999 ' FROM CASSETTE TO RS232
3000 IF PROMPT AND WAIT THEN 3190
3005 IF NOINPUT THEN 3190
3010 IF EOF(3) THEN CR=0:CLOSE #3:GOTO 3190
3020 INPUT #3,C$
3025 IF LEN(C$)=0 THEN 3190
3030 PRINT #2,C$
3040 PRINT C$
3050 IF ASC(C$)=13 THEN IF LFREQUIRED THEN
     PRINT #2,CHR$(10);ELSE IF PROMPT THEN
     WAIT=-1
3190 RETURN
3990 'FROM VIRTUAL SCREEN TO RS232
4000 IF PROMPT AND WAIT THEN 4190
4005 IF NOINPUT THEN 4190
4100 PRINT #2,CHR$(PEEK(VSPTR));
4120 IF PEEK (VSPTR)=13 THEN IF LFREQUIRED
     THEN PRINT #2,CHR$(10); ELSE IF PROMPT
     THEN WAIT =-1
4130 VSPTR=VSPTR+1
4140 IF VSPTR=CSR THEN PRINT "VR DONE":
     VR=0:WAIT=0
4190 RETURN
4990 'FROM VIRTUAL SCREEN TO CASSETTE
5000 PRINT #3,CHR$(PEEK(VSPTR);
5010 VSPTR=VSPTR+1
5020 IF VSPTR=CSR THEN VC=0:CLOSE 3:PRINT
     "VC DONE"
5190 RETURN
5999 'FROM RS232 TO CASSETTE
6000 GOSUB 2000
6010 PRINT #3,C$;
6190 RETURN
6999 'FROM CASSETTE TO VIRTUAL SCREEN
7000 IF EOF(3) THEN CV=0:PRINT "CV DONE":
     CLOSE #3:GOTO 7190
7010 INPUT #3,C$
7020 PRINT C$;
7190 RETURN
7990 ' DIRECTION CHANGERS
8000 DC$=INKEY$
8010 DC$=DC$+INKEY$
8020 IF DC$="VR" THEN VR=-1:CV=0:CR=0:
     VC=0:GOSUB 8100
8030 IF DC$="CR" THEN CR=-1:VC=0:RC=0:
     GOSUB 8200
8040 IF DC$="RC" THEN RC=-1:CR=0:CV=0:
     GOSUB 8300
8050 IF DC$="VC" THEN VC=-1:CR=0:CV=0:
     GOSUB 8400
8060 IF DC$="CV" THEN CV=-1:RC=0:VC=0:
     GOSUB 8500
8070 IF DC$="??" THEN GOSUB 8600
8080 IF DC$="RE" THEN GOSUB 8700
8085 IF DC$="TS" THEN IF TEST THEN TEST=0
     ELSE TEST = -1
8087 IF DC$="QQ" THEN PRINT #2,CHR$(17);
8090 AT=0
8095 RETURN
8100 'FIND VSCREEN START AND CURSOR LOC
8110 VSPTR=PEEK(&H270)*256+PEEK(&H271)
8120 CSR=PEEK(&H274)*256+PEEK(&H275) +
     PEEK(&H278)+PEEK(&H279)*20
8130 GOSUB 8800
8190 RETURN
8200 PRINT "Reading From Cassette"
8210 INPUT "Wind# or 9999?",W
8220 IF W=9999 THEN CR=0:CV=0:GOTO 8290
8230 WIND W
8240 IF TAPCNT <> W THEN PRINT "Tape Error":
     CR=0:CV=0:GOTO 8290
8250 INPUT "File name?", INFILE$
8260 OPEN "I",#3,INFILE$
8270 IF CR THEN GOSUB 8800
8290 RETURN
8300 PRINT "Writing to Cassette"
8310 INPUT "Wind# or 9999?",W
8320 IF W=9999 THEN RC=0:VC=0:GOTO 8390
8330 WIND W
8340 IF TAPCNT<> W THEN PRINT "Tape Error":RC
     =0:VC=0:GOTO 8390
8350 INPUT "New file name?",OUTFILE$
8360 OPEN "O",#3,OUTFILE$
8390 RETURN
9399 '
8400 GOSUB 8100
8410 GOSUB 8300
8490 RETURN
8499 '
8500 CLS
8510 GOSUB 8200
8590 RETURN
8599 '
8600 LPRINT "Direction List"
8610 LPRINT "PF1 = VIRTUAL SCREEN TO
     CASSETTE"
```

```
8620  LPRINT "PF2 = RS-232 TO CASSETTE"
8630  LPRINT "PF3 = CASSETTE TO  VIRTUAL
      SCREEN"
8640  LPRINT "PF4 = CASSETTE TO RS-232"
8650  LPRINT "PF5 = VIRTUAL SCREEN TO  RS-
      232"
8660  LPRINT "PF6 = PRINT THIS LIST"
8670  LPRINT "PF7 = RESET"
8680  LPRINT "PF8 = SET TEST ON/OFF"
8685  LPRINT "PF9 = CTL/Q"
8690  RETURN
8699  '
8700  CLOSE #3
8710  VR=0:CR=0:VC=0:RC=0:CV=0
8720  FOR A=1 TO 3
8730  A$=INKEYS:NEXT A
8790  RETURN
8799  '
8800  PRINT "Prompt = (N=none)?";
8810  PCHAR$=INKEY$:IF PCHAR$=" " THEN
      8810
8820  IF PCHAR$="N" THEN PROMPT=0
8890  RETURN
8899  '
9000  NUMPH=2
9010  DIM PH$(NUMPH)
9020  PH$(1)="452-8930 CIS 27E12"
9030  PH$(2)="585-7095 NEWSNET 27E12"
9040  'Add more phone numbers here
9050  ' and change line 9000
9100  PRINT "Duplex (H/F)?"
9110  A$=INKEY$:IF A$=" " THEN 9110
9120  IF A$="H" OR A$="h" THEN HALFDUPLEX
      =-1:FLOWCTRL=0 ELSE FLOWCTRL=-1
9200  PRINT "Receiver need    linefeeds (Y/N0"
9210  A$=INKEY$:IF A$=" " THEN 9210
9220  IF A$="Y" THEN LFREQUIRED=1
9240  FOR I=1 TO NUMPH
9250  PRINT PH$(I)
9260  NEXT
9300  INPUT "Enter RS-232 parameters";COM$
9310  OPEN "I",#1,"COMO:"+"("+COM$+")"
9320  OPEN "0",#2,"COMO:"+"("+COM$+")"
9325  WIDTH "COMO:",255
9510  KEY 1,"@@VC"
9520  KEY 2,"@@RC"
9530  KEY 3,"@@CV"
9540  KEY 4,"@@CR"
9550  KEY 5,"@@VR"
9560  KEY 6,"@@??"
9570  KEY 7,"@@RE"
9580  KEY 8,"@@TS"
9585  KEY 9,"@@QQ"
9590  PRINT CHR$(22)
9600  ON ERROR GOTO 10000
9990  RETURN
10000 PRINT CHR$(139);
10010 RESUME NEXT
```

Available software packages

The HX-20 owner has several alternatives among communications programs on the market. Talbot Computers, Transam Microsystems, Epson (UK) Ltd, Kuma Computers, and Warburton Franki are some of the companies that have programs out.

Talbot Computers provides three packages that can be used for a combination of communications and editing: Intext, Dialtext, and Comtext. Intext is primarily a word processing system with communications capabilities and is described in Chapter 9, Word Processing. Dialtext is a complete hardware/software package while Comtext is software only.

Dialtext

Dialtext is a complete system for the remote printing of text prepared on the HX-20. The sending system is composed of an HX-20 with microcassette drive, an acoustic coupler, and Dialtest editing/sending software (£756). The receiving system is another HX-20, an Epson FX-80 printer, an auto-answer modem, Dialtext software, interface box, and cables (£1972).

Once text is entered and edited on the first HX-20, the operator dials into the second HX-20 which is set to answer the phone automatically. This second HX-20 takes the message and stores it in memory. The message can then be printed out on the printer, with a log of all incoming messages written to the microprinter. Error checking is provided. Transmission speed can be 300 or 1200 bps.

The receiving system can alternatively be configured so that messages are saved to a microcassette drive instead of to a printer. Actually, the receiving device need not be an HX-20 and Talbot will assist in customizing its software to run on other systems.

Comtext

Comtext is a communications program with text editing capabilities. It provides for both file transfer to a remote system or interactive communication with that system. Any communications mode setting can be selected. Text can be displayed as it is being sent or received or the display can be suppressed for maximum speed. Editing and communications are done via machine language for speed. Price is £80.

ITE+

The Intelligent Terminal Emulation and Full Text Editor (£50) ROM from Transam allows connections to other computers, including the emulation of standard video terminals with 24 lines by 80 columns. File transfer capabilities are also provided. (A downloaded file is saved as a RAM file, which can then be stored on microcassette or processed with a BASIC program.) Full screen editing with word wrap is possible, and any text can be prepared/edited before or after transmission/reception. Parallel printer routines are available so that printing can occur while data is being received over the modem (with use of Transam's parallel interface unit).

The included editor can also be used in standalone fashion, and the RS-232 routines can also be used to drive a printer as well as talk to another computer. Full control over RS-232 parameters is available, as well as XON/XOFF flow control. For debugging purposes, the usually non-printable ASCII codes can be displayed. An unusual additional feature will allow you to print your document sideways on the internal printer, giving you an 80-column printout.

The program is written in machine code and programmed into a ROM which can be installed either in the spare internal ROM slot of the HX-20 or in the expansion unit.

Epson (UK) Communications ROM

According to Transam, an Epson dealer, this ROM (£30) allows the HX-20 to send and receive ASCII text in blocks according to ISO Standard ISO/R1745-1975, an ACK/NAK type protocol. The program, which is internally fitted in the HX-20, can be called from BASIC programs. (Note to US readers: this protocol is not used in the US and would only prove useful in transferring data between two similarly equipped HX-20s.)

Desk Master 4

This machine language program from Kuma Computers (£29.50) allows communications between an HX-20 and another computer, via the RS-232 interface at 100–300 bps. All RS-232 parameters are set as a result of questions asked during initialization.

Data is viewed on the LCD one screen-full at a time. (To get the next 'page', the user hits RETURN.) The manual states that incoming data will not be lost while the display is stopped, because the I/O routines will continue to save the data in the buffer.

The printer can be used while online, but only when the remote site has stopped transmitting. (Otherwise data will be lost.)

The user is optionally notified if CTS or DSR is dropped, which is a nice feature. Parity errors and carrier detect errors will also be displayed.

This program sets aside quite a lot of memory for the RS-232 buffer — MEMSET must be changed to 14336 before running — so other programs in the machine may have to be deleted before DM 4 will run. The manual provides the storage location in which the buffer size is maintained, for those who might want to change it, but leaves it to the reader to take it from there.

Dumbterm

Another communications program is DUMBTERM from Warburton Franki. This is a machine language program 'POKED' from BASIC to convert the HX-20 to a dumb terminal mainly for use with a 300 bps modem. Keyboard data is sent via the RS-232 port and incoming RS-232 data is displayed on the virtual screen which has been set at 64 lines of 20 characters. Full support of Backspace, Arrows, and CR/LF is provided, allowing multiple physical screens to be sent to the HX-20 and perused at will using the HX-20 arrows. RS-232 communications settings and virtual screen size may be altered by the user ($20).

Trouble-shooting tips

When a BASIC program has a bug in it, it usually evidences itself as incorrect output or a function error or something else easily noticeable. Not so with communications. When devices aren't talking to each other correctly, you may not see anything at all, or you may see garbage being output. Communications problems require their own set of trouble-shooting techniques.

Here's a simple checklist to run down before calling for help.

Is the cable inserted? Particularly on the HX-20, the RS-232 cable may look like it's inserted but may not be in all the way. Is the other end in tight?

Using the right cable? The black HX-20 cable is used for Data Communications Equipment (DCE) like modems, the grey cable for printers, most other computers, and anything else that looks like Data Terminal Equipment (DTE).

All components powered on? This seems obvious, but it happens to everyone and you'll be embarrassed if you call for help and this is what the problem is.

Is the printer online?

All components working on the same parameters? If one side is 7 data bits, 1 stop bit, 300 bps, even parity — the other side had better be the same. The HX-20 is unusually particular about parity errors, signalling an I/O error when it gets data with the wrong parity.

Are you getting input, but the number of characters being sent does not match the number of characters being received? You probably have a speed matching problem.

If you get the right number of characters, but none is recognizable, and particularly if you are getting graphics characters when you're not supposed to, check that the right number of data bits is set.

If the right number of characters is being received, but you can only recognize some of them, then check that parity is the same.

If you are losing characters occasionally, try communicating at a lower speed, e.g., 300 instead of 1200, or 100 instead of 300. Or try using flow control to slow down the faster end of the communications link.

If nothing at all is getting across, you may have a hand-shaking problem. Check that CTS, RTS, DSR, DTR, RxD, and TxD are wired correctly. (You may need an RS-232 tester to verify this.)

Getting or sending extra line feeds (or no line feeds at all)? TRS-80 computers and some others interpret a CR (carriage return) as both a CR and a LF. If this is the problem, modify the program at either end. If one end is a printer with internal switches like the Epson MX-80, check that the appropriate internal switch is set correctly.

Finding that a carriage return is being inserted every 80 characters into the data you're uploading? Check that you've set the WIDTH 'COMO:' parameter correctly.

Finding a bad character at the beginning of a file after uploading from the HX-20? This happens due to the hardware design of the HX-20 and there is no easy circumvention. One solution: change your processing on the receiver to throw away all characters until a particular character is found — and put this character at the beginning of your HX-20 file.

If your problems don't seem to respond to easy solutions, you'll need additional equipment that will show you the status of the signals on the line. This equipment runs from $40 LED displays that show the 7 most used RS-232 signals on up to $15,000+ data monitors. The device you'll most frequently see is called a breakout box. Selling for $150–$200, it shows the status of the RS-232 signals and also allows you to connect or disconnect any signals via switches and jumper wires. It's a standard piece of equipment for anyone seriously involved with communications trouble-shooting and makes diagnosis of hand-shaking problems a simple matter.

A new device that has just started making its appearance is the 'smart' connector. This is a connector, from IQ Technologies and Mikrocomputertechnik, that looks at what signals are on the line and automatically does any necessary signal switching. It can be left in the line permanently — though this is an expensive solution — or it can be removed for use elsewhere once the signal configuration is known.

9

WORD PROCESSING

This chapter covers:
Word processing on the HX-20
Ffosswriter
SkiWriter
Intext
Other WP/Editing programs

At first glance, word processing might seem to be the HX-20's weakest point *vis-à-vis* competitive portable computers. After all, screen size is quite important for entering/editing/reviewing text, and the HX-20 has a screen smaller than many of its newer competitors.

But there are two good reasons for using an HX-20 for word processing:

1. The availability of Ffosswriter.
2. The fact that you already own an HX-20 (which you may have bought for another purpose).

In the US the HX-20 comes with SkiWriter, a very nice mini word processor. It's provided in the spare EPROM slot inside the computer, which is a strange place for it, as SkiWriter is best used for occasional word processing use and this 'wastes' the slot. Our guess is that the decision to put it here was strictly a marketing one, as a defence against the Radio Shack Model 100's built-in word processing.

But the Model 100, as delivered, is better suited to word processing than the HX-20, as delivered. My own attempt to use SkiWriter to do part of this book while on vacation sent me scurrying back to pen and paper. The basic failings are: the screen is too small to see what you've just written, which I find important when creating text; and document file handling is clumsy and slow. Documents can't be named either.

There's really nothing much that can be done about the screen. Horizontal scrolling just doesn't make it. Vertical scrolling is a necessary part of any word processing program, but you really don't want to have to scroll continually manually just to look back at the beginning of the sentence you're writing. (The sentence you just read would take nearly three full screens to display on the HX-20. Perhaps that'll teach me to write shorter sentences.)

But many of the other limitations of the HX-20 (and Skiwriter) can be overcome via software.

Our original intention in this chapter was to discuss what readers should look for in a word processing program. But a product from England, Ffosswriter, seems to have the features we want — so we can achieve our purpose by describing a real product rather than some ideal one. Back to SkiWriter and other text editing programs later.

FFOSSWRITER

Caution We have not seen this program in action. But it comes from a highly respected and very technically sophisticated software development company — and the recipient of an award from the HX-20 Users' Group. We would guess, though not guarantee, that this program does what the developers say it does. (By the time you read this, it may do even more.)

Ffosswriter uses a proprietary random access cassette system (RAX) to provide a range of features previously only found on disk-based units, according to the authors. (Ffoss's RAX is used in several programs sold by Epson (UK) Ltd.) These features include the ability to edit only a part of a document, and renaming/deleting/creating documents without the user having to think about what the tape is doing at all.

Ffosswriter features can be broken down into document handling, document layout, printing, transmission, and editing.

Document Handling Features

- Twenty character volume names including space characters, punctuation, etc.
- Check tape volume, name, etc.
- Edit a document or part of a document.
- Twelve character document names.
- List the directory of documents.
- Rename a document.
- Delete a document.
- Create a new document from parts of old documents.
- Add parts of a document to any other document.
- Edit parts of a document before moving them to a new document, without altering the original.

(Frankly, I wish I could do all this with my Scripsit equipped TRS-80 Model III.)

Ffosswriter automatically divides a document into sections; so a user need only edit part of a document without having to read the whole thing from tape.

Document layout commands

- For the internal microprinter:
  - new page
  - new paragraph
  - new lines
  - word wrap when formatted print is done
  - draft (all commands shown)
- For external printers:
  - page width
  - page length
  - margin: top, bottom, left, right
  - indent from left margin (12 possible offsets)
  - indent from right margin (12 possible offsets)
  - tab definitions (12 possible columns)
  - paragraph indent (from margin or indent position)
  - centre line (line or paragraph end)
  - push line right (line or paragraph end)
  - unjustified (as laid out, usually used for tables)
  - ragged right-hand side
  - justified right and left side
  - single- or double-spaced lines
  - force a new page
  - force a new paragraph
  - force a skip of any number of lines
  - page numbers on or off
  - page starting number (0–99)
  - bold start and end
  - compressed start and end
  - expanded start and end
  - underline start and end
  - super/subscript character
  - discretionary hyphen
  - non-break hyphen
  - non-break space
  - system page layout defaults
  - user definable page layout defaults

Print Facilities

- Print draft.
- Print formatted.
- Print unjustified (as laid out on screen).
- Print marked block.
- Print multiple copies.
- Override page layout commands at print time.
- User definable printer characteristics.

Transmit Facilities

- Local or remote via the RS-232 port
  - draft
  - formatted
  - stripped (text only).
- User definable RS-232 characteristics.

Edit facilities

- Insert characters.
- Delete characters, in insert: left, in overwrite: right.
- Cursor up, down, left, right.
- Move cursor to start or end of buffer.
- Command verification on input (not edit).
- Whole command delete on input (not edit).
- Select variable screen width.
- Save current edits and resume or exit edit session.

The function keys are dedicated to word processor functions with an overlay provided.

Additional word processing functions

- Automatic word wrap.
- Word/sentence/paragraph cursoring.
- Delete (to end from cursor).
- Extend block.
- Additional text block functions
  - block delete
  - block extend
    - by moving start or end
    - by word/sentence/paragraph extend
  - add (or move) blocks to the 'Scratchpad' for editing, move, copy.
- Scratchpad
  - full edit features
  - sub block move or copy back to main
  - clear.
- Search and replace (or insert)
  - define one or the other or both
  - global search and replace (or insert)
  - confirmed (manual) search and replace (or insert)
  - single case
  - constant replace string (no search required).

Future enhancements will include display adaptor software to allow Ffosswriter to be used with a separate larger screen for desktop operation.

One Ffosswriter drawback we noticed from Ffoss documentation: if the CHARGE BATTERY warning message comes on while you are using the program, the document tape may become corrupted. Hitting RESET or BREAK also causes unpredictable results. Ffoss suggests re-running Ffosswriter to the point when the interruption occurred as the best (only?) means of recovering your text.

Up to 18 documents can be held on each side of an MC30 microcassette, with the maximum size of a single document limited to 15,000 characters. (Total storage on each side of an MC30 cassette is 33,000 characters.) Each document may have up to thirty 500-character segments.

Ffosswriter supports the Epson MX80/RX80/FX80 and the Qume Sprint 9 as standard, but allows you to configure permanently the program for two other printers of your choice plus a remote computer link. XON/XOFF and ETX/ACK handshaking is supported, as well as no protocol (other than RS-232 control lines).

Ffosswriter is supplied with a 50-page manual, a plasticized reference card, and an overlay for the function keys. It comes on microcassette with the RAX EPROM for the internal slot. (This EPROM is compatible with the Ffoss, mailing list, card index, diary, and RAXUTILS programs.) Alternatively, users with the expansion unit can order Ffosswriter on EPROMs. Price is £95 (plus VAT for UK residents).

SKIWRITER

Considering that SkiWriter comes free with USA machines, it's a very nice bonus. It's better than most mini word processors. It has all the basic word processing functions and is very easy to use.

Features include:

- Auto word wrap.
- Hard carriage returns.
- Tab (set permanently at 5).
- Vertical scrolling (not horizontal).
- Top/bottom.
- Insert/delete character.
- Delete block.
- Delete document.
- Insert key toggles (like Wordstar); stays in insert mode till you take it out.
- Copy block (*not* move block)
  - must go back and delete block symbols, but this method allows multiple copies to be made.
- End of page marker.
- Search (case sensitive).
- Set print spacing.
- Set right and left margins.
- Prints to microprinter or external printer.
- Save/load using microcassette. (To use an external cassette, a short BASIC program must first be run.) The load function works as an insert, adding text to wherever the cursor is rather than replacing what may already be in memory.

The search feature (called FIND) will look for a character string of up to 20 characters, including carriage returns, format characters, the page character, etc. The NEXT command allows you to repeat a previous FIND without having to re-enter the character string. Each time a string is found, the previous three lines are displayed, along with the line containing the string. You can then update the text manually. (More sophisticated word processing programs provide a combined search and replace feature.)

SkiWriter is supplied on ROM, for the spare internal EPROM slot. Like anything you put in here, this has the drawback of using space that could otherwise be used for 8K of optional RAM in the expansion unit. Since RAM in the HX-20 overrides ROM at the same address, SkiWriter will not work if you have a fully expanded 32K machine. You must first deactivate RAM locations $6000–$8000 by manually setting a switch inside the expansion unit.

Having SkiWriter always resident in the machine is an advantage to those with heavy word processing requirements. Yet, SkiWriter seems designed for people who do only occasional note-taking. As an occasional note-taker, it exhibits a major design flaw: in an unexpanded system, it leaves only 1201 bytes of RAM for anything else.

On a cold start, the HX-20 comes up with 2624 set as the lower limit for BASIC. Via the MEMSET command, this can be changed to provide space for machine language programs and data areas. When run, SkiWriter automatically resets this value to 14,320. This gives you enough room to hold an 11,696 byte document ($A40–$37F0), but puts a crimp into anything else you want to do. If you have BASIC programs in memory that total more than 1201 bytes, SkiWriter won't initialize. We haven't tried it, but assume the same for RAM files.

If you change MEMSET yourself so that you can work in BASIC after using SkiWriter, you'll find that any document left in memory will be overwritten. It's also possible to lose the MENU memory map when going back and forth from SkiWriter to 1000+ byte BASIC programs — a situation that requires a cold start to fix.

SkiWriter can talk to the RS-232 port for transmission to an external printer or even another computer. A program in BASIC is supplied to change the RS-232 port parameters. Control codes can also be sent to a printer, but only before starting a print or after finishing a print — printer codes cannot be embedded in a document.

SkiWriter saves all documents on tape with a file name of DOCUMENT.WP1. These are saved at the fixed tape locations of 50, 1335, 2410, 3500, and 4600. (The last two are only for C60 cassettes.) The user is responsible for keeping track of what file is where and what it holds. The SkiWriter manual suggests that BASIC programs saved in ASCII form with the DOCUMENT.WP1 filename can be read in, but we have not got this to work. There may be a fix available by the time you read this.

SkiWriter is a middle ground between sophisticated word processors and simple text editors. It performs that function nicely enough, within the limitations previously mentioned.

SkiWriter comes with a clearly written instruction manual and an overlay for the function keys. There is a function key for each of the SkiWriter functions, including a context-sensitive HELP key for telling you what's wrong with what you just tried to do.

INTEXT

Talbot Computers Ltd sent us a microcassette copy of their text editor, Intext (£50).

Intext allows the user to create a file of up to 5500 characters long (21,600 on the 32K HX-20). Text can be inserted and deleted either by overwriting or by using the INS/DEL key. Text can also be deleted 20 characters at a time using a special PF key. The cursor is fully controlled by the standard cursor keys. In addition it can be rapidly moved to the top or bottom of the text file using the HOME/CLEAR key. The screen can also be scrolled up or down four lines at a time using the SCRN key. The GRPH and CAPS LOCK keys function as they normally do.

Any text or graphic string can be searched for from the cursor position — and the search can be repeated throughout the text without retyping the string. Frequently used words or short phrases can be programmed into three PF keys for insertion into the text via a single keystroke. The writer can return to the Intext main menu at any time using the MENU key — the cursor will remain at the last position.

Text is stored in memory and can be printed at any time on the integral microprinter or on any external serial printer. Alternatively, it may be output via the RS-232 interface to any other computer or terminal or saved on microcassette. All of the usual communications parameters such as parity, transmission speed, etc., can be set. In addition, Intext supports hand-shaking via DTR/RTS, ETX/ACK, and XON/XOFF. An 'echo back' function is also provided for error checking.

Intext provides considerable control over a variety of different external printer functions. The user simply enters a PF5 followed by a standard mnemonic right in the text file to indicate underlining, condensed printing, form feeds, italics, superscripts, subscripts, expanded printing, boldface, entering tabbing, etc. These mnemonics are set up for Epson printers, but can be changed by the dealer to work with other printers. The user can also directly insert any control characters, including escape sequences, right in the text.

Print parameters that can be set by the user are: printer line width, page length, bottom margin, line spacing, and tab column positions. Line centring, left margin positioning, and commands to remote typesetting machines are supported via control characters that can be placed in the text.

One thing that differentiates Intext from SkiWriter, as an example, is that Intext is not a 'see what you get' type editor. There is no word-wrap or paragraph creation on the screen, only on printouts. Personally, we prefer to have the screen look like the printout, but there is an advantage to the way Intext does it — rather than having to print out on the HX-20, the text can be transferred to another computer system in raw form. It also saves memory space not to have blank areas in the text display.

Intext is written primarily in machine language for speed, with a BASIC start-up initialization routine. Like SkiWriter, it is not intended to function simultaneously with any other programs in memory — you'll have to save onto tape anything else in the system before starting Intext.

A Mailmerge feature has recently been announced as an add-on for Intext. The price is £30.

ITE+

ITE+, from Transam Microsystems, has been described in Chapter 8, Communications. While originally designed for communications use, it can be used as a stand-alone text editor. At least one feature, the ability to print an 80 column document on the microprinter sideways, is unique.

DESK MASTER 2

Desk Master 2 by Eclectic Systems is available from Kuma Computers. This is a line-oriented text editor, written in BASIC. By line oriented we mean that each line of the display is processed separately from any other line. For instance, there is no automatic word-wrap; you must hit return at the end of each line — words that don't completely fit must be placed on the following line. If you've ever used a screen editor, like SkiWriter for instance, you're going to hate to go back to a line editor.

DM2 uses the PF keys, INS/DEL, TAB, NUM, and the cursor keys to assist in text entry, scrolling through the document, and modification. Documents can be saved on tape or printed on either the microprinter or on an external printer. (The program comes set up for the Okidata Microline-80, but 'it may be possible for expert users to make the necessary modifications' to the program for other printers not compatible with the Oki.) Margins, page length, and spacing are supported.

DM2 takes up 7K for the program, leaving you about 5K for your document.

OTHERS

There are many BASIC programs even simpler than DM2 on the market for memo-taking. Generally, these are also line oriented and clumsy to use. This writer feels that, for most of them, even if they're free, they're overpriced. After all, if you want a line editor you can just use BASIC. Put a line number and a REM sign (') in front of each paragraph and *voilà*, you have a text editor. You won't have search and replace on character strings, but you will have:

- insert/delete;
- scrolling;
- editing by paragraph number;
- search for paragraph number;
- copy paragraph capability (by overtyping the line number with a different line number);
- save and load from microcassette, external cassette or other computer;
- printing drafts on the microprinter or external printer.

And the editor doesn't have to be loaded from tape, it doesn't take up RAM space, and your document won't get in the way of BASIC programs or RAM files stored in the machine.

Among the other programs on the market are: Make-Text (King Sofware), APWriter (AP Systems), WP20 (Warburton Franki), and Correspondent 20 (Epson UK). Check Chapter 11, Software and Systems, for further details.

For information on Comtext, see Chapter 8, Communications.

10

INVENTORY/ STOCK TRACKING

'If it's not in the computer, it doesn't exist'

This chapter covers:
What to look for in an inventory system
Available inventory software
Bar code readers/software
Technical information on bar codes
Bar code printing program

Inventory programs are a common computer application and one particularly suited to portable computers. Basically, these programs fill three purposes:

- continuously maintaining information on stock items;
- providing re-order alerts;
- counting items, i.e., 'taking inventory', for reconciliation against computed totals.

The HX-20 can be used in inventory control in two ways:

- as a stand-alone computer;
- as a data entry device, with the data transmitted to a central computer.

Inventory programs must fulfil different functions in different types of businesses. A manufacturer, for instance, must keep track of 'raw goods' as well as the same item at different stages of the production cycle. A more simple inventory system, and more commonly seen, tracks only 'finished goods'.

Stock tracking programs can also be broken down into two categories based on the type of processing they do: invoicing or order entry. An inventory program based on invoicing is the simplest — when an invoice comes in, the order is filled and that quantity of the item subtracted from the inventory count. More complex systems, running on desktop computers, will keep orders and billing separate, handle open orders, etc. Your business requirements should be the determining factor in selecting the type of system to use.

An invoice-type of inventory system should probably maintain the following information: (we found *Evaluating Small Business Software* (Business Computing Press, 1979) by Greg B. Scott helpful in preparing this list.)

Master file
- item number
- item description
- product category (may be embedded in the item number)
- price
- cost
- quantity on hand
- quantity on order
- re-order level
- re-order quantity
- quantity sold month-to-date and year-to-date
- cost of sales month-to-date and year-to-date

Optional transaction file:
- item number
- date
- quantity
- price
- purchase order number

Optional invoice file
- invoice number
- date
- customer number
- shipping address

- purchase order number
- terms
- price with discounts
- sales tax
- freight
- total charge
- comments

Optional back-order file
- item number
- customer number
- invoice date
- purchase order number
- quantity back-ordered
- unit price
- discount percentage

Note that the transaction file is optional. The inventory system can be organized in two ways: online or batch. In an online system, quantity changes are made directly into the database. In a batch system, all changes are grouped together and run against the database at one time. The advantage of an online system is that you always know the exact state of your stock. The advantage of a batch system is that the operator is not slowed down by the database update process.

Different businesses need to handle inventory in different ways. For instance, if you have been doing your inventory accounting on a LIFO basis, you don't want a program that prices inventory on an average cost basis. The link to your other accounting programs should also be considered. You'll probably want the invoices generated by the inventory program to appear in your customers' statements. To do this, you'll need to look into what kind of input your Accounts Receivable program is expecting.

Any inventory system must provide the capability of permanently changing any database item information. You must be able to add and delete items. You must be able to modify prices, quantities, item descriptions or any other field, if for no other reason than to fix something that had been entered incorrectly. Changing entries may seem a natural function, but there have been inventory programs written where changes could not be made — an item was required to be deleted and then re-added *in toto.*

The following is a general checklist of features for an inventory program, or almost any data management program for that matter. It's recommended that you use this just as a starting point for making up your own checklist to suit your own needs. (Some of this material has previously appeared in *In Business* magazine.)

CHECKLIST

System design
- easy to use?
- integrated into other business packages?

Data storage capacity and record format
- maximum file size (larger than memory?)
- maximum number of records per file
- maximum record length
- maximum number of fields per record
- maximum number of characters per field
- variable length fields allowed?
- variable length records allowed?

Field format
- alphanumeric
- numeric
- date
- special items

Speed
- time required to start up the system
- time to bring it down
- average access time per record
- maximum access time for any record
- time to add a new record
- sorting time

Indexing/sorting
- by any field?
- maximum number of simultaneous indexes
- ascending/descending order
- numeric order
- sort multiple fields?
- sort within a selected range?

Selection criteria
- number of keywords
- equal
- not equal
- greater than
- less than
- instring
- AND/OR
- wild card masking

Data entry
- ease of adding/changing data
- easy entry of common data?
- global edit?
- data validation?
- keyboard only?

Printed report generation
- report heading (multi-line)
- column heading (multi-line)
- automatic page numbering?
- justification (left, right, decimal)
- variable column widths?
- variable column separation?
- column totals?
- subtotals?
- maths operations allowed?

Flexibility
- can file design be changed after the data is entered?

Miscellaneous
- programmer's interface?
- sample data tape?
- documentation
- audit trail?
- cost

Let's go over the checklist items. In evaluating a package, the first thing to look at is the system design. Is the system designed to be used by an experienced staff member or can it be used by an employee who's less knowledgeable? How easy is the system to use? Is all the usage information available online, or will the operator have to look in a book for inventory part numbers, instructions, etc? Particularly on a machine with limited memory, we can't have all the functions we'd like to have, so which ones did the designers leave out?

Any software package contains design compromises. Keep in mind that it's more than likely you and the designer will have a difference of opinion over what is important. So you *must* know as completely as possible what it is you need from the package before making up your own checklist and grading your prospective purchase.

A word to the wise: don't assume anything. Just because most packages seem to give you a particular function, don't assume they *all* will — make sure you see that function in operation before you make your purchase. Keep in mind that the designer is not in the same business you are in and therefore is not only likely not to know the needs of your business, but is also capable of making remarkable oversights.

Every package you're thinking of buying should be tested under conditions in which it will actually be used. If you plan to keep a 1000-item (called records) inventory file, with 15 pieces of information (called fields) on each item, don't let the dealer demonstrate the package with just a 30-record file. This may mean doing the preliminary functional check at the dealer's, but then visiting a real user to see it in action. It'll be worth it though.

Integration — the ability of a program to interact with other programs — is very useful. If all of your business data is stored the same way, it becomes easier to transfer data from one application to another, e.g., from your inventory manager to your spreadsheet, to your word processor, etc.

The second category on our checklist, after system design, is capacity and format. How much total information can we store — the file size? How many records (items) in our file? How much total data on each item can we have — the record length? How many separate pieces of data (fields) can we have for each record? How much data in each field?

Do you need a system with fields and records that are not all the same length? Because this is harder to program, many developers don't do it, but it can really save space. Otherwise, each record/field must be as long as the longest possible record/field.

Regarding the format of the fields, any kind of data can be stored in a computer. But if you've stored numeric data in a field without telling the program that it's numeric, don't expect the program to let you do computations on that field later.

How about speed? This is critical on a machine like the HX-20 with a relatively slow processor and a slow means of data storage — tape. Do you need a system that is online and can be queried interactively or a system designed to be updated overnight producing a daily report?

If looking at an online system, how long does it take the system to be brought up for the day? How long does it take to bring down if you want to use the computer for something else? How long does it take to access the average record? An inventory system typically runs on the 80/20 rule, i.e., 20% of your stock accounts for 80% of your activity — so you may care about high speed for part of the database only.

If the time varies with each record, what's the longest you have to wait for any record? How long does it take to add a new record? This will be fast for a system that runs entirely in the HX-20's memory, but could be quite slow for a system that stores current data on tape. If the file is not automatically kept in order, how long does the program take to sort the file into the order you need it in?

Indexing/sorting: How is information actually retrieved from the database? Can you ask for items by part number only, or in order by price?

Selection criteria: Can you ask for a list of inventory items that are below their re-order points *and* require more than one month to re-stock? Complex search arguments are not beyond the HX-20's capabilities, though most software developers may not have provided this function.

Data entry: Is data entered one item at a time, or can the operator move about on the screen adding/changing information? Is the form of data entry fixed, i.e., 'hard-coded' in the program, or can you vary it to suit your needs? If repetitious data is included in many records, is there some easy way to enter it just one time? Can a change be made for multiple records at one time, or must each relevant record be changed individually? Can data be validated as it is entered, so no bogus information (like 31/4/84) is inadvertently entered? Can data be entered from another file (an older database, for instance), or only from the keyboard?

Report generation: Can you get out the reports looking like you'd really want them to look? Can you specify your own report and column headings? Can you specify the format of each printed item? Can reports be produced that draw on information from different sources? Can fields be totalled, subtotalled, multiplied by other fields, etc?

Flexibility: After you've completely planned your database design, entered all your data, and started using the system — inevitably you'll want to go back and change something. Will you be able to? Can the definition and order of the fields (called a schema) be changed without having to re-enter the data?

Miscellaneous: Can programs be written to access the database in ways that the designers hadn't intended, but which would be helpful to your business? Does the system come with sample data to help you get started? Can you pick up the documentation and start experimenting with the system, or is the manual too complex to understand without help?

If a critical part of the system fails, how easy will it be to recover? Will you be able to operate your business manually in the interim? Does the system provide an audit trail, a permanent record of all activity on the file? How does the cost relate to other packages and, more importantly, to the benefits you'll get out of it?

Desk Master 11

DM11, the Mobile Stock Recorder by Country Software, is available from Kuma Computers. It allows the tracking of 100 stock items on the 16K machine or 425 items on the 32K machine. Item records can be added, deleted or changed. Other functions include Find, List, Save/Load to tape.

The program asks, for each item, whether you want to compute VAT. US users will want to remove that line from the program. (The program is in BASIC, so this shouldn't be hard to do.)

A more serious caveat is that this program may just be too simple for someone with a serious need for inventory control. Typically, an efficient inventory program lets the operator type in part number and quantity. This program, on the other hand, appears to give you the items in the order that they're in the machine, which is just not as useful. We haven't seen this program run, but we'd be afraid that, for a few items, it would be more productive to track the inventory with a pencil and paper. And for many items, this program may be clumsy compared to programs for other machines on the market or other (and future) inventory programs for the HX-20.

Some other stock control programs

Measurement Control and Displays Ltd (£50), Healey Management Systems Ltd (£75), MST Consultants (£30), Gemini Marketing Ltd (£20), Computronics ($29.95) all publish stock control programs of varying functions. Typically, they can handle somewhat under 1000 items on a 32K machine.

Check Chapter 11, Software and Systems, for similar programs, such as general database programs that could be used for inventory purposes.

BAR CODE READERS

Bar codes have been used for years as a method of counting items. Years ago, this writer worked for a company in the business of counting stock for other companies. Employees worked in pairs, one calling out the count for an item to the second who would write it down. In a later development, employees were given portable tape recorders and so a single employee could both count and record the count. Later came hand-held terminals with keypads and then the same terminals with bar code readers. Having a terminal with a keypad makes part number and quantity entry easy — having a terminal with a bar code reader makes entry of that same part number automatic. Only the quantity must be entered by hand.

Besides counting stock at reconciliation time, bar code readers can continuously maintain item information as well. Some supermarkets in the US have converted to a bar code system at checkout counters. With each grocery product having a UPC imprint, the checkout clerk needs only to run the product over the bar code reader in order for the supermarket to have an automatic update of quantity on hand, quantity sold, etc. It also assures the supermarket that the item has been sold at the correct price. A further bonus is that items don't have to be individually marked with prices (though some consumer groups object to this) as the store's central computer is the only place where prices need be kept.

Different types of codes have been developed over the years to meet the needs of different industries. Consumers are most familiar with the UPC code. Interleaved 2 of 5 code, a strictly numeric code, is used by heavy industry and warehouses. Code 39, or 3 of 9 code, is an alphanumeric code used by the US Department of Defense to tag its property and also by many private companies. Codabar code is often used in libraries and in the health field. Unlike Interleaved 2 of 5 or 3 of 9, Codabar can be scanned in either direction. Check the November 1983 issue of *80 Micro* magazine for several good articles on bar codes. One of these, by Davey S. Thronton, has been reprinted at the end of the chapter. It includes a more technical discussion of bar codes and a BASIC program to print bar codes on an Epson MX-80 printer.

AVAILABLE BAR CODE READERS AND SOFTWARE

Interface Solutions has a complete package with bar code wand and software on microcassette. The software provides for bar code decoding as well as printing bar codes on an Epson FX-80 printer. Bar codes supported are:

Code 3 of 9
UPC
Codabar
Interleaved 2 of 5
Plessey

Another package comes from Transam Microsystems. The bar code wand is £82, the software alone is £40. The software consists of a set of machine code routines to decode bar codes, callable from your application program written in either BASIC or assembler. The package supports many popular codes:

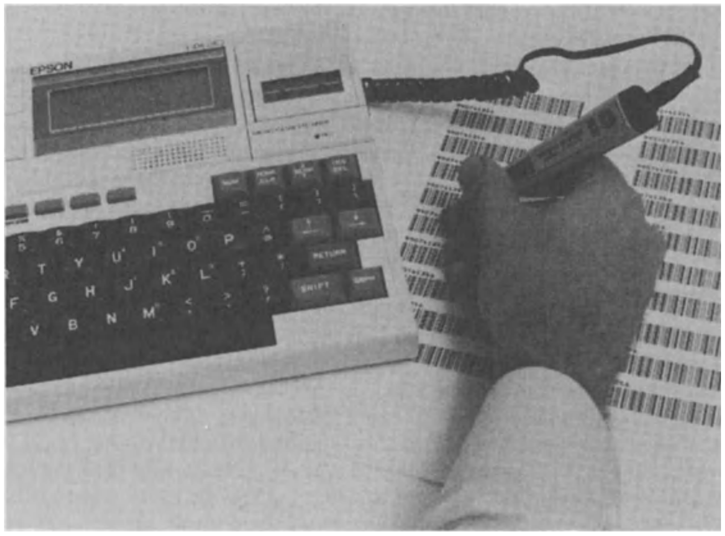


Fig. 10.1 Bar code reader from Interface Solutions, Inc.

Code 3 of 9
Interleaved 2 of 5
EAN 8
EAN 5
UPC A to UPC E
Modified Plessey
Monarch Codabar

Epson (UK) Ltd can supply a similar program on an internal ROM, for £45.

Warburton Franki has a print/read package for 3 of 9 code (Code 39). The print program can produce bar codes in five different heights — 0.17 and 0.83 in 4.3 to 21 mm — on an Epson printer. A BASIC demonstration program is also included ($60).

BiTech Enterprises was kind enough to loan us one of its bar code packages. This package includes a wand, a sample HX-20 program to read 3 of 9 code, and an 11-page booklet with technical and operational information on using a bar code wand with any computer.

BiTech gave us some sample imprints to test the package against, but because the printed samples were not the best we had many failures to read the data. That brings us to one of the best points of bar codes, and one of the worst. Every bar code contains a checksum in it which prevents false reads. You just can't read the data incorrectly — you'll either be able to read it right or not be able to read it at all. The bad part is that the bar code must be well printed to be clear enough for the reader to pick up.

If the BiTech package had a drawback, it was that no other software was provided, nor were there any suggestions on how to write any. Perhaps the package was intended for OEMs who would add their own software. But the end-user who wants a code other than 3 of 9 needs to get software elsewhere.

Another way to hook up a bar reader is via the RS-232 port. Analog and Digital Peripherals makes one unit, AC powered, that will read 3 of 9 code plus up to three user-selectable codes. This unit is $794 for the slot version, $884 for the wand version. The Datawand from MSI is a battery-powered — no external wires — unit with its own CMOS memory. Another unit, from Skan-a-Matic, reads Interleaves 2 of 5, 3 of 9, Codabar, UPC A and E, and EAN 8 and 13. Still another is marketed by Altek.

Note Perhaps you don't want a complete hardware/software package. Any combination of software for the HX-20 and any bar code reader for the HX-20 should work, though you should get either/both on a return basis.

The following article has been reprinted from the November 1983 issue of *80 Micro*, Copyright 1983 by *80 Micro*. It was originally intended for TRS-80 Model III owners, but is equally applicable to the HX-20 user. The technical information on bar codes is followed by a BASIC program to print bar codes on an Epson MX-80 printer.

Bars and Stripes Forever

by Davey S. Thornton

Most people think of bar codes as the striped labels on grocery items that identify a product and its cost. But bar code applications are more extensive — they are used in both industrial and commercial sales, inventory control, and equipment and product status accounting.

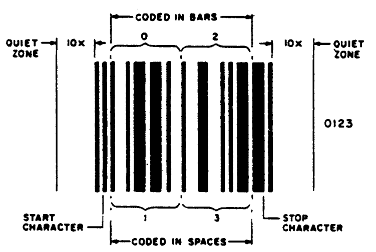
Several different types of bar codes exist. Grocery stores use the Universal Product Code (UPC); a description of UPCs along with a program to produce them appears on page 114. This article explains how industrial bar codes work and provides a Model III program to generate the standard bar codes: Interleaved 2 of 5, 3 of 9, and Codabar bar codes.

How the Codes Work

A bar code is a self-contained message that rapidly transmits data between independent systems with relative security and minimal hardware. Since bar codes interface with computers, binary notation is the basis for the algorithms used to encode and decode data.

The Interleaved 2 of 5 code has a maximum of 2 of 5 bits as binary 1s (wide bar or wide space) in any code sequence. The Interleaved 2 of 5 code consists of a set of start and stop bits with a maximum of five groups of bars and spaces representing 10 numeric characters (see Fig. 1). A narrow bar or space represents a logic zero and a wide bar or space a logic 1. In each group of bars and spaces, the bars represent the first character and the spaces represent the second character. Figure 1 gives the code sequence for the Interleaved 2 of 5 code.

Interleaved 2 of 5

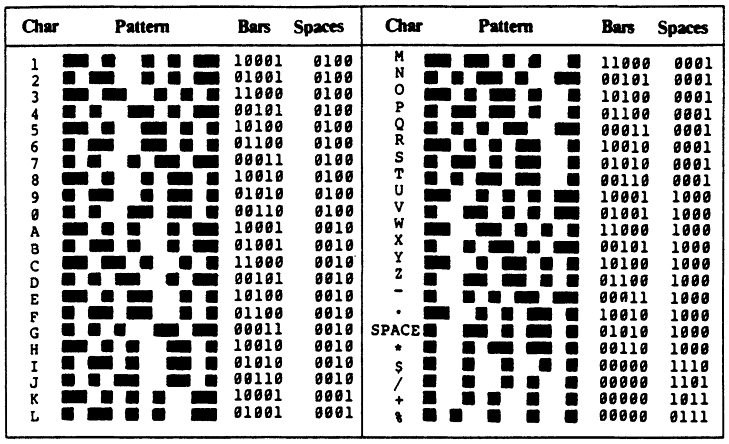


Data Character Set

Wide bars and spaces = Binary 1
Narrow bars and spaces = Binary 0
Each data character contains 5 binary elements, 2 of the 5 are binary 1s

Data Character	Weighted Position 1	2	4	7	P
0	0	0	1	1	0
1	1	0	0	0	1
2	0	1	0	0	1
3	1	1	0	0	0
4	0	0	1	0	1
5	1	0	1	0	0
6	0	1	1	0	0
7	0	0	0	1	1
8	1	0	0	1	0
9	0	1	0	1	0

Figure 1. Interleaved 2 of 5 code sequence.



Char	Pattern	Bars	Spaces	Char	Pattern	Bars	Spaces
1		10001	0100	M		11000	0001
2		01001	0100	N		00101	0001
3		11000	0100	O		10100	0001
4		00101	0100	P		01100	0001
5		10100	0100	Q		00011	0001
6		01100	0100	R		10010	0001
7		00011	0100	S		01010	0001
8		10010	0100	T		00110	0001
9		01010	0100	U		10001	1000
0		00110	0100	V		01001	1000
A		10001	0010	W		11000	1000
B		01001	0010	X		00101	1000
C		11000	0010	Y		10100	1000
D		00101	0010	Z		01100	1000
E		10100	0010	-		00011	1000
F		01100	0010	.		10010	1000
G		00011	0010	SPACE		01010	1000
H		10010	0010	*		00110	1000
I		01010	0010	$		00000	1110
J		00110	0010	/		00000	1101
K		10001	0001	+		00000	1011
L		01001	0001	%		00000	0111

Figure 2. 3 of 9 code sequence used as start/stop code only.

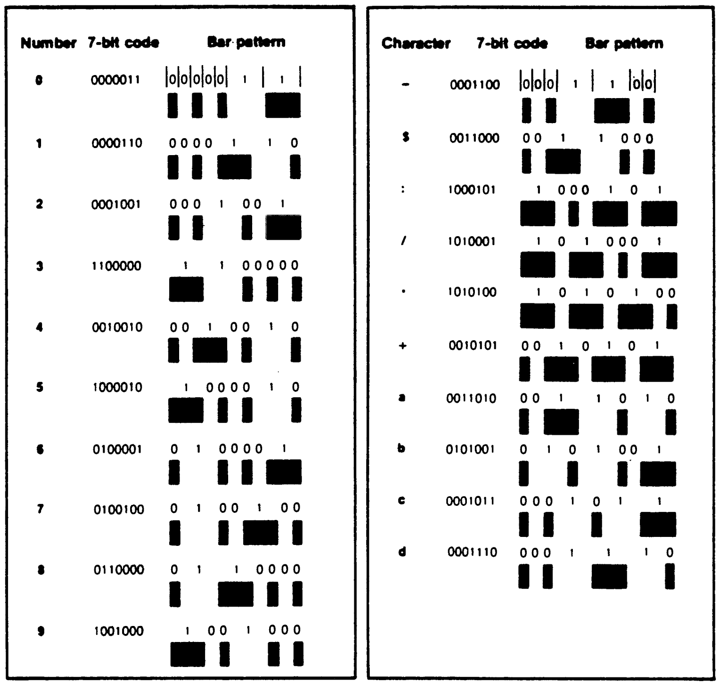
Interleaved 2 of 5 code represents only numeric characters while the 3 of 9 code represents both numeric and alphabetic characters. Each character in a 3 of 9 code consists of 9 bits with five bars and four spaces. The 3 of 9 code is so named because no more than 3 of 9 bits can be logic 1s (wide bar or wide space) in any one sequence. Further, the space between the characters is not significant because of the discrete nature of the code.

The 3 of 9 code uses the spaces between bars to point to one of five character groups. Within the group, the bar identifies the specific character. These bar codes use the same binary sequence as the Interleaved 2 of 5 code (see Fig. 2). This pattern holds for all but four special characters represented by bars equal to logic zero and spaces with alternate three logic 1s. Figure 2 gives the code sequence for the 3 of 9 code.

Codabar code includes a numeric set, six special characters, and four interchangeable start/stop codes. Unlike the 2 of 5 and the 3 of 9 codes, you can scan Codabar codes in either direction. The length of the Codabar character isn't fixed as in the 2 of 5 and 3 of 9 codes shown below.

0 through 9, –,S	L1 = 5*X + 2*N*X
:, /, ., + A, B, C, D	L2 = 4*X + 3*N*X

The difference in L1 and L2 lengths shows up as (N – 1)*X, but is not significant and can be made up in the intercharacter gap. Since the code is discrete, the intercharacter gap change doesn't affect code readability. Figure 3 gives the Codabar sequence.



Number	7-bit code	Bar pattern
0	0000011	0 0 0 0 0 1 1
1	0000110	0 0 0 0 1 1 0
2	0001001	0 0 0 1 0 0 1
3	1100000	1 1 0 0 0 0 0
4	0010010	0 0 1 0 0 1 0
5	1000010	1 0 0 0 0 1 0
6	0100001	0 1 0 0 0 0 1
7	0100100	0 1 0 0 1 0 0
8	0110000	0 1 1 0 0 0 0
9	1001000	1 0 0 1 0 0 0

Character	7-bit code	Bar pattern
-	0001100	0 0 0 1 1 0 0
$	0011000	0 0 1 1 0 0 0
:	1000101	1 0 0 0 1 0 1
/	1010001	1 0 1 0 0 0 1
.	1010100	1 0 1 0 1 0 0
+	0010101	0 0 1 0 1 0 1
a	0011010	0 0 1 1 0 1 0
b	0101001	0 1 0 1 0 0 1
c	0001011	0 0 0 1 0 1 1
d	0001110	0 0 0 1 1 1 0

Figure 3. Codabar code sequence. (Reprinted with permission from Bar Code News.*)*

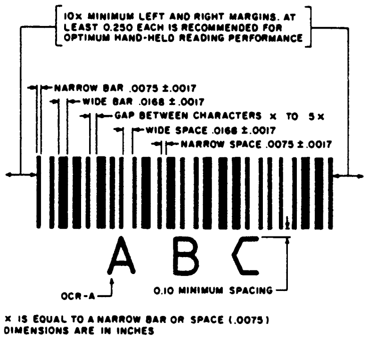

10x MINIMUM LEFT AND RIGHT MARGINS. AT LEAST 0.250 EACH IS RECOMMENDED FOR OPTIMUM HAND-HELD READING PERFORMANCE
NARROW BAR .0075 ±.0017
WIDE BAR .0168 ±.0017
GAP BETWEEN CHARACTERS x TO 5x
WIDE SPACE .0168 ±.0017
NARROW SPACE .0075 ±.0017
A B C
OCR-A
0.10 MINIMUM SPACING
x IS EQUAL TO A NARROW BAR OR SPACE (.0075)
DIMENSIONS ARE IN INCHES

Figure 4. MIL-STD-1189 code dimensions.

Printing Code

Next, determine if you can use a specific dot-matrix printer as an inexpensive and convenient bar code printer. Using MIL-STD-1189 (see Fig. 4) and the ANSI bar code specifications (see Table 2), you can determine the density of printable bar code characters, and whether a specific printer can print bar codes. However, the vast majority of dot-matrix printers with graphics capability are capable of printing bar codes of some density. (For a more rigorous evaluation of this subject, I recommend an article by Wellman Hoff in the winter issue of *Computer Technology Review, the System Integration Source Book,* West World Productions Inc. This article includes a Basic program for evaluating dot-matrix printers.)

Since my system includes an Epson MX-80 FT printer, my evaluation is limited to this printer. The MX-80 prints 120 columns per inch in the high-resolution mode. This equals a horizontal spacing of .00833 inches. The vertical spacing is restricted to the minimum platen shift, which equals .00463 inches. The dot separation, .01388 inches, defines the dot diameter.

	UPC/EAN	3 of 9 Code	Interleaved 2 of 5	Codabar
Character Set	Numeric	Alphanumeric plus −.*$/ + % and space	Numeric	Numeric plus $−:/.+
Number of Characters Encoded	10	43	10	16
Start and Stop Codes	Unique, both (101)	Unique, both (*)	Start NB/NS/ NB/NS Stop WB/NS/ NB	4 possible a/t,b/n,c/*, d/e
Number of Module Combinations Used	4	2	2	2
Maximum Substitution Error Rate without Check Digit (CD)	CD required	1 in 10^6	1 in 10^6	1 in 10^6
Maximum Substitution Error Rate with Check Digit (CD)	1 in 10^4	1 in 10^9	1 in 10^8	1 in 10^9
Ten-character Length for .010-Inch Module (Nominal)	.70 inch	1.4 inch	.90 inch	1.0 inch
Variable Length	No	Yes	May be w/CD	Yes
Discrete	No	Yes	No	Yes
Self-Checking	Yes	Yes	Yes	Yes
Date Introduced	1973	1974	1972	1972
Codified in Standards	UPCC/IAN	USD2&3/ANSI/ DOD/AIAG	USDI/ANSI/ AIAG	USD4/ANSI/ CCBBA
Market Influence	Retail	Industrial & Government	Industrial	Medical/Photo/ Libraries

Table 1. Comparisons of popular bar-code symbologies.

CODABAR CODE

* 1234567890−$ *

INTERLEAVED 2 OF 5 CODE

* 1234567890 *

3 OF 9 CODE

* CODE TEST SEQUENCE *

Figure 5. Sample bar-code printout.

There are two relationships you must examine to determine if a specific printer is capable of printing a bar code.

The first is the overall ratio of narrow to wide elements. This depends heavily on the ratio of the dot diameter to the vertical and horizontal spacing. With the Epson MX-80 this difference is approximately 1.6-to-1 (.01388/.00833 = 1.666) and 3-to-1 (.01388/.00463 = 2.998) for the vertical spacing.

This amount of overlap also tells you that the second relationship — the dot gap between both vertical and horizontal dot placement is not significant because of the size of the dot diameter. With a dot radius equal to or greater than the separation between dot positions, there is little or no dot gap.

BAR CODE	ELEMENTS	ELEMENT WIDTH TOLERANCE T	WIDE-TO-NARROW ELEMENT RATIO N*
INTERLEAVED 2-OF-5	NARROW BAR W NARROW SPACE W WIDE BAR WIDE SPACE	$\pm\left(\frac{18N-21}{80}\right)W$	2:1 TO 3:1 (MUST EXCEED 2.2:1 WHEN-EVER NARROW ELEMENT <0.02-IN. WIDE)
3-OF-9	SAME AS ABOVE	$\pm\frac{4}{27}(N-2/3)W$	SAME AS ABOVE
CODABAR	9 BAR WIDTHS 10 SPACE WIDTHS	$\pm\frac{0.0015}{0.0065}\times$ ELEMENT WIDTH	DOES NOT APPLY

N = THE RATIO OF THE WIDTH OF THE WIDE ELEMENT TO THE WIDTH OF THE NARROW ELEMENT

(NOMINAL RATIO:N MUST BE HELD CONSTANT WITHIN AN INTERLEAVED 2 OF 5 AND 3 OF 9 BAR CODE SYMBOL).

FOR ALL THE ABOVE BAR CODES:

BAR CODE HEIGHT MINIMUM IS 0.25 IN. FOR HAND SCANNING OR 15% OF THE BAR CODE LENGTH, WHICHEVER IS GREATER; MINIMUM OF 1.25 IN. OR 25% OF THE BAR CODE LENGTH, WHICHEVER IS GREATER, FOR TRANSPORT PACKAGES.

MINIMUM NOMINAL WIDTH OF NARROW ELEMENTS IS 0.0075 IN. EXCEPT FOR DIRECT PRINTING ON CORRUGATED CONTAINERS, WHERE 0.040 IN. IS REQUIRED.

VOIDS OR SPOTS MEETING EITHER OF THE FOLLOWING ARE PERMITTED:
(1) CONTAINED WITHIN A CIRCLE WHOSE DIAMETER IS 0.4 TIMES THE NOMINAL WIDTH OF THE NARROW ELEMENT.
(2) OCCUPIES NO MORE THAN 25% OF THE AREA OF A CIRCLE WHOSE DIAMETER IS 0.8 TIMES THE NOMINAL WIDTH OF THE NARROW ELEMENT.

MINIMUM PRINT CONTRAST SIGNAL IS 75% IN THE B633 SPECTRAL BAND.

Table 2. Summary of ANSI bar code specifications. (Reprinted with permission from Computer Technology Review.*)*

Dot Row Narrow Elements	Dot Row Wide Elements	Density in Characters/Inch
* 2	5	4.1379
** 2	7	2.8583
3	8	2.6667
4	9	2.1827
4	10	2.0698
4	11	1.9680

*Program Listing in standard mode.
**Program Listing in compressed mode.

Table 3. MX-80 FT printer evaluation.

Program Listing. Bar code print routine.

```
10 '*******************************************
20 '*      BAR CODE PRINT ROUTINE              *
30 '*               by                         *
40 '*        Davey S. Thornton                 *
50 '*        8128 Brucar Court                 *
60 '*      Gaithersburg MD. 20877              *
70 '* Print Interleaved 2 of 5, 3 of 9         *
80 '*      and Codabar codes                   *
90 '*******************************************
100 CLEAR2000:DIM R(255),D$(59),E$(10),C$(33),A$(4)
110 FOR I=1 TO 59
120 '*****             LOAD BINARY CODE    ****
130 READ D$(I):NEXT I
140 DATA "011000100",,,,"010101000","000101010",,,,,"010010100","0
10001010",,"010000101","110000100","010100010","000110100","100100
001","001100001","101100000","000110001","100110000"
150 DATA"001110000","000100101","100100100","001100100",,,,,,,,"10
0001001","001001001","101001000","000011001","100011000","00101100
0","000001101","100001100","001001100","000011100"
160 DATA"100000011","001000011","101000010","000010011","100010010
","001010010","000000111","100000110","001000110","000010110","110
000001","011000001","111000000","010010001","110010000"
170 DATA"011010000"
180 FOR I=1 TO 10:READ E$(I):NEXT I
190 DATA"00110","10001","01001","11000","00101","10100","01100","0
0011","10010","01010"
200 FOR I=1 TO 33:READC$(I):NEXT I
210 DATA"0011000",,,,,,,"0010101",,"0001100","1010100","1010001","
```

Listing continued

Thus, for the Epson printer, the dot over-print is the controlling factor and affects the narrow-to-wide element ratio. Table 3 gives a list of narrow- to wide-element dot-row widths and the density in characters per inch.

Listing continued

```
0000011","0000110","0001001","1100000","0010010","1000010","010000
1","0100100","0110000","1001000",,"1000101",,,,,,"0011010"
220 DATA"0101001","0001011","0001110"
230 FOR I=1 TO 4:READA$(I):NEXTI
240 DATA"0101100","1001010","1101000","0111000"
250 CLS:INPUT "ENTER 'L' FOR STANDARD FORMAT OR 'C' FOR COMPACT FO
RMAT";Q$
260 IF  Q$="L" THEN C1=3:C2=4 ELSE IF Q$="C" THEN C1=2:C2=3 ELSE G
OTO 250
270 CLS
280 'INPUT SEQUENCE          *****
     "." "A,B,C,D" SEQUENCE "A,B,C,D" CODABAR
     "." SEQUENCE                      2 OF 5 CODE
290 ' SEQUENCE                        3 OF 9 CODE
     WHEN ENTERING CODABAR CODE THE STOP CODE MAY
     BE LEFT OFF IF THE SAME CODE WAS USED AS START CODE.
300 'EXAMPLE
        CODABAR           .A1234567890-$:/+D
        2 OF 5            .1234567890
        3 OF 9            ABC-XYZ1234567890.$/ +%
310 INPUT"ENTER CODE SEQUENCE";Q$
320 IF LEFT$(Q$,1)="." THEN 440 ELSE T$=""
330 '*****             PRINT 3 OF 9 CODE     ****
340 IF LEN(Q$)>20 AND C1=3 OR LEN(Q$)>30 AND C1=2 THEN PRINT"STRIN
G TO LONG REENTER":GOTO310
350 PRINT"3 OF 9 CODE"
360 LPRINT"3 OF 9 CODE":LPRINT
370 FOR I=1 TO LEN(Q$)
380 T$=T$+D$(ASC(MID$(Q$,I,1))-31)+"0":NEXT I
390 T$=D$(11)+"0"+T$+D$(11)
400 FOR I=1 TO LEN(T$)
410 R(I)=VAL(MID$(T$,I,1))
420 NEXT I
430 GOTO600
440 T$="":S$="":Q$=RIGHT$(Q$,LEN(Q$)-1)
450 '*****             PRINT 2 OF 5 CODE     ****
460 IF ASC(Q$)>=65 THEN 650
470 IF LEN(Q$)>11 THEN PRINT"STRING TO LONG REENTER":GOTO310
480 PRINT "INTERLEAVED 2 OF 5
490 LPRINT "INTERLEAVED 2 OF 5 CODE":LPRINT
500 FOR I=1 TO LEN(Q$)-1 STEP2
510  T$=T$+E$(ASC(MID$(Q$,I,1))-47): S$=S$+E$(ASC(MID$(Q$,I+1,1))-
47)
520 NEXT I
530 S$=S$+"      ":R$=""
540 FOR I=1 TO LEN(T$)
550 R$=R$+MID$(T$,I,1)+MID$(S$,I,1):NEXT I
560 R$="0000"+R$+"100"
570 FOR I=1 TO LEN(R$)
580 R(I)=VAL(MID$(R$,I,1)):NEXT I
590 T$=R$
600 N1=LEN(T$):K=0:N6=0
610 FOR I=1 TO LEN(T$):IFK=0 THENK=1ELSEK=0
620 N6=N6+R(I)*C2+C1:NEXT I
630 GOSUB 770
640 GOTO310
650 Q=ASC(Q$):Q$=RIGHT$(Q$,LEN(Q$)-1):T$=""
660 Q1=ASC(RIGHT$(Q$,1)):IF Q1>=65 THEN Q$=LEFT$(Q$,LEN(Q$)-1) ELS
E Q1=Q
670 '*****             PRINT CODABAR CODE    ****
680 IF LEN(Q$)>17 THENPRINT"STRING TO LONG REENTER":GOTO310
690 PRINT "CODABAR CODE"
700 LPRINT"CODABAR CODE":LPRINT
710 FOR I=1 TO LEN(Q$)
720 T$=T$+C$(ASC(MID$(Q$,I,1))-35)+"0":NEXTI
730 T$=C$(Q-65)+T$+A$(Q1-65)
740 FOR I=1 TO LEN(T$)
750 R(I)=VAL(MID$(T$,I,1)):NEXTI
760 GOTO 600
770 '*****             PRINT BAR CODES       ****
780 FOR M=1 TO 8
790 LPRINT CHR$(27)"A"CHR$(4);
800 N4=FIX(N6/256)
810 LPRINT CHR$(27)"L"CHR$((N6/256-N4)*256)CHR$(N4);
820 K=0
830 FOR J=1 TO N1
840 IF K=0 THEN K=1 ELSE K=0
850 FOR I=1 TO R(J)*C2+C1
860 LPRINT CHR$(127*K);
870 NEXT I
880 NEXT J
890 LPRINT
900 LPRINT CHR$(27)"@";
910 NEXT M
920 LPRINT CHR$(27)"@"
930 LPRINT CHR$(27)CHR$(14)"* " Q$" *"
940 RETURN
950 END
```

The Program Listing provided here produces 2 of 5, 3 of 9, and Codabar codes. The program asks for a code sequence and the input specifies the type of code you want produced. If you desire 3 of 9 code, you must use a sequence of up to 30 alphanumeric characters. (The 3 of 9 code has a restriction of 43 characters but the listing prints only 30 characters in compressed mode and 20 characters in standard mode.)

If you desire 2 of 5 code, then a period precedes the code sequence and you can use only numeric characters. With the 2 of 5 code, the sequence must be less than or equal to 10 characters.

If you want the Codabar code, then in addition to preceding the code sequence with a period, you must include a stop/start code after the period and before the code sequence. Since the Codabar start/stop codes are interchangeable, you can use a different code at either end. If you desire a different code at the end of the sequence, include it during the entry; otherwise, the program assumes that you want the code specified as a start code as a stop code.

Scanning the Code

To help those of you who are ready to jump up and write the definite scanning routine, here are a few tips.

First, assuming that you use a TRS-80 Model III, you can make some assumptions about the speed of the scanning algorithm and the problems you're likely to encounter. Assume that the algorithm takes 50 microseconds to execute, then estimate that the routine has a sampling rate of about 20,000 samples per second (sps).

If you wave the reading wand at the average rate of about 30 inches per second (ips), you find that each sampling period corresponds to a wand travel of about .0015 inches (30 ips/20,000 sps). As discussed earlier, the narrow-bar spacing is .01666 inches based on a dot row equal to two dot rows for a narrow bar or space. Thus, you can see that a narrow bar or space consists of approximately 10 samples — each representing about 10 percent of the narrow-bar/space width.

This tells you that a read system, based on the TRS-80 Model III, is susceptible to errors in acceleration/deceleration and variable speed of wand motion. Incorrect readings used in calculations contribute to additional errors.

Bar code scanners have circular viewing areas or apertures (which vary from .0045 to .017 inches in diameter). This increases the chance for error since reflected light entering this aperture is converted from an analog signal to a binary digit. The scanner diameter adds additional error as a result of the amount of light admitted, and the size of the bar/space reflecting that light.

The elements scanned are represented by two widths. The ratio of narrow-to-wide should be between 2-to-1 and 3-to-1. The algorithm should compare neighboring elements in consecutive fashion. Compare bars to bars and spaces to spaces. Comparing these elements to their nearest neighbor minimizes errors resulting from speed changes (acceleration/deceleration).

Use the start/stop code as a known to identify the start of a code sequence as well as to define the narrow and wide bar/space widths as determined by the wanding rate. Use these values to evaluate successive code bits. All of these calculations can't be made during the scanning process without further reducing the scan rate, which is unacceptable. It is possible, however, to store the scan data and perform the calculations after the stop code is received.■

11 SOFTWARE AND SYSTEMS

'When all else fails, read the instructions'

This chapter covers:
Packaged software and complete systems from the US, the UK, and Australia

We've collected information on a good number of software packages and systems available for the HX-20. Except where noted, all of the following descriptions are based on information provided by the vendors, often in the vendors' own words. Addresses of all vendors can be found in Appendix A. All prices exclude any taxes or freight charges. Prices and product descriptions are subject to change, so check with the vendor for latest information before ordering.

Additional software descriptions can be found in Chapters 3,6,8,9, and 10.

QUICKVIEW SYSTEMS

Paul Heckel of QuickView has pioneered a new method of displaying information on small screens. By using text compression/decompression, QuickView can display data in a more usable fashion than other programs running on small screen computers.

Viewdex, an Electronic Rolodex File, is QuickView's first product for the HX-20. Much of Heckel's philosophy is explained in his book: *The Elements of Friendly Software Design* (Warner Books, 1984). We'll just give a quick overview of Viewdex here.

Each of the electronic index cards in the Viewdex system can have a name, address, and phone number — in a compressed format. Enough compression is used to enable the entire record to fit on the HX-20 screen. But even though compressed, the user gets a 'visual image' of what's on the card. On command, Viewdex will decompress a specified field and show its full contents. If the field is longer than 20 characters, it can still be read because of a unique technique that deletes vowels on the left as it adds additional characters on the right.

The user can quickly flip through the cards with the up/down arrow keys. Powerful search and full-screen editing functions are also included.

Viewdew will likely be marketed by Epson America. No price has yet been set.

FFOSS LTD

Ffoss has written a number of programs which are being marketed by other companies — Epson UK Ltd, for instance. The ones marketed are Ffosswriter by Ffoss (discussed in Chapter 9, Word Processing), LO-RAX, and RAXUTILS.

Lo-RAX

LO-RAX is a set of low-level cassette driver routines that can be integrated in with application programs. It provides random access to data files on microcassette rather than the sequential access supported by the standard HX-20 operating system.

LO-RAX works with your file or database handler, keyword search algorithm, alphabetic sort or whatever. Ffoss Ltd uses LO-RAX as the basis for some of its other software packages, including: Card Index/Mlist (keywords, alphabetic sort), Diary/Raxutil (direct logical access), and Ffosswriter (linked segment files).

You could just put a directory on tape and WIND away yourself. But for total reliability and the maximum speed, LO-RAX provides:

- No loss of data (LO-RAX used double write/CDC checks).
- No corruption of tape positioning, even under *continuous* random access use.
- No user intervention required.
- The maximum amount of data that can be stored on a tape.
- Fast search algorithm.
- Record access time always the same.
- Compensation for variations in users' tapes and machines.

LO-RAX comes with clear documentation plus demonstration programs. It can be located anywhere in RAM, in the internal EPROM slot or in the expansion unit. An object code licence (only) is provided, as technical details remain proprietary.

Raxutils — £15

RAXUTILS is for users with RAX (random access cassette) data tapes. According to the authors, features include:

- Copy a RAX tape to/from other RAX tapes or external cassette tapes.
- Rename a RAX tape.
- Format a blank tape for later use as a RAX tape.

KUMA COMPUTERS LTD

Kuma publishes programs created by independent software developers. All programs are available on microcassette, and all come with concise four to eight page manuals, unless otherwise specified. All will run with the basic 16K HX-20, though the expansion unit may be helpful in some cases.

Desk Master 1 by Eclectic Systems — £29.50

This program turns the HX-20 into a printing calculator. Operations are:

addition
subtraction
multiplication
division
reverse a sign
print sub-total
print total
chain calculations
clear keyboard entry
clear machine except for memory
memory addition
memory subtraction
recall memory
recall memory with clear
print item count
print time/date
percent
square root
enter a constant
add comments

Options available are:

print off/on
number of decimal places
floating or fixed decimal point
accumulate grand total for all calculations

If an entry error is made, the program displays an appropriate message.

Desk Master 2 by Eclectic Systems — £29.50

Described in Chapter 9 Word Processing.

Desk Master 3 by Nick Riordan — £29.50 (basic 16K version), £39.50 (enhanced 32K version)

This is a VisiCalc style spreadsheet. The publisher warns that due to restricted memory space and the fact that the program is in BASIC, this program is slow and functions are limited compared to its desktop cousins.

The format of the commands and of the cell formulae is very reminiscent of VisiCalc, as much as we can tell from the instruction booklet without seeing the actual program. Functions include save/load to/from cassette, printing all or part of a spreadsheet, replication, recalculation on demand, help. What you will find most noticeably missing are brackets for calculations and built-in functions, like SUM.

Desk Master 4 by AG Microsystems — £29.50

This is described in Chapter 8, Communications.

Desk Master 6 by Eclectic Systems — £19.50

DM6 helps the user make decisions by quantifying the choices available. If you have to make a choice among several options — buying a car, for instance, or picking a marketing strategy — and you can assign a numerical rating to each choice, this program can show you the 'best' choice. Up to 7 options for each decision can be evaluated against 2–20 different criteria — fuel economy, luggage space, etc. The output is in the form of matrices that show the numerical relationships of the choices to each other.

Some explanation of the theory behind the program is provided, but the serious user would be best advised to locate a business management book if he really wants to understand the concepts. The casual user, on the other hand, can easily operate the program by simply responding to the prompting questions.

Desk Master 7 — Editor/Assembler — by Appollo — £14

This is described in Chapter 6, Assembly Language.

Desk Master 8 by Eclectic Systems — £19.50

This mailing list program allows the user to create, maintain, and print a list of names and addresses. By using adhesive paper in the printer, mailing labels can be printed out directly. Up to 60 addresses can be held in a 16K machine, over 200 if the maximum 32K RAM is used.

The addresses are stored as if they were each on index cards. The cards in the file are each the same length, holding from 4–8 address lines per card. (The length of the cards is pre-set by the user.) Functions available include: create a file, add new entries, change existing addresses, list addresses, print the file, save/load a file. The PF keys are assigned to special functions during text processing. Files are saved in RAM, but must be saved to tape before running any other program that uses RAM files.

Desk Master 9 — RAM Database — by Southwestern Consultants — £29.50

This program makes it easy to save and retrieve data to/from the HX-20's RAM files.

The first step in using this program is one common to most data management programs: defining how the data is to be organized. You can define a 'column', i.e. field, to be either alpha or numeric. Alpha fields can be any length from 1 to 255 characters, numeric fields can be any length up to 999,999.99. The 16K version of this program allows 8 fields; the 32K allows any number of fields up to the limits of memory. A typical example might be: 4 alpha fields of 10 columns each plus 4 numeric fields of 99.99 form. (Apparently, numeric fields always have two decimal places.) This would allow for 73 records in a 16K machine or 237 records in a 32K machine. (Other combinations can yield less or more numbers of records.)

Once the file is defined and the definition saved to tape you are now ready to enter data. The available functions are:

Add — insert new records into the file.
Change — modify existing information by record number.
Delete — remove a record from the file, giving record number.
Find — locate a particular record using either the 'key' (first) field in a sorted file or any alpha or numeric information in any field.
List — display a record (or just the first field) on the screen or microprinter; allows scrolling through the file.
Sort — put the file in order based on the contents of the first field.
Total — sum up each of the numeric fields.
Write — save the file to tape or print on an external printer.

Desk Master 10 — Labeller-by Eclectic Systems — £19.50

This program prints custom-designed labels, using the sticky-back paper available from Kuma. Like Desk Master 8, also written by Eclectic, Labeller allows the user to edit label contents using the PF keys. Any character can be designed into the label, including graphics characters. Labels can be printed out sequentially, or just a single label can be printed, or multiple copies of a single label.

Desk Master 11 — Mobile Stock Recorder — by Country Software

This is described in Chapter 10, Inventory/Stock Tracking.

Desk Master 14 — Expenses — by A.J. and S.E. Pack

This program looks much like Home Budget by the same authors (see below). This is a 'rolling budget' program for maintaining up to 16 expense categories and 9 income categories. Data is kept on a weekly basis with the user being able to 'see' 12 months into the future. Data can be printed in numerical form or output as histograms.

Cash Exchange by G. Greensall — £19.50

This program allows the user, according to the publisher, to detect trends in currency rates, supply listings of chosen currencies, and convert amounts to/from chosen currencies using either current or historical rates of exchange.

Currency information for the period Jan 1981 to June 1983 is supplied in pounds sterling, US dollars, Japanese yen, West German Deutschmarks, French francs, and Dutch guilders. Currency information for other countries can be added. The program is designed to be used to compare the pound with one other currency, but one of the menu options allows cross-comparisons between any two other currencies.

A graphing feature — to screen or printer — shows the rate of exchange between chosen currencies over the past 12 readings (1 year) or the past 24 readings (2 years). The rate of exchange for any month within this time frame can also be listed directly.

To keep the currency rates up to date the user must change DATA statements within the program. This is clearly described in the manual.

Home Budget (Version 2) by A.J. and S.E. Pack — £17.35

Up to 30 user-definable categories can be maintained for keeping track of home expenses, or other similar financial data. Besides tracking income and expenditure, rolling forecasts can be made. Results can be displayed by categories, with totals, or in histogram format.

Note This program is intended to work with pounds sterling, but should work with dollars just as well.

Most programs of this type are too simple to be of much use, but this one looks like the authors gave some forethought to what people would really want in a budgeting program.

User-Definable Graphics — £17.50

Several times in this book we've pointed out that the HX-20 has 32 user-definable graphics characters that can be displayed on the screen. But it's more than a casual operation to create those new characters. HX-20 UDG from Kuma gives you an easy way to do it. (See King Software for a similar program.)

All of the UDG characters can be viewed at one time by selecting that option off the menu. You can then select one for creation/editing. The editing function shows you an enlarged image of the character as well as the same character in real size. You can then make changes to it via the cursor keys and the space bar. You can also copy a previously defined character (one of yours or one of the pre-defined Epson ones) for further modifications. Other functions include: mirror imaging, inversion, dot reversing, printing, and saving/loading to tape.

Directions are given in the three-page manual on using your UDGs in other programs and maintaining them in a different part of memory. (As delivered, the program will store them at $0A40, requiring that MEMSET be set to 2817.)

H5 Horse Race Forecast Program — Professor Frank George — £24.50

Professor Frank George is a leading British cyberneticist who has served as a consultant to NATO, taught at universities in Britain and the US and is recognized, according to Kuma, as 'an expert in all aspects of forecasting'.

We give you that information in lieu of any hard evidence as to how well this program works.

H5, also available for the Apple, Sharp, Commodore, and NewBrain computers, looks like a useful aid for the serious handicapper, judging from the 13-page instruction booklet, but not, as the instructions warn, a complete replacement for individual judgement.

For input, the program uses data found in specialized horse racing publications, such as *Sporting Life* (UK) or *The Daily Racing Form* (US). This includes, for each horse, the results of its last four races, the weight-adjusted rating, and a speed figure. For each race, you enter the number of runners, the prize money, and the distance.

There are three outputs: a list of the horses running in the race, a display of all the information entered for the race, and, most importantly, a forecast for the race whose details are currently in memory. The forecast takes the form of recommendations, e.g., excellent bet, very good bet, possible bet, etc.

Data can be printed, saved to tape, or recalled from tape and modified.

Entertainment Pack-1 — £17.50

Classic games like Moonlander, Biorhythm, Blackjack, 39 Steps.

Stampout — £9.95

Game: avoid the hazards in an expanding pattern.

Computax — £49.50

This program is designed to calculate personal tax liability for use by accountants and life assurance consultants.

A.P. SYSTEMS LTD

As a retailer, APS can provide many of the programs available from Epson UK or Kuma or other major publishers. The company also publishes its own line of software.

APWriter — £25

A suite of three programs allowing the user to create/edit text, create/file/update a name and address file, and merge the two to produce 'personalized' letters. Programs are written in BASIC.

The Receipt Program — £25

Provides itemized receipts to be given to customers with the details of each transaction stored in memory. At the end of the day, a report is produced showing the invoice number, VAT, and totals. Summary information is written to microcassette tape. A second program can read the tape data and produce a report by day, showing the date, VAT, gross, and net. (Note to US readers: State sales tax can probably be substituted for VAT, but ask first.) For businesses with less than 500 transactions per day.

The Restaurant Program — £25

This is a variation of the Receipt Program, intended for those in the restaurant business, with under 500 customers per day. In addition to the data collected above, this program also maintains service and cover charges.

The Housekeeper

This is a package of 8 utility programs for £25:

microcassette tape directory (2 programs)
calculator (3 programs)
measurement conversions
number and alpha sort
jotter

The directory program is designed to be recorded on each side of a microcassette tape and will help you keep track of what's on the tape. It appears to be similar in intent to the Skier programs in the BASIC manual that Epson America distributes. In any case, this writer feels that, due to the slow tape speed, it is far more efficient to use pencil and paper to keep track of what's on your tapes then to try to automate the process.

Calculator programs 1 and 2 provide the usual arithmetic operations plus percentages and memory. Calculator program 3 provides mathematical functions that are usually only found in scientific calculators: powers, roots, exponentials, natural logarithms, pi, sines, cosines, tangents, etc.

Linear measures, areas, volumes, weight, temperature, currency, and other units of measurement can be converted back and forth with the converter program.

The sort utility provides a set of subroutines to do numerical and alphabetical sorting, along with documentation showing how to incorporate those subroutines in your own programs.

The Jotter is a memo-writing program that allows the user to save a note of up to 50 lines on tape, for later printing.

KING SOFTWARE

King Software is run by attorney Larry Johnson and produces software for both the Epson HX-20 and the Radio Shack Model 100. The primary development effort is presently going towards a time and billing keeper for lawyers. Consultants who charge clients for their services on an hourly basis could probably use the system as well. This system will allow the user to edit and adjust the hours after they've been entered.

King Software sent us five program packs of miscellaneous programs for the HX-20. These are $29.95 each or $89.95 for all five on one microcassette. A sixth program pack is also available.

In general, these programs are not terribly sophisticated. However, we didn't find any bugs in any of the programs tested — and we tested all programs on all of the five packages we received.

King maintains a list of purchasers and sends out correction-sheets periodically. The documentation is clear, though perhaps overly simple. The programs are often marked by inconsistent input: sometimes the return key is required, sometimes it isn't. Sometimes another key instead of return is used to terminate input. There is also an annoying habit of re-running the copyright message before going back to the menu. But perhaps the worst defect is that many of the programs take so long to start up compared to how long it would take you to do them on a calculator or with pencil/paper, that you have to wonder whether it's worth it. The saving grace is that King's later programs, Draw and Maketune, for instance, are quite decent, and improvements are constantly being made to the earlier ones.

Program Pack 1 — Word processing and expense accounting

This package requires use of all five program areas in BASIC. The drawback here is that it is going to take a few minutes to load all files — and you'll need all five if you want to do word processing plus expenses; only four if just one of these functions is desired.

'Maketext'

This is a simple word processing program which can handle different types of non-continuous data. There is no word-wrap on input, the backarrow is destructive which makes changes harder, and the program doesn't honour carriage returns when printing. There is word-wrapping on output, though. But if you try to print continuous data, there is a loss of some characters, probably due to the program's trying to word-wrap. As with all HX-20 BASIC programs, there is no loss due to fast typing — it's all buffered.

Expenses program

The user defines expense categories and the program keeps running totals for each category. Instructions are given for changing the program to use user-selected categories.

Program Pack 2 — Phone directory and message recorder

Again, this pack uses all five program areas and since the programs are all interconnected, all five programs will have to be loaded — a process that can take several minutes.

This system lets you add/change/delete items in a list of phone numbers. It lets you log in calls, will give each a time–date stamp, elapsed time for each call, and your notes on each call. The idea is to start a log entry when you make/get the call, take notes, and end the entry when the call is over. You can erase the log and leave the phone number list intact or print out the log or phone list.

Program Pack 3

This is made up of three independent programs, each using a single login area.

RENO 21 — 'Shuffles' and graphically displays a 52-card deck for blackjack. One suggested use is as practice in counting cards.

Biorhythm — Given a birthdate, this program produces a little microprinter chart of physical, emotional, and intellectual cycles, for any specified period of time.

Dater — A calendar calculator, can compute days between dates, the day of the week for any date or the date that is a given number of days away from any other date.

Program Pack 4

This pack includes four independent programs, each using a single login area.

CONVERT$ — Converts from any currency to any other currency. Has built-in table of exchange rates, which can be updated or added to.

ALARM — Works as an alarm clock, displays the time and sounds a tune when the selected time is reached. **Note** the author warns that this can run down the battery.

TIMER — Shows time elapsed between a begin time and an end time.

MORSE — Converts an entered message into code, or vice versa; displays and sounds notes (long and short beeps); tests you on code letters/words (going in either direction).

Program Pack 5

This package is composed of two independent programs, but because of memory constraints, only one can be in memory at any one time.

DRAW — This program lets you draw with the numeric keypad and the space bar. Dots can be put down in any of eight directions, with the space bar erasing the dot just drawn. You can save your drawing to a RAM file or dump it to tape or send it out over the RS232 port. Three caveats: the program takes quite a while to save a drawing, it uses quite a lot of memory, and you can only keep one drawing in memory at a time.

MAKETUNE — Compose a tune by keying in the note value/octave and duration. Save to RAM, tape, RS-232.

King Software also markets the programs listed below. We haven't seen them, so the descriptions are based on information supplied by King.

PP6 — User-Defined Character Generator — $33.95

This allows the user to create his own characters. The cursor keys allow you to edit boxes on a 6 × 8 grid. Characters newly created can be saved to tape or transferred to another computer in data form. (See Kuma Computers for a similar package.)

Interval

For music students. Uses the HX-20 sound generator to test your ability to identify intervals within a given scale or between randomly selected notes.

AUTOMATED TIME INC.

One system designed specifically for lawyers is the LTR-IV Legal Time Reporter, from Automated Time Inc. The LTR-IV is sold as a total package, i.e., hardware + software. Basically, it keeps track of the time spent on behalf of a client, producing a log describing the work, along with any comments. By using the real-time clock in the HX-20 and accounting for office interruptions, the system eliminates time slips. An interface into end-of-month time/billing is provided. Similar systems, according to the company, are in the works for accountants, engineers, and physicians.

EPSON (UK)Ltd

These products are marketed by Epson UK and are available from A.P. Systems, Onestop, Transam Microsystems and other British dealers. Descriptions given are those obtained from dealers, from the Epson software catalogue or from the developer. Prices are from A.P. Systems and, as usual, exclude freight and VAT. The first price is for the program on micro-cassette, the second is for the same program on a ROM cartridge.

DIY — developed by Ffoss Ltd — £25/£70

This is a program generator system for the user who wishes to write data capture and data handling systems without learning BASIC. At generation time, the user can define a series of prompts that the operator will see, define validation characteristics for the expected input, set up special checks on the data and define external printer information. The user defines the screens and the order in which they will appear, starting anywhere and allowing many alternative routes which are followed when data is entered by the operator. At run-time, the operator enters data which can be validated, printed and/or stored on tape. The operator also has the full capability of skipping entries and returning to them later as well as reviewing/modifying previously entered data with the cursor keys.

ECalc — £30/£85

A VisiCalc-style spreadsheet system to create financial reports, models and 'what-if' forecasts by use of a work area arranged into rows and columns. Spreadsheet size is 14 × 15 or 26 × 35 with the HX-20 expansion unit. Written in BASIC and reputed to be slow. The LCD screen acts as a window on the spreadsheet, with the current cell always at the top, left-hand corner. Commands are entered with one key for ease of use. External Epson printers may be used to print an entire spreadsheet, which may also be stored on tape. ECalc was reviewed by Mike Liardet in *Personal Computer World*, 7/83. Liardet concluded that ECalc was a usable system for lightweight applications.

Card index system

The card index system is designed for anyone who needs to store and easily retrieve data, notes, price lists, customer information, estimates, addresses or other details. Up to 100 'cards' may be stored on each side of a tape, with up to 200 characters per card. Data is stored free-format with four, 10-character fields making up the card name for selection purposes. Screen editing plus selection criteria.

Diary — developed by Ffoss Ltd

The Diary package uses the RAX ROM to record events and appointments over a three to four month period. Appointments that are entered for any day covered within the time frame will be automatically sorted into time order. Appointments can be moved or copied from day to day and printed on the microprinter. Periodically recurrent events such as weekly meetings, birthdays, etc., can be stored such that their occurrence will be noted whenever an applicable day is accessed. The diary is accessed via a calendar displaying one week at a time which also displays summary information for each day and allows scrolling backward or forward one day or week at a time by single key depressions.

Correspondent 20 — £40/£80

This text editor allows the user to create up to 7 pages of text 60 lines by 60 characters on each side of the tape. The main functions are displayed in menu format with the editing facilities installed on the function keys. No word wrap, no right justification. May be linked to an external printer.

Mailing list — developed by Ffoss Ltd — £40/£80

This is a complete mailing list system, using the RAX ROM (which is included in the price). Mailing list items may be selected on one or two keywords, each up to 10 characters long. Subsets of items may be selected and manipulated, using the keywords as the search criteria. Wild card and partial keyword searching are available. Up to 250 items may be stored on a 30 minute tape, each item holding up to 254 characters. Items may be examined and modified with on-screen edit commands.

Sales Order Entry System — £40/NA

Allows the collection of sales orders on the customer's premises. Ties in with your price, stock, and description file (up to 399 records), downloaded from a central computer. An extended acknowledgement for the customer, showing discounts, VAT, etc., is also produced. Orders stored on tape may subsequently be transmitted to the central computer. Uses the Epson UK communications ROM. (Not appropriate for US users.)

Communications — £30

Described in Chapter 8, Communications.

Bar code reader software — £45

Described in Chapter 10, Inventory/Stock Trading.

Epson Games 1 — £18

Bomber — bomb the city and land the plane.
Lightcycle — avoid crashing into the wall.
Blackjack — popular card game.

Epson Games 2 — £18

Cavern — find your way around the caves and steal the treasure from the dragon.
Hangman — guess the word.
Sketchit — draw pictures on the screen and copy them to the internal printer.

SOFTWARE RICHES

Software Riches is a part-time effort of 17-year-old Jason Rich. Probably a better entrepreneur than programmer, his efforts for the HX-20 appear to be converted pocket computer programs for daily chores, such as keeping track of appointments. He's also done some games for the LCD screen.

Programs include:

Appointment Keeper
Calcu-list
Galactic War
Paddle Ball
Bingo

AMERICAN MICRO PRODUCTS INC.

The following programs, developed by AMPI, are expected to be marketed by Epson America. Reportedly, they were adapted from software written for programmable calculators.

Mathematics Library
- Prime Factors
- Gamma Function
- Polynomial Multiplication
- Hyperbolic Functions

- Relative Minimum
- Matrices
- Ordinary Differential Equations
- Roots of an Equation
- Simultaneous Equations
- Non-Linear Systems
- Integration
- Interpolation
- Coordinate Transformation
- Convolution

Electrical Engineering
- Active Filter Design
- Passive Filter Design
- Phase-Locked Loop
- Series to Parallel Conversions
- Smith Chart Calculations
- Signal Detection
- Decibels, Nepers, Power, Voltage, Current Ratio Conversions
- Transistor Parameter Conversions
- Complex Arithmetic
- Fourier Analysis
- Network Analysis
- Bode/Nyquist Calculations
- Multiplication of Polynomials
- Reactance Chart

Statistical Distribution Analysis
- Multiple Linear Regression
- Histogram
- Normal Distribution
- Weibull Distribution
- Exponential Distribution
- Binomial Distribution
- Poisson Distribution

General Statistics and Related Distributions
- Students' t Distribution
- F Distribution
- Chi Squared Distribution
- Means and Moments
- t-Test for Paired Observations
- t-Test for Unpaired Observations
- One-Way Anova
- Two-Way Anova
- Contingency Table
- Rank-Sum Test

Finance
- Annuities/Annuities Due
- Rent or Buy Decisions
- Cash Flow Analysis
- Bond Analysis
- Machinery Analysis
- Days between Dates
- Depreciation
- Portfolio Analysis

Graphics Development Library
- Locate
- Scale
- Fix Decimal Places
- Draw X Axis
- Draw Y Axis
- Draw Axes
- Label Axes
- Draw Grid
- Label Grid
- Plot
- Frame

Business Graphics Library
- Pie Chart
- Bar Chart
- Line Chart
- Point Chart

LONGDIN AND BROWNING

Portable Survey Computer — £800

This is an integrated menu driven program suite which is capable of carrying out all normal survey processing, plotting, and computation. It processes and plots detail survey observations from the keyboard, tape or total station storage devices. It computes and adjusts traverses on various projections and spheroids, and computes alignment with offsets and setting out schedules.

At any stage, the user can select whether results are to be displayed on the screen, listed on the microprinter or routed to an external printer. During any program, the user can skip back to the previous menu or input to correct errors or to abort the current function. Both an immediate calculator mode and a batch processing mode are provided. The programs use a data file management system to provide full input and editing features for all observations and results.

Three cassettes containing 19 programs with a comprehensive user manual together with a test data tape are offered for £800. For £1750, a full hardware/software package is offered: HX-20, RX-80 printer, Sanyo auxiliary microcassette recorder.

WARBURTON FRANKI

Warburton Franki is the Epson distributor for Australia. Currently (11/83), the company is considering handling the full range of UK HX-20 software. Contact the company for further details. It also has available the following programs:

Protect — $A40

This program expands HX-20 BASIC to include a protection facility for BASIC programs. When installed, programs saved on microcassette using the 'SAVE file.ext,P' command will cause a 'protection error' if they are loaded and an attempt is made to SAVE, LIST or alter them or LOAD them with the Protect feature not installed.

DBLFUN — $A20

This program expands HX-20 BASIC so that it can perform arithmetic operations on the following functions with double precision:
ATN COS EXP LOG SIN SQR TAN

Tapecopy — $A75

This machine language program will selectively copy files from the microcassette of one HX-20 to the microcassette of another via the RS-232 ports. File and tape counter information is printed on the microprinter of each HX-20. An option allows communication via 300 bps modems. Up to 16 filename specifications, which can include '?' and '*' wildcards, may be entered as selection criteria.

HXGames — $A20

A microcassette with the following games is available:

Blackjack	Poker	Artillery
Match	Bombtank	Bombcity
Follow	Towers	Hangman
Sequence	Tic-tac-toe	Biorhythm
Rifle	Catch	Horse
Walls	Omega	Cavern
Sketchit	Lightcyc	Bandit

TRACE RESEARCH & DEVELOPMENT CENTER

Trace Director Gregg Vanderheiden: '. . . the vast majority of the software being developed for disabled individuals is limited to providing for a special need, rather than allowing the use of common general-purpose software'.

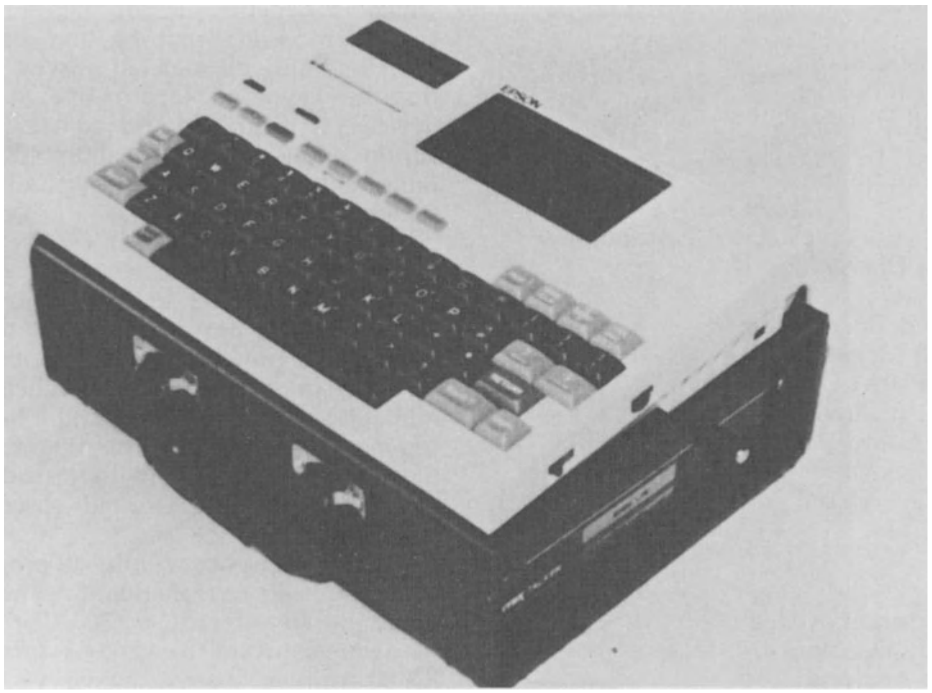


Fig. 11.1 The Portable Voice combines a voice synthesizer with the HX-20

The software being developed at Trace for the HX-20 lets the individual use a shorthand method of typing — a single keystroke, for instance, might call up an entire word. Producing shifted characters by hitting one key at a time is another feature.

Words +

A text-to-voice synthesizer has been combined with the HX-20 to produce a device for persons unable to speak. Input to the Words+ Portable Voice can be either through the keyboard or via special switches. Up to 99 sentences that are often used can be stored and recalled by number. These sentences can be easily changed by the user and can be previewed prior to speech output to ensure that the desired sentence has been selected. The computer can also produce hard copy instead of speech.

A Morse Code program allows the Portable Voice to be used with a single switch of any kind. As an example, Words+ notes that a 14-year-old boy with a high spinal cord/brain stem injury is now using the Morse Code system with an eyebrow switch that Words+ devised. The price for the Portable Voice, which includes all necessary hardware and software, is $1449 for the keyboard version, $1459 for the Morse Code version, and $1499 for the flexible switch version. There is a raft of additional options such as larger knobs for the power switch and/or to hold down the shift key for single-handed operation. The unit comes in a carrying case with a handle.

The Words+ Flexible Portable Voice, according to the company, enables the unit to be used by persons with varying degrees of motor control. This is a switch array that plugs into the computer. The array ranges from a panel with four 5 × 5 in (127 × 127 mm) switches up to one with 64 or 100 1 × 1 in (25 × 25 mm) changed at any time and can represent a single letter or symbol, a word or even a complete sentence. Price is about $350.

Words+ has also developed larger, non-portable systems based on the Radio Shack TRS-80 III/IV for communications, entertainment, appliance control, education, etc. Contact the company for further details.

WORMALD INTERNATIONAL SENSORY AIDS

Viewscan Text System — £3735

Wormald's VTS combines the HX-20 with a large screen portable display and a miniature video camera. This system, for the partially sighted, is designed to aid in reading and typing text.

The palm-sized camera may be hand-scanned over any printed material, generating a bright image on the display. The size of the displayed letters can be adjusted to the needs of the user. Help messages and prompts are presented for guidance.

The components of the VTS are all integrated so that text, for instance, can be created on the HX-20, viewed as it is created on the large screen, and saved to microcassette tape. The entire system is battery powered and easily set up.

Some suggested uses for the VTS are: for students to take notes in class or in the library, for programmers to use as a terminal to another computer system, for secretaries and clerical people to use on the job, for receptionists to access phone numbers and take messages, and for home use to read newspapers or do correspondence.

All of the normal functions of the HX-20, such as running any of the other software in this chapter, are still available. Program output can be directed to either the HX-20 screen or the VTS screen. With the expansion unit, 24K of memory is available for other programs.

The VTS is available world-wide. The US distributor is Sensory Aids Corporation of Chicago.

EPSON AMERICA

Epson America plans on marketing a full range of software for the HX-20 in 1984. These packages will include programs from QuickView Systems, AMPI, and others. As we write this (11/83), software from three vendors has already been released: Kriya, SkiSoft, and Artsci.

Kriya's Learning Lab has been described in Chapter 3, HX-20 BASIC. The company also has a Typing Tutor program. SkiSoft's SkiWriter has been described in Chapter 9, Word Processing.

The 'Personal Productivity' programs from ArtSci Inc. consist of 10 programs on cassette for $12.95, with a 20-page manual. Epson America was nice enough to send these out free of charge to all early HX-20 buyers. A collection of games from ArtSci was sent out similarly. Unfortunately, we have to report that, even when free, these programs are overpriced. After testing out the productivity programs, we declined to even load the game tape, figuring they would prob-

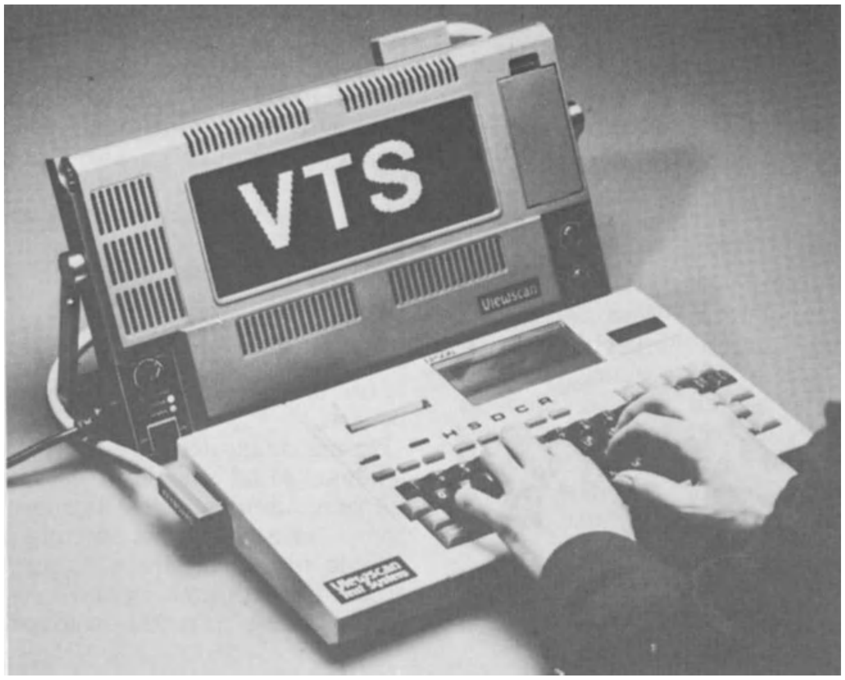


Fig. 11.2 Wormald Viewseam text system

ably be as simple-minded as the productivity programs were.

Calculator program

Four-function calculator: add, subtract, multiply, divide. Uses Reverse Polish Notation, i.e., enter the number and then enter the arithmetic operation.

Digital clock

Displays the time, month, and day of the week. Attempts to display the date, but doesn't always manage it. The numbers are drawn graphically on the screen, which looks nice but takes too long in our opinion. Like most of the other programs in this package, we can't imagine anyone actually using this program.

Event timer

This is a stopwatch program. Set a starting time, then when you stop it you can see how much time has elapsed. Children might like it.

Memo writer

Type a line, hit return or get to the 22nd column, and the line is sent to the printer. Will 'word rapp' [*sic*].

List maker

Just what it says: add things to a list (16 items maximum — 30 characters per item), delete an item from the list, print the list. Great for grocery shopping, if you don't eat much.

Will put items in priority order if you like. The program asks for a priority number before inserting each item in the list. Therefore you should write down the items on a piece of paper first so you know what order they should be in — you can't change the order once they're in the computer. Of course, if you've already written them down on paper. . .

Phone list

Holds up to 16 names and phone numbers — 30 characters total per entry. Can add/replace names, and search for all names that start with the characters you enter.

Amortization/payments

Given principal, interest rate, term of the loan in years, and the number of payments per year, this program will print: payment per month, and for each month, the current interest paid, the accumulated interest paid, the current principal paid, the accumulated principal paid, and the amount of loan remaining. Printer output only. (Hope you don't have a 30-year mortgage.)

Days between two dates

Asks for starting and ending date. Gives you the number of days between those dates, day of the week each of the dates falls upon.

Future value of investment

Given: starting principal, interest rate, compounding term, number of periods to calculate, the number of years in the future to come up with a value for, this program will give you: the future value of the investment times the number of years into the future. Now, this might be useful.

Statistics

Using your choice of frequency distribution or sampling, with grouped or ungrouped (ungrouped: 1 frequency per observation, grouped: multiple frequency per observation) data, this program gives you: mean, variance, and standard deviation. But if you really need this kind of statistical information, you should spend a little more for a better program.

COMPUTRONICS

Computronics has released a set of 'mini-pacs' for computers running Microsoft BASIC, including the HX-20 (so the company says). We have not seen these and have not been enamoured of its software in the past, so *caveat emptor*. Titles available are:

Small Business Accountant	$49.95
Small Business Payroll	$49.95
Small Business Receivables	$49.95
Small Business Invoicing	$29.95
Small Business Inventory	$29.95
Small Business Payables	$29.95
The Income Tax Assistant	$49.95
The Mortgage Calculator	$29.95
The Real Estate Analyst	$99.95
The Financial Calculator	$99.95
Mail Pac 1	$29.95
Index Card Writer	$29.95
Transogram Writer	$29.95
Telephone Directory	$29.95
The Personal Check Register	$29.95
Home Budget Manager	$49.95
Multiple and Linear Regression	$29.95
Arithmetic Teacher	$29.95
Multiple Choice Quiz Writer	$29.95
The Horse Computer	$29.95
Thoroughbred Distance Analyzer	$29.95

OBJEX LTD

Pro-Write

Pro-Write, from the CN*CX group at Objex Ltd, is a computer-aided means of preparing CNC machine tool programs. It allows on-site program editing, with printouts, and data transfer to the machine tool controller. Coupled with a paper tape punch, it can also edit and update NC machine tapes on site.

System CN*10 provides a complete hardware (32K HX-20) and software package for £1200. It is upgradable to the £1600 CN*20 which has more sophisticated software. Various add-on hardware options are also available from the company: video monitor with interface (£250), twin disk drives (£498), plotters, printers, digitizing tablet, acoustic coupler.

There is also a CN*CX time study program that takes as input moves, feeds, and speeds entered by a user from the component drawing. It can then print a summary of the moves made and time taken for each tool operation. The program can further summarize the totals for each tool and output a cycle cost for the part. £50.

PHARMACEUTICAL (CHEMISTS') SYSTEMS

Several programs for pharmacists are available, from P.D.S., Independent Retail Computer Systems, P & M Data Services, Orange Computers, and BPA Computer Systems.

Chemlabel — P.D.S. — £749

Prints out labels giving the chemist's (pharmacist's) name, drug name, etc., produced by code. The chemist chooses from the menu, selecting from 400 drugs already set up in the system, although up to 700 may be stored. Labels are held on a roll holder for ease of use. Price includes die stamp with chemist's name, 1000 pre-printed labels, label roll holder, expansion unit, microcassette, and case.

Pharmaceutical Labelling Package — Independent Retail Computer Systems Ltd (developed by Ffoss Ltd) — £320

This system produces container labels, records orders and communicates with suppliers' machines for placing of orders. Mostly written in assembler, it provides:

- over 800 drug records
- automatic warnings, which may be suppressed when necessary;
- quick entry facilities via the function keys;
- automatic dosages;
- user ability to add, delete, or amend drugs, dose, and warning records to match local needs;
- dose entries used by doctors;
- use of internal or external printers;
- free form drug and dose entry facilities;
- help facility giving guidance to new users;
- uses PIP code for order transmission;
- matching on two or more letters of drug name or on quick match code or on PIP code

All RAM files are controllable. A fast entry system keeps pharmacist's effort to a minimum, printing 3 to 4 labels a minute on the micro-printer. A 'resilient and reliable system, tested thoroughly in practising dispensaries', it is guaranteed for 12 months. Expansion unit required. Support-System help available Monday-Saturday.

Pharmaceutical Labelling Package — P & M Data Services Ltd — £45

This labelling package holds 410 drug descriptions on the 16K machine and 1200 with the expansion unit. Eleven line labels are produced with full details.

Oralabel — Orange Computers — £150

This labelling program can store over 900 drugs, with over 120 coded dosages, and 10 automatic warnings. Features include:

- use of abbreviations to speed up retrieval of stored information;
- visual display of text before printing;
- stock control with reports on monthly consumption, average daily dispensed;
- ability to change any data to suit any requirement;
- calculation of private and retail prices;
- automatic dating;
- written in machine code for speed.

Like other labelling systems, Oralabel works best with pre-printed, self-adhesive labels available from printers like Briggs & Bamforth.

Pharmassist — BPA Computer Systems — £695

This system has a drug file of 1000 descriptions stored. It prints warnings, strengths, dosages, and instructions. Written in machine code for speed. Price includes: 1000 pre-printed labels, label roll holder, microcassette drive and case.

MISCELLANEOUS

The following descriptions are primarily taken from the Epson UK software catalogue. The vendors have not sent us any further information.

Barstocks — by Valldata Ltd — £99

This system incorporates data capture and full stock analysis.

Datacraft/Barstock — by EBOR Computer Services Ltd — £75

The Barstock system allows the landlord or auditor to evaluate wet stock and to compare takings. There is a facility to roll over current closing stocks so that they become open for the next period. Full listings may be made of all stock items. VAT calculations are catered for. Up to 120 items may be stored on the basic 16K machine.

Tapsum — by Time & People — £31.05

This is a calculation package including memory functions, percentage and all the normal calculator functions plus printing.

Currency Conversion — P.D.S. — £149

This program selects which currency is to be bought and then calculates the conversion rates for the day, deducts the agent's commission, breaks down the division of coinage, and calculates the conversion to the currency to be sold. It's designed for use by travel agents all over the world. There is also a weekly analysis of currency stock.

Rocfile V42 Database — ROCON Ltd — £35

This is a live data program using the HX-20 memory to allow access to all the records at once. Used for updating telephone lists, printing address labels. Unique fluid entry facility.

Data Capture — Valldata Ltd — £40

User-definable data recording system, with selectable record and file sizes for use as a batch input program.

MST Portable Filing — MST Consultants — £30

This filing system allows the user to choose card headings, and sorts, adds, deletes and saves them. The time and date are used to index printouts. Alpha or numeric sorts may be carried out on headings.

Database — Gemini Marketing Ltd — £19.50

This database system allows the user to set up files using menu options such as add, change, calculate, sort, find, print and browse. It is possible to set up as many dedicated databases as you wish with as many datafiles as you wish.

Deskdata — Leeds Computer Centre — £40

This is a flexible automatic data recording system. A format program allows the user to set up each individual record. Full mathematical functions are available for each field. Data can be accessed by full or partial key search. This may be used for names and addresses, stock inventory, sales/purchase ledger information or as a card index.

Games — Pocket Computers — £12

Little Brick-Out, Animals, Hangman, 3D Graph.

Big Bang — Time & People — £20.70

This is an educational game involving skill in extinguishing sticks of dynamite before they blow up.

Epcalc Financial Planning/Modelling — Healey Management — £20

This is a financial modelling package for 'What If?' situations.

PS Investor — Practical Software — £20

This is a tool for the professional as well as the lay investor for tracking and analysing an investment portfolio.

Golf Handicapping — P & M Data Services Ltd — £45

This package handles 250 members' records, expanding to 950 handicap records. It is menu driven and automatically updates and maintains records required by the new standard scratch handicapping scheme.

Golf Handicapping — EBOR Computers Ltd — £75

Up to 200 members' records may be accommodated using this program. Designed for the golf club secretary to maintain records as required by the standard scratch scheme.

PQS — BPA Computers Ltd — £160

PQS is the Personal Quotation System designed for the life assurance and pension agent or broker to assess and display a client's requirements. A variety of compound interest and income tax calculations may be performed.

Mailing List — Gemini Marketing Ltd — £19.50

A system incorporating search and 10 selection parameters. Labels may be printed in a variety of formats.

Mini-Mainopt Maintenance Optimization — C.M. Mainopt Ltd — from £510 to £1310

This is a powerful tool for evaluating the financial impact of maintenance decisions. Mini-Mainopt can be used to calculate operating costs, energy consumption, and many other factors. The results are printed as graphs and in numerical form.

The Navigator — Microtek Ltd — £95

A comprehensive navigation package for small craft. Log-on enters details of crew, destination, etc. Log entry gives input of current details: log reading, compass course etc., and calculates dead reckoning position. Full auto-correction for deviation and tidal streams. Fixes by several methods. Course calculations and racing details, VMG true wind, etc.

PAYE — Pocket Computers — £45

A filing and calculation system for the maintenance of the PAYE tax scheme. Wage slips and working sheets are printed. Running totals for the end of year returns are maintained.

COMTIME — Healey Management Services — £95

COMTIME allows the work study engineer to carry out his study, input RA and CA element frequency and do all the necessary editing. It produces an immediate printout of the study with full elemental description and the standard minute value.

BITWeigh Portable Weighing System — Berisford Information Technology — £1700 (hardware + software)

BITWeigh is a portable production measurement and reporting tool to conform to the requirements of Weights and Measures. It stores details of all products and statistical records about packing plants, taking periodic weighing samples and entering them directly into BITWeigh's memory. All the weights are printed out, negative tolerance values are highlighted, and one can choose the level at which samples are rejected.

12

PERIPHERALS

'The great thing about standards is that there are so many of them.'

This chapter covers:
Different peripherals available: what they do, how they connect, where to get them

ABOUT PERIPHERALS

The beauty of a portable, and particularly the HX-20, is that everything you need is contained in one small package. Well, almost everything. There are some additional functions that can be added to the HX-20 that may be worth considering, depending on your needs. The trade-off is that you'll probably be sacrificing some of the portability of the unit.

Most of the following peripherals are designed to operate off the RS-232 port. However, the high-speed serial port on the HX-20 uses signals with a voltage within the permissible RS-232 range. This means that with a little tinkering, a cable, and some software, an RS-232 device should work on this port.

This high-speed serial port is the 'other' DIN connector on the back of the HX-20. Were you to hook up the Epson floppy disks on the TV adaptor, it would connect here. It can also be used as it is for high-speed data transmission between HX-20s (using Epson cable #717), with the proper software. The software is key though, as the serial port cannot be controlled via BASIC alone.

The Serial Port

Description	DIN connector
Signal Ground (SG)	1
Set Transmit Mode (POUT)	4
Transmit Data (PTX inverted)	2
Set Receive Mode (PIN)	5
Receive Data (PRX inverted)	3
Frame Ground (FG)	E

(Above are in order clockwise)

Also, keep in mind that, unlike other computers, the HX-20 has no parallel port — it can only transmit data a bit at a time.

Note With many of the following devices, the user is responsible for providing his own software to 'drive' the device.

The information that follows has been obtained from vendors. We have not reviewed these products. Addresses of all companies mentioned can be found in Appendix A.

TV INTERFACE

HX-20 BASIC provides a number of statements designed to be used in conjunction with a color CRT. This overcomes the disadvantage of the HX-20's small screen size, but at the expense of portability. However, once you've hooked up the TV and the disk drives, you have a true office computer (with not as much memory, though, as other computers designed for the office).

Oval (UK) manufactures a color display controller for the HX-20. This unit is intended only for TV systems on the PAL Standard (not the US). According to Oval, the unit is fully compatible with all of the display commands available from HX-20 BASIC. The input comes off the HX-20's high-speed serial interface. (The Epson TF-20 disk can be daisy-chained off the controller.) There are two outputs: UHF on channel 36 and direct video via the video recorder socket. The text display is 32 columns by 16 lines. The graphics display is 128 × 64 pixels in 4 colours or 128 × 96 pixels monochrome. (For comparison purposes, an Apple II display is 280 × 192 pixels, a TRS-80 Model III is 128 × 48.) Power requirement is 110 or 220 or 240 volts. Size (HxWxD) is 50 mm × 155 mm × 215 mm. Weight is 1.5 kg. Price is £150. Information on this unit was supplied by OneStop and by the HX-20 Users' Group.

T.L. Ronson of the HX-20 Users' Group also notes that a monochrome monitor can be hooked directly to the RS-232 port of the HX-20. Contact him for details.

In the US, Epson America plans on bringing out a TV adaptor, but no details are available as yet. We would suspect, however, that specs will be similar to that of the Oval unit.

SPEECH SYNTHESIZERS

If you've seen the movie *War Games*, you've heard a sample of speech output from a computer. This is a technology, unheard of a few years ago, which can now be purchased by anyone. It's still a young technology, though, and the user is mostly on his own in doing anything with it.

Originally, computer hobbyists took the Texas Instruments 'Speak 'N Spell' educational toy and hooked it up to the computers. Now, the most popular speech synthesis device is probably 'Type-'N-Talk' by Votrax. This box, which can be added to any computer with an RS-232 port, allows the user to enter English text and hear electronic speech. Not for novices. $299. Requires AC power.

A similar unit is the Echo GP, from Street Electronics. Another company in this market is Centigram Corp. Unless you're an engineer, beware of low-cost devices that require the user to do his own hardware interfacing, in addition to writing his own software.

For a packaged speech synthesis system for the handicapped, check the description of Words+ in Chapter 11, Software and Systems.

HOME CONTROL SYSTEM INTERFACE

Controlling appliances and other electrical devices in the home does not, at first sight, sound like an application for a portable computer. But the HX-20, because it is so self-contained, can actually be a very cost-effective way to do it.

The Heath Company's Heathkit catalogue shows one such device: an interface to connect the BSR home control system to a computer via an RS-232 interface. $130. AC powered.

Circuit Science has its own controller box which can accept ASCII codes directly from the HX-20's RS-232 port. This allows the HX-20 to talk directly to the BSR module that operates the appliance. The controller is $170; BSR modules extra.

Contact HyperTek for a complete, programmable, full-control system for $1400. With this system, you can even dial in while on vacation and get a status report.

VIDEODISKS/VIDEOTAPES

Many videodisk players and videotapes can be hooked up to computers via an RS-232 interface, using an adaptor. These adaptors are advertised in computer hobbyist magazines at prices starting at $300. (Adaptors requiring a parallel interface can still be used with the HX-20, by purchasing a serial-to-parallel converter.)

For most video equipment, these adaptors provide computer control over the same functions available on the VCR's front panel or remote control. For videodisk players, this includes random access.

There are a few commercial applications available using this technology, but by and large it is very new and has not yet found a wide market. Prospective users should be prepared to do their own development.

MODEMS

See Chapter 8, Communications.

PRINTERS

There are all kinds of printers on the market. There are probably as many different printer models as there are different computer models. So, a discussion of their respective merits is really beyond the scope of this book (especially since we've found only one that is battery-operated and thus really suitable for a portable). We did talk a little about fully-formed versus dot matrix in Chapter 1, Which Computer? and that will have to do. We even hesitate to name some specific models for you to check out, because we will (justifiably) be accused of slighting equally capable models. Well, we'll give you one name anyway.

Computer Peripherals markets a 160 cps dot matrix thermal printer with graphics, automatic autowrap, 40 column printing, RS-232 interface. Its dimensions are 4 × 10 × 2 in (102 mm × 254 mm × 51 mm), approximately. That makes it a good size for a portable. Unfortunately, like all but one of the other printers on the market, it requires AC power. Price is $145.

The one battery-powered portable printer we've heard about is a new unit from Teletex (TTX). It's a thermal printer, 40 cps, with 132 column width.

PRINT SPOOLERS

An increasingly popular peripheral found connected to more and more printers these days is a separate printer spooler or buffer. This is a box that installs between your computer and your printer. It takes the data to be printed and feeds it to the printer when the printer is ready for it. Normally, your computer would have to wait for the printer — because printers are very slow compared to computers. This way, it's the spooler that waits, while your computer is free to do other things.

Print spoolers were originally simple devices with few or no controls. And these still exist. But they have been joined by others that allow multiple copies and editing. Prices range from $150 to $500. Most computer dealers can demonstrate at least one model.

PLOTTERS

Picture a human hand making a drawing. A plotter is a device designed to replicate that activity, though with far more accuracy in placing the lines on the paper. It's this accuracy, measured in thousandths of an inch, that gives plotters their big advantage over graphics printers.

Plotters can draw on a flat sheet of paper (flatbed plotters) or on paper that rolls backwards and forwards under the pen (drum plotters). The cheapest plotters have a single pen that moves back and forth over the paper under computer command. To get multi-color drawings, all the lines that are to be in one color are drawn first, then the pen is manually changed and the lines of the second color are drawn. Some of these plotters have add-on attachments to store pens of other colors. Upon command, the pen holder goes over to the rack and swaps pens. Houston Instrument is a leading manufacturer of this type of plotter. Its machines are sold in many computer stores, with prices starting under $1000.

In another type of plotter, the pen holder can revolve to position a different color pen (up to four colors) over the drawing. One of these, made by Yokogawa Corporation, is relatively small and lightweight (13 lb, i.e., 6 kg).

TERMINALS

If you think printers and computers are ubiquitous, check out terminals. A terminal has traditionally been a device to let you operate a computer from a location other than the one where the computer is. Terminals typically consist of keyboards and displays, and can be wired into the computer or used via dial-up facilities. In essence, we've already talked about terminals in Chapter 8, Communications: we've used the HX-20 as a terminal to access other computer systems.

But you can also use another terminal to access the HX-20. Talbot Computers' Dialtext system, for instance, uses one HX-20 to transmit files to another HX-20. But you can go beyond file transfer. With the right software, you could operate an HX-20 remotely. The HX-20 can run a program that will listen on a phone line and then take particular actions depending on what data came across that line, just like a large timesharing computer would do.

As an example, let's say the HX-20 could be recording data in memory or on tape. You could call it up and ask it to transmit back to you all the data it has recorded up to that point. You'd need an auto-answer modem on the receiving side and an ordinary modem on the sending side. And you'd also need a terminal to talk to the HX-20. This terminal could be anything that can talk over a phone line, including another HX-20.

DISK DRIVES

Floppy disk drives are in widespread use as a medium of storage for desktop microcomputers. For battery-powered portables, they are somewhat less useful. The amount of power required plus the weight makes them more suitable for office use.

Programs and data can be stored in another computer and transferred back and forth from/to the HX-20 by means of an RS-232 connection. But a floppy disk attached directly to the HX-20 offers an advantage in convenience. There are also disk functions built directly into HX-20 BASIC that can be taken advantage of when a direct connection is available.

Epson (Japan) manufactures a dual double-sided, double-density unit, called the TF-20, that is expected to be available for sale world-wide in 1984. These disks are minifloppies (5.25 in) (133 mm), with 40 tracks/side (48 TPI), with 16 256-byte sectors/track. That's how they look physically; logically (to the computer) the internal disk software uses the cylinder concept: 2 tracks/cylinder, 64 128-byte sectors/cylinder. However you look at it, total formatted storage is in the neighbourhood of 650 Kbytes. Power requirement is said to be 40 W (at 220 V), temperature range is 5 to 28 °C (41 to 82 °F), size (HxWxD) is 6.5 in × 4.7 in × 13.8 in (165 mm × 120 mm × 350 mm). Connection is via the HX-20's high-speed serial port. According to One-Stop, the TF-20 can reportedly be used simultaneously with the TV hook-up.

The TF-20, comes with a disk BASIC that is loaded into the HX-20's RAM. This BASIC adds additional features to the normal HX-20 BASIC.

As we write this, Epson has just begun manufacturing a 3.5 in (90 mm) battery-powered disk. This disk will be sold to OEMs and is not planned for any present Epson products. However, it may turn up in a future Epson portable and perhaps, if there were enough user interest, it might be brought out as an add-on for the HX-20.

One thing to note The serial interface does not transfer data as quickly as the parallel interface on most microcomputers: 38.4 Kbps v. 256 Kbps. This is still fast enough for many applications, but keep it in mind.

Another disk system is that from Analog & Digital Peripherals. For $1050, the company has a system that includes two 5.25 in (133 mm) double-sided, double-density drives (2M unformatted). This AC-powered unit connects via the RS-232 port. Systems with less capacity are available at lower cost, according to the company.

A company making RS-232 connectable disks but with a different approach is Western Telematic Inc. Its $2195 Datamate II unit is an intelligent device with built-in editing and searching, with a formatted capacity of 328 Kbytes. A 164K version is $1995. From the same company, the $1735 Minimate II does not have quite as many features, but still retains character editing, flow control, 110–9600 bps operation, single searches, 328K capacity. A 164K version is $1535.

While looking at these prices, it may have occurred to you that it would be just as cost-effective to buy a complete disk-based computer. Nearly any system that expected a console to be connected via an RS-232 port would work with the HX-20. In this type of configuration, a communication program would run on the HX-20 allowing the HX-20 to talk interactively over the RS-232 port. A 'smart' terminal program, such as the one in Chapter 8, Communications, could transfer data from memory or tape into the other computer. Several CP/M computers with disks but without consoles, such as the Morrow Micro Decision, sell for under $1500. Actually any cheap computer with disks and an RS-232 port, even an Atari 400/850, could serve as auxiliary storage for the HX-20.

TAPE SYSTEMS

Need additional online tape storage? Or want a faster tape? Or want to make tape-to-tape copies? Or need to have two tape drives online at the same time? A number of manufacturers sell microcassette systems easily interfaced to the HX-20 via the RS-232 port. One such company is Analog & Digital Peripherals Inc. (ADRI), which has several units, both DC and AC powered, at prices starting from $540. Another manufacturer is Techtran Industries.

A different type of system is manufactured by Exatron. It is called the 'stringy floppy', and uses wafer-shaped cassettes with high-speed access.

If you need much more storage than microcassettes or wafers can give you but still want to opt for tape rather than disk, then consider cartridge tape drives. These units are not inexpensive and require line power. But they can hold many megabytes.

For $2895, ADPI has a tape cartridge system which can hold up to 5 megabytes. For $4250, two drives are hooked together, providing 10 megabytes of storage. Connection is via the RS-232 interface. Transfer speed runs from 300 to 38,400 bps. While the maximum speed on the

RS-232 port is 4800 bps, the top speed on the high-speed serial port is 38,400 bps. As we noted earlier, it may just take a bit of custom cabling and some software to make the hook-up.

Some other companies in this market are Advanced Digital Information Company and Digi-Data Corp.

For even more storage — or just for compatibility with minicomputers and mainframe computers — you can even hook up a full-blown, vacuum-type, reel-to-reel tape drive. Black Hole Technology and IBEX make interfaces that will connect off the RS-232 port.

BAR CODE READERS

See Chapter 10, Inventory/Stock Tracking.

MEMORY

The typical way to add additional memory is via Epson's expansion unit. This is described in Chapter 2, The HX-20.

There are a couple of other possibilities for adding additional read–write memory. Analog & Digital Peripherals has a bubble memory unit that will store 500K bytes and uses an RS-232 interface. Western Telematic has a 16K editing buffer attachable to an RS-232 port. This $650 buffer has a numeric keypad for command entry. Transfer rate is 75–9600 bps.

A cross between internal read-only-memory and external storage like tape cassettes is the ROM cartridge. This cartridge, manufactured by Epson (Japan) and available from dealers like Transam Microsystems, plugs into the spot normally taken up by the microcassette drive. It is the same size and shape as the drive and is powered off the HX-20 batteries.

The ROM cartridge is often referred to in Epson technical notes as a 'PROM cassette'. This is because the 8K ROM held in the cartridge is not part of main memory. Its contents must be read into storage serially, one bit at a time, much as the contents of a tape cassette must be read into storage. The advantages of the cartridge over tape are much improved speed of loading, improved reliability, less power drain, and less susceptibility to damage. The price (from Transam Microsystems) is £45.

MULTIPLEXORS

A Western Telematic unit allows up to four devices to share one RS-232 port, individually selectable to 300 or 1200 bps: $295. A unit with additional features, such as software selection of a port, speeds from 75 to 19.2 Kbytes, ACK/NAK handshaking, etc. is $395.

Bay Technical Associates has a similar unit, allowing four devices to share one RS-232 port, each with individually settable configuration parameters. Buffering is an option. Advanced Systems Concepts is another company in this business.

Simpler units are available from many computer dealers. If you have trouble locating them, try the Black Box Catalog or, in Australia, Cable-Tectronics.

PARALLEL INTERFACES

Some peripherals on the market — some printers, some plotters, some laboratory equipment, etc. — only operate off a parallel interface. While Epson doesn't directly support the use of such a device, the HX-20 can be configured to communicate with them.

The most widely advertised way to add a parallel interface to a computer that doesn't have one is to convert the signal off a serial port. This is often included as an optional feature on print spoolers, though, in that case, the data only goes in one direction: from computer to peripheral. Comsolink is one company that makes these; there are many more.

Print spoolers are more expensive than simple converters because they include memory. If what you want is serial to parallel conversion and don't need this memory, check the products sold by Binary Devices and The Black Box Catalog. Another source is JC Sytems, which has a programmed device to control 1–2 16-bit parallel ports off an RS-232 port. Devices can be cascaded to provide up to 100 ports.

Transam Microsystems manufactures a unique parallel interface box for the HX-20. This THX-01 connects to the memory expansion interface, rather than to one of the serial ports. (If you have an expansion unit already fitted, you'd disconnect it, plug the Transam connection cable in, then plug the expansion unit into a special socket in the THX-01.)

The THX-01 allows the user to operate an external printer at the same time that the RS-232 port is tied up doing something else, such as communicating with a modem.

The THX-01 provides:

- one 8-bit bi-directional port with control signals;
- two 8-bit output ports with output enables;
- two 8-bit input ports with latching ability.

By using CMOS chips the power requirements of the THX-01 are kept low (0.1 mA). It can be powered directly off the HX-20's batteries or off any other 5 V supply. The user manual, while quite short, provides the necessary details for the unit to be put to use: description of the 50 output pins, memory addresses to use for I/0, customization suggestions, electrical schematics.

For users with Transam's ITE+ ROM, no additional software is necessary. Others would need to supply their own software. Users are also responsible for supplying their own cables, though Transam may stock cables for popular devices. The price for the THX-01 is £85, exclusive of freight and VAT.

DATA ACQUISITION/CONTROL

Innumerable electronic devices exist to measure 'real-world' events. These devices include temperature sensors, potentiometers, radiometers, probes of all types, etc. Unfortunately, they were not designed to feed data into a general purpose computer like the HX-20. But it's possible to take that data and convert it into a form that the HX-20 can use. If the data is in analogue form, than an analogue-to-digital (A–D) converter is necessary. But even digitized data can't be read by the HX-20 unless it meets certain requirements (e.g., RS-232 voltage levels), so further conversion may be necessary. But once that conversion has been done, the HX-20 becomes a very useful data collection, monitoring and control device.

DarkHorse by Breakthrough ($995) is one such conversion device. Functions include analogue-to-serial conversion, parallel-to-serial conversion, control outputs that can switch relays, wake-up timer to switch on/off the HX-20. Of particular interest to HX-20 users is the fact that the DarkHorse is battery-powered (50 hours on alkaline batteries) and lightweight (3 lb or 1.36 kg). The user manual includes programming instructions. Additional software assistance is available from the vendor.

Model TR ($1500) by Panasonic is a basic data collection system designed to gather time and attendance data. It incorporates an optical badge reader and an RS-232 interface. Model SA ($1650) is a stand-alone system, incorporating a printer. Model DS 500 ($2400) can store and transmit data originated by a number of multipoint data entry terminals.

Starbuck Data has a less expensive unit, for $540, that will hook up directly to the HX-20's RS-232 port, receiving ASCII instructions via that connection. The Model 8232 provides 24 channels of analogue and digital data acquisition and digital output. The 8232 can store data for later transmission. Its communication port can also be remotely disabled, which allows the HX-20 to share its single RS-232 interface among several devices.

Starbuck reports that some of the applications currently in use are home control, computer control of an automobile, data acquisition/ control in a nuclear power plant, industrial quality control, and sports physiology.

Starbuck's 8232 uses 9 volt DC power. The user has 2K of RAM in the 8232 for his own machine language programs and data storage. Machine language programs for the 8232 can be developed on the HX-20, as the 8232 uses Motorola's 6802 processor, a close cousin of the Hitachi 6301 used in the HX-20.

Other companies with various data acquisition devices are: Measurement Equipment Corp., Niagara Scientific, Taurus Computer Products.

MORE ON DATA ACQUISITION: SPECIFIC DATA COLLECTION DEVICES

For those who have to use paper tape, our sympathies. For paper tape readers and punches that can be hooked up to the HX-20 via the RS-232 port, try Addmaster Corp., Facit Inc., GNT Automatic, and Trend-DLC. All units are AC powered.

For punched card readers, contact: Cardamation.
For OCR, try Hendrix Electronics and Key Tronic Corp.
For mark sensing equipment, try Input Business Machines and True Data Corp
For MICR, try Input Business Machines.
For badge readers, try Peripheral Dynamics and True Data Corp.
For a camera interface, try Dunn Instruments.

CARRYING BAGS/CASES

The Madson Line is a maker of soft luggage for computers, terminals, and printers. Its bag for the HX-20 is made of tearproof, water-repellent cordura nylon with sewn-in high-density foam panels, velcro openings, outer pockets, leather grip, adjustable/removable shoulder strap. Price is $45.

The Chip-Tote, by Kangaroo Video Products, is a carrying case that doubles as a desk. It's a foam-padded, cordura nylon bag that opens up into a one-piece work station. Features include a zippered pouch, shoulder strap, and hand strap. $59.95.

Other companies with carrying cases are American Tourister and Computer Case Co.

13

OPERATING TIPS

This chapter covers:
Taking back-ups
Keeping tape directories
Travelling with the HX-20

TAKING BACK-UPS

Here are three important tips in operating your HX-20.

1. Take back-ups
2. Take back-ups
3. Take back-ups

The HX-20's microcassette drive is more reliable than an audio cassette player, but it still has problems, especially with cheap tapes. Unlike disk-based systems, there is currently no utility program that can read your file and pull off the good parts, leaving you with the job of only having to re-create the one block that was damaged.

If any part of a BASIC program on tape is damaged, then all of it is lost. The BASIC interpreter will cheerfully say 'IO Error' and not give you any of your program back, even if the error occurred on the very last block of the file. You can make out better with data files because you can read (and save) everything up to the error. (You'll have more trouble reading the file after the bad spot, even if you try to WIND past it. See Chapter 4, Using and Writing BASIC programs, for an explanation of why.)

So, the best way is to make frequent copies of important data and programs. For programs, you can just SAVE or SAVEM to a different tape. For data used by a program, the program should allow the data to be read in all at once and written back out to another tape. If it doesn't, you can write a short BASIC program to do it. (OPEN, INPUT, INPUT, . . . CLOSE, OPEN, PRINT#, PRINT# . . . CLOSE)

Here's one procedure for back-ups that could be used. It should be equally applicable to disk systems.

This procedure is based on the fact that you are likely to have three types of files:

1. Program tapes, which you need always but which rarely (if ever) change.
2. Data tapes which change frequently.
3. Data tapes which are not used currently, but which you need to save for historical purposes

For category 1, make a back-up copy now. Take another back-up only when a change is made.

For category 2, make back-ups on a regular schedule, such as daily, or any time major changes have been made. When data reaches a pre-defined point, treat as category 3.

For category 3, make one more copy and store separately.

For 'Work in progress' (category 2), the following three-tape back-up cycle could be used:

1. Take your current data tape, which we'll call tape A and make a back-up copy of it on tape B. Store A and use B as your current tape.
2. When it's time for another back-up, create tape C as a copy of B. Use C and store B.
3. For the next back-up, re-create tape A as a copy of C. Use A and store C.
4. The next back-up will re-use B as a copy of C, using B and storing C and so forth.

Note that we always switch to our new copy as our current work tape. This way, we'll know right away if there's anything wrong with the copy. If we had just stored the new copy away each time, we might not find out till we had to use the back-ups that they were no good.

If you follow this procedure, you'll always have one copy of work done in the previous work period and one copy of the work done in the work period before that. You can expand this system to use as many tapes as fit your work cycle. If you work a five day week and take a back-up at the end of the day, then a six-tape system would work nicely. For more critical applications, make two copies of each tape at back-up time.

It's best to take the back-ups on a regular schedule. It's human nature not to do something that seems to have no immediate benefit. By making it a habit, you'll increase your chances of having that back-up around when you need it.

One final note on the topic of bad tapes. If you have consistent problems reading older tapes or commercially prepared tapes, you may have a problem with the HX-20's internal electronics. We noticed, for instance, that our two HX-20's could each read and write their own tapes, but could not read each other's tapes. Even when we pulled the microcassette drive out of one and put it into the other, we still had a problem. So, keep at least one 'test' tape around so that when you have a 'tape' problem, you can verify that it is not in fact a hardware problem.

KEEPING TRACK OF WHAT'S ON EACH TAPE

Epson America, in its BASIC books, supplies a program that will create a file directory on the front of each tape. Every time you create a file, you re-run the directory program and the directory will be updated. This kind of thing might work OK with disks, but the speed of tape access is such that pencil and paper is a much more efficient means of keeping track of tape files.

It doesn't take much time to keep a written log of what's on which tape. One method is to assign each tape a number, writing the number right on the cassette (not on the cassette box or anything else that might become separated from the tape). Then, when a file is created, write down on a separate log the name of the file, the starting/ending addresses, and the tape number. You can keep the file names grouped by function or grouped by tape number. Either way, you'll know quite quickly where the file you want is located. You'll also know, if you group the files by tape number, which tape has room for more files even though nothing except the tape number actually gets written on the tape.

If you don't mind having a lot of tapes on hand, use a new tape for each file. Unlike disks, there is nothing physically written on the tape that tells the machine how far down in the tape it has gone (unless you're using RAX). The WIND routine is related to how much the spindle has turned — and nothing else. Thus if the tape slips on the spindle, the tape counter will get thrown off. The best way to avoid this problem is to start every file at the beginning of a tape.

If you have a problem finding a file, try searching for it earlier than the tape count where it's supposed to be — just to account for any positioning errors that may have cropped up.

TRAVELLING WITH THE HX-20

With some aid from American Tourister, we can offer several tips on selecting a case for a portable computer:

1. Don't get an instrument case. They usually look as if they hold valuable equipment and so are often targets for thieves.
2. Get a case that's the right size for the computer, rather than one that was made for something else.
3. Get a hard-sided case if you want greater protection against theft and damage.
4. Get a soft-sided case if you want convenience and light weight.
5. Get a waterproof case.
6. Get a case with comfortable handles.
7. Get a case that has pockets for papers, cassettes, this book, etc.
8. Get a case that seems to be well constructed.
9. Don't leave the case where there are temperature extremes, such as in the boot of your car.
10. Place permanent identification, such as an engraved Social Security number, on both the HX-20 and on the case.

11. Use door-guard locks in hotel rooms.
12. Get insurance coverage. Save the serial number of the machine. (Columbia National is one agency that advertises insurance on computers.)

APPENDIX A: VENDOR LIST WHERE TO GO TO BUY SOFTWARE, HARDWARE, SUPPLIES

ALL ADDRESSES ARE IN THE USA EXCEPT WHERE NOTED

Addmaster Corp.
416 Junipero Serra Drive
San Gabriel, CA 91776
213/285-1121
Paper tape readers

Advanced Digital Information Company
723 9th Ave, Bldg A
Kirkland, WA 98033
800/638-0818
Cartridge tape drive (RS-232)

Advanced Systems Concepts, Inc.
435 N. Lake Avenue
Pasadena, CA 91101
213/793-8979
RS-232 switching devices

Altek (CW)
1 Green Lane
Walton-on-Thames
Surrey, England
0932 244110
Bar code reader (RS-232)

American Micro Products, Inc.
705 North Bowser
Richardson TX 75081
214/238-1815
Software, including financial, engineering

American Tourister Inc.
Special Products Division
91 Main Street
Warren RI 02885
Soft-sided and hard-sided carrying cases

Analog & Digital Peripherals, Inc.
815 Diana Drive
Troy, OH 45373
513/339-2241
Tape and disk units

A.P. Systems Ltd
90-100 Brighton Road
Surbiton, Surrey
England
01 399 1257
Software, including mailmerge

Automated Time Inc.
1008 140th Avenue NE, Suite #102
Bellevue, WA 98005
206/643-5558
Legal time reporting system

Avocet Systems, Inc.
804 South State Street
Dover, DE 19901
302/734-0151
6801 Cross-assembler, EPROM programmer

Bay Technical Associates
PO Box 387
Bay St Louis, MS 39520
601/467-8231
Smart RS-232 switch

B & B Electronics
Box 475
Mendota, IL 61342
815/539-5827
Low-cost RS-232 testers, taps, adaptors

Berisford Information Technology
Berisford House
King Street
Middlewich, Cheshire
England
0606 845611
Portable weighing system

Binary Devices
11560 Timberlake Lane
Noblesville, IN 46060
317/842-5020
Serial-to-parallel converters

BiTech Enterprises
10 Carlough Road
Bohemia, NY 11716
516/567-8155
Bar code reader

Black Box Catalog
PO Box 12800
Pittsburgh PA15241
412/746-2910
Expensive, but extensive line of communications gear

Black Hole Technology
225 East Street
Winchester, VA 01890
617/721-7690
9 track tape drives (RS-232)

B.P.A. Computer Systems
London House
14C High Street
Buntingford, Herts
England
0763 73056
Software, including life assurance

Breakthrough, Inc.
PO Box 230
Logan, UT 84231
Data acquisition systems

BRS/After Dark
1200 Route 7
Latham, NY 12110
518/783-1161
Information retrieval service

Cable-Tectronics
PO Box 126
Brookvale 2100
Australia
(02) 938 5211
RS-232 switches

Cardamation
PO Box 746
Frazer, PA 19355
215/647-8260
Punched card readers (RS-232)

Circuit Science
4 Townsend West
Nashua, NH 03063
603/880-4066
Appliance control (RS-232)

C.M. Mainopt Ltd
139 High Street
Wootton Bassett
Swindon, Wilts
England
0793 853000
Software, including maintenance optimization

Columbia National General Agency
88 East Broad Street
Columbus, OH 43215
Computer insurance

COMET
BL Systems Ltd
Grosvenor House
Prospect Hall
Redditch, Worcs
England
0527 28515
Electronic mail

Consolink Corp.
1840 Industrial Circle
Longmont, CO 80501
303/651-2014
Print spoolers

CompuServe
5000 Arlington Centre Boulevard
Columbus, OH 43220
614/457-8600
800/848-8990
Information retrieval/time-sharing service

Computer Case Co.
5650 Indian Mound Court
Columbus, OH 43213
614/868-9464
HX-20 carrying case

Computer Peripherals
1117 Venice Boulevard
Los Angeles, CA 90015
213/298-1297
800/854-7600
Portable printers

Computronics
50 North Pascack Road
Spring Valley, NY 10977
914/425-1535
Software, including accounting

Computype
2285 West Country Road C
St Paul, Mn 55113
800/328-0852
612/633-0633
Prints bar code labels

Cuadra Associates
2001 Wilshire Boulevard
Suite 305
Santa Monica, CA 90403
213/829-9972
Database directory

Dataccount Inc.
Box 14706
Portland, OR 97214
503/232-0490
Software

Data Composition
1099 Essex
Richmond, CA 94801
800/227-2121
415/232-6200
Prints bar code labels

Digi-Data Corp.
8580 Dorsey Run Road
Jessup, MD 20794
301/498-0200
Tape systems (RS-232)

Dow Jones News Retrieval
PO Box 300
Princeton, NJ 08540
609/452-1511
800/257-5114
Information retrieval service

Dunn Instruments
544 Second Street
San Francisco, CA 94107
415/957-1600
Camera interface (RS-232)

Duplex Communications Ltd
2 Leire Lane
Dunton Bassett
Near Lutterworth
Leicestershire LE17 5JP
Low-cost RS-232 analysers, adaptors

EBOR Computer Services
5 Regent Buildings
Acomb, York
England
0904 791595
Software, including barstocks

Engineering Specialties
1501-B Pine Street
PO Box 2233
Oxnard, CA 93030
Serial-to-parallel converter

Epson America Inc.
3415 Kashiwa Street
Torrance, CA 90505
213/534-0360
Natl tech support: 800/421-0184
HX-20 Marketing: 213/539-9140

Epson (UK) Ltd
Dorland House
High Road
Wembley, Middlesex
England
Freefone Epson 2730, or 01-902 8892

Exatron Corp.
181 Commercial Street
Sunnyvale, CA 94086
408/737-7111
Wafer tape drives (RS-232)

Facit Inc.
66 Field Pt Road
Greenwich, CT 06830
203/622-9150
Paper tape reader (RS-232)

Ffoss Ltd
112 Bath Road
Slough, Berks
England
0753 820277
Software, consulting, training

Gemini Marketing Ltd
9 Salterton Road
Exmouth, Devon EX8 2BR
England
03952 5832
Software, including data management

Global Computer Supplies
9133 Hemlock Drive
Hempstead, NY 11550
800/645-6393
RS-232 equipment and miscellaneous supplies

GNT Automatic
1560 Trapelo Road
Waltham, MA 02154
617/890-3305
Paper tape equipment (RS-232)

Graphnet
8230 Boone Boulevard, Suite 330
Vienna, VA 22180
800/336-3729
703/556-9397
Telex/TWX/Mailgrams

GTCO Corp.
1055 First Street
Rockville, MD 20850
301/279-9550
Digitizer

HBT Associates
15 Downing Drive
Covington LA 70433
504/892-4045
Manufacturer's rep for RS-232 devices

H.C.C.S. Associates
533 Durham Road
Low Fell
Gateshead
Tyne & Wear NE9 5EY
0532 821924
Software, including Forth, Pascal

Healey Management Services
14 Hemmells
Laindon North Trading Centre
Basildon, Essex
England
0268 416155
Software including financial modelling

Heath Company
Benton Harbor, MI 49022
616/982-3411
800/253-0570
Home controller interface plus other electronic devices

Hendrix Electronics
670 North Commercial Street
Manchester, NH 03101
603/669-9050
OCR (RS-232)

Houston Instrument
8500 Cameron Road
Austin, TX 78753
512/837-2820
Plotters

HyperTek
PO Box 137
Rt 22 East
Salem Industrial Park
Whitehouse, NJ 08888
201/874-4773
Home control (RS-232)

IBEX Computer Corp.
20741 Marilla Street
Chatsworth, CA 91311
213/709-8100
IBM type reel-to-reel tape drives (RS-232)

Independent Retail Computer Systems Ltd
Emery Down
Lyndhurst, Hampshire
England
042 128 2452
Software, including pharmaceutical

Inmac
2465 Augustine Drive
Santa Clara, CA 95051
408/737-7777
Communications/computing supplies, custom cables

Input Business Machines Inc.
1370 Piccard Drive
Rockville, MD 20850
301/869-9222
OCR, MICR, Mark Sense (RS-232)

Interface Solutions Inc.
PO Box 11167
Memphis, TN 38111
901/372-3764
Bar code reader/software

IQ Technologies Inc.
11811 NE First Street
Bellevue, WA
206/451-0232
'Smart' RS-232 cable

Janus Manufacturing Co.
PO Box 4004
Grand Junction, CO 81502
RS-232 interfacer

JC Systems Inc.
4360 Viewridge Avenue
San Diego, CA 92123
619/277-6585
Serial-to-parallel conversion

Kangaroo Video Products
9190 Manor Drive
La Mesa, CA 92041
619/698-0230
Carrying bag

Key Tronic Corp.
Building 14
Spokane Industrial Park
Spokane, WA 99214
509/928-8000
Data entry equipment, including OCR (RS-232)

King Software
442 Central Building
Seattle, WA 98104
206/623-8225 or 206/567-4809
Software, including legal time reporting

Knowledge Index
Dialog Information Services Inc.
3460 Hillview Avenue
Palo Alto, CA 94304
800/227/5510
415/858-3796
Information retrieval service

Knowledge Industry Publications
2 Corporate Park Drive
White Plain, NY 10604
914/328-9157
Database directory

Kuma Computers Ltd
11 York Road
Maidenhead, Berks SL6 1SQ
0628 71778
Broad range of software

The Leeds Computer Centre
55 Wade Lane
Merrion Centre
Leeds
England
0532 458877
Software, including data management

Longdin & Browning Surveys Ltd
Old Castle
Llanelli
Dyfed, UK, SA15 2SR
(05542) 57401
Surveyor's system

The Madson Line
255 Fourth Street
Oakland CA 94607
800/851-1551
415/465-3760
Computer and printer carrying bags

MCI
2000 M Street NW
Washington DC 20036
800/MCI-2255
202/833-8484
Electronic mail

Measurement Control & Displays
8 North Street
North Square
Guildford, Surrey
England
0483 574659
Software, including stock control

Measurement Equipment Corp.
2304 Main Street
Evanston, IL 60202
312/866-7940
Data acquisition (RS-232)

Micro Business Centre
17–19 Lichfield Street
Wolverhampton, West Midlands
England
0902 29907
Software, including invoicing

Microcosm, Inc.
1679 Enterprise Plaza
PO Box 624
Hillsboro. OR 97123
503/648-6500
Microprocessor emulation (8086/8088/80186) via RS-232

Micronet 800
Scriptor Court
155 Farringdon Road
London, England EC1R 3AD
01 837 3699
Information retrieval service

MicronTechnology Inc.
2805 East Columbia Road
Boise, ID 83706
208/383-4000
Image-sensing camera (RS-232)

Microtec
505 West Olive Avenue Suite 325
Sunnyvale, CA 94086
408/733-2919
Cross software for 6301

Microtek Ltd
15 Lower Brook Street
Ipswich, Suffolk
England
0473 50152
Software, including small craft navigation

Mikrocomputertechnik
Winchenbachstr 3A
D-5600 Wuppertal 2
West Germany
0202 51044
Serial-to-parallel converter, smart RS-232 interface

Motherboard Corp.
5328 21st Street South-West
Seattle, WA 98106
303/484-8602
Trucker's software

MST Consultants
Newton Road
Bovey Tracey
Newton Abbot, Devon
England
0626 832617
Software, including data management

Newsnet
945 Haverford Road
Bryn Mawr, PA 19010
215/527-8030
Information retrieval

Niagara Scientific
6716 Joy Road
East Syracuse, NY 13057
315/437-0821
Data acquisition (RS-232)

Object Ltd
Ock Mill
Marcham Road
Abingdon, Oxford
England
0235 24301
Machine tool programming system

Orange Computers Ltd
117 King Street
Knutsford, Cheshire
England
0565 53417
Software, including pharmaceutical (chemists') system

P & M Data Services Ltd
29 Wigan Centre Arcade
Wigan, Lancs
England
0942 497123
Software, including golf handicapping

Panasonic
One Panasonic Way
Secaucus, NJ 07094
Data collection systems, other electronic equipment

P.D.S.
Carne House
Markland Hill
Bolton, Lancs
England
0204 493816
Software, including currency conversion

Pelikan
200 Beasley Drive
Franklin, TN 37064
615/790-6171
Ribbons

Peripheral Dynamics
1850 Gravers Road
Norristown, PA 19401
215/825-7090
Badge readers (RS-232)

Pocket Computers Ltd
Cartwright House
39 Monument Hill
Weybridge, Surrey
England
0932 57808
Software, including games

Practical Software
5/62 Shepherds Hill
London N6, England
01 340 0237
Software, including investment analysis

QuickView Systems
146 Main Street, Suite 404
Los Altos, CA 94022
415/965-0327
Software, including data management

REP Systems
5328 21st Street South-West
Seattle, WA 98106
206/762-6434
Software

Rocon Ltd
Radley Road Industrial Estate
Abingdon, Oxon
England
0235 24206
Software, including data management

Scan-Tronic Corp.
3398 E 70th Street
Long Beach, CA 90805
213/633-4051
Data acquisition (RS-232)

Sensory Aids Corp.
Suite 110
205 West Grand Avenue
Bensenville, IL 60106
312/766-3935
Wormald VTS distributor

S.J.R. Software
71–73 Middlewich Road
Northwich, Cheshire
England
0606-48462/45731
Software

Skan-a-Matic
PO Box S
Elbridge, NY 13060
315/689-3961
Bar code readers (RS-232)

SkiSoft, Inc.
Cambridge, MA
617/861-1190
Software development

SM Software (UK) Ltd
Raglan House
56 Long Street
Dursley, Glos.
England
Software

Sofsearch International Inc.
PO Box 5276
San Antonio, TX 78201
512/340-8735
800/531-5955
Software locator service (by function)

Software Riches
Riverview Terrace
Irvington, NY 10533
914/591-6470
Software, including games

The Source
1616 Anderson Road
McLean, VA 22102
800/336-3300
703/734-7500
Information retrieval

Starbuck Data Co.
PO Box 24
Newton Lower Falls, MA 02162
617/237-7695
Data acquisition/control (RS-232)

STC Multicomponent
Edinburgh Way
Harlow, Essex, CM20 2DF
England
Software dealer

Street Electronics
1140 Marsh Avenue
Carpinteria, CA 93013
805/684-4593
Speech synthesizer (RS-232)

Talbot Computers Ltd
61 Heathwood Road
Talbot Park
Bournemouth, Hants
England
0202 519282
Software, including communications

Talbot Microsystems
5030 Kensington Way
Riverside, CA 92507
714/781-0464
Software, including Forth

Taurus Computer Products
PO Box 911, Station 'U'
Toronto, Ont. Canada M8Z 5P9
613/226-5361
Analogue/digital I/O (RS-232)

Teletex Communications Corp.
3420 East Third Avenue
Foster City, CA 94404
415/341-1300
Portable printer

Time & People
30 Cursitor Street
London EC4, England
01 405 2565
Software, including games

Trace Research and Development Center
University of Wisconsin
1500 Highland Avenue
Madison, WI 53706
608/262-6966
Systems for the communicatively handicapped

Transam Microsystems Ltd
59/61 Theobald's Road
London WC1X 8SF
England
01 404 4554
Software, hardware, including parallel interface

Transdata Ltd
Engineering Division
Attn: Ivor Smith
South Street
Havant, Hampshire PO9 1BU
England
0705 486556
System design, programming, hardware engineering

Trend-DLC
280 Midland Avenue
Saddle Brook, NJ
201/791-1414
Paper tape reader/punch (RS-232)

Triad Distributing
2770 South Harbor Boulevard
Santa Ana, CA 92704
714/662-3733
Portable modem

Trilog
17391 Murphy Avenue
Irvine, CA 92714
714/863-3033
Bar Code printer

True Data Corp
17092 Pullman Street
Irvine, CA 92714
714/979-4842
Mark sense, badge readers (RS-232)

Tymnet
20665 Valley Green Drive
Cupertino CA 95014
408/446-7000
Electronic mail

Universal Data Systems
5000 Bradford Drive
Huntsville, AL 35805
205/837-8100
Modems

US Robotics, Inc.
1123 West Washington
Chicago, IL 60607
312/733-0497
Modems

Valldata Ltd
Oakwood
Spa Road
Melksham, Wilts
England
0225 705957
Software, including data management

Warburton Franki
199 Parramatta Road
Auburn, NSW 2144
Australia
Auburn: 648-1711
Melbourne: (03) 699-4999
HX-20 distributor

Western Telematic
2435 South Anne Street
Santa Ana, CA 92704
714/979-0363
Intelligent tape and disk units

Western Union Electronic Mail Inc.
Products and Service Coordination
PO Box 185
McLean, VA 22101
703/821-5800
Mailgrams

Wintek Corp.
1801 South Street
Lafayette IN 47904
317/742-8428
6801 development systems

Words+ Inc.
622 So. Fair Oaks
Sunnyvale, CA 94086
408/730-9588
Systems for the communicatively handicapped

Wormald International Sensory Aids
7 Musters Road
West Bridgford
Nottingham NG2 7PP
England
0602 820600
Viewscan Text System for the visually handicapped

Yokogawa Corp. of America
2 Dart Road
Shenandoah, GA 30265
404/253-7000
Plotters

APPENDIX B: WHERE TO GO FOR MORE INFORMATION

MAGAZINES, NEWSLETTERS, USERS' GROUPS

Bar Code News
North American Technology
Strand Building
174 Concord Street
Suite 23
Peterborough, NH 03458
603/924-6048
Information on bar code applications

Breakthrough Inc.
PO Box 230
Logan, UT 84231
Publishes the Field Portable Computer Newsletter

Byte
70 Main Street
Peterborough, NH 03458
603/924-9281
General magazine for professional computerists

Computer Food Press
31754 Foxfield Drive
Westlake Village, CA 91361
213/462-0888
Information on CBBSs

Creative Computing
39 East Hanover Avenue
Morris Plains, NJ 07950
201/540-0445
Applications-oriented micro magazine

Dr Dobb's Journal
Box E
1263 El Camino Real
Menlo Park, CA 94025
415/323-3111
Technical micro journal

80 Micro
Peterborough, NH 03458
603/924-9471
Magazine for TRS-80 owners, but useful for others

Epson Computing Group
400-2 De Young
Marian, IL 62959
618/993-3600
Users' group, with newsletter for QX-10/HX-20 owners

Forth Interest Group
PO Box 1105
San Carlos, CA 94070
Information on Forth

Hitachi America Ltd
Semiconductor and IC Sales and Service Division
1800 Bering Drive
San Jose, CA 95112
408/292-6404
Manufacture of the 6301 microprocessor

HX-20 Users' Group
28 Sawyer's Lawn
Drayton Bridge Road
Ealing, London W13
England
Users' group with world-wide membership, newsletter

In Business
PO Box 323
Emmaus, PA 18049
215/967-4135
General magazine for small business

Interface Age
13913 Artesia Boulevard
Cerritos, CA 90701
General micro magazine with portables column

Link-Up
6531 Cambridge Street
Minneapolis, MN 55426
612/927-4916
Communications magazine

Microcomputing
(formerly Kilobaud Microcomputing)
Elm Street
Peterborough, NH 03458
603/924-3873
General micro magazine

Miller Services
61 Lake Shore Road
Natick, MA 01760
617/653-6136
Books on Forth

Mountain View Press
PO Box 4656
Mountain View, CA 94040
415/981-4103
Books on Forth

The On-Line Computer Telephone Directory
J.A. Cambron Co.
PO Box 10005
Kansas City, MO 64111
816/756-1847
Updates on CBBSs

Other Networks' Newsletter
PO Box 14066
Philadelphia, PA 19123
Updates on CBBSs

Personal Computer Word
53 Frith Street
London W1A 2HG
England
General micro magazine

Personal Computing
50 Essex Street
Rochelle Park, NJ 07662
201/843-0550
Novice-oriented applications magazine

Phoenix Publishers Association
14 Vernon Road
Bushey, Herts
WD2 2JL
England
Book on HX-20 BASIC

Portable Computer
Miller Freeman Publications Inc.
500 Howard Street
San Francisco, CA 94105
415/397-1881
Occasional HX-20 articles; directed towards novices

68 Micro Journal
3018 Hamill Rd
PO Box 849
Hixson, TX 37343
Occasional 68xx articles

System-68
PO Box 310
Conyers, GA 30207
404/929-0606
Occasional 68xx articles

YES! Bookshop
1035 31st North-West
Washington, DC 20007
202/338-2727
Large line of computer books

APPENDIX C: PORTABLE COMPUTER MANUFACTURERS

ALL Computers Inc.
110 Bloor Street West
Suite 501
Toronto, Canada M5S 2W7
416/960-0111

Casio Inc.
15 Gardner Road
Fairfield, NJ 07006
201/575-7400

Convergent Technologies Inc.
2500 Augustine Drive
Santa Clara, CA 95051
408/727-8830

Datec Inc.
2860 South Circle Drive
Suite 2114
Colorado Springs, CO 80906
303/579-0166

DVW Microelectronics Inc.
PO Box 135
Coventry, England CV6 5RW
0203 668181

Gavilan Computer Corp.
240 Hacienda Road
Campbell, CA 95008
408/379-8005

GRiD Systems Corp.
2535 Garcia Avenue
Mountain View, CA 94043
415/961-6873

Grundy Business Systems Ltd
Grundy House
Somerset Road
Teddington, TW11 8TD
England

Hewlett-Packard
Personal Computer Division
1010 North-East Circle Boulevard
Corvallis, OR 97330
503/757-2000
800/547-3400

MicroOffice Systems Technology Inc.
35 Kings Highway East
Fairfield, CT 06430
203/367-2525

Microwriter USA Ltd
17 East 71 Street
New York, NY 10021
212/288-7778

MSI Data Corp.
340 Fischer Avenue
Costa Mesa, CA 92626
714/549-6000

NEC Home Electronics Inc. (USA)
Personal Computer Division
1401 Estes Avenue
Elk Grove, IL 60007
312/228-5900

Olympia USA Inc.
Box 22
Somerville, NJ 08876
201/722-7000

Panasonic Co.
One Panasonic Way
Secaucus, NJ 07094
201/348-5337

Quasar Company
9401 West Grand Avenue
Franklin Park, IL 60131
312/451-1200

Radio Shack
One Tandy Center
Fort Worth, TX 76102
817/390-3011

Sarasota Automation
1500 North Washington Boulevard
Sarasota, FL 33577
813/366-8770

Sharp Electronics Corp.
10 Sharp Plaza
Paramus, NJ 07652
201/265-5600

Sony Communications Products Co.
Sony Corporation of America
Sony Drive
Park Ridge, NJ 07656
201/930-6432

Teleram Communications Corp.
2 Corporate Drive
White Plains, NY 10604
914/694-9270

Texas Instruments
Consumer Relations
PO Box 53
Lubbock, TX 79408
800/TI CARES

Toshiba America Inc.
Information Systems Division
2441 Michelle Drive
Tustin, CA 92680
800/457-7777
714/730-5000

Universal Data Inc.
3960 Ortonville Road (M-15)
Clarkston, MI 48016
313/625-0158
800/521-1056

Xerox Corp.
Stamford, CT 06904
203/329-8711

APPENDIX D: SYSTEM REFERENCE

Table AD1
Devices

KYBD:	keyboard
SCRN:	screen
LPTO:	microprinter
COMO:	RS-232 port
CASO:	microcassette
CAS1:	external cassette
PACO:	ROM cartridge
BRCD:	bar code reader
A:	disk drive A
B:	disk drive B
C:	disk drive C
D:	disk drive D

Table AD2
OPEN device modes

'I'	input
'O'	output
'R'	read/write (disks only)

Table AD3
File types

0 = BASIC	A = ASCII
1 = Data	B = Binary
2 = Object	

Table AD4
BASIC reserved words

ABS	ALL	AND	ASC	ATN
AUTO	BASE	CDBL	CHR$	CINT
CLEAR	CLOSE	CLS	COLOR	CONT
COPY	COS	CSNG	CSRLIN	DATA
DATE	DAY	DEF	DELETE	DIM
ELSE	END	EOF	EQV	ERASE
ERL	ERR	EXEC	EXP	FILES
FIX	FN	FOR	FRE	GCLS
GET	GO	HEX$	IF	IMP
INKEY$	INPUT	INSTR	INT	KEY
LEFT$	LEN	LET	LINE	LIST
LLIST	LOAD	LOCATE	LOF	LOG
LPRINT	MEMSET	MERGE	MID$	MOD
MON	MOTOR	NEW	NEXT	NOT
OCT$	OFF	ON	OPEN	OPTION
OR	PCOPY	PEEK	POINT	POKE
POS	PRESET	PRINT	PSET	PUT
RANDOMIZE		READ	REM	RENUM
RESTORE	RESUME	RETURN	RIGHT$	RND
RUN	SAVE	SCREEN	SCROLL	SGN
SIN	SOUND	SPACE$	SPC(	SQR
STAT	STEP	STOP	STR$	STRING$
SUB	SWAP	TAB(	TAN	TAPCNT
THEN	TIME	TITLE	TO	TROFF
TRON	USING	USR	VAL	VARPTR
WEND	WHILE	WIDTH	WIND	XOR

Table AD5
BASIC error messages

Error Code	Error #	Message
/0	11	Division by zero
AO	52	File already open
BD	58	Bad data in file
BF	51	Bad file mode
BN	50	Bad file number
BO	61	Buffer overflow; RS-232 data lost
BS	9	Bad subscript; bad RAM file rec#; space found after TAB or SPC keywords
CN	17	Can't continue
DD	10	Duplicate definition
DS	56	Direct statement in file
DU	60	Device unavailable
FC	5	Illegal function call
FD	55	Bad file descriptor
FN	23	FOR without NEXT
ID	12	Illegal direct
IE	54	Input past end
IO	53	Input/Output error
IU	59	Device in use
LS	15	String too long
MO	22	Missing operand
NE	63	File does not exist
NF	1	NEXT without FOR
NO	57	File not open
NR	19	No RESUME
OD	4	Out of DATA
OM	7	Out of memory
OS	14	Out of string space
OV	6	Overflow
PP	62	Protected program
RG	3	RETURN without GOSUB
RW	20	RESUME without error
SN	2	Syntax error
ST	16	String too complex to evaluate
TM	13	Type mismatch
UF	18	Undefined USR function
UL	8	Undefined line number
UP	21	Unprintable error

The following two errors are flagged by HX-20 BASIC, though neither the WHILE nor WEND statements will actually execute.

WE	24	WHILE without WEND
WH	25	WEND without WHILE

Table AD6
HX-20 character set

Dec	Hex	ASCII	Key(s)	
0	0	NUL	CTRL @	
1	1	SOH	CTRL A	
2	2	STX	CTRL B	
3	3	ETX	CTRL C	
4	4	EOT	CTRL D	or CTRL →
5	5	ENQ	CTRL E	
6	6	ACK	CTRL F	
7	7	BEL	CTRL G	
8	8	BS	CTRL H	or DEL
9	9	HT	CTRL I	or TAB
10	A	LF	CTRL J	
11	B	VT	CTRL K	or HOME
12	C	FF	CTRL L	or CLR
13	D	CR	CTRL M	or RETURN
14	E	SO	CTRL N	
15	F	SI	CTRL O	
16	10	DLE	CTRL P	or SCRN
17	11	DC1	CTRL Q	or SHIFT SCRN
18	12	DC2	CTRL R	or INS
19	13	DC3	CTRL S	or CTRL ←
20	14	DC4	CTRL T	
21	15	NAK	CTRL U	
22	16	SYN	CTRL V	
23	17	ETB	CTRL W	
24	18	CAN	CTRL X	
25	19	EM	CTRL Y	
26	1A	SUB	CTRL Z	
27	1B	ESC	CTRL [	or SHIFT PAUSE
28	1C	FS	CTRL /	or →
29	1D	GS	CTRL]	or ←
30	1E	RS	CTRL^	or SHIFT ←
31	1F	US	CTRL _	or SHIFT →
32	20	SP	Space bar	
33	21	!		
34	22	"		
35	23	#		
36	24	$		
37	25	%		
38	26	&		
39	27	'		
40	28	(		
41	29	)		
42	2A	*		
43	2B	+		
44	2C	,		
45	2D	–		
46	2E	.		
47	2F	/		
48	30	0		
49	31	1		
50	32	2		
51	33	3		
52	34	4		
53	35	5		
54	36	6		
55	37	7		
56	38	8		
57	39	9		
58	3A	:		
59	3B	;		
60	3C	<		

61	3D	=	
62	3E	>	
63	3F	?	
64	40	@	
65	41	A	
66	42	B	
67	43	C	
68	44	D	
69	45	E	
70	46	F	
71	47	G	
72	48	H	
73	49	I	
74	4A	J	
75	4B	K	
76	4C	L	
77	4D	M	
78	4E	N	
79	4F	0	
80	50	P	
81	51	Q	
82	52	R	
83	53	S	
84	54	T	
85	55	U	
86	56	V	
87	57	W	
88	58	X	
89	59	Y	
90	5A	Z	
91	5B	[	
92	5C	\	
93	5D	]	
94	5E	^	
95	5F	_	
96	60	'	GRPH [
97	61	a	
98	62	b	
99	63	c	
100	64	d	
101	65	e	
102	66	f	
103	67	g	
104	68	h	
105	69	i	
106	6A	j	
107	6B	k	
108	6C	l	
109	6D	m	
110	6E	n	
111	6F	o	
112	70	p	
113	71	q	
114	72	r	
115	73	s	
116	74	t	
117	75	u	
118	76	v	
119	77	w	
120	78	x	
121	79	y	
122	7A	z	
123	7B	{	
124	7C	¦	
125	7D	3	
126	7E	~	GRPH /
127	7F	DEL	GRPH]

Pre-Defined Graphics Characters

128	80	┼	GRPH S
129	81	┴	GRPH X
130	82	┬	GRPH W
131	83	┤	GRPH D
132	84	├	GRPH A
133	85	─	GRPH T
134	86	│	GRPH R
135	87	┌	GRPH Q
136	88	┐	GRPH E
137	89	└	GRPH Z
138	8A	┘	GRPH C
139	8B	▒	GRPH J
140	8C	█	GRPH F
141	8D	▄	GRPH G
142	8E	▌	GRPH H
143	8F	●	GRPH Y
144	90	○	GRPH U
145	91	♠	GRPH I
146	92	♥	GRPH O
147	93	♦	GRPH P
148	94	♣	GRPH @
149	95	♪	GRPH K
150	96	☎	GRPH V
151	97	✈	GRPH ,
152	98	⌂	GRPH M
153	99	Ψ	GRPH N
154	9A	🏃	GRPH B
155	9B	↑	GRPH ;
156	9C	↓	GRPH .
157	9D	×	GRPH :
158	9E	÷	GRPH /
159	9F	±	GRPH L

User-Defined Graphics Characters

224	E0	GRPH 0
225	E1	GRPH 1
226	E2	GRPH 2
227	E3	GRPH 3
228	E4	GRPH 4
229	E5	GRPH 5
230	E6	GRPH 6
231	E7	GRPH 7
232	E8	GRPH 8
233	E9	GRPH 9
234	EA	CTRL *
235	EB	CTRL +
236	EC	CTRL ,
237	ED	CTRL –
238	EE	CTRL .
239	EF	CTRL /
240	F0	CTRL 0
241	F1	CTRL 1
242	F2	CTRL 2
243	F3	CTRL 3
244	F4	CTRL 4
245	F5	CTRL 5
246	F6	CTRL 6
247	F7	CTRL 7
248	F8	CTRL 8
249	F9	CTRL 9
250	FA	CTRL :
251	FB	CTRL ;
252	FC	CTRL <
253	FD	CTRL =
254	FE	CTRL >
255	FF	CTRL ?

Table AD7
Special key functions

CTRL key and:

A	Display leftmost portion of virtual screen
C	Interrupt automatic numbering
D or	CTRL → Tab right
E	Delete to end of line
F	Move cursor to rightmost portion of screen
H	Delete character to left of cursor
I	Tab left 8 cols
K	Home the cursor
L	Clear/home
M	Carriage return
P	Move window up
Q	Move window down
R	Insert
S or	CTRL → Tab left
V	Make cursor visible
W	Make cursor invisible
Z	Delete to end of virtual screen
/	Move cursor right
]	Move cursor left
^	Move cursor up
_	Move cursor down

Table AD8
Program function key assignments

	BASIC	SkiWriter	Microcassette	Ffosswriter
PF 1	AUTO . . .	Examine Format	Fast Forward	Word
PF 2	LIST	Next	Slow Forward	Sentence
PF 3	LLIST	Start Block	Stop	Paragraph
PF 4	STAT . . .	End Block	Rewind	Block
PF 5	RUN	Page	Quit	To Main
PF 6 (SHIFT 1)	?DATE$:?TIME$	Change Format	Clear	Def
PF 7 (SHIFT 2)	LOAD . . .	Find		Search
PF 8 (SHIFT 3)	SAVE . . .	Delete Block		Rplc
PF 9 (SHIFT 4)	TITLE . . .	Copy Block		
PF 10 (SHIFT 5)	LOGIN . . .			To Pad
	Operating System			
PF 11 (CTRL 1)	Cassette			
PF 12 (CTRL 2)	Copy			
PF 13 (CTRL 3)				
PF 14 (CTRL 4)				
PF 15 (CTRL 5)				

Table AD9
International characters

Country	Hex Code
Spain	10
Italy	11
Sweden	12
Denmark	13
England	14
Germany	15
France	16
USA	17

Table AD10
SELECTED MEMORY ADDRESSES
(All addresses in hex)

7E	— Set to $80 to unprotect low memory
7F	— Character set
106-107	— Address of Trap routine
115-116	— Address of IRQ handler
118-119	— Address of Software Interrupt (SWI) handler
11B-11C	— Address of non-maskable interrupt handler
11E-11F	— Address of table of user-defined graphics characters

The following six addresses do not apply to BASIC programs

120-121	— Address of routine executed on BREAK key
122-123	— Address of routine executed on MENU key
124-125	— Address of routine executed on PAUSE key
126-127	— Address of routine executed on CTRL PF3
128-129	— Address of routine executed on CTRL PF4
12A-12B	— Address of routine executed on CTRL PF5
12C-12D	— Amount of RAM
270-271	— Address of beginning of virtual screen
272-273	— Address of end of virtual screen
274-275	— Address of beginning of physical screen
276	— Width of virtual screen
277	— Height of virtual screen
278-279	— Cursor position on the physical screen (X,Y)
2D0-2D3	— Cassette block identification field
2D4-323	— Cassette header block
324-327	— Microcassette block identification field
328-377	— Microcassette header block
378-47B	— Microcassette buffer area
4FE-4FF	— RAM file size
5A2-5A3	— Address of RAM file
657	— Start of Device Control Block table
69C-6B5	— DCB for KYBD:
6B6-6CF	— DCB for SCRN:
6D0-6E9	— DCB for COMO:
6EA-703	— DCB for CASO:
704-71D	— DCB for CAS1:
71E-737	— DCB for PACO:
738-751	— DCB for LPTO:
AA5-AA6	— SkiWriter RS-232 parameters (e.g., AD38 is 600 bps, no parity, 8 bits)
4000-5FFF	— RAM in expansion unit
6000-7FFF	— RAM in expansion unit or internal ROM
8000-BFFF	— BASIC or ROM in expansion unit
C000-FFFF	— Operating system
DFFA	— Trap routine
E000	— Reset routine
EF49	— External interrupt handler
FF6A	— Keyboard scan routine

Table AD11
Format of Blocks on Cassette/Microcassette

Synchronization field — 80 bits of 0
Preamble — FFAA (2 bytes)
Block identification field — 4 bytes

Byte 0:	Block type
	H = Header
	D = Data
	E = End of file

Bytes 1-2:	Block number (0000-FFFF)
Byte 3:	Block identification number (00 or 01)

Data field — 80 bytes for header and end of file blocks, 256 bytes for data
BCC field — Block Check Character for CRC error checking
Postamble — AA00 (2 bytes)

Table AD12
Format of header block

Displacement Dec	Hex	Size Dec	Item	Description
0	0	4	ID field	HDR1 (indicates a header block)
4	4	8	Filename	
12	C	8	File type	
20	14	1	Record type	2 (fixed-length, written twice)
21	15	1	Interblock gap length	blank (long gap) or S (short gap)
22	16	5	Block length	00080 or 00256
27	17	4		empty
32	20	6	Creation date	MMDDYY
38	26	6	Creation time	HHMMSS
44	2C	6		empty
50	32	2	Serial number	01
52	34	8	System name	HX-20 (& 3 blanks)
60	3C	20		empty

Table AD13
Format of End of File Block

Displacement Dec	Hex	Size Dec	Item	Description
0	0	4	ID field	EOF (& 1 blank)
4	4	76		empty

Table AD14
Device control block format

Displacement Dec	Hex	Size Dec	Description
0	0	4	Device name in ASCII
4	4	1	I/O mode $10 — sequential input $20 — sequential output $30 — sequential input/output
5	5	2	Entry point for the OPEN routine
7	7	2	Entry point for the CLOSE routine
9	9	2	Entry point for the routine to input one byte into ACC A.
11	B	2	Entry point for the routine to output one byte from ACC A
13	D	2	Entry point for the EOF routine
15	F	2	Entry point for the LOF routine. Store number of characters remaining into ACC D
17	11	4	Device information
21	15	1	Position of next character to be output
22	16	1	Width of output line/block
23	17	1	Size of the print zone (comma field)
24	18	1	Position of last print zone
25	19	1	$80 = WIDTH illegal on this device $01 = no LF after CR on a PRINT

Table AD15
Decimal/Hex conversion

From hex: locate each hex digit in its corresponding column position and note the decimal equivalents. Add these to obtain the decimal value.
e.g. Hex FFFF=61,440 + 3,840 + 240 + 15 = 65,536

From decimal: Find the largest decimal value in the table that will fit into the decimal number to be converted, and note its hex equivalent and hex column position. The find the decimal remainder. Repeat the process on this and subsequent remainders.
e.g. Dec 2635 = A & 4 (Hex 2635-2560) & B (Hex 75-64) = A4B

byte 1				byte 2			
4		3		2		1	
Hex	Dec	Hex	Dec	Hex	Dec	Hex	Dec
0	0	0	0	0	0	0	0
1	4,096	1	256	1	16	1	1
2	8,192	2	512	2	32	2	2
3	12,288	3	768	3	48	3	3
4	16,384	4	1,024	4	64	4	4
5	20,480	5	1,280	5	80	5	5
6	24,576	6	1,536	6	96	6	6
7	28,672	7	1,792	7	112	7	7
8	32,768	8	2,048	8	128	8	8
9	36,864	9	2,304	9	144	9	9
A	40,960	A	2,560	A	160	A	10
B	45,056	B	2,816	B	176	B	11
C	49,152	C	3,072	C	192	C	12
D	53,248	D	3,328	D	208	D	13
E	57,344	E	3,584	E	224	E	14
F	61,440	F	3,840	F	240	F	15

Note: 1K=1024

Table AD16
HEX addition table

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10
2	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11
3	4	5	6	7	8	9	A	B	C	D	E	F	10	11	12
4	5	6	7	8	9	A	B	C	D	E	F	10	11	12	13
5	6	7	8	9	A	B	C	D	E	F	10	11	12	13	14
6	7	8	9	A	B	C	D	E	F	10	11	12	13	14	15
7	8	9	A	B	C	D	E	F	10	11	12	13	14	15	16
8	9	A	B	C	D	E	F	10	11	12	13	14	15	16	17
9	A	B	C	D	E	F	10	11	12	13	14	15	16	17	18
A	B	C	D	E	F	10	11	12	13	14	15	16	17	18	19
B	C	D	E	F	10	11	12	13	14	15	16	17	18	19	1A
C	D	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B
D	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C
E	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D
F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E

APPENDIX E: COMPUTER TERMINOLOGY

GLOSSARY OF THE FAMILIAR BUT DIFFERENT, AS WELL AS THE TOTALLY UNFAMILIAR

abort halt processing before completion

access time time required to locate and retrieve data

accumulator register used for arithmetic/logical operations

ACK in data communications, a positive acknowledgement given by the receiver to the sender (ASCII 6). Cf. NAK

acoustic coupler a modem with cups to hold a telephone handset

address the location of a byte in memory

addressing modes the different ways in which the processor can locate the information necessary to execute an instruction

algorithm set of steps for solving a problem

alphanumeric the set of alphabetic characters plus special characters plus numeric characters

analog continuous as opposed to discrete. Example: a clock face measures time in an analog fashion, while a digital clock measures time in discrete steps

AND a Boolean operation in which two binary values are compared. If one operand has the same bit on as the other, the result is on, otherwise the result is off.

ANSI American National Standards Institute, a private organization that often sets standards for the computer industry.

application program software to fulfill a particular user requirement, as opposed to utility programs, for instance, that may be run only to prepare the computer for the application program.

architecture the structure of a computer; how the different internal pieces of a computer fit together.

archive move a program or data to where it will be permanently kept, without expectations of its ever being used except in emergencies.

argument a value that is passed to a function and processed by it. Example: the SQR function to find square roots might be passed a '4' as its argument.

array a table. Unlike printed tables of values, an array may have any number of dimensions

ASCII a widely-used means of expressing text and control characters in 7 data bits

assembler a program to convert assembly language source code into executable machine code

assembly language a programming language in which statements have a 1 : 1 correspondence with actual machine instructions

asynchronous not within the normal chain of events; not synchronized with the main actions of the system *asynchronous communication* transmission in which there is no fixed time period between successive data characters

audit trail a log of activity during a program run

auto-answer see *Modem*

auto-dial see *Modem*

automatic repeat ability of a key to continue producing the same character for as long as the key is held down

back-up a duplicate copy, usually made for safe keeping

bar codes a system of recording data, using printed bars of varying thicknesses

bar code wand a hand-held scanner used for reading bar codes

base number system. Binary, octal, decimal, and hexadecimal are base 2, base 8, base 10, and base 16 number systems

BASIC a popular programming language for small computers

batch non-interactive; a collection of data processed together instead of singly

baud in common usage, synonymous with bps

BCD binary coded decimal: a way of storing decimal numbers in memory

Bell-compatible see *Modem*

benchmark test producing recordable and reproducible results of specific functions running on a specific system

beta test field testing of a product before its sale to the public

binary an either/or, on/off condition. Used as the base for a number system on most computers

binary search method of locating an element in an ordered list by continually discarding that half of the list that is out of range

bit binary digit; the most basic unit of information that a computer handles. Expressed as a 1 or a 0

bit map memory area in which each bit represents the state of some condition

black box a device that performs a specific function without requiring (or possibly even allowing) the operator to see or understand what's going on

block data that has been physically grouped together, usually for ease of processing, storage or transmission

Boolean logic method of combining together conditions that are either on/off, true/false

boot or bootstrap see *cold start*

bps bits per second, a measure of transmission speed

branch a jump to another part of a program

break (1) a key that interrupts program operation
(2) in communications, a signal on the link that tells the other end to stop current processing

bubble memory see *memory*

bubble sort method of ordering a list of items by exchanging or 'bubbling' each item through the list

buffer a temporary storage area, usually used for holding data in transit

bug software or hardware fault

bundling marketing a computer system with components that would normally be sold separately

bus a hardware path that serves as a communications link among the computer's internal components

byte commonly, an 8-bit binary string operated upon and addressable as a single unit. On most computers, characters are stored one to a byte. However, some larger minicomputers store characters one to a word (which can be any number of bits), and which is sometimes redefined as a byte

carriage return the code generated by pressing RETURN or ENTER; receipt of this code moves the cursor to the left margin (ASCII 13)
carrier signal the phone line tone upon which the data signals are superimposed
CBBS Computerized Bulletin Board System
Centronics a type of parallel interface that has become an industry standard for data transfer to printers. A 36-pin connector is often used, though this is not part of the standard. (Only the signals are standardized)
character in general usage, any of the 256 allowable combinations of 8 bits. This includes letters, numbers, punctuation marks, etc.
character string a connected sequence of characters
checksum a number computed for error-checking purposes
chip an integrated circuit, the basic building block of computer systems
clear change to zeros or blanks
clock (1) the internal system time which activates the processors on every tick. On the HX-20 it operates at 0.6 MHz, producing a signal about every 1600 nanoseconds
(2) the 'real-time' clock: keeps track of time and date
code (1) program instructions; source code is what the user enters; object code is what is output from an assembler or compiler; machine code is what the computer actually executes (synonymous with object code on many machines)
(2) a particular number with a defined meaning, e.g., a decimal 13 (hex OD) is the code for a carriage return
cold start bringing up a computer system from a completely uninitialized state, e.g., using the CTL/@ sequence on an HX-20
command a direction to a program to do something, such as FIND in SkiWriter or RUN in BASIC
comments notes in a program's source code that give information about the program but which are bypassed by the compiler/-interpreter/assembler
compatibility capability of two products to perform the same functions in the same manner, producing the same results. Example: many personal computers have varying degrees of IBM PC compatibility
compiler a program that converts high-level source code into assembly language or machine code, for later assembly or execution. COBOL and Fortran are compiler languages, BASIC is normally an interpreted language but compilers are available on some machine. Compiled programs (because the end result is object code) run faster than interpreted programs
complement to take the one's complement of a value, reverse its bits. The two's complement is the one's complement plus 1. Used by computer hardware instead of direct subtraction, which is more difficult to implement
configuration particular way in which a computer system has been assembled from its components
connect time period during which a remote terminal user is logically connected to a computer system
connector the plug/socket that physically makes a connection. Example: DB-25 connectors are either male (with pins) or female (with holes). The special connectors that are sometimes necessary to mate plugs of the same sex are called changers or gender changers
constant a named memory location whose contents are set once by a program and not changed thereafter
contiguous physically adjoining
control block a set of memory locations in RAM that give information about the system

control character a code that when received by a device, causes it to perform some activity

conversion process of taking something designed for one computer system and modifying it for use on another computer system

core originally memory composed of magnetic rings, but now synonymous with computer memory in general

counter memory location that tracks the number of occurrences of some event

CP/M a widely-used operating system for business applications. Officially, known as CP/M-80. Other variations, e.g., CP/M-86 and CP/M-68K are not in general use

CPM Critical Path Method: a project management tool

cps characters per second

cpu central processing unit. The heart of the computer system. On a microcomputer system, synonymous with mpu. Example: the 6502 in an Apple, the 8088 in an IBM PC, the 6301 in the HX-20

crash an abrupt termination of processing,usually due to a hardware failure, and usually requiring a restart

CRT a picture tube plus associated electronics; usually found as the display screen of a desktop computer. Synonymous with monitor. (Though the term is often used interchangeably with VDT and terminal, those units are typically composed of a CRT plus a keyboard)

CTS Clear To Send. An RS-232 control signal indicating that the DTE end of the connection is allowed to transmit

cursor an indicator that specifies where on a display screen the next character to be input will go

cut and paste see *Word Processing*

cycle (1) the length of time it takes the processor to complete an action. On the HX-20, this can vary between 1.6 and 10 microseconds, depending on what action — read, write, etc. — is being done
(2) one complete clock period. On the HX-20, 1.6 microseconds

daisy chain method of connecting peripherals one to the other as opposed to connecting each directly to the computer

daisy wheel printer letter-quality printer producing fully-formed characters by means of an impact upon a petal of the print element

data raw input, intended to be processed into information

data base a means of organizing stored data, so that specific information can be retrieved or updated

data communications the method by which data is digitized, transmitted, and received by other devices

data set (1) a file
(2) a modem

DBMS Data Base Management System; the set of programs that maintain a database. Usually used to refer to software that does more than just file management

DB-25 type of connector, with 25 pins in 2 rows, used primarily as the 'standard' plug for RS-232 connections. Note: not all RS-232 interfaces use DB-25 connectors (e.g., HX-20); not all DB-25 connectors indicate an RS-232 interface

DCE Data Communications Equipment. The part of an RS-232 link, usually a modem, that provides the mechanism by which communications occur. The DCE side will transmit on pin 3 and receive on pin 2

debug locate and resolve problems with an element of a computer system

dedicated single-purpose

default the value that is automatically generated with any operator intervention
delay loop a series of computer instructions the sole purpose of which is to slow down the program, e.g., so that the user can read a display
delete remove
delimiter a code that a program can recognize as the start/end of something, e.g., double quote marks at the beginning of a literal
device driver program written to allow communication between a peripheral device and another program. Example: software to read the input from a bar code wand
digital referring to signals that are in discrete, rather than continuous states
digitize convert analog (continuous) data into digital (discrete) data that a computer can use
DIN connector a type of connector, with the pins arranged in a circle, often used for connecting a computer to a cassette player. The HX-20's RS-232 interface terminates in a DIN socket
direct access see *random access*
directory list of files
disk see *floppy disk*
documentation instructions, printed or online, that explain a product
DOS disk-based operating system
dot graphics a method of drawing in which every point on the display or printer can be programmed
dot matrix a means of forming characters by grouping dots. On the HX-20, each character is made up of a set of 5 horizontal dots and 7 vertical dots
double density a disk recording technique that can produce approximately 170K on each side of a 40-track 5¼ in. (133 mm) diskette
download transfer a file into another computer system that acts as the terminal
downtime period during which a computer system (or subsystem) cannot be used
driver see *device driver*
DSR Data Set Ready. An RS-232 signal that indicates the DCE side of the link is ready, e.g., the modem is turned on
DTE Data Terminal Equipment. The end of an RS-232 connection, usually a computer or a terminal, that originates data or acts as the final receiver of data. The DTE side will transmit on pin 2 and receive on pin 3
DTR Data Terminal Ready. An RS-232 signal that indicates that the computer or terminal is online.
dual density being able to operate at either of two densities, e.g. a dual density disk drive can read either single or double density diskettes
dump move data from one device to another without any processing
duplex see *full duplex*

echoplex in data communications, transmitting back the same code you just received so that the operator can visually verify that it was received correctly
EIA Electronics Industry Association, sometimes used as a synonym for RS-232, one of their standards
8-bit computer type of computer in which most operations are internally handled in 8-bit units, external data is accessed 8 bits at a time, with the maximum memory accessible at any one time usually being 64K

EL Sharp's electro-luminescent display

electronic mail terminal-to-terminal transmission of messages which are computer-readable, but which are meant for human eyes

emulate imitate the operation of another device. Example: many communications programs emulate TTY terminals

end of file condition condition when there is no further data that can be read on a sequential file

end of file marker code placed after the data in a file signifying that there is no further data

ENTER key synonymous with RETURN key

entry point location within an object code program at which execution begins

EPROM see *memory*

error a condition that arises when the operator or the hardware does something a program doesn't understand, expect or appreciate

ETX in data communications, a character indicating the end of a text block being transmitted (ASCII 3)

execute do

FF form feed. A control character telling the receiver to do a page eject (ASCII 12)

field an item of data in a record. Example: zip code in an address record in a mailing list file

file a collection of related data, usually on a storage medium, logically and/or physically contiguous. Examples: a word processing document, a BASIC program

firmware term commonly used to refer to software in ROM

flag a variable set aside in memory to indicate the status of some function

floating point a way of representing data, also called scientific notation, in which the decimal point is not in a fixed position

floppy disk a device that holds data on flat, flexible, removable magnetic platters (diskettes) allowing both sequential and non-sequential access to the data

flow control the use of codes or signals to control data transfer, e.g., XON/XOFF or DTR/DSR

FORTH programming/operating environment

FORTRAN a programming language

friction feed a type of printer mechanism that, similar to that of a typewriter, presses the paper against the platen. Used primarily for single sheet printing

full duplex communications connection allowing simultaneous transmission in both directions

function in BASIC, a pre-defined routine that, when called, returns a value

garbage collection a technique used by interpreters to reclaim memory space no longer in use

gender changer see *connector*

GIGO 'garbage in, garbage out'. Refers to the fact that computers have no intuition and will process your data in accordance with your program, even when the data is meaningless

global known throughout the system. A global variable, for instance, can be used by any part of the program; a local variable can be used only by the subroutine it's defined in

half duplex communications connection allowing transmission in only one direction at a time

hand-shaking exchange of signals on a communications link to control data transfer *hang* a state of non-activity which can only be ended with a restart, usually caused by a failure in the system software which in turn is often caused by the overwriting of system pointers in memory

hard-coded use of actual values within program code, rather than variables which can change with conditions

hard copy printout

hard disk a disk drive in which the recording surface is rigid, inflexible; typically capable of much higher storage capacities than floppy disks

hardware the physical components of a computer system

hard-wired directly wired point to point, as opposed to going through the dial-up telephone system

head part of a drive that does the actual reading/writing on magnetic media

header identification bytes placed at the beginning of the data

hex hexadecimal (base 16) numbering system. A way to organize binary (base 2) information so that it is more readable. Uses the characters 0–9 and A–F to represent the quantities 0–15

high-level language a programming language in which one statement converts into many actual computer instructions

host computer in a communications link, the computer that accepts commands from the remote user

housekeeping initialization routes, usually found at the start of a program or power-on of a device

I/O Input/Output

IC Integrated Circuit

icon graphical symbol

IEEE-488 a standardized means of connecting I/O devices, used by Hewlett-Packard and a few other computer manufacturers

IF condition a true/false test

index (1) in programming, a number used to locate a particular entry in an array

(2) in assembly language programming, a number added to a memory address to form a new address

(3) in file management, a number used to locate a particular record on a file index

index register on the 6301, a 16-bit register often used to hold addresses

information data that has been processed into something people can find meaningful

initialize make a program or device ready for use

ink jet printer type of printer that avoids the noise of an impact upon the paper by shooting the ink onto it

input what gets put into a program or device. As a verb, synonymous with 'read'

instruction a specific operation for the computer to execute

instruction set the list of all possible instructions that have been implemented on a particular computer

integer a whole number. HX-20 integer arithmetic uses whole numbers from −32,768 to 32,767

interface the point at which two dissimilar elements meet to transfer data. This could be two programs, two physical devices, a program and a physical device, etc.

interpreter a program that scans source code and performs the indicated operation one program statement at a time

inter-record gap the space between successive physical blocks on a recording medium

interrupt an event that presents a signal to the processor, causing it temporarily to suspend normal program execution

jump an instruction that sets the program counter to a specific location, as opposed to just letting the program counter increment to the next sequential instruction
justify see *Word Processing*

K usually 1024, but also used as 1000 on occasion, Example: a 9.6 Kbyte line transmits data at 9600 bits per second, but 64 Kbyte of memory is actually 65,536 bytes.
keyword text string around which some function revolves, e.g., zip code might be a keyword for sorting a mailing list
kludge a device or program that looks as if it was produced and designed at the same time, using components from disparate sources; usually a prototype for a later product

label in assembly language, the name given to a program statement so that it can be a target for a branch/jump
language a means of describing a problem in such a way that a program, called an interpreter or compiler or assembler, can turn the description into something a computer can understand
LCD liquid crystal display, used in digital watches and portable computers
letter-quality typewriter-quality
LF line feed. A control code that advances the cursor to the next line without changing the character position (ASCII 10)
LIFO last-in, first-out. A type of item management in which the last item to be entered into the system is the first one to be retrieved. Example: the stack maintained for subroutine calls in HX-20 machine language is maintained in LIFO fashion
linked list list of items, contiguous or non-contiguous, whose elements are chained to each other via address pointers
linking loader a system utility that brings into memory a relocatable object module, assigns absolute addresses to it, and transfers control to it
literal characters in a program that are meant to be taken as they are, at face value, rather than as a symbolic name for something else
load bring into memory
local known only in one section of the system. Cf. global
lockout a situation in which a resource is needed for continued processing, but cannot be obtained
logical the way the computer looks at things as opposed to the way they physically are. Example: a printer can be physically connected to the HX-20, but may not be recognized by the computer as such, until hand-shaking has begun, i.e., it's not logically connected. Example: a line in a BASIC program with a remark is treated the same way by the interpreter as the same line without the remark; while physically different, the lines are logically the same
log-in sign in
loop a list of instructions executed in a repetitive fashion, until some condition has been satisfied
lpm lines per minute
LSB least significant bit/byte; the furthest right bit in a byte (b0) or the furthest right byte in an address. Cf. MSB.

M in general usage, 1 million. If referring to memory, then equals 2^{14} or 1,048,576.
machine code see *code*
machine language see *code*
macro in assembly language, a statement that expands into multiple instructions
mag magnetic

main memory a computer's internal storage
mainframe (1) a large-scale computer, such as an IBM 370
(2) the backplane into which computer circuit boards are plugged
map layout
mark in communications, a '1' state
mark sense a method of recording data by means of pencil marks
mask framework into which data is placed, with only a selected part of the data remaining visible
master copy original version from which duplicates are made
megabyte one million bytes, actually, 1,048,576
membrane keyboard flat, touch-sensitive, sealed keyboard useful in hostile environments, but unsuitable for touch-typing because of the lack of key-travel
memory the part of the computer that is used to store instructions and data
volatile memory memory that requires a constant application of power in order to retain its contents
non-volatile memory memory that will retain its contents without a power source
ROM Read-Only Memory; memory that is 'burned-in' at the factory with permanently set instructions/data. Once burned-in like this, it cannot be changed. Many computers have a BASIC interpreter in ROM
ROM cartridges ROM packages, containing programs, that can be plugged into a computer. Used on many inexpensive computers and video game machines
PROM Programmable ROM. Originally a write-once memory device, now usually used as a synonym for EPROM
EPROM Erasable, Programmable ROM. Usually used in custom applications, this is a ROM that can be erased through ultra-violet light and rewritten by use of widely-available 'PROM-burners'
EEPROM EPROM that can be both erased and rewritten via electrical current
RAM scratchpad memory that can be written to by the user. If data is to be processed by the computer, it must be loaded here first. If programs are kept on an external medium, e.g. tape cassettes, then they must be moved (loaded) into RAM first in order to be run
dynamic RAM volatile semiconductor RAM that must be continually accessed (refreshed) by the processor for it to continue to hold its contents
static RAM volatile semiconductor RAM that doesn't require a refresh cycle
CMOS RAM a particular type of static RAM that requires low power, i.e., batteries can be used instead of line power
RAM cartridges RAM that has its own power supply, allowing it to be removed physically from the computer. Similar to a ROM cartridge, but in this case the data or programs to be stored are determined by the user
bubble memory a form of magnetic (as opposed to semiconductor) non-volatile RAM that keeps its contents without any power. But access time is lower than that of semiconductor memory
memory location a particular byte in memory
menu list of choices on the screen, from which one may be selected
menu-driven system in which the operator directs the functioning by selecting choices from menus
MICR Magnetic Ink Character Recognition; method of recording data on paper. Used for bank checks
microcomputer a computer with its central processor on a single chip

microprocessor (MPU) the chip that serves as the central processing unit in a microcomputer, e.g., Z80, 8088, 6301

microsecond $^{1}{}_{1,000,000}$ of a second

millisecond $^{1}{}_{1000}$ of a second. Example: the time required for a floppy disk drive to move the head from one track to the next may be 6 milliseconds or 6 ms

minicomputer a term that has become increasingly vague; currently refers to anything between microcomputers and mainframes

mnemonic name assigned to a machine operation code by an assembler

mode a software/hardware framework which restricts the operator to only a subset of the computer's total functions at any one time

modem a device for converting a computer's digital signals into the type of analog signals that can be carried across an ordinary telephone line

> *originate-only modem* a modem that can transmit and receive only on the frequencies specified for the modem originating a call
>
> *auto-dial* the capability of a modem to send pulses or tones that simulate a handset dialing a number
>
> *auto-answer* the capability of a modem to respond to an incoming call
>
> *Bell 103-113 compatible* a type of asynchronous, full-duplex modem in common usage in the US. Normally operates at 110 or 300 bps
>
> *Bell 202 compatible* an early 0–1200 baud, asynchronous modem. Can operate in full-duplex only on private lines; operates half-duplex on dial-up lines (unless equipped with the reverse channel feature)
>
> *Bell 212A compatible* a modem capable of operating at any two of the following three combinations: 300 baud asynchronous, 1200 baud asynchronous, 1200 baud synchronous. Most commonly used to communicate with remote time-sharing services at 1200 baud asynchronous
>
> *null modem* not a modem, but a cable that is wired to allow direct communication between two devices in situations where a modem would normally be required. (The signals are crossed so that each DE device thinks it is talking to a DCE device)
>
> *modem eliminator* usually a device that functions as a null modem with the added feature of amplifying the signals for traversing longer distances

module hardware or software that is self-contained, but used as part of something else

monitor (1) CRT
(2) a primitive operating system

MSB most significant bit/byte; furthest left bit (b7) in a byte or furthest left byte in an address. Cf. LSB

MS-DOS a popular operating system for IBM PC type computers

multiplex combine two signals so as to share one path

NAK in data communications, a negative acknowledgement given by a receiver to the sender (ASCII 21), Cf. ACK

nanosecond one-billionth (10^{-9}) of a second

nested entirely within. Example of a nested IF statement in BASIC: IF . . . THEN IF . . . ELSE . . . ELSE . . .

network simultaneous logical and physical connections among a number of computers, terminals or computer peripherals

nibble half of a byte; a single hex digit

null void, empty

object code see *code*

object module a program in object code form
OCR Optical Character Recognition; a system that turns human-readable text into machine-readable data
octal base 8 number system, using the digits 0–7
OEM Original Equipment Manufacturer; someone who markets a system built from one or more separately obtained components, to which the OEM may or may not 'add value' in the form of in-house developed software/hardware
offline logically disconnected from a computer
one's complement see *complement*
online logically connected to a computer
op code in assembly language, that part of the instruction which tells the cpu what function is to be performed
operand in assembly language, the part of the instruction that tells the cpu what is to be acted upon by that instruction
operating system (OS) software that acts in a supervisory manner, controlling the other software and hardware components of a computer system
OR in Boolean logic, turning the result bit on (true) if either of the operand bits is on, otherwise turning the result bit off
output what comes out of a computer. As a verb, synonymous with 'write'
overflow condition that results when something is stored in a container not large enough to hold it, e.g., storing a number greater than 32,767 in an integer variable
overlay a program segment loaded in on top of a previous segment that is no longer needed
overwrite store data at a location where there had previously been other data

packet (1) in data communications, a message unit
(2) a set of flags/pointers that are passed to a system component; also called a parameter list
pad add spaces to extend the size of
page usually, 256 bytes ($100) of memory
parallel interface a connection in which multiple data bits are transmitted at one time. Cf. serial interface
parameter information passed from a program to another program or subroutine
parity a simple means of error detection
odd parity the number of data bits plus the parity bit that are in a '1' state equals an odd number
even parity the number of data bits plus the parity bit that are in a '1' state equals an even number
no parity if there are 7 data bits, the parity (8th) bit is ignored. If 8 data bits are used, there is no separate parity bit
mark parity the parity bit is always on
parse interpret
Pascal a programming language
patch fix a program, usually a program in object code form
peripherals external devices added to a computer system to provide additional functions, such as I/O
pin feed printer mechanism that advances the paper by means of holes in the margin
pixel a dot on a dot graphics screen/printout
plotter a computer-driven device for mechanically drawing on paper
pointer a memory location that contains the address of another memory location
poll a request for data, usually directed towards a device

port I/O interface

portability (1) for hardware, the ability to compute while on the move
(2) for software, the ability to run on diverse computers

POS Point of Sale

postfix arithmetic system in which the operator follows the operands. Example: 23 + rather than 2 + 3

precision accuracy to which a numeric operation can be resolved

primitive in software, the lowest level of operation

process control supervision of an industrial or other real-time process by a computer

processor the part of a computer system that interprets and executes programs

program set of instructions that direct the activities of a computer system

program counter register that contains the address of the next instruction to be executed

programming the process of writing computer instructions

PROM see *memory*

prompt a short message usually put out when the operator is expected to input something

protocol set of rules by which data communications occurs. Asynchronous protocol, also called start–stop, requires that each data character be framed with a start bit at the beginning and a stop bit at the end.

push-down/pop-up see *LIFO*

QWERTY keyboard a common type of keyboard, named for the arrangement of letters

RAM see *memory*

RAM file a collection of data stored contiguously in RAM

random access method of retrieving/updating data in non-sequential fashion

real-time wall-clock, 'people' time; unrelated to the system clock which controls what happens within the computer

real-time program a program that must continuously process data as it is received

record set of information in a file. Example: a customer entry in a mailing list file

register special, high-speed memory within the processor set aside for arithmetic/logical operations and program control

relocate move a program within the computer's memory

remark a comment in the source code

remote in data communications, a terminal or computer physically distant from the other end of the link

report-generator program that can produce a formatted printout according to the user's specifications, without programming

reserved words words that are permanently assigned to a specific usage and cannot be used for any other purpose

reset return the system to a starting condition

response time time elapsed from when a command is entered until a response is received

ROM see *memory*

routine generally synonymous with subroutine

RS-232 a standardized electrical definition used in data communications

RS-449/422/423 US government supported standard for data communications interconnections, intended eventually to replace RS-232

RTS Request To Send. The RS-232 signal that indicates the DTE device wants to transmit

save store in a safe place
scan examine
scrolling the act by which new data appears on one side of a display screen, displacing data on the opposite side
search/replace see *Word Processing*
secondary storage external data storage, as opposed to main memory
sequential file a file that is ordinarily accessed continuously from beginning to end
serial interface connection in which bits are transmitted one at a time. Cf. parallel interface
sex changer see *connector*
single-step in debugging, facility that enables the user to stop after every program instruction to view/alter memory
single-threaded accomplishing one process at a time
slave component under the direction of another component, called the master
software programs
source code see *code*
space (1) a character — ASCII 32 — that displays as a blank
(2) in communications, a '0' state
spreadsheet way of placing numerical data in tabular format, with arithmetic relationships between the cells
stack a sequentially retrieved list of items, usually addresses, in memory. Many microcomputers, including the HX-20 have a particular register, the stack pointer, set aside just for stack operations
stack pointer a register whose contents contain the address of the next free entry in a stack
start bit in asynchronous transmissions, a '0' bit that signifies the beginning of data
statement in programming, an instruction to the computer
stop bit in serial data, a '1' bit that signifies the end of a data byte
storage usually, an external, permanent area for holding data
store save for later use
string sequence
string packing a technique of storing machine code within a BASIC statement
subroutine self-contained section of code within a program, called from another part of the program
symbolic use of a name to represent a numeric quality
synchronous communication a form of transmission in which data characters are strung or blocked together and in which each character begins and ends at a fixed point in time from the beginning of the block
syntax rules, especially those that apply to the format of commands and programming language statements
system any cohesive product that contains all of the necessary components to perform a specific function
system integration putting components from different manufacturers together to form a coherent finished product

table look-up find a specific entry in a table
teletext one-way, regular dissemination of information to multiple users
terminal a keyboard plus display or printer, used for communicating with remote computers

text editor a program to manipulate alphanumeric characters

thermal printer a type of printer that 'burns' characters into special heat-sensitive paper

time-sharing a means of splitting computer time over a number of terminal users

toggle flip from one state to another. Example: pressing the INS key on the HX-20 toggles insert mode on/off

token in BASIC, the code assigned to a reserved word

tpi tracks per inch; the spacing among the circular tracks on a diskette. 48 tpi usually yields 40 tracks per disk side; 96–100 tpi usually yields 80 tracks per side

trace step through a program, producing a description of what happened at each step. In Microsoft BASIC, a feature that displays what program lines are being executed

track the channel in which data is recorded on a magnetic surface

tractor feed a mechanism for moving paper through a printer via sprocket holes in the paper's margin. As opposed to pin feed, tractors generally adjust to the width of the paper

transaction a single interaction between human and computer

tree structure a method of organizing data that relates the elements like branches on a tree

truncate chop off

truth table a table that shows all possible results of a Boolean operation

TTY an early ASCII printing terminal; most terminals now in use are programmed to 'look like' TTYs for compatibility reasons

two's complement see *complement*

update modify; usually data or program source code

upload transfer a file into another computer system acting as the host

upward compatible any product that can accept input designed for a pre-existing product, and produce the same output

user from the programmer's point of view, anyone who uses his program; from the manufacturer's point of view, anyone who uses the computer once it has left the factory (including programmers)

user-friendly easy to learn/use

user interface the part of a computer system or program that interacts with the operator

variable a named memory location whose contents are changed during program execution

VDT Video Display Terminal; a CRT + keyboard

videotex a two-way form of electronic communication, usually where a user interacts with a large computer system to retrieve information selectively. Sometimes the definition is restricted to systems that provide graphics as well as text, though other times it can refer to any two-way system providing information to consumers and business

virtual a means of logically extending a physical device beyond its limits, by using all or part of a different device to simulate it

virtual display using memory to simulate a display larger than what can be shown at one time

VisiCalc the original spreadsheet program

warm start in general, any restart which allows at least some program or some data previously being processed to be recovered

wild card a special character that stands for 'any characters'. Example: a wild card search for COMPUT? might return a match on COMPUTER or COMPUTES. A wild card character might also stand for multiple characters in some programs, and could then return: COMPUTING

window use of the display screen as a viewing area within a section of memory

word a specific number of bytes, varying with the kind of machine, that can be manipulated by the computer in one instruction. Most 8-bit computers have 8-bit words, IBM mainframes have 32-bit words

Word Processing any system of document creation, modification, storage, retrieval, and printing. Some WP features are:

search/replace find a particular character string and exchange it for a new character string

cut/paste move a section of a document (block) to another part of the document or, in some cases, to an entirely different document

word wrap prevents words from being split up between successive lines, i.e., if a word can't be finished in one line, it is automatically moved *in toto* to the next line

justification lining up of text so that all characters touch the margin

Wordstar the first major microcomputer word processing program, and which serves as a standard against which others are compared

word wrap see *Word Processing*

write put out

write protect a mechanism used to prevent the alteration of a storage medium, e.g., the plastic tab on a cassette

X-OFF DC3, CTL/S, $13. The code sent on a communications line to tell the other end to pause transmission

X-ON DC1, CTL/Q, $11. The code sent on a communications line to tell the other end to resume transmission

zap manually change memory or files at the byte level, bypassing the normal procedure to do so

INDEX