Bell Laboratories Cover Sheet for Technical Memorandum



Title- **Experience with the Mergenthaler Linotron 202 Phototypesetter, or, How We Spent Our Summer Vacation**

Date- **January 6, 1980**

TM- **80-1270-1, 80-1271-x, 80-1273-x**

Other Keywords- **reverse engineering**

| Author | Location | Extension | Charging Case- **39199** |
|---|---|---|---|
| **Joe Condon** | **MH 2C-525** | **6694** | Filing Case- **39199-11** |
| **Brian Kernighan** | **MH 2C-518** | **6021** | |
| **Ken Thompson** | **MH 2C-423** | **2394** | |

*ABSTRACT*

In the summer of 1979, Center 127 purchased a Mergenthaler Linotron 202, a CRT-based digital phototypesetter. This paper discusses our experience with the device, some of what we have learned about how it operates, and the hardware and software we have developed to permit users to take advantage of its capabilities.

Pages Text 11 Other 2 Total 13

No. Figures 5 No. Tables 0 No. Refs. 3

Bell Laboratories

Subject: **Experience with the Mergenthaler Linotron 202 Phototypesetter, or, How We Spent Our Summer Vacation**
Case- 39199 -- File- 39199-11

date: **January 6, 1980**

from: **Joe Condon**
**Brian Kernighan**
**Ken Thompson**

TM: **80-1270-1, 80-1271-x, 80-1273-x**

*MEMORANDUM FOR FILE*

## 1. Introduction

Bell Laboratories has used phototypesetters for some years now, primarily the Graphic Systems model CAT, and most readers will be familiar with *troff* and related software that uses this particular typesetter.

The CAT is a relatively slow and antiquated device in spite of its merits (low cost, and until recently, high reliability). Most newer typesetters use digital techniques, rather than the basically analog approach of film stencil and optical plumbing used in the CAT. These typesetters store their characters digitally, using some representation of the character outline, and print on photographic paper by painting some area with a CRT. Figure 1 is a block diagram of a typical digital typesetter.

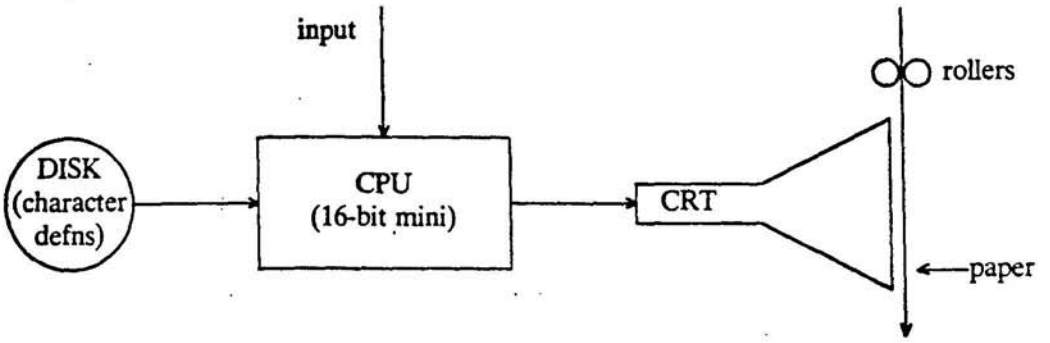



**Figure 1:** Basic Digital Typesetter

Digital technology has some obvious benefits. First, the typesetter can be much faster, since size and font changes, horizontal positioning, and perhaps some vertical positioning, are all done electronically instead of mechanically. This may well make the typesetter more reliable too.

Second, since characters are stored digitally, more sizes and fonts can be used, and there are no restrictions on the number of fonts that one can use concurrently.

Third, in principle at least, one can create new characters for special needs, or indeed entire character sets. Locally, the most obvious special character is the Bell System logo ⓐ.

Finally, these devices typically have much higher resolution than purely mechanical typesetters do, which opens up the interesting possibility of using them as plotting devices for line drawings

(such as Figure 1), graphs, and similar figures.

## 2. The Mergenthaler Linotron 202

In mid-1978 we learned of the recently announced Linotron 202 from Mergenthaler, one of the oldest firms in the business. (Otto Mergenthaler invented the linotype in the 1880's.) The 202 has a list price of about $45,000, and, with necessary options, still costs less than $60,000. This was the best balance of price and performance that we could find. Other contenders considered briefly included the Autologic APS-5 (which is used now by the Murray Hill Computer Center and even then by the Direct II project at Piscataway) and the Harris Intertype 7000. Both of these machines are twice as expensive as the 202, and thus out of our range.

Typesetter speeds are usually given in "newspaper lines per minute" (which says something about the primary market). The 202 is rated at 450 lines/minute, the CAT at 50, and the APS-5 variously from 1500 to 3000. In addition, since size changing is electronic, the size-changing operation that so much degrades the CAT's performance is not a problem.

The horizontal resolution of the 202's CRT is 972 scan lines per inch; the vertical resolution is stated to be 576 (but read on). This is plenty for drawing characters, and even for lines and figures. For comparison, the CAT resolution is 432 horizontally and 144 vertically; the APS-5 is 720 in each direction.

The 202 contains a commercial 16-bit minicomputer of modest capabilities, a Computer Automation Naked Mini. There are two floppy disks that store a formatting program and character definitions for about 60 fonts of about 100 characters each. Characters are brought into a fast "font memory" for actual typesetting; extra font memory can be added to reduce the amount of character set swapping.

Characters are represented by their outlines in an encoding that Mergenthaler will not reveal to customers; a special-purpose processor in the typesetter converts the outline into a sequence of vertical strokes (scaling the definition to the proper height and width) by which the CRT paints the character from left to right. Characters may be printed from 4½ points to 72 points high in steps of ½ point (one point is 1/72 inch); the width may be set independently in the same range, to achieve some rather striking effects. Characters may also be slanted 12 degrees forward.

Our main hesitation (apart from finding $60K) was the secrecy surrounding the representation of characters, since from the beginning we intended to create our own. Repeated discussions with Mergenthaler representatives, both in the United States and in their head office in England, made it clear that they would not divulge the principles of operation, not even under a trade secret or license agreement. Nonetheless, it appeared that it would be quite possible to use the 202 as just another typesetter using the programs that Mergenthaler provided to position Mergenthaler characters on paper. Furthermore, it seemed feasible to use the machine to draw figures, again with Mergenthaler's formatting program resident in the 202.

## 3. Preparations

*Troff*, the standard typesetting program, is remarkably wedded to the CAT. Accordingly, well before the 202 arrived, one of us (BWK) modified *troff* to be more typesetter-independent.

This required finding everything in *troff* that depended explicitly or implicitly on the CAT and making it into a parameter. Since the number of characters in each font and the position of particular characters within a font also depend on the typesetter in question, major changes in *troff*'s internal data structures were required. The new version of *troff* accepts an argument that names the desired typesetter, and reads a file of data appropriate to that typesetter to set all the parameters. These include horizontal and vertical resolutions, legal sizes and fonts, paper width, character names, and the widths of all characters.

The output from *troff* is no longer device codes specific to the CAT, but an anonymous intermediate language intended for a post-processor that will drive a particular typesetter. Post-processors currently exist for the 202, the CAT, the APS-5, and the Tektronix 4014. More details may be found in (1).

At the same time, JHC designed an interface to connect the 202 to our PDP-11 via a DR11-C. When the machine actually arrived, this was built and debugged. A description of this interface is given in Appendix A.

KLT wrote a device driver for the PDP-11. It is worth noting that since the typesetter is an active element, it is possible to read from it as well as write to it, a fact that we have made considerable use of.

## 4. Operational Aspects of the 202

The 202 turned out to be a considerable pain in the posterior from an operational point of view, at least until we learned to live with it. This section has no technical content, but may provide some entertainment. It consists of a slightly edited copy of a letter sent to the regional manager of Mergenthaler about six weeks after the machine arrived.

August 22, 1979

This letter is to ask for your assistance in solving the host of problems that we are experiencing with the Mergenthaler Linotron 202 which Bell Labs purchased early in July.

In its current state, the machine is completely unusable in any sort of production environment. Through some combination of hardware and software difficulties, we are unable to print any document of any complexity whatsoever. Furthermore, we have in hand several extremely small test cases which produce erroneous results. Mergenthaler personnel are unable to explain why they fail.

Perhaps the best way to explain the difficulties is with a chronology of events.

Friday, July 6:

The machine was installed early in the afternoon and passed all of the installer's tests. Not everything we ordered was delivered; among the missing were the regulated power supply, the extra cassettes, the extra diskettes, the slave and binary byte formatting programs, and information about warranty, customer course, etc. We had to scrounge an extra diskette from the installer before we could do anything at all. The only documentation on the programming of the machine was a manual that the installer told us "applied only to England", and some preliminary drafts of manuals supplied earlier by Michelle M.

The machine failed early in the evening; it printed only random black dots all over the paper.

Wednesday, July 11:

Ed P, a Mergenthaler applications person, determined (as we had already) that our problems were indeed a broken machine.

Thursday, July 12:

(Yes, a week after delivery.) The machine was repaired; the problem was a defective code converter board.

With the machine running again, we progressed far enough to learn that the paper tape reader on the 202 would not read the light-colored tape punched by our computer. This problem (whose possibility had never been mentioned by anyone at Mergenthaler) took a day to resolve, because it showed up as apparently random behavior; I thought that perhaps it was only that I was not using the font-handling utilities properly.

We also learned to our dismay that letting the machine sit with paper in it for more than a few hours caused the paper to stick to the roller; the next time the paper was advanced, it created an incredible paper jam that could only be removed with a screwdriver.

In the meantime, repeated phone calls still had not produced any action on the missing parts of our order.

Saturday, July 14:

When copying the pi fonts to our customer disk, I learned that several were defective; in particular the upper case Greek pi [sic] was missing from the universal math #1 layout; half of the characters were missing from universal math #2; and the widths of a variety of characters were entered incorrectly.

On the same day, program loading became erratic; by noon it was impossible to load programs from tape or disk.

Sunday, July 15:

The program-loading problem cured itself spontaneously. This uncovered a new difficulty: the typesetter simply stopped printing in the middle of certain characters, quite reproducibly.

Monday, July 16:

The program load problem re-surfaced, this time absolutely. The New York repair service assured me (by telephone) that the problem with loading was a defective CPU, and that a new one would be shipped from Chicago immediately.

At the same time, Michelle M told me that the problem with printing only half a character was probably a defect in the font handler program that had been shipped with the machine; she promised a new one, and as well replacements for the defective fonts.

The new cassettes arrived.

Thursday, July 19:

The new CPU board arrived, but failed to solve the program load problem. Our own detective work demonstrated that the problem had actually been a defective program memory board.

Friday, July 20:

A new memory board was installed; this cured the program loading problem. The problem of printing half-characters turned out to be a broken code converter board, not the font handler program. This was repaired.

You should note that by this point we had had the machine for two weeks, during which time it had worked for a total of less than 24 hours.

Sunday, July 22:

Our software had been ready to drive the 202 for some time. When the machine finally settled down, one of our early test jobs caused the 202 to give time-varying results: each run would cause a different amount of 202 output before it stopped with an error message. The error message bore no apparent relation to the input. A tape of this problem was sent to Michelle M for explanation.

Wednesday, July 25:

The machine broke again, this time printing parts of characters, random gibberish, vertical lines in place of characters, and similar effects. The intensity dropped substantially at the same time. Lyle R, our (by now regular) repairman, suggested wiggling the chips [sic] on the code converter board. This apparently nonsensical suggestion in fact cured the problem for a while.

Thursday, July 26:

The same chip-wiggling exercise became necessary every few hours, from this day forward.

We were also assured this day that the replacements for our defective fonts had been sent from Long Island several days before via UPS. The new font handler program was received, but not used, since it was not related to the problem. Somewhere around this time the power supply and diskettes finally arrived.

Thursday, August 2:

The machine had been running erratically for nearly a week. Throughout this time, we had had very severe problems with apparent light leaks that caused dark smudges every few inches down the paper, with regular paper jams, and with paper tearing as it came out of the cassettes into the developer. I have already mentioned that if the machine were left with paper in it overnight, by morning the paper would be stuck to the roller so strongly that it would be necessary to wash it off. We were told that the problem was that we were using stabilization paper and the machine was designed for resin-coated paper. Page 12 of the brochure "An Inside Look at Linotron 202" lists "Grade S photomechanical paper" (that is, stabilization paper) as the first choice of output material.

As it turned out, by disabling some of the mechanism that maintains pressure on the roller, we were able to eliminate the paper sticking. This also turned out to be the "light leak": the pressure was so high that it sensitized the paper.

The paper tearing problem has still not been solved; the Mergenthaler cassettes are too wide, the paper slops around inside them, and then tears if it is pulled out even slightly off straight.

Friday, August 3:
The machine died finally; no amount of chip-wiggling had any effect.

Sunday, August 5:
The machine spontaneously cured itself, at least for a while.

Monday, August 6:
Early in the morning (in fact at 4 am -- one works as hard as possible whenever the 202 actually functions) it died again. Erratic behavior continued through August 7, 8 and 9.

Thursday, August 9:
A new code converter board was installed. This had the interesting property that it printed 10-point letters at 9-point width. The repairman took it back with him, leaving us with the earlier (and still non-working) board.

Friday, August 10:
A new code converter board was received (I haven't been counting, but it was either the fourth or the fifth) that seemed to solve the problems for the time being.

At this time, a mechanical design problem surfaced. The braided cable that connects the anti-static plate to the frame is uninsulated; it can be (and was) set down in such a way as to short out the no-film sensor. This caused the front panel lights to short out too. We added insulation to the cable so this problem won't happen to us again, but I commend it to you as a worthwhile modification.

Sunday, August 12:
The intensity became erratic -- one job would come out properly black; the next would be too faint to use.

Monday, August 13:
One of the fans failed. They are soldered in rather than plugged in; one has to be a contortionist to replace them, so it was left for the next repair trip.

Tuesday, August 14:
For some time, we have been having troubles getting certain jobs through; in fact, any job more than a couple of pages long or more than slightly complicated is certain to fail. One standard symptom is that the typesetter begins to flash the RUN light and signals error 5 ("off-range height or width"), even in text that contains no height or width commands. This is problem P1.

A second problem is called P2: in the middle of apparently innocuous text, random garbage appears. This problem is completely reproducible, and, since it contains fragments of characters, obviously caused by some hardware or software error in the 202. Michelle M had no suggestions about what could cause either of these problems.

I have not named the myriad other ways in which it fails.

Friday, August 17:
Michelle M says that the tape which produced time-varying results on our machine worked properly and identically each time on her machine. I consider this to be prima facie evidence that our machine is defective.

The intensity is still erratic.

Tuesday, August 21:
We produced some very small example tapes that cause inexplicable and wrong behavior from the machine. (We used tapes so that it can't be blamed on the on-line interface.) A new repairman, Dave V, replaced the fan. He found a font memory board that did not pass his diagnostics, but did not replace it. He suggested replacing the high voltage supply to cure the variable intensity problem. He verified the erratic behavior of the tapes, and carried them off to Long Island.

He also gave us the slave program instead of the H+J, in the hopes that it would solve problems P1 and P2. Interestingly enough, when the slave program is used, the data that used to cause problem P1 now causes problem P2 -- garbage. The data that used to cause P2 works fine. Strangely enough, the slave program will not draw vertical rules; Michelle M assures me that it is supposed to, so I infer that the rule-drawing function is tripping over some hardware problem as well.

There any number of niggling little things that I could continue with; for example, I just this day received the replacement fonts. They were addressed to "Bell Labs" only, and so sat in our Receiving department until I stumbled over them accidentally. I have not yet received the information about the customer course that was promised a

month ago. No one can explain to my why a 1-em dash is actually 1.04 ems long (which is not a lot; but it makes it hard to draw lines with it). But I'm sure you see the general theme.

As you can see from this tale of woe, our particular 202 is not "packed with reliability and convenience features", as your brochures suggest. In fact, it's an unmitigated disaster. In the first two weeks of ownership, the machine worked for about 24 hours. Thereafter, it has been so erratic that at best we can only test our code from time to time; when we do, we can't tell if errors are our fault or yet another problem with the 202. There is no way that we can put this expensive lemon into production.

The only positive comment that I have is that the individuals that I have dealt with at Mergenthaler have without exception tried to help; each seems genuinely concerned. In particular, Michelle M and Lyle R have been remarkably patient with my daily phone calls seeking advice and help.

But no one person at Mergenthaler seems to have the combination of knowledge and time that it will surely take to make this machine function properly. I would appreciate it if you would inform me of what action you can take to convert our 202 into a working machine.

In subsequent weeks, we had to replace another memory board and myriad light bulbs. We never did find the light leaks. The machine is highly susceptible to static electricity; the tiniest shock pushes it into catatonia, from which the only escape is to power off and on again. (Better grounding helps, but does not eliminate it.) We discovered that the addressing switches on one of our memory boards had been set incorrectly by the repairman, so we had not been using it at all. Mergenthaler finally notified us that our problems with time-varying output had been a bug in their formatting program. And so on.

## 5. Reverse Engineering

In parallel with our efforts at making the device work at all, we spent quite a bit of time trying to figure out *how* it works, and what the character set encoding is.

The Achilles heel (or foot in the door, depending on one's taste in metaphor) of the 202 turned out to be its archaic paper tape orientation. All programs are supplied on paper tape. One of them makes a directory listing of a floppy disk, printing it on the typesetter itself. Since this program has to print something regardless of what it finds on the disk, it follows that it must contain an actual character set; this observation was made by all of us concurrently. Accordingly, KLT set about writing a disassembler for the 202 CPU. (Fortunately we had thought to obtain manuals for this well in advance.)

The disassembler was working within a few hours, and by late that evening, we had found the character definitions. An entire night of staring at them (in hexadecimal) produced only limited insight, but within 48 hours, we could print characters whose form was simple enough. The alphabet on the paper tape was upper case Helvetica, RATHER LIKE THIS. This was fortunate, for the simplicity of Helvetica gave us a toehold: letters like I and L are good places to start decoding.

More characters were needed for scrutiny and checking hypotheses, however. By studying the manual for the CPU and the disassembled listing program KLT was able to write a program that would copy a floppy disk into the PDP-11 where it could be examined more easily. It took only an evening to figure out the "file system" on the floppies, and thus to extract all of the character definitions into UNIX files.

There followed about four weeks of intense and generally frustrating introspection on bit patterns, trying to infer how they could possibly represent the character that they did. Although the details are not particularly germane, the broad outline is this.

Characters are represented by vectors that, taken in the proper order, define the lower and upper limits of inked areas as the hardware steps across the character definition from left to right. Figure 2 illustrates this point.

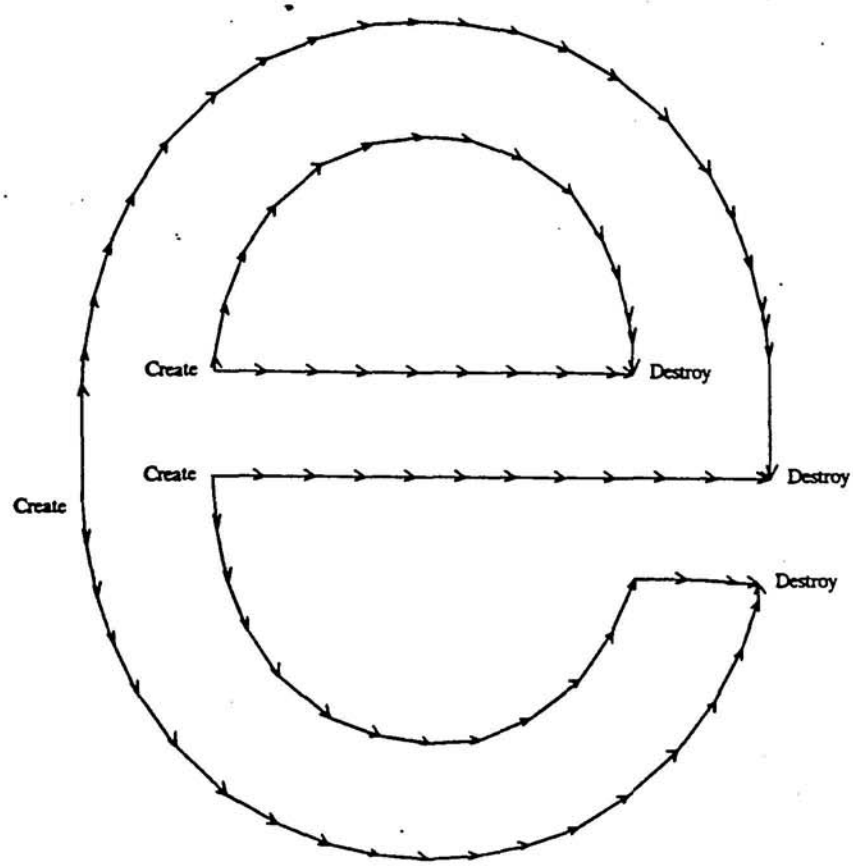



Figure 2: Character representation in the 202

The conversion process is best thought of as a finite state machine that reads a sequence of bytes describing the character. The first few bytes describe the position of the character—how much space is to the left of it, and how far up its first component is. Subsequent bytes are primarily the $x$ and $y$ components (4 bits each) for the next vector. (There are a lot of vectors in the e above because 4 bits does not allow for very large steps.) One $x$ value is special, however, and indicates an "op-code" for the machine. For example, one opcode is used to indicate a change in the sign of subsequent $y$ values, since the 4 bit chunks are unsigned. Several others indicate the initiation or termination of one or more pairs of vectors, such as at the start and end of the horizontal bars on a character like

e

There are also opcodes for giant vertical steps, word alignment, and continuation of horizontally disconnected characters like the double quote ".

By far the most difficult part of the job was not deducing what individual op-codes mean but determining the order in which the hardware processes the set of vectors that are "active" as it steps across the definition of the character. The individual bytes are not labelled in any way; what they apply to depends on the state of the machine when they are encountered.

The encoding used by the 202 is quite compact. The table below shows the average number of bytes per character needed to store the letters and digits in several fonts of varying complexity. Bear in mind that a single representation is sufficient to define the character up to at least 72 points

(one inch).

| | |
|---|---|
| Helvetica | 91 |
| Memphis Medium | 104 |
| Times Roman | 113 |
| Old English | 168 |

While the character set decoding was going on, we also wrote a program to read paper tape from the 202 so we could study the other programs provided by Mergenthaler, and find out how to drive the device with our own software. We wanted to do this mainly because it was too hard to defeat their software—it believed very strongly that it was a general purpose formatter and knew what it was doing; heroic measures were needed to disabuse it of this notion. Furthermore, it became clear that it was riddled with bugs, and that we simply could not go into production with it.

Again, the details are uninteresting. The job required figuring out how to put characters into font memory and get them back out again, how to control the panel, and (hardest by far) how to drive the CRT. For the most part, this was accomplished by poring over listings of disassembled Mergenthaler programs, coupled with abortive experiments on the 202.

We did ultimately determine that Mergenthaler had not represented the capabilities of the machine accurately. Although the range of points sizes is officially 4½ to 72 in steps of ½, the hardware itself will run from ½ to 127. (Large characters may not fit on the CRT, however, which is only 96 points high.) Character widths are limited to between 4½ and 96, somewhat greater than the "official" 4½ to 72. The machine is capable of both forward and reverse slants; the reverse slant is not accessible from Mergenthaler software and thus not advertised, perhaps because it seems a dubious feature anyway.

Most important for our hopes of doing graphics, however, was to determine the true value of the vertical resolution. Various Mergenthaler documents list this as 288 per inch, 576 per inch, and even 1152 per inch. Which is correct? Surprisingly, none of the above. The true value is 486 per inch, a number that occurs in no document that we have seen. 486 is convenient, however, since it divides the horizontal resolution, 972.

### 6. Our 202 Software

As we mentioned above, we wanted to get rid of Mergenthaler's software as quickly as possible. The first step was an assembler; this was written by KLT shortly after the disassembler was, partly as a test of the disassembler.

A second and vital step here was to decode the boot program in the 202, which was contained in a 512×4 PROM in the CPU. The four-bit chunks from the PROM were simply combined into eight-bit bytes in a variety of plausible ways and run through the disassembler until some combination produced assembly code. Once the boot sequence was understood, we modified it so we could boot from the PDP-11 instead of from paper tape.

Although assembly language is adequate and efficient, it is not fit for most people to use. Accordingly, KLT resurrected the B interpreter.[2] B is a good language for a 16-bit word-oriented minicomputer, and most of our subsequent 202 programs have been written in B. (The assembly language run-time support has since grown somewhat to make things run faster.)

The current software arrangement is as follows. Any program may be loaded from the PDP-11. Normally, however, a simple boot program is stored on one of the floppy disks, and the machine is booted from that. This program in turn will load one of four standard programs from the floppy, selected from the front panel.

The program that is normally run is the "dumb slave" that we wanted originally. It accepts commands to move up and down and back and forth by specified amounts, to change size and font, and to print a character. It also does some device control functions, primarily flashing the lights to indicate progress. Since we began using this program instead of Mergenthaler's formatter, most of our "hardware" problems have disappeared.

Other programs list the contents of disks, copy a disk to or from the PDP-11, and read a paper tape. There is also a memory diagnostic sufficiently better than Mergenthaler's that the repairman asked for a private copy (which we provided, on paper tape).

The format of floppy disks is a simple two-level file system. A "super block" points to entries for up to 256 fonts; each font in turn points to up to 256 characters. Font 0 is special: it contains the programs that can be selected after booting. The search for a font is done in a particular order, so a new font can overlay an old one merely by being on the other disk. Disk images are created on the PDP-11 and copied over by a simple program running on the 202.

The formatting program loads characters into font memory from the floppy disk individually, rather than a font at a time. In effect, it maintains a cache of recently used characters. The replacement strategy for the cache is to flush it completely when it fills; since the font memory holds about 900 characters, this is not as silly as it sounds.

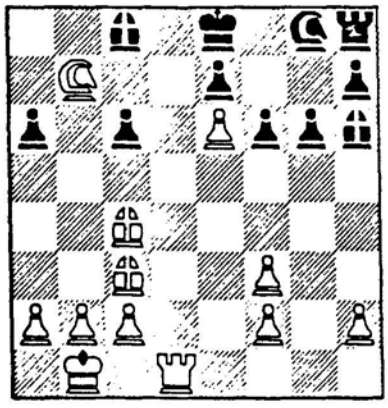
Our formatting program does not drive the 202 as fast as Mergenthaler's does, at least for "newspaper lines"; we run at about 320 lines/minute on a sample text for which they run at 425. We think that the discrepancy is because we are using an interpreter, not because we don't fully understand the machine, but this remains to be proven. We are much faster on material that involves special characters (mathematics, for instance), because loading and retaining individual characters instead of entire fonts is superior to the Mergenthaler algorithm of swapping in an entire font each time a special character is referenced.

Our disk format is less flexible than Mergenthaler's, since it does not reclaim storage if a font is updated. But it packs about 50 percent more characters onto a disk, so it has some merit.

### 7. Preparation of Character Sets

As mentioned, we had hoped from the beginning to be able to create our own characters for the 202, either from artwork or algorithmically with a system such as METAFONT(3).

The first character we created was ⊙. The original artwork was scanned on a Dest scanner (thanks to Peter Denes), and converted into its Mergenthaler representation. With this much experience in hand, we turned to other necessary character sets. KLT, whose avocation for chess is well known, contributed characters for figures like this one:



**Position after 23 N×b7**

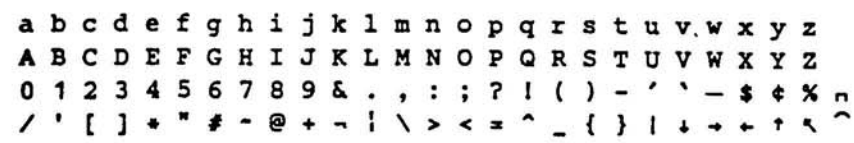
The "printout" font (i.e., constant width) supplied with the 202 was missing many of the ASCII special characters, so we added them by creating vector representations directly (with no artwork); this provided us with a complete ASCII alphabet suitable for printing programs and the like:

```
a b c d e f g h i j k l m n o p q r s t u v w x y z
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
0 1 2 3 4 5 6 7 8 9 & . , : ; ? ! ( ) - ' ` — $ ¢ % ¤
/ ' [ ] * " # ~ @ + ¬ | \ > < = ^ _ { } | ↓ → ← ↑ ↖ ^
```

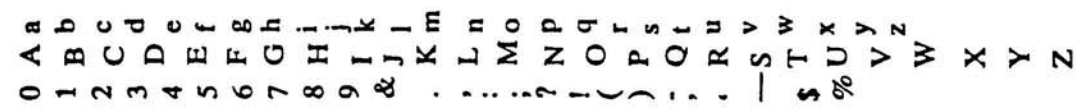
Similarly, we created the bracket-building piece-parts that the CAT provided:

by editing the definitions of Mergenthaler characters like { and (.

Recently we obtained artwork for the Courier-like constant width font that had been available on the CAT; this was scanned and converted, into




Although the 202 has no capability for rotating characters, it is possible to perform such transformations on fonts to make new ones. For example, we have a rotated Times Roman font for labelling axes on graphs, although so far no non-frivolous applications have arisen.



The software for character manipulation is rough and ready, since it was thrown together as quickly as possible. Accordingly, we will not attempt a precise description of any of it, but instead simply indicate the capabilities that exist. More details may be obtained from the authors.

Given a character specified by a set of vectors describing its closed contours, two programs *merg* and *font* convert it into a 202 representation: *merg* converts arbitrary vectors into ones that satisfy the limited range (4 bits) in the Mergenthaler internal representation, and *font* converts this into actual 202 codes. Another program (historically called *wec* [sic]) simulates what we believe the 202 does, and prints the character on a Tektronix scope. Yet another program (*bec*) converts from the 202 representation into closed-contour vectors—in effect, the inverse of the transformation done by *font*. Most of our other software deals with these vectors.

There are also a handful of programs for dealing with data from a scanner. These include cropping, rotating, scaling, and flyspeck eliminating.

## 8. Some Conclusions

On balance, we believe that our experience with the 202 has been favorable, although there have been some frustrating and anxious times. Since we stopped using Mergenthaler software, the machine has behaved well, except for the paper-tearing problem (not as severe now but still extremely irritating) and occasional difficulties with static electricity.

We have developed a useful repertoire of software for defining new characters. Most of this is geared towards scanned analog data, but we have some *ad hoc* tools for creating simple characters out of nothing. Much remains to be done to make this software accessible to non-specialists. We also have no convenient way for users to load private character definitions on the 202 temporarily, although this would be easy to support.

The speed and resolution of the 202 have already served as the impetus to improve our capabilities in typesetter graphics. Several programs now exist that permit us to draw simple figures like the ones in this memo, and *troff* itself has been modified to support line and arc drawing. Much remains to be done here, too, both in algorithms and in user languages. Fortunately, whatever is done in this area will not be limited to the 202; most of it ought to be applicable without change to the APS-5 and perhaps the FR-80.

We also have to drive the 202 better; although it can deal with arbitrary up and down motions, it is faster and more accurate if its input has already been sorted vertically. We also make no use of the full area of the CRT; many motions that could be purely electronic (and thus fast) are done mechanically (and thus slowly).

One interesting portent for the future is that Mergenthaler has just announced a new machine called the Omnitech 2000. It is a laser typesetter with high resolution (723/inch), reasonable speed (probably about one third of a 202), apparently sensible hardware design, and a cost of about $30,000. We believe that it uses essentially the same character encoding that the 202 does. This machine might well be a reasonable alternative to the CAT.

Micro-5

**Joe Condon**

**Brian Kernighan**

**Ken Thompson**

**References**


1. Brian W. Kernighan, A Typesetter-independent TROFF, TM-80-1273-xx.
2. K. Thompson, *Users' Reference to B*, TM 72-1271-1.
3. Donald E. Knuth, "METAFONT, a system for character shaping," Stanford University Department of Computer Science Technical Report (1979).

**Appendix A: Hardware Interface to the Linotron 202**

The Linotron 202 has a parallel byte with parity interface to connect to a computer. A simple circuit was made to connect that to a parallel interface unit (DR11-C) on the PDP-11/70. The Mergenthaler unit uses optically coupled isolators, and a different handshake than DEC hardware does. Schematic and sequence diagrams are given in Figures A1-A3.

Figure A1 shows eight bits of signal J1 coming from the 11/70, a parity generator (which the Linotron requires) and nine open collector drivers with protection/termination resistors that provide signals to the Linotron. Also shown is the data path in the other direction which consists of Litronix IL100 optically coupled gates and drivers into the 11/70.

The top of Figure A2 shows the handshake for the data path from the 11/70 to the Linotron; the lower half is for the other direction. Signals csr0 and csr1 are used only for debugging; they are normally high. In the discussion that follows we shall talk as if the interface were coupled back onto itself by connecting the signals sc1 to sc2 and connecting ac2 to ac1.

The sequence begins with the data receiver signaling that fact to the sender by asserting ac2. The sender is either already waiting and gets an interrupt via rqsta right then or he gets the interrupt when he becomes ready and turns on his own interrupt enable. The interrupt handler of the sender then presents new data onto the data lines and pulses the line ndry (new data ready) which causes ff0 to assert signal sc. The receiver detects that and asserts rqstb which causes an interrupt into the receiving machine. When that interrupt is serviced that machine pulses the line dmxt (data transmitting). The trailing edge of that pulse indicates that the receiving machine has taken the data, and so it is used to de-assert the signal ac2 by clocking a zero into ff1. This causes the direct setting of ff0 which causes the de-assertion of sc, and so ff1 is also set. This removes the setting signal from ff0, causing another interrupt rqsta, and removing the setting of ff1.

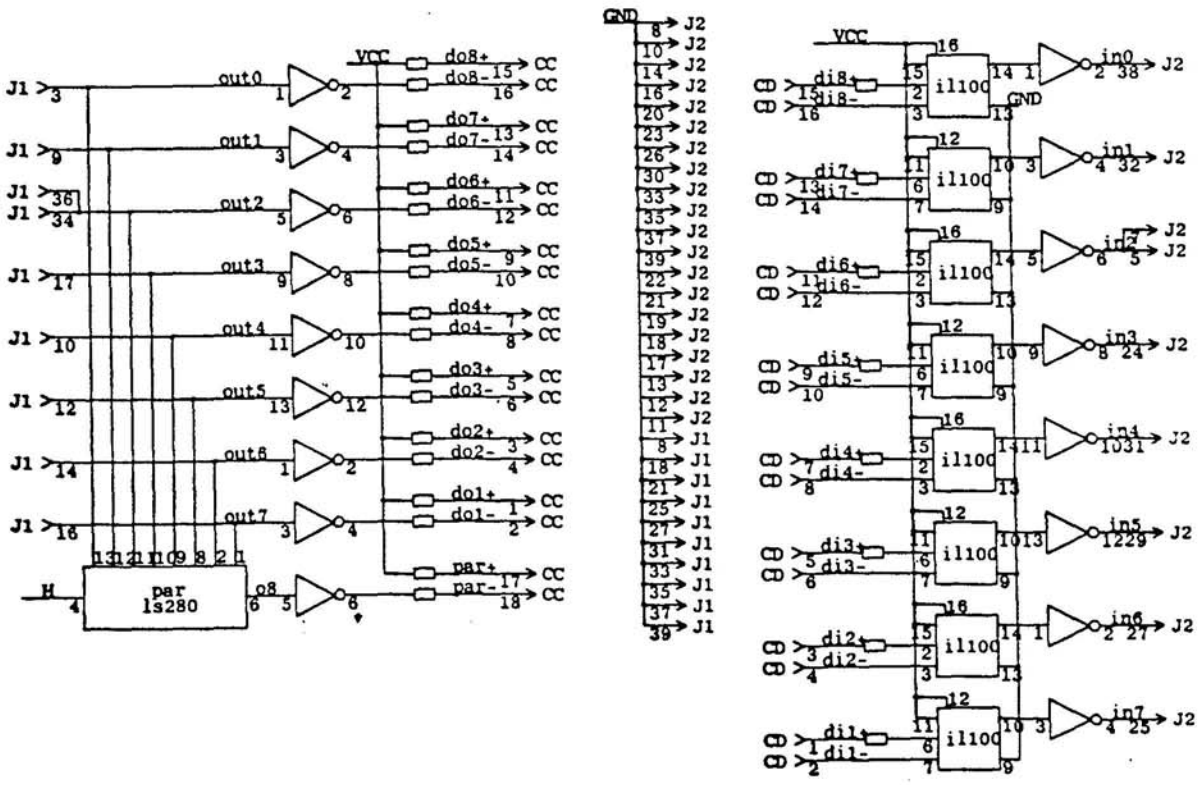



Figure A1: Interface between Linotron 202 and DR11-C

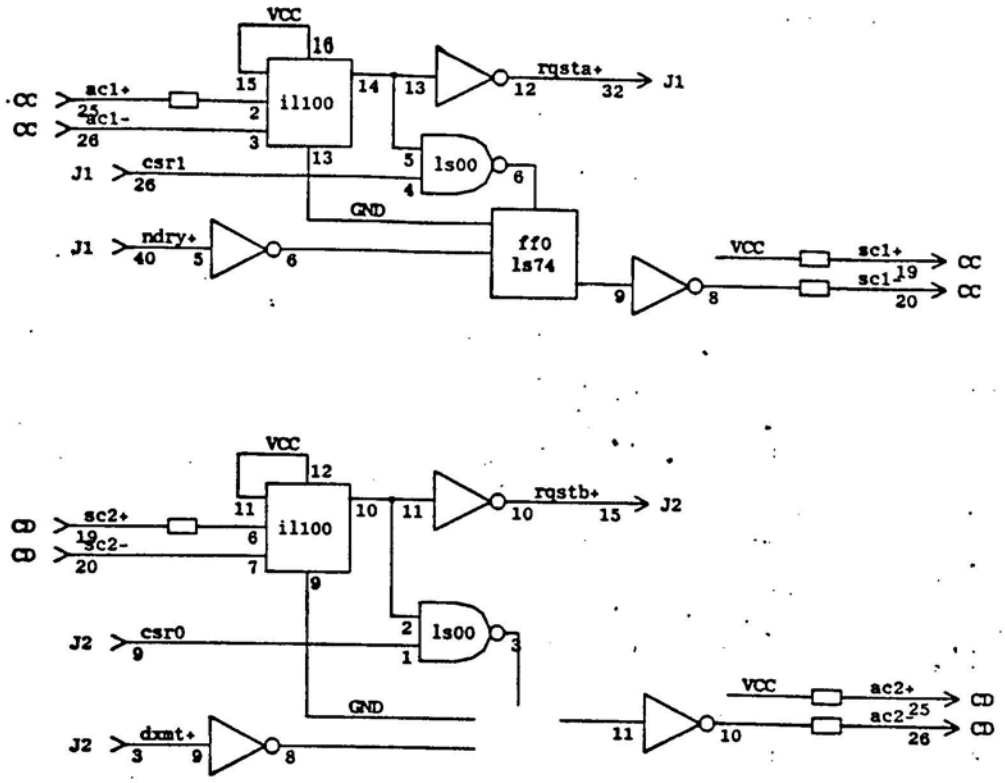



Figure A2: Interface Signals

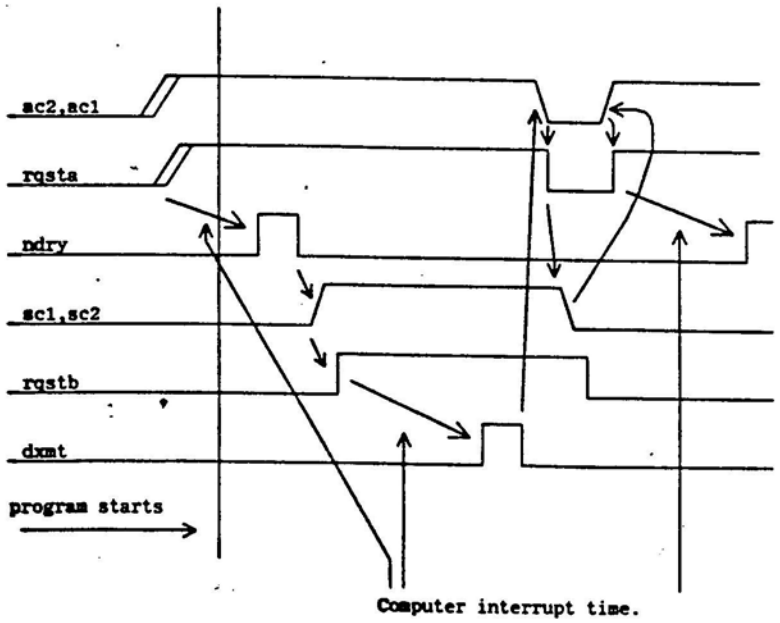



Figure A3: Interface Timing