July-September 1988 Volume 2. Issue 3.

Scorpio News

Scorpio Systems
P.O. Box 286 · Aylesbury · Bucks · HP22 6PU

Contents

Editorial

Just a very brief Editorial this time. Thank you to the people who submitted the articles for this issue, especially the 'new' authors. We have in fact ended up with a small amount of material in hand for the next issue. But don't let this stop YOU (yes, YOU) sending something in.

The reason for the brief Editorial is that there seems to be no 'news' to report. We haven't received a Gemini Dealer Newsletter since December, and have received no other communications from any other related manufacturer.

On the New Product front, we have heard from one subscriber that he has both a Gemini GM880 64180 and a Gemini GM890 Z280 board on order, but has not yet received either. However, we do understand that the 64180 is becoming imminent (RSN — Real Soon Now, as they say in Byte). And, as stated in the last issue, we are told that the Gemini GM886 80286 board is in production. So perhaps we can look forward to reviews of these two latter boards in the next issue of Scorpio News. Any volunteers? Until then, *"Be seeing you"*. •

Letters to the Editor

Norway Calling

Dear Sir

Here is also a letter from Breda, the Netherlands. In brief the description of my Gemini computer system. 19" Vero frame with 8 slots motherboard, in deep (40 cm) 19" case, 5U high. The boards are GM811, GM802, GM802 (drive M:>), GM832, GM829, Arfon lightpen! The keyboard is GM827 with 87 keys. The drives are 2 x Teac FD55-F. The PSU is GM843. Printer: Citizen 120-D. Software is e.g.: Graphpac, Gempen VG:1.

I'm using Gempen for letters, stamp collection, LP collection and so on. In text I always use the <ESCAPE>s and <cntrl>s for the printer's character-set. I found out it was possible to control the printer, in the Gempen manual there was (is) a lack about it. (Manual Graphpac >>> FILL ?). The cursor is flickering instead of blinking, how can I alter that? I am not a great programmer, but I am a hobbyist. And that's why I ask for a little program for the <lightpen - GM832> in MBASIC.

In early times I had a Nascom 1 with RAM A. The story is well known. I learned many things with it..... Then I saw Gemini. Many, many years later the Scorpio News came and I was happy again....

Something about IBM (...): I do not like IBM at all and all that kind of things like those IBMs and clones.

Yours sincerely, G.C.M. Vissers, Breda, The Netherlands. •

"Real Programmers"

by V.A.R. Ious

At a party, the Real Programmers are in the corner talking about operating system security and how to get around it. At a football game, the Real Programmer is the one comparing the plays against his simulations printed on 11" by 14" fanfold paper. At the beach, the Real Programmer is the one drawing flowcharts in the sand. •

Adding a Winchester to your System

by I.M.Cullen.

Some 18 months ago I was being given a guided tour by a fellow Nascom-Nut of his system, and its latest addition "the Winchester". Whilst studying the thing enviously, I asked where he obtained the finance, Bank Robbery? Pools win? Rich aunt died? No! He insisted, it's much cheaper than you imagine. It is indeed, and not as difficult to install as you might think.

"but they're costly"

A look through the Gemini price list reveals that you would need to lash-out something like £1000, but using second-hand components it can be done for perhaps a fifth of that price. This is about the same as you might pay for a good size RAM-Disk card.

"I don't mind changing disks"

So what are the advantages that a winnie gives you? The answer can be broken down into two things, speed of access to your files and a large amount of storage. The two things are of course interlinked, but let's look at speed first.

With a normal floppy drive, data is transferred to your system at the rate of 256 Kbits/sec (500 Kbps for 8" drives), with a winnie using the SASI/ST506 interface the rate of transfer is 5 Mbits/sec, over 20 times faster than a floppy drive. The step up to a winchester system from floppies is like stepping up from Cassettes to floppies.

In addition although the stepping rate of most winchester's is nominally 3mS, most modern microprocessor controlled winnie's can accept step pulses much faster than this (known as 'Buffered Seek') in some cases separated by as little as 16uS, these are buffered by the on-board micro and the stepper motor is driven faster, until it gets close to its destination where-upon it switches back to 3mS for the last few steps. However it must be admitted that this much higher rate of stepping can be somewhat offset, since most winchesters have considerably more cylinders to step over than even an 80 track disk.

Perhaps the biggest boon to having a winnie based system, is the large amount of storage which can be accessed (at high speed). Even a small winnie with perhaps 5Mb can hold many (all?) of your programs and data, so avoiding the need to swap

disks, or even find them! Programs with overlays are no longer a pain to run, since the overlays load quickly, and you'll never have the wrong disk in drive A: again.

"there must be some disadvantages"

Apart from having to buy and install the winnie system, there is a down side to running them. The drive motor in the winnie runs all the time and at 3600rpm, so there is a certain amount of noise, but the more modern the drive (say 1984 on) the less noise, it may even be less than the fan already in your system. In addition they do take a lot of power, which could mean a different PSU is needed.

"so what bits will I need"

This depends very much on what you have already in your system. Assuming a minimal CP/M disk based system, the CPU card should be fine, be it Nascom II, Gemini or MAP-80. 64K of RAM is probably not essential, but preferable. Shown below is a shopping list of items you will/may need:

- SASI Interface – GM829/GM849 Floppy Controller or MAP-80 MPI card
- Controller Card – Xebec S1410 Winchester Controller
- Winchester – almost any ST506 type interface drive.
- PSU – must have the AMPs on +5v & +12v
- Cables – 3 ribbon, two power & connectors
- Software – Gemini CP/M 2.2, upgrade required. MAP80 CP/M 2.2 or 3.0 can be re-assembled

The problems may start with the Floppy Controller card, since one with a 'SASI' interface is required. The GM809 will be no good (no an GM805 won't do either), but a GM829 or GM849 will be fine, also a MAP-80 MPI card will do well.

The next item is the Hard disk controller, which connects to the SASI interface from the floppy card and operates the winchester drive. Typically the card to choose is the Xebec S1410, as used by Gemini. These are available Second-hand, but you may have to scan the adds for a while. There are alternatives such as the Konan 'David Junior' model, but Gemini & MAP-80 CP/M software does not support it. I believe there is a version of SYS which supports the Konan running with a Nascom II.

Now to a winchester, the Xebec card (& Konan) is designed to connect to Winchester's using the Seagate ST506 interface which is very popular. This means there is a wide assortment of winchesters of all capacities to choose from. Your CP/M software may however limit your choice of winnie for reasons I shall explain shortly.

The third item of hardware required, is some means of powering the beast. When a winchester is started, the motor draws a large amount of current from the +12v supply. In addition both the winnie & Controller card guzzle amps from the +5v supply. Typically amperages are shown below:

	Winchester:	Controller:
	Rodime 200 Series	Xebec S1410
+12V	4.0 amps at start, reducing to 66 mA (max)	2.4 amps (max) after 30secs
+5v	0.65 amps (max)	2.5 amps (max)

In addition there is the rest of the system to power. As a general rule of thumb something like a 100w supply is needed. However the later Gemini GM817's (85w) are able to supply high amperages for a short while and will run a winnie system quite happily.

Last, and certainly not least is the software. How do you tell your CP/M implementation about the winnie. Well let's deal with Gemini CP/M 2.2 users first. It will be necessary to acquire a new CP/M with a winnie BIOS. Gemini can no doubt supply an upgrade if you return your Master disk. The later Gemini CP/M's BIOS 3.3 for example have a SYSTEM.CFG file which is edited to indicate the type of winchester and what configuration is required. It will as far as I understand only cope with Rodime 200 Series drives, and some NEC and Miniscribe drives, so this limits your winchester options. The Gemini winchester CP/M does have a nice feature in allowing you to Boot direct from the winchester.

If you are running a MAP-80 CP/M 2.2 or CP/M 3.0, no further purchases are necessary, you may merely edit your BIOS files and place details of your particular winchester into them, then re-assemble and generate a new system. Apart from saving some cash, the MAP route has another advantage in that most any type of winchester can be catered for. If you need to upgrade your CP/M to cope with the winchester a phone call to MAP-80 might be worth your while.

Putting it all together

Apart from arranging the winnie & controller's place in the card board box, the rest is all down to links & cabling. I shall assume you have manuals for your CP/M and Floppy/SASI card, it's useful to have one for the winnie, and not really essential for the Xebec controller.

Both the winnie & controller card use standard Floppy drive power connectors for +5v & +12v, good quality cable & short runs (2 feet or less say) is the order. Do check correct polarity before connecting winnie or controller, and check good voltages & ripple after connecting them.

Three ribbon type cables are required. A 50-way cable connects the SASI port to the controller card, which should have a 220/300 ohm pack terminating near the connector. A 34-way ribbon cable using a similar layout to that of 5.25" Floppy drives, takes control signals to the winnie (or winnie's), the last winnie on this cable should have a termination resistor pack. A separate 20-way ribbon cable takes read/write data to & from each winnie drive, the first drive should connect to J2 on a Xebec.

Up to 8 controller cards can run from one SASI port!! The address for your one controller should be 0, linked near the SASI connector. In addition the Xebec can store data in 256 byte or 512 byte sectors on the winnie, the link should be set to 512 for Gemini/MAP CP/Ms, the link is typically three pin, marked with 2 & 5.

The winnie should be set as drive 0, on Rodimes this is set by a switch accessed through the side casing.

Configuring the software is the final job, for Gemini CP/M you merely select which type of drive you're using in the SYSTEM.CFG file and configure a new system onto a new disk (i.e. keep the old floppy system handy in case things don't go!). For a MAP CP/M, where you are using a winnie not in the MAP data, you need to know the following winnie details:

Cylinders
This is the number of tracks on one platter surface (if the drive has 152 cylinders & 2 surfaces, it has 304 tracks).

Heads
This tells the BIOS how many surfaces are in use. (A drive with one platter has 2 surfaces, hence 2 heads.

Reduced Write Cylinder
Many drives require the write current to be reduced beyond a certain cylinder. Micro controlled ones often do this automatically, hence this may not be relevant.

Write Precompensation Cylinder
The data is written in a different manner beyond a certain cylinder (some times all cylinders, hence enter 0) by the Controller to improve readability.

Fast Step

Older drives require their step pulses at 3mS intervals, newer ones with 'buffered seek' accept step pulses in a quick burst. The FAST step equate is hidden away in the MAP BIOS. Normally it's set for fast stepping (a value of 5), but for slower drives such as the Seagate 506 a value of 1 is needed.

For any ST506 type winnie just 4 (possibly 5) values need setting, there's no format info.

Lets Go !!!

With all the new bits plugged up, and a new system on the disk, it's time to try it out. The winnie should start spinning with power on & take about 20 secs to get to speed. Meanwhile the boot disk should start loading the CP/M, then nothing will appear to happen until the winnie reaches speed and self-tests in some cases.

With Rodimes the power-on light will flash while the winnie spins- up. It will then stay on after correct speed and a second light should flash as the CP/M accesses the winnie. If the power-on light appears to flash in morse code, that's because it's flashing an error message, you will need to look-up the error in the Rodime manual and take any action that Rodime prescribe.

If all's well after 30 secs or so you should be able to type DIR A: (or B:/C: if the winnie is partitioned) and get "no file", or perhaps a list of juicy files left on by the previous owner.

But what if all does not go well? Firstly check all cables for correct orientation, there's several. If reversed it should not do any damage, except power cables (smoke may accompany this fault!!). Check you have booted off the winnie CP/M disk. If the system hangs (wait a minute or so) it indicates that the BIOS cannot talk to the Xebec. If an error message appears it probably means the Xebec cannot talk to the winnie. Run a winnie test/format program this will indicate if the Xebec/winnie are OK and give any errors. It maybe necessary to re-format the winnie before it will run properly. Try this, if formatting appears to work, things can't be too bad.

"Hummmm"

Has all the above set you thinking about winnie-ing your system? If so, give it a go, because I think you'll enjoy the performance increase and it's not so difficult. If you need further help I will be quite happy to assist, and can supply a program ('wot I writ') to format/test the winnie & controller, for most any winnie.

Ian Cullen, High Hall Farm, Nettlestead, Nr Ipswich. IP8 4QT. (0473) 831353 •

The Use of Overlay Files

By R. Pearce

Undoubtedly one of the biggest advantages of the newer 16-bit micros over our old Z80 systems is the sheer size of the available memory. This extra memory allows more complex programs to carry out more involved operations on larger blocks of data with fewer disk accesses. The shortage of memory space can be a major drawback on 8-bit systems, but there are ways around it.

Perhaps the classic example of memory shortage is the text editor/word processor. Here we have a program with many tasks to perform on a large amount of data. The more options we put in the less text can be held in memory. Some cheap editors such as HiSoft ED80 or GEMPEN simply restrict the facilities and limit the size of text file that can be handled. Research Machines TXED allows long files by editing one section at a time, which works but is very ugly and rather inconvenient. Wordstar, however, manages to do just about everything you could ask of it (except fit on an SDSS disk) and handles 32Mbyte files to boot. It achieves this by continuously juggling both code and text between memory and disk.

In this article I intend to consider only the juggling of code, since data juggling using random access files seems a more popular subject and thus there is much more written about it.

A few months back I wrote a review in this journal of a printed circuit board design package. It so happened that as I was writing the review I was in the process of developing a similar package of my own. Both packages had run into the problem of memory shortage but our approaches differed. I must confess that at least part of the reason for this was that the compiler used for the program I reviewed supported a chain command which my compiler does not have. I would argue, however, that this is fairly irrelevant since my single file solution breaks the file down into only 8 parts compared with some eleven files in the other.

The question to be asked here is not so much how the two programs go about loading the required part but why the divisions occur where they do. Specifically, of course, why do I think my version is better. The logic that led to my approach went something like this:

- The program must have facilities for editing, saving, loading and printing the file. Also useful would be a tidy-up facility for plotter dumps.

- Each of the above listed facilities is self contained and does not inherently need any of the others.
- Each facility takes a significant time, and each one also runs to completion (and will thus not be called up twice in succession except in unusual circumstances).
- Therefore the time taken to load one of these parts would be a change- round time which is only encountered once for each unit (within a reasonable time).
- Thus it is very reasonable to break the program up in this manner. We will require a menu section to drive them from, and we must ask whether they all fit in memory with a good size data area remaining.

In practice I found that the editor and the dot matrix printer dump were the only ones which took a significant space, and both could be fitted in quite acceptably on a 40K system. There was one extra consideration, which was that I had decided to provide help messages which take a large amount of space. These I put in a separate file to be read in and displayed as required.

The mistake (in my opinion) made on the other version was breaking down the individual editing functions. Now I am quite aware that Wordstar does this to some extent, however Wordstar has all the common functions available instantly and those which it loads from disk remain resident for immediate access until some other function overwrites them. The situation I found with the PCB program was that if I asked for a DIL pack pad set, I have to wait while the disk whirrs before continuing and if the next thing I want to do is put another DIL pack on I have to wait again. Placing and deleting tracks was slightly better in that several tracks can be placed or deleted in one go, but then a specific instruction had to be given to return to normal mode and - guess what - the disk starts up again.

Now let us suppose for a moment that it is in fact necessary to break down the code further. Perhaps the editor won't fit all in one (a very possible problem for my current project). How do we approach the problem now. Well, let's look at Wordstar again. (Incidentally the reason I refer to Wordstar so often is that I am using it to enter this text, so it's an easy example.) Looking at the main editing menu I see 21 commands listed. These are the common commands and only one of them (delete line) is non-resident. In documents, however, this command is fairly rare so that is not a problem. Now let's look at the other menus. The block menu contains disk and file handling commands which can be non-resident without making a noticeable difference. It also contains the block handling routines, some of which will, in long files, often require disk transfers so it would be difficult to tell that they are non-resident. Of the 19 commands in the quick menu only find and replace are non-resident and then only the opening bits which are used once only. The print menu has only one function during editing so it is resident. The on-screen menu has only centre text non- resident. The help menu is entirely non-resident since it is all messages which occupy vast amounts of memory, and (perhaps more importantly) the sort of experienced user who is concerned about speed will not often use it.

So what are the lessons here? Well firstly we see that very few of the editing functions are non-resident (from which we can surmise that the overlay file is broken down mostly as I described above). Secondly we see that those which are tend to be infrequently used ones or ones which are inherently slow due to disk access in any case.

While on the subject of Wordstar, there are a couple of other points to make. Wordstar stores most of its messages on a separate file rather than in memory. Because this means they take time to access, a policy is adopted whereby non-essential messages are only printed at all if the operator seems uncertain. For example, menus are not displayed if the next key is pressed immediately. Another aspect of this message system is that individual messages are picked out from the file. Thus when the operator requests help, only the relevant information is printed. Returning to the PCB example, this means that if I request help while moving a track I should not be told how to save the file, print the file, define DIL packs, swap sides or do anything else not relevant to moving a track, especially if I can't actually do it at the moment! If I ask for help when moving a track I should be told how to move the track, how to deposit it and how to abort (if I can).

Another useful lesson from Wordstar involves what happens when the non- resident code I needed is finished. If it was a major item returning to the opening menu (or completion of a PCB printer dump for example) then it is quite acceptable to reload the opening menu code. However if you have just deleted a line (perhaps 200 bytes of code) you don't want to have to wait while the 15K of the main editor is reloaded. This is where chained files are generally rather inappropriate. What we need to do is load the special code whilst keeping the existing code in memory. Research Machines BASIC has a command to do this. It is called MERGEGO, and operates by reading a disk file onto the current program without deleting any of the program which does not clash. The big problem here (and it is quite significant) is that the program data is erased in the process since the program is being modified. Also, MERGEGO is very slow since the input file has to be in ASCII listing format. In any case, RM BASIC is interpreted so the program can be instantly made to use less memory by compiling.

In any program or suite of programs there will be a considerable amount of common code (e.g. I/O, disk handling, standard runtime routines). In many compilers much of this code occupies standard locations, and in almost all compilers it is possible to arrange for the majority of it to occupy standard locations. Once we have done this, at no penalty in terms of size or speed, it would seem very silly to reload all this code every time. In my PCB system, there are runtimes from 100h to 1100h (the compiler always puts them there) and a set of global functions to overcome compiler limitations which reside at the top of memory. These blocks of code are loaded at the start and remain resident until the user returns to CP/M. Among the code at top of memory are two routines to handle the disks. One routine loads and executes

a section of the overlay file, the other prints a section of the help file. Both routines take a reference number on entry and look up in a table stored at the beginning of the relevant file. Since both these tables are also constant, and since both files are constantly open, the tables and the file control blocks are resident also.

This raises another drawback of chained files. As I mentioned, I leave the overlay file open. This can cause problems if the disk is changed, but this is the case whenever code is juggled. If each section of code is in a separate file, it will obviously be necessary to open the file before loading the code, which means extra disk accesses.

To recap then, if we cannot fit the whole program in memory (along with a reasonably large data area) our first step should be to attempt to identify distinct, non-related, self-contained units which run to completion so as to be used only occasionally and never twice in succession. If we cannot do this, or if we need to break one of these sections further, we should separate only the most infrequently used parts and leave space for one at a time so that they can return instantly. We should also keep a record of which one was loaded last so that we do not load it again if it is still present. Also, we should avoid any other situation where we are reloading code which is still resident such as common runtime support. If our program contains long messages we should remove these to another file and load them only when needed. And one very important point. In many programs the initialisation and startup messages etc. account for a significant portion of code which is used only once so as soon as it has been used throw it away.

Well so much for the basic theory. Let's now take a look at some details. I shall start with the simplest improvement I suggested above, disposing of the startup code. The Gemini CP/M BIOS does this with its introductory message simply by storing it in what will become the sector deblock buffer. This approach is very useful for the assembler programmer, who can very often put such messages in disk I/O buffers or above the other code in an area which will become data storage. Unfortunately, high level languages do not often lend themselves to this sort of treatment. If, however, we are going to use overlays then we can always put the opening bits in the overlay patch area. In the case of my PCB package I had to write assembler code to operate the overlay which then had to be relocated on loading. This code, which forms the opening .COM file, is some 18K in total of which only 1K is resident. Another 1K is the relocator, system checks, and code to open the overlay files and analyse the command line. This leaves 16K of introductory graphics which may seem excessive, but it does not significantly delay the system and takes no room after it has been used.

As I mentioned earlier, I do not consider chained files to be a satisfactory solution for a final version. Since I know of no compiler which provides any other method, it will be necessary to include a small amount of resident machine code with which

the main program sections must interact. The simplest approach in the long run is to relocate this code to the top of the TPA on loading and move the BDOS vector (which is used to define RAMTOP) to below it. Because the C compiler I used for the PCB package does not provide facilities for transferring global data from one program to another, I also put the two arrays on which the program works above the BDOS vector. The machine code includes routines to find the arrays and the C code then addresses them using pointer type variables. This approach also means that a computer with more RAM actually can hold more data rather than just having more wasted space. Unfortunately it is not nearly as easy in languages other than C or assembler.

An incidental note for users of HiSoft C. I use this version and as I stated above it does not directly provide for global variables to remain between programs. My solution in the PCB program works very well but it would not be suitable for one of my forthcoming projects. Fortunately, I have since discovered a neat little trick. HiSoft C usually places all static variables at the top of the TPA as found by the compiler, but this can be changed using the #data directive as described in the manual. What the manual does not say is that it is possible to have two or more #data directives in the source file. Variables always occupy the data area defined by the last #data before their declaration and, providing none of the areas overlap (and that there were no variables declared before the first #data), the program will compile and run correctly. However, only the last data area defined will be automatically filled with zero bytes when the program is run so all the other variables will be retained (providing they are defined in the same order in all programs). Note that the last data area should be the highest, otherwise the stack checking will not work properly.

The task of actually interfacing the high level code with the (relocated) machine code can also be tricky. My approach in C, which is totally portable and really quite neat, was to define a data type "func_ptr_ptr" as

```
typedef char * (* * func_ptr_ptr)();
```

This type code comes out as a pointer to (an array of) pointers to functions which return a pointer to character. I then defined a variable of this type which was loaded (indirectly) from the BDOS vector. Since the machine code creates a table of function addresses with a pointer just after the new BDOS vector, it is now possible to call these functions directly.

```
static func_ptr_ptr global_function;
global_function = gbase();
(*global_function[ GET_DATA ])( &pads, &padpos, &tracks, &trackpos );
```

(In this case, gbase() is a function returning a func_ptr_ptr whose value is stored three bytes after the address pointed to at location 6 (in C code this would be : *(cast(int_ptr)(*(cast(int_ptr)6) + 3)). GET_DATA is a defined constant corresponding to the vector for a particular global function.)

This is all very well if you program in C, and providing you know how your particular compiler passes arguments etc. If your compiler uses a post- linker (for instance if it generates .REL files), then it should be possible to write some machine code to do the linking and link it with the compiler output. Once again it will probably be necessary to know how the compiler calls functions or procedures and how the parameters are passed. If the manual does not say (and most don't) the best way to find out is to write a very simple program which calls a function with arguments, compile it, and dis-assemble it (ZSID or GEMDEBUG are good tools for this, or MDIS if you prefer to wade through a listing). Finding your bit of code among all the runtime support can be a pain, but there are ways around it. Compilers which don't use a linker often tell you where the runtimes end. If not you can be fairly certain anyway that the compiled code is the last part of the object file. If your compiler uses .REL files, then you can instruct L80 to put your bit exactly where you want it by issuing /D: and /P: switches.

Most compilers pass arguments on the stack with the first argument being lowest on the stack (which is the highest address on the Z80). Sometimes the number of arguments will be pushed on just before the call. Some compilers pop off the arguments after the call, but some expect the routine that was called to do so before returning. Some compilers use special routines at the start and/or end of a function. For instance, HiSoft C starts every function with a call to 141h (or 13Eh) where IX is pushed and then loaded with SP before moving SP down to make room for local variables, and exits from functions by jumping to 17Bh (or 178h) which loads SP from IX, pops IX and removes DE bytes of parameter data from the stack. This sort of information is essential for writing machine code to link with compiled code.

If your compiler does not generate intermediate files (i.e. if it directly generates .COM files) then this linking process is not so easy. Some compilers allow small sections of machine code to be defined directly within a function (e.g. HiSoft use inline(...), some have mini-assemblers etc.), and although this facility is not usually suitable for any significant code it should be possible to encode a call to some previously loaded resident machine code. It is simplest if the resident code is at a fixed address, but above the BDOS vector is quite easy to use. There should obviously be a jump vector table at some easily found address. The relevant jump address can, of course, be found using high level code and stored in a static variable for the low level code to read. This overcomes one difficulty, namely finding the function parameters. There remains one problem, which is how to return to the main code. It may be possible to push a return address and allow the compiled function to do the work, but this can involve some rather messy code. I prefer to put the

correct return system into the main machine code routines. This obviously requires knowledge of the compiler's workings, but it means the code can be jumped to (using JP (HL) for instance) which is considerably simpler than calling. An alternative would be to call a fixed address which is known to contain a JP (HL) instruction. There will almost certainly be one somewhere in the runtimes. In PASCAL this might look something like:

```
FUNCTION GLOBAL ( FUNC, PARAM : INTEGER ) : INTEGER;
   VAR
      ADDRESS, PSTORE : INTEGER;{Providing these will not be stack relative}
   BEGIN
      {Work out the address to call. base is a variable or constant which points
      to the jump table.}
      ADDRESS := BASE + (3 * FUNC);
      PSTORE := PARAM;  {So the machine code can get it}
      INLINE (    42, ADDR(ADDRESS),              {ld hl,(address)}
                  237, 91, ADDR(PSTORE),          {ld de,(pstore)}
                  205, JPHL,                      {call a jp (hl)}
                  34, ADDR(PSTORE)                {ld (pstore),hl} );
      GLOBAL := PSTORE
   END;
```

So how do we actually go about loading sections from our overlay file? First we must define a structure whereby the various code sections in the overlay can be identified. It would be highly inappropriate to enforce a standard length or standard positions for the sections, so it will clearly be necessary for the file to contain a reference table giving the relevant data on the sections. We will probably need the following :

Load address	–	where to put it
Entry address	–	where to run it from
File position	–	where in the file to find it
Length	–	how much to load

Often it will be useful to have a section of code broken down into several segments which load at different memory addresses. Obviously there is no reason why these should not occur contiguously in the file, so we should only need one file position reference. Load address and entry address cannot really be standardised because we may well have both major program sections (such as the editor) in the same file with small re-entrant routines (such as delete line). A possible table structure (the one I actually used for my PCB package) would consist of:

First Sector Number	–	16 bits absolute sector
Entry Address	–	16 bits absolute address
Load Address Block 1	–	16 bits absolute address
Length of Block 1	–	7 bits sector count + flag
Load Address Block 2	–	16 bits absolute address
Length of Block 2	–	7 bits sector count + flag

Load Address Block 3	–	16 bits absolute address
Length of Block 3	–	7 bits sector count + flag
Load Address Block 4	–	16 bits absolute address
Length of Block 4	–	7 bits sector count + flag

This structure takes 16 bytes and allows each section to be broken into up to 4 blocks of up to 16K (128 CP/M records). The length bytes have a flag in bit 7 to indicate the end of the section. Since I used one logical sector to contain the entire table, I could have up to 8 sections, but it would obviously be possible to use more sectors and have more sections. To make life simple I wrote a short program to combine standard object files (.COM files) to make up an overlay file with addresses provided from the console, although a submit file with XSUB can be used to control it.

The machine code to operate this system takes about 70 bytes once the file is open and the reference table is loaded. I have included a listing of it. It will not allow returns to the caller since this was not necessary in my application but it should be easy to do. Also it does not keep a record of the last section loaded but this should not be too difficult either. Naturally it uses random access file handling.

```
chain:  ;  Chain next program - non re-entrant version

        pop     hl          ;Drop return address
        pop     hl          ;Get program number (HiSoft C system)
        ld      a,l
chain1: ld      h,0         ;Entry point used by machine code
        add     a,a         ;Calculate reference address
        add     a,a
        add     a,a
        add     a,a
        ld      l,a
        ld      de,ovrctl   ;Reference table has been loaded here
        add     hl,de       ;HL now points to required entry
        ld      e,(hl)      ;Get sector number
        inc     hl
        ld      d,(hl)
        inc     hl
        ld      (ovrfcb+33),de      ;Set random record number
        ld      e,(hl)      ;Get entry address
        inc     hl
        ld      d,(hl)
        inc     hl
        push    de          ;Push it
chain2: ld      e,(hl)      ;Get load address
        inc     hl
        ld      d,(hl)
        inc     hl
        ld      b,(hl)      ;Get length
        res     7,b         ;Strip off flag
        push    hl          ;Save hl
```



```
chain3: push    de          ;Save registers
        push    bc
        ld      c,26        ;Set DMA address
        call    bdos
        ld      de,ovrfcb   ;Read next sector
        ld      c,33        ;(Read Random code)
        call    bdos
        ld      hl,(ovrfcb+33)      ;Increment sector number
        inc     hl
        ld      (ovrfcb+33),hl
        pop     bc          ;Restore registers
        pop     de
        ld      hl,80h      ;Adjust load address
        add     hl,de
        ex      de,hl
        djnz    chain3      ;Loop back
        pop     hl          ;Restore reference pointer
        bit     7,(hl)      ;Check for last
        inc     hl
        jr      z,chain2    ;No = loop back
        ret                 ;Jump into new code
```

As it stands this system of reference will serve well for most applications, but it could be improved. Firstly, where re-entrant sections are used it may be desirable to have two or more routines in a common section. An example would be where they have considerable common code which is not used elsewhere, or where one would often be followed by the other. This could be achieved using a dual table system where the routine number would be used as a reference to a table of entry addresses and section numbers, and the section number found would reference a table of disk allocations.

An obvious second change that could be made concerns the length bytes. An accuracy of 128 bytes in the length of blocks is good enough for most cases, but it would obviously be possible to use 2 bytes to give the exact length. Of course it would then be necessary to buffer the sectors as they are loaded.

There remains one very difficult question if we are going to have re- entrant sections in the overlay. How, using a high level language compiler, do we arrange for the relevant code to occupy a suitable free block of memory? The situation is not as troublesome as it could be because we have removed the need to link into the main code by our overlay management system. Firstly we must allocate the memory block we are going to use. If the compiler produces relocatable object (.REL files) we will probably have issued /D: and /P: link switches to ensure the data area is constant, and we should be able to leave a suitable gap between code and data to take the extra bits. Then when we compile and link these we can arrange for them to be where we want. If the re-entrant sections need to access runtimes or any of the main code, it should be possible to load the host program into the linker for the required references. Such a process is naturally very involved and will take some time and

effort, particularly as we need to know the entry addresses. It is well worth trying to ensure that each section of the program works before creating a complex overlay structure.

The problem of locating re-entrant code is even more difficult if there is no intermediate stage in the compilation. One option would be to compile the host program several times, with different patches each time. Providing that the patch is always the last item, and that the compiler does not put anything significant immediately after the compiled code, we should then have all the patches at the same location with the necessary space allocated for them. We would obviously have to find out where this location is, which could prove somewhat tricky. I have noticed that HiSoft C has an extra (undocumented) directive #code, which I presume works like #data. Unfortunately there seems to be good reason for its being undocumented, namely that it totally confuses the compiler. Presumably it works on the non-CP/M systems for which the compiler is also available.

Operating a message overlay file is much less demanding. Once again we will want to have a reference table at the start of the file so that the program can call up the message it wants, but it can be much simpler. The only information we really need about the message is where to find it, since we can use a standard end-of-string code to stop the printing. Of the three programs where I have used a help file so far, none has needed more than 12K of help. This makes life very simple, to the extent that the first one does not even use random access. The reason is that there is less than one extent of the file, so all that is necessary is to place the sector number in the FCB current record byte and then read sequentially. Once again I arrange for each message to start at the beginning of a sector, so I only need one byte to address the start of the message. Thus by using one sector for the reference I could have up to 128 messages.

In practice I use a slightly more complex system which includes a coding seed to make the file unreadable to over-inquisitive hackers. It would be possible, of course, to use some form of data compression to shorten the file, but I did not consider this worth while. I also store the reference table backward, again for no particularly good reason, and to add extra complication each message starts with a heading which is automatically centred on screen when it is read. If a form feed is found in the message, the controlling software stops and awaits a key press before continuing. As with the program overlays I wrote a short program to generate the help file. It simply reads in a Wordstar text file and breaks it every time it finds a comment line (one which starts with two dots). The comment becomes the title and all that follows is the message up to the next comment (or end of file). At the moment the message numbers have to be assigned from the console because I wanted them to be non-contiguous and I could not be bothered to devise a way to declare them in the source.

Well I hope this article has been of some use to somebody and at least of some interest to others. If anyone has any further thoughts or comments, I am sure the editor would be pleased to receive them (he may even publish them!). I have submitted the two programs I mentioned (GENHLP & GENOVLY) for possible inclusion in Scorpio Systems utility disks along with the source of the code to handle their output. Alternatively, they are available from myself at the address given below. Please send £5 to cover postage etc. and specify the format you require (Gemini/Nascom 5.25" formats only).

Robert Pearce
7 Connaught Road
Newbury
Berks
RG14 5SP

Public Domain Software — a note of caution

by P.D.Coker

Some readers will have made use of freely available, low cost software obtained from user or special interest groups, or purchased through one or other of the firms which advertise in the glossier magazines. In most cases, no guarantee is given that the software which you have purchased will run on your particular machine and the amount of information which is available in the advertisements or disk contents listings is inadequate.

Some of this program material is truly public domain - donated in a spirit of generosity by the author and available for the cost of a disk and a copying charge. Other software is known as Shareware, and is either fully-functional (but not always the most up-to-date version) or restricted in some way. To get the maximum benefit from Shareware, you are usually invited to register by sending in an appropriate amount of cash after trying it out. You will then receive an updated or de-restricted version and, in many cases, a detailed manual. In some cases, the author relies on your good nature to send a donation if you like his/her program and intend to use it. It is a sad fact that some so-called PD software distributors charge far more than the copying and handling charge but equally, one should be careful of so-called low-cost offers since the quality of the disks can be poor and may damage your drives. The majority of such PD software is for MSDOS although one or two sources also offer CP/M programs.

The quality of PD software is extremely variable and the majority of the vendors haven't much idea either. So-called 'clubs' offer discounts for quantity but little else. Having been caught out by one or two of these organisations, I felt that it might be useful to give a few names and addresses of firms and user groups which I have found to be reliable:

Shareware, 87 High Street, Tonbridge, Kent (0732 771344).
Advertise widely; useful booklet for £1 in stamps listing all their PD and Shareware MSDOS programs with ratings and compatibility.

Seltec Computer Products Ltd., Farley Hall, Wokingham Road, Bracknell, Berks RG12 5EU (0344 863020).
Widely advertised, but generally more expensive than the previous firm. No guarantee given that software will run on your system but most will run on IBM or near compatibles. I believe that they used to and may still offer some CP/M software.

Some of the other PD Software houses may be OK - but I haven't tried all of them - there seem to be dozens.

PC Independent User Group (0732 63157) mainly for Amstradophiles but offer all (or nearly all) PC public domain software at £3 - £4 per disk; very useful organisation but costs £22 per year to belong. They do offer many disks of useful software which has been tested and does work on Amstrad and similar PCs) - and offer a useful help line service as well.

For completeness, I ought to add the **IBM PC User Group** (01 620 2244), although I don't belong to it (it costs £25 per year and a £5 joining fee), and what used to be called the **CP/M User Group** (0322 22669). This organisation caters for MSDOS as well as CP/M users and was excellent. I used to belong to it but regretfully decided not to renew my subscription when they suddenly stopped producing informative journals/newsletters/library catalogues last year. Members have access to virtually all PD software which is commonly available for a copying charge of £2.30 or so per disk (send your own formatted disks) but you will need to know which library and disk numbers you need. Both CP/M and MSDOS libraries are available and CP/M disks can be produced in a variety of formats.

It might be useful to see if your potential source of software issues a catalogue (printed or on disk). I was impressed by the **Shareware** booklet and I understand that their bulletin board (0732 770539) is quite useful. •

Various Points

by Peter Hand

Who, me?

Me write for **Scorpio News**? Well, I suppose somebody has to, otherwise it will disappear ... so I'll have a go. I've always held that if a person doesn't have anything to say, the least they can do is to shut up; therefore this is really against my principles. Maybe once I get started I'll think of something.

Confessional Box

I do not own a Gemini system, and never have. I do, however, have a Nascom 1 sharing an uneasy existence with a California Digital S100 mother-board and a bunch of unreliable hand-made cards. This system will live for ever, because it includes an 8748 programmer, though looking at it now I am amazed that it ever worked at all. Dynamic RAM skyscrapers, Veroboard interfaces and more wire links on the Nascom than tracks ... to think that a mere six years ago it was the envy of my neighbourhood. The 8-inch disk system died last year, so data goes in and out via the RS232 link and a little program called NASCOMMS by which my other computers pretend to be a cassette deck.

I have an interest in Gemini, though, and it is commercial. Some years ago I was designing an industrial monitoring system and needed a graphics video display. After an inordinate amount of research I decided that the Pluto represented the best value for money bearing in mind the system requirements, and the whole design was therefore based around the 80-bus. That's not to say that I have been entirely satisfied; Pluto is extremely SLOW and if there's any technical data on it, I never saw any.

IO Research

IO Research have always been supremely unhelpful to me. I have never, ever, got past the receptionist with a technical enquiry. Listen, chaps, if you're reading this, customers make pay-days possible. The only reason I didn't design you out last year is that I was too busy. I came close, though. The contents of your trash bucket arrived at my premises in 1987 and some of it is still here — like the Pluto marked "IOR 5" which writes half its text sideways down the screen. Could this be your fifth X-model

prototype? It certainly looks old and dirty enough. You should be ashamed, and so should Gemini who sold it for the full price.

Help Wanted

Having mentioned Gemini, there is also their retail arm, Quantum, and I have something to say about them too. While I cannot complain about their standard of service, I must point out that I pay a lot for it. In spite of transferring five-figure sums to their bank account I still have to pay retail prices. Is there any dealer who'd like to take over this business in exchange for 30-day credit and a modest trade discount? This year I've got "loads-a-money" to spend and I'm totally corrupt.

New Pluto Review-ette

The new Plutos have arrived at last! The first thing I noticed is that the user manual is dated 1986, and the cards have "1988" etched into the metal. I expect that's why they have so many undocumented links and features. Still, they work; a process of elimination revealed that the extra six pins on the TTL video connector carry only the new brightness line, and can be ignored in my application. It's a good job I never used the 50-pin header because it's gone without trace.

A very nice board, altogether. It has no less than five layers of metal and ninety(!) ICs including for the first time a serial interface (undocumented) and space for a palette DAC. The silk screen now carries the legend "GEMINI COMPUTER SYSTEMS LTD." as well as the IO Research trademark, and joy of joys! a full set of 78 edge fingers. So at last I can have a "Ready" interrupt, once I figure out where to patch it on the board. But wait! what is this link marked "INT" connected to bus pin 22? I don't know, because it is not mentioned anywhere in the manual. Neither is the DIL switch, though fortunately here the silk screen gives some help. It apparently selects serial baud rates, a diagnostic mode, palette on or off and the polarity of the sync signals. A reset push button has been fitted, though the board still doesn't reset from the bus, doubtless for very good reasons. Resetting the Pluto has always given me problems. If the host system is turned off and on again quickly, Pluto hangs up. My customers can't understand this, and I don't think they should have to, as it could be cured by the addition of a single diode.

There's an extra 64k of memory on board, which implements the brightness attribute as a fourth colour plane. This means you can have sixteen colours without a palette, instead of the eight formerly available. Personally I don't find it that much of an advantage, and I think most serious users will want the palette, but it's there if needed. Tests with a stop-watch show that this extra plane slows things up, but not by as much as expected. I am a bit mystified as to why the board is stuffed up with 64k RAM chips in these days of megabits, but I suspect it has to do with availability. So long as I get my cards on time I'm not going to complain. The ROM has taken a

giant leap from a 2764 to a 27256 to support the extra features and interfaces, and some of the logic has condensed into three PALs. An AMD 8088 still runs the show, with a Motorola 6845 video chip and now a Signetics 2651 serial controller. No brand loyalty at Io/Gemini, obviously.

Well, that's about it, really. All the quirks and features of the old Pluto are there, including the NASIO and DBDR lines; it appears to be totally compatible with my old software. Because the palette can go on the same card, that evil piggy-back is a thing of the past. Now if they could just fix the reset, update the manual and make the @$**&? thing go a bit faster, I'd have to say it's a very good product.

Operating Systems

At my company we use mainly Micromint computers, as advertised in Byte, with 64180 CPUs running ZCPR3 and ZRDOS+, and on the few occasions I have to use a CP/M computer my blood pressure rises to a dangerous level. ZRDOS is the biz! I am totally sold on it. It makes MSDOS look like Nasbug by comparison. In the States it has an enthusiastic user base well-supported by over fifty bulletin boards from which tools and updates can be downloaded, but in this country the support is indistinguishable from zero. I guess this is only to be expected, given that a million Brits thought that a disk was something in your back that slipped when you tried to carry all the bits of a BBC upstairs at once. Well, help has arrived. J B Designs (15 Market Place, Cirencester, Glos. GL7 2PB, tel. 0285 68122) have become UK dealers for Micromint and related goodies; I am happy to say that they base their pricing on a more sophisticated algorithm than changing the dollar symbol to a pound sign. In fact, you'd be hard pressed to pay less by going direct. "But", I hear you ask, "what has this got to do with me?" Lots, if you want to use Z-System on your hand-cranked hardware, because a UK user register is being put together and soon there'll be a bulletin board (or Z-Node, as we initiates call it) on line at a telephone near you.

Zee what you can get

The Z-System is not of course a Micromint product. It is the property of Echelon Inc and is definitely not in the public domain, though its status is a little indistinct because parts of it are freely available over the phone. I suspect this is pure philanthropy on the part of the copyright holders but it works very well, because many of the users are very clever guys indeed and upload their own software tools pro bono publico ("for the general good", to those of you who went to comprehensives). 'Tis a pity all our good software people have been working on Nimrod and similar systems, and are too busy to do anything useful. Or they were; maybe they're free now (dig, dig). Anyway these gents, whose names all seem to be Irv, Jay or Wayne, give the Z-System exemplary support through the network of Z-Nodes. Once logged on you can download newsletters, help files and manuals,

software updates and bug fixes, assemblers (I know where there's one for the Z-280), editors, BIOS patches for all kinds of machines, even games - you name it, they've got it and it's free. Without access to a node, the Z-System is only twice as good as CP/M. We need a local node coz it costs a pound a minute to call the States. (Tip: if you dial '83' between the '0101' and the area code, you get a wire connection without that irritating satellite gadget which suppresses the other guy's carrier on V22. Not a lot of people know that.)

To get started with Z-System you need ZCPR3 and ZRDOS. You can use ZCPR3 (which is free) with CP/M but you don't gain much. ZRDOS costs money, though not much with the Dollar at its present level, and also you won't get far without the tools, books and manuals. Don't be content with ZCPR1/2, ZCMD or BDOSZ; the latest versions are ZCPR ver 3.3 and ZRDOS + ver 1.9 and can be obtained in a variety of disk formats (excluding Gemini) from Echelon Inc, PO box 705001-800, South Lake Tahoe, California 95705. If you telephone them on 0101-916-577-1105 a nice answering machine will take your name, address and credit card number and send you what you need, providing you know what that is. And there's the catch. It's not an easy matter installing the system for the first time, although the documentation is pretty comprehensive. The easy way, using what they call Zee-dot-com, doesn't give you a real system. You have to do it yourself. Maybe if enough people are interested the editor of this esteemed publication will give us space to work through it together. Oh, and if you do phone Echelon, say "Zee" not "Zed" or they get confused.

Finally, Brethren

Gemini may not have their 64180 card up yet, but I do. I'm writing this on it. It has 1 meg memory, a disk controller, two RS232 and a parallel port and full 80-bus compatibility, with Z- System and a BIOS that supports the SVC as system console. With a 9-meg clock it runs getting on for six times faster than an (ahem!) appliance and a bit faster than my colleague's ST. If I may delve into the controversy between ST and Amiga enthusiasts for a moment, though I don't possess either, I feel that the Amiga is to the ST what Betamax was to VHS - i.e. very good, possibly better in some ways, and doomed. No doubt Dr. Dark will dispute this, but time will tell. I have a vision of the future that goes something like:

"I've got a fantastic bit of software! You'll have to borrow it sometime when you've got a free evening."
"Yeah, love to, but unfortunately I've got an ... Amiga."
"Oh ..."
(Commiserations and drinks all round)

Mind you, these 68000 machines will never go the way of the Atom or the Nascom. They make great terminals!

Well, that's about it. Looks like there was more to say than I expected, even if every bit offended someone different. Now, how can I get it to **Scorpio News**? My system supports nearly two hundred disk formats courtesy of Uniform, but I don't even bother to look for Gemini ... it won't be there. And that's a cue for another whinge. How come there are so many formats? Multiplying the disk sizes (4) by the possible sides (2) and densities (2) and numbers of tracks (2) gives a number not unadjacent to 32. What spirit of perversity causes every different manufacturer to select their own unique format? Pride, I suppose. If anyone knows whether the Gemini format matches any computer ever sold in America, I'd be most grateful for the information. For now I'll send this in on an IBM 360k PC disk, which is as standard as a 3/16 UNF bolt. ●

Private Adverts

GRAND CLEAR-OUT.
813 CPU–£82, 832 SVC–£82, 829 FDC–£82, 809 FDC–£40, MAP80 VFC–£145, 805 FDC–£13, 848 Multi-Serial I/O (nearly new) – £87, 863-32 Static RAM card–£65, 803 EPROM board–£40, 802 64K RAM–£45, Nascom RAM B–£40, IO824 A/D Board–£43. Seagate ST506 (5Mb)–£62, Rodime 204 (20Mb) – £140, CDC Wren 9415-36 30Mb Wini (Voice coil stepping)–£185. Controller cards: Xebec S1410–£138, Adaptec ACB4000–£225. Streamers: Tandberg TDC3200, Archive 5945L–£140 ea., with tapes. PSUs: 843 140w–£40, 817 85w (late type)–£45, 817 75w–£36. KBDs: 827 un/cased–£37/£47, 852 Low Profile case/cable–£80. Vero Frames: with front/back panel suits N2/Pluto etc.–£55, for 80-BUS boards + enclosure & floppy/wini frame–£75, 8 slot (5 conn's) backplane–£25.
All bits GWO, ANY offers ,phone Ian (0473) 831353.

For Sale: GM837 Colour Card, GM816 I/O Card, GM870 MODEM Card. All with manuals, software etc.
Offers. C.Bowden. 0209–860480 (evenings).

For Sale: Nascom 2 CPU Board, CP/M, Nasdos, £85. 256K MAP80, £50. AVC, £50. Dual disk, £50. FDC, £25. IEEE, £35. A/D, £10. Maplin MODEM, £25. 64K Memory Board, £10. Digitalker Board, £10. Chassis style green screen monitor, £15. Numeric key-pad, £6. Kenilworth case, PSU, backplane, £25.
Phone Romford (0708) 48836 after 6pm.

For Sale: Nascom 2, 2 x RAM A cards, PSU, extended keyboard, documentation. Not greatly used as I upgraded to GM813 when it first appeared – £150. Nascom IMP matrix printer, IMPRINT, unused ribbons, documentation – £150.
Tel: Steve Willmott, Little Chalfont (02404) 4892, evenings. ●

A Review of the Gemini GM6202 'AT' Compatible

by C. Bowden.

Some months ago I received some literature with details of some of Gemini's new products. In addition to the Challenger series, and the standard 80-BUS products, several interesting new products were listed and described. These included new 80-BUS cards, utilising Z180 (aka 64180), Z280 and 80286 processors, a new RAM-disk with battery backed memory, a Hayes compatible MODEM, and a series of colour cards with a high specification and commensurate price.

In addition, two new systems were detailed. One series is based on the Inmos transputer, and it has a very impressive specification. The other system is more likely to be of general interest since it is an IBM 'AT' compatible system, and should be able to run all of the standard PC software.

This system is listed as the Gemini GM6000 series, and the prices cover a range of systems, starting with a main system card at £595, and progresses up to a very comprehensive system with a 23ms 140Mb Winnie, VGA Graphics, and a 14" Colour Monitor, at £3785. These systems (GM61xx) run at 8MHz, but equivalent models with 12MHz processor and compatible RAM (GM62xx) are available at £200 additional cost.

The range includes systems without a Graphics card and Monitor, allowing purchasers to fit their own options. Thus a 'Basic' system is available with 1 Mb, 8Mhz card, 49Mb Winnie, 1.2Mb Floppy, and Keyboard, for £1865. (The 44mb, 28ms Winnie now seems to have been superseded by a 49mb, 35ms unit.)

(The prices quoted are based on a Gemini list issued in April 1988.)

Operating systems are extra, and several options are available, with MSDOS, Concurrent DOS, Xenix, and BOS listed.

The prices quoted may seem high, if one is comparing this system with some of the systems advertised in the magazines, but I think that it would be fair to say that the extremely high quality of the product and its performance must be taken into account. There are also advantages in supporting manufacturers based in the UK. It has never been any problem, for example to get 80-BUS products serviced, and Gemini have usually been helpful over problems.

I have had an opportunity to use one of these systems, and the notes that follow are based on my impressions of the system after using one for several days and trying various software packages out.

The system described was originally listed as the GM6102, but would now be very similar to the GM6202. This system comprises an enclosure with a 12Mhz, 1Mb system card, 44Mb 28ms Winnie, 1.2Mb floppy and Keyboard. A Taxan 770+ Multi-Sync Monitor and a Paradise Auto-Switch 480 EGA graphics card were used in conjunction with the GM6102.

Hardware

The system card is available in 8 and 12 MHz versions, but the one fitted to the system had the following specifications:-

- 12 MHz 80286 CPU and 1Mb of 80 nano-second parity checked RAM, running without wait states.
- Battery backed Real Time Clock.
- 101-key AT style keyboard, and AT compatible keyboard interface.
- Phoenix BIOS.
- 6 expansion slots (two of them 'short' slots).
- Socket for a 80287 maths co-processor.
- One serial and one parallel port.
- A system EEPROM holding system default parameters.

This card is a multi-layer card of very high quality, utilising pre-insertion tested components. It is made in the UK, by British Aerospace, at Bristol.

The case is a very well screened metal enclosure, and it is stated to conform to the RFI requirements of BS6527. The PSU is separately screened, and appears to be a very robust unit. It contains a fan, and provides two CEE22 mains outlets to power auxiliaries such as monitors or printers. Judging by the photographs that appeared in the 'PCW', at the time of their review of the CHALLENGER, the case appears to be virtually the same as is used for that machine. The front panel is of a moulded construction, and apart from the Gemini logo, only displays a Power indicator, and the Floppy drive(s).

A recessed area on the lower rear panel holds the parallel and serial port connectors, keyboard socket, and makes provision for a Video O/P socket, should a system card contain 'built-in' graphics facilities. The rear left hand side of the case provides access to the expansion slots, by removal of a narrow blanking plate for each expansion card fitted. It is of course necessary to remove the enclosure surround to fit expansion cards, or to access the system box. Since the enclosure is secured by only four screws, this is not difficult.

A greater problem is that it is necessary to remove the PSU box to obtain access to a system configuration switch, and this needs eight screws to be removed, as well as any expansion cards (Video, and Disk at this time). Luckily the switch in question mainly describes the Video configuration, and does not need to be accessed very often.

The machine contained one expansion card - a Xebec combined Hard Disk and Floppy controller, capable of supporting two hard disk drives, and two floppies of between 180K, and 1.2Mb. This was not expected, since the 'News Release' information stated that the disk control system would be located on the system card. The main disadvantage of this card is that it ties up one of the expansion slots, but there is also some advantage in having a pluggable disk controller, since this improves system flexibility. I understand that the system will not now be available with on-card graphics or hard disk support, since the features provided tend to change quickly, and plug-in cards are more adaptable. The floppy controller will be on the main card.

The Floppy Drive was a half height Teac High Density unit, providing 1.2Mb Formatted Capacity, but able to read 360 Kb format, (and also to write it - this is not advisable due to mismatch of head and track widths between drives, and can give rise to read errors when the data is accessed later). The Hard Disk was a half-height Micropolis unit with 44Mb formatted capacity.

The machine contained one floppy drive, and one hard disk drive. There is room (and metalwork) in the case for another floppy drive, but the area below the Winnie appeared to be occupied by a card, so presumably some re-siting would be necessary if a dual Hard Drive system were required. In order to fit another floppy, it would be necessary to get a new front panel, since there is no provision to easily remove the necessary moulded area of the existing panel. In general one floppy of the type fitted is adequate for normal processing, but a second drive can be a great help at times, and my feeling is that this drive should be external to the main system. I have more to say about this later. The hard drive(s) however is/are totally enclosed within the case.

User expansion of floppy drives would appear to be difficult since no 'spare' drive power connectors emerge from the PSU, and only one 34 way Floppy connector is apparent, mounted on the main circuit board, just below the Floppy. There is another 34-way floppy connector on the controller card, but I am not sure whether this can be used. There are connectors for a second hard disk drive on the Xebec Controller Card.

The keyboard is a 101-key unit typical of 'AT' keyboards, and felt nice in operation. There is not much more that one can say about it.

The general standard of construction is very high, and the unit is quite heavy. Personally I prefer this, unless the unit is to be moved around a lot, since it makes the machine less prone to knocks and bumps, and to external fields. The Fan, Floppy and Winchester drive ran quite quietly, and the machine also felt quite cool, with the air emanating from the fan being just slightly warm. As an Electronics Engineer I have always preferred 'cool chips'. The temperature at which some equipment runs never ceases to amaze me. I have burnt my fingers on chips in some equipment, and I wonder how long they are going to survive.

I would however have liked to have been able to have seen some minor alterations such as:

a) A hardware 'Go slower' switch, on the rear panel.
b) Provision for bringing out drive connector cables at the rear. This may be non-standard in compatibility terms, but it would allow temporary connection of alternative drive types, to give ease of inter-format copying.
c) A more easily removable second floppy panel cut-out.

These points are minor however, since it is possible to get software to make the system run slower, or one can re-configure the EEPROM, and one can find ways of bringing out a lead to connect an external drive - see below.

Software and Documentation

Possibly because the system is a new product and things are not yet fully organised, documentation and software was conspicuous by its absence. The only software supplied was a disk with Hard Disk support, and a 'UTILITIES' disk, with just ONE program - SETUP.COM. Documentation consisted of three A4 sheets with brief details of switch settings, I/O port addresses and connections. There was also a manual on the Hard Disk unit. There was no information on the I/O devices used and their programming in low level languages, nor on any other hardware. Most important, there was no programme to allow user configuration of the system defaults held in the EEPROM.

I understand that these problems will be rectified in the future.

Initially, MSDOS 3.2 was not available but I was able to 'borrow' a copy of PCDOS version 2, to start testing the system, and later a copy of PCDOS version 3.2, which allowed me to format floppies correctly and to access the features of the system.

My first exercise was to format and partition the Hard Disk, and to edit CONFIG.SYS to allow the Hard Disk to be recognised. Since DOS cannot access disks larger than 32Mb, the hard disk was set up as two drives, C: and D:, each of 22Mb. System files and DOS files were then copied to the ROOT directory of C:,

and it then became possible to BOOT from the Winnie. CONFIG.SYS was also edited to give the system 40 buffers, use ANSI.SYS, and set COUNTRY = 44 for the UK. AUTOEXEC.BAT was set to load the 'KEYBUK' file in order to set-up the keyboard for UK codes, and also to set a suitable DOS PROMPT. Subsequently, other commands were added to AUTOEXEC.BAT in order to improve system startup configuration.

Performance and Appraisal

The News Release information describes three methods of configuring the 1Mb of system RAM:

a) 0-512k, 1024-1536k
b) 0-640k, 1024-1408k
c) 0-1024k, to allow a RAM resident BIOS to be loaded.

I had assumed that the card would have been set for system b), since this would appear to be the most 'normal' arrangement, but in fact, attempts to set up a RAM-disk in extended memory above 1024k failed, and I can only assume that the RAM was currently set to system c) since the system provides 640k of standard RAM. The software necessary to re-configure 12MHz systems was not available at the time of this review, but will shortly be available. Gemini say that it is possible to configure the system for non-valid parameters, and need to get the software right before releasing it, which is a sound approach, but the inability to be able to re-configure the system caused some problems due to inability to use spooler or RAM-disk facilities, except in system RAM.

The two versions of PCDOS tried ran without apparent problems, and I was also able to try out a number of software packages on the machine, including a number of more expensive commercial packages. All of the programs tried seemed to run without problems at the full system speed. None of the programs tried relied on timing in the way that certain games do, except possibly Chess. (I am not sure if the game is completely tied to the system real time clock, or runs faster, and therefore performs more evaluations in the allowed clock time, which would of course, increase its power.)

I was not able to try 'Flight Simulator' however, which I understand is a particularly good test for compatibility. The commercial packages tried included Wordstar 3.41, Supercalc 3, Xtree, Turbo Pascal, Turbo Basic, Microsoft Basic 5, Norton Utilities, PC II Tools, Cyrus Chess, and Sidekick. A number of 'Free' or 'Shareware' programs were also tried, without problems.

In addition to the standard 25 by 80 format, the EGA card used supported 43 lines and 132 columns, and drivers were available for certain programs. Wordstar was

tried in the 132 by 43 mode, in colour, and results were quite acceptable as far as screen definition was concerned, although this is obviously more a function of the EGA card and Monitor, than the computer itself. There was not sufficient time to try to word-process with WS in this mode to see whether the program behaved correctly.

The system speed is most impressive. Hard disk performance is very good. Wordstar Overlays load from the hard disk almost before one has had time to look up from the keyboard, and programs load extremely quickly. I have not carried out any accurate quantitative tests on this, but Supercalc 3, for example loads almost instantly, compared with several seconds on an 'XT'. On a Gemini Galaxy 2 system, Supercalc 2 took about 7 seconds to load to the first screen, from floppy, and about 2 seconds from Hard Disk.

After I had used the system for a short while, I found it tedious compacting software from 360K disks on to 1.2Mb disks by copying onto and then off of the Winnie, and I also wanted to write to and format 360k disks, so I decided to try to use a Teac FD55B that was available, as drive B:. I replaced the short cable running from the 34 way connector on the main PCB, to the 1.2Mb floppy, with another longer cable.

This new cable was fitted with two 34 way card edge (floppy type) connectors, one of these connectors being about 3 feet away from the first. The cable to this 'remote' connector was placed through one of the unused expansion slot covers, which was then replaced and lightly tightened, and the connector plugged in to the FD55B, (which had a separate PSU). I then re-ran SETUP to tell the system that drive B: existed as a 360k drive, and rebooted. After sorting out a couple of wires, all was OK. and I had a 360k drive online. The sorting out of the 'couple of wires' was a bit of a problem for a while.

IBM, in their wisdom, decided that line 10 should be motor 1 on/off, line 12 drive 1 select, line 14 drive 2 select, and line 16 motor 2 on/off. In order for this to work, some modification of the standard 'Shugart' arrangement was needed, and was solved by Gemini splitting the ribbon cable between lines 9 and 10, and 16 and 17, and then twisting lines 10-16 so that these seven lines are reversed where they enter the first floppy drive.

When the system asserts line 10 for motor, or line 12 for select drive, these arrive on pins 16 and 14 of the first drive respectively, and provided that the correct drive select jumper is inserted in the drive, the drive works correctly. In extending the cable to another external drive, I then had to get line 10 (= line 16 before the twist = motor 2 on/off) back to line 16 so as to operate the motor in the second drive. This was accomplished by cutting lines 10 and 16 of the cable, and cross-connecting them as needed.

Initially I also cross-connected line 12 to line 14. This did not work, since both drives were selected simultaneously, so I used a DVM to find which of the lines 12, 14 or 16 became active when B: was selected, and I found that line 14 (line 12 ex main PCB) was the one. This line was therefore used, but I was left rather confused, since this should have been drive select 1. Without a fair amount of dis-assembly of the machine, I could not see what drive select link was in place on the floppy in the machine, but I suspect it was drive select 1! If this was the case, then line 14 (drive select 2) must have been being asserted when drive 1 was required, and drive select 1 for drive 2!!

The addition of this drive made things much easier. It would also have been possible to have connected say a FD55F, or a FD35F (with suitable connectors), for 720 Kb format.

The Norton 'SI' - System Information program gave the following results when tried on three PC systems, all running PCDOS 3.2 :-

1) IBM PC XT (4.77 MHz 8088) 1.0
2) IBM PS2/50 (10 MHz 80286) 10.0 (Wait states ?)
3) Gemini AT (12 MHz 80286) 13.5 (No wait states)

the above figures are averages over several runs with 'SI'. The Gemini on occasions gave figures as high as 15.9, but I suspect that the results may get more variable as speeds increase. I do not know how accurate this program is, and it is obviously only testing one aspect of the system, but it is indicative of the relative machine performance, excluding disk I/O. It is supposed to indicate how much faster a system is than a bog-standard 'PC', and the first result above verifies this.

Conclusions

It is difficult to criticise the Gemini AT in respect of the standard of construction and performance. It appears to be capable of carrying out its tasks quickly and without problems.

It was let down initially by absence of Documentation and Support Software, non-standard memory-mapping, and lack of support for user expansion of the Floppy Drives, but these aspects are easily remedied. Fundamental design errors in the hardware are a different matter, and are not easily overcome, but the Gemini 'AT' did not appear to suffer in this respect.

Provided Gemini supply adequate documentation and support software, the machine should be capable of providing the standard of performance and reliability needed by more demanding enterprises, and it can be recommended. ●

Digital Imaging Systems — Part 2

by D.R.Hunt

The Optical Disks.

The final stage of the process is to save the compressed image and that's usually done on optical disks. I've seen and heard more confusion and mis–information on this subject than any other, so perhaps you'll forgive my 'going off' at length.

Firstly there's the type of optical disks which may be used. Currently there are four diameters of optical disks, 5.25", 8", 12" and 14", of which 5.25" and 12" are the most common. Now some people mix up the type of optical disk with the information which is stored on it; and some confuse the fact that optical disks fall into two types, those which may be written to, that is information placed on an otherwise blank disk; and those which are manufactured with information already on the disk, and which can not be written to. Some people manage to confuse all types together.

The first type are available in all sizes and are commonly known as WORMs (WORM meaning 'Write Once Read Many'). This is a feature of the technology of the disks. As we shall see later, once an area on the disk has been written to, it is not (yet, they've just crept outside the laboratory — Tomorrow's World — BBC TV — 25th February 1988) possible to write to the same area again. On the other hand, once written to, it is possible for the computer to read the information as many times as it likes — hence Write Once, Read Many.

Another aside — the use of acronyms. Many acronyms like WORM are used in the computer world. They just sort of grow up. But there is a tendency for people to invent acronyms simply because they sound 'technical'. For instance these new Erasable Optical Disks — they haven't found an acceptable acronym yet and EOD sounds distinctly odd. An acronym for these won't take long to fall into common usage. But beware the person who talks in acronyms — 'Computerese', more often than not these people are simply trying to sound clever and don't even know what the acronyms they use actually mean half the time. Never be afraid to ask. If these people really know what they are talking about, they won't mind speaking plain English, and if they don't know what they're talking about, the English will come across as rubbish and you'll be able to recognise an idiot when you see him !!

The second type of optical disk have the information fixed in them during manufacture and are most commonly of the 5.25" size although there are some 12"

types in use. These are most definitely 'Read Only' disks and are often referred to as 'CD–ROMs', CD–ROM meaning Compact Disk Read Only Memory. CD–ROMs are easily mass produced, using a photographic technique, where the digital patterns to be reproduced are etched on to a plastic disk and the disk is then coated with a very thin film of aluminium to make it reflective. The final touch is to coat the surface with a tough plastic to protect the mirror finish of the disk.

Where the confusion arises, is that the information contained on an optical disk is *not* defined by the disk itself. Digital information is digital information, and although there are different types of digital information, computer programs, digitised music, digitised pictures, digitised television – and many others; it must be understood that the optical disk is simply a means of storing digital information, it doesn't imply what the information is.

Let's get that across by an analogy. Take a glass bottle – milk bottle, beer bottle – any old bottle. Now the one thing we know about a bottle is that it will hold most fluids. The shape of the bottle might imply what sort of fluid is in the bottle, but doesn't guarantee it. Take those sad stories a few years back of thoughtless gardeners storing Paraquat in cola bottles and the like. The fact that the unsuspecting thought that the bottles contained Coke didn't mean that the bottle must contain Coke.

It's the same with optical disks. They are simply a means of storing digital information, they don't imply what the information is.

The real confusion is about the 5.25" disks, CD–ROMS. These are small, cheap to manufacture and have enormous capacity, and because of this have been adopted by the publishing world for the new technology of 'Micro–Publishing'. Here the CD–ROMS contain a mixture of text and pictures, in other words, books. The thing is that the books are not stored as images of pages, they are stored in two separate ways on the disk, text is stored letter for letter, and pictures are stored as patterns of dots. With Digital Imaging Systems, as we've already discussed, the information is stored as images only, regardless of whether the image is of text or of a picture. I might add that 5.25" CD–ROMs are identical to those you buy as Compact Disks, and there the information stored is music. Can you see how each is different, despite the disks being identical ? Another fallacy. Measure a Compact Disk, you'll find it's 120mm across, now that's as near as damn it, 4.75" – so why 5.25" disks ? That's another story.

The same confusion goes for 12" CD–ROMs (which actually *are* 12" in diameter); they are an offshoot of the ill–fated TV disks which appeared about 8 years ago. A TV disk contains digital information (although the system used was not truly digital, but a form of frequency modulation encoding). These days 12" CD–ROMs are most

often encountered in micro–publishing the classic and most well known example being the BBC Domesday project which is contained on two 12" CD–ROMs.

So that gets over that a disk is a disk and not what you put on it. Why optical disks at all ?

Optical disk technology has been around in laboratories for some 15 or more years, and things like the TV disk have been commercially available for 8 years, so it's not exactly new, it's actually a well tried technology. Like all things which work, it works very simply. It's only required the improvement in the manufacturing technology of the component parts to reduce the price to make it viable for mass acceptance. The heart of optical disk technology is the strange properties of the Laser (Light Amplification by the Stimulated Emission of Radiation, and don't let anyone spell it with a Z or tell you it means something different). The Radiation referred to is light, and not anything radio–active. In this instance, the Laser is not a large cumbersome machine out of Star Wars, but a small electronic component about 5mm in diameter by about 5mm long. Laser light is known as coherent light and has different properties from the light emitted by – say a torch. For optical disks the two properties of most interest are the ability to focus the light to a pin–point (in fact a pin–point is the proverbial blunt instrument compared to the focal point of the Laser beam used by an optical disk drive) coupled with the fact that the beam can be controlled (turned on and off) very fast electronically.

Ok, to write to a WORM, the Laser is switched on and off very fast in the required binary code and focused on the mirror smooth surface of the optical disk. The disk is of course rotating, and a mechanism moves the Laser light mechanically (either using a mirror or by moving the Laser) so that the Laser light never falls on the same place twice. It's following a course rather like a gramophone record except that it starts at the inside of the disk and works outwards towards the edge, and it's not a single spiral, rather a number of concentric rings (called tracks). Every time the Laser comes on, it burns a minute hole or pit in the surface of the disk. Where the Laser is turned off, there isn't a pit, the disk surface is unaffected and remains mirror smooth. So the pits could represent the '1's and the absence of pits represent '0's. Simple as that !

The clever bit is reading it back and is the same for either a WORM or a CD–ROM. This also uses the Laser but with the power turned down so it doesn't burn the surface of the disk. The Laser follows the tracks previously written, and where the Laser light falls in a pit, the light which is reflected goes off at odd angles. Where the Laser light falls on the mirror surface, the light is reflected directly back towards the Laser. It would go all the way back to that Laser if it weren't for a special mirror in the way called a beam splitter. This diverts some of the reflected light into a light sensor. The light sensor sees flashes of light where the Laser plays on the unpitted surface of the disk, and no light where the Laser plays on a pit. The light sensor

simply converts the flashes of light into an electrical signal and the rest of the electronics in the disk box amplify and shape the signal into a digital signal, exactly the same as the digital signal which wrote to the disk in the first place.

Notice how a WORM can only write once. Once an area has been pitted by writing to it there is no way of smoothing out the pits to allow the same area to be written to again.

Optical disks – Pros and Cons.

That's the way it works, what are the advantages ? The biggest advantage relates to the small size the Laser can be focused down to. Something like 16,500 pits to the inch, that's also 16,500 tracks to the inch across the disk, and as the usable surface of a 5.25" disk is close to an inch and a half, that's one hell of a lot of pits, representing a very large amount of information. If the tracks on a disk were unwound and joined together, the resulting single track would stretch between 5 and 30 miles depending on the size of the disk !! So the overriding advantage of optical disks is the information storage capacity, it's vast ! Far greater than the other competing mediums around. Any other advantages, well CD–ROMs are fairly robust and hard to damage, although sandpaper is effective, and WORMS are pretty strong, although not as robust as CD–ROMs, as they are made of a glass sandwich – WORMS are usually totally enclosed in a tough plastic case and the disk can not be removed (unlike many advertising pictures which show the disks without their cases).

There must be some snags. CD–ROMs and WORMS are not as fast to read or write as the hard disks on a computer, but come a close second. The optical drives used to spin the disks and control the laser are more expensive than computer hard disks, but the trade off between capacity and cost doesn't so much favour the optical disk, it screams it – double the price for 10 to 20 times the capacity. WORMs are Write Once media. Ok, that's bad news if a WORM was to be used as a hard disk in a computer, but then a hard disk is not usually replaceable like the WORM and certainly not secure against someone altering the information on the disk. Security on a WORM is far greater than a hard disk. Once information is written it's virtually impossible to change it, and definitely impossible to change it invisibly.

So all in all, optical disks are used in an Imaging system because of their high capacity, the difficulty of altering data once written, the fact that optical disks are replaceable media and the technology is such that the data integrity is guaranteed for at least 10 years, and this period is getting longer.

Optical Disk Standards.

I hear a lot of rubbish about optical disk 'standards', but when I ask anyone to be specific I find they're usually unclear as to what they mean, or are worried about the wrong things. Now 'standards' are creeping in for CD–ROMs, they've got to. You can't publish books for mass distribution if everyone wants the data laid down on the disk in a different way. The standards come about in the same old way. Standards Committees go away in a huddle for a few years, and during that time several vested interests put competing standards on the market. Slowly one system gains dominance, the others falling by the way side. Eventually a 'de–facto' standard is established and then the Standards Committee emerges from it's huddle and pronounces the winner. Simple really !

Even with CD–ROM, we've still got two standards, 5.25" and 12". People even worry about that – what's it matter. Outside I've got two cars, a Ford with 13 x 175 tyres and an Opel with 14 x 185 tyres. It's obvious I can't take the wheels off one and put them on the other, even if the wheels studs would fit. But it doesn't worry me, so why should only two physical standards to CD–ROM worry you.

Another interesting fact – all the manufacturers of WORM optical disk drives and media are very large multi–national companies, keep this fact in mind for what follows.

With WORMs the story is somewhat different. The process of standardisation is nowhere near as complete. There are several competing types of disk drive and optical media and although they all work the same way, they aren't interchangeable. A couple of early drive and media systems have already dropped out of sight and the world is polarising into 5.25" and 12" camps. The big information handlers favour 12", the lesser information handlers go for 5.25". Some imaging systems hedge their bets and offer both.

Does this actually matter in the context of an imaging system ? The data stored is not like micro–publishing where the information is to be disseminated to a large number of users who have to be compatible. Any profits to be made from a micro–publishing system are in the dissemination of the disks and that can only be profitable if standards exist. No, an imaging system user is highly unlikely to want to publish his paperwork and so the need for compatibility between one imaging system and the next is not there. The things an imaging system user should worry about are twofold. First will the media used by his system remain available during the lifetime of the equipment in the event of the optical disk system he has chosen falling out of favour ? Secondly, if his optical disk system is obsolete at the time of replacement, can the existing data be transferred to the new system ?

All I can say here is that this business of establishing a 'de facto' or official standard for that matter takes years, typically two or three generations of equipment. Given the typical life of office equipment, that's 14 to 20 years. But positive signs of a standard becoming established occur at the end of the first generation. So in the case of WORMs, the pointers are already there. Even during this process, the losers have to capitalise on their investment, and the only way to do this if the hardware sales are lagging is through the sale of media. Being large companies, they don't want to get a bad name in any area, so the manufacture of media continues regardless of the popularity of the system. It's not until the end of the second generation that any large manufacturer will entertain discontinuing a product, because a reputation for letting people down costs more in the long run than keeping uneconomic production of a particular type of media going.

So, the bad news has happened, some 10 to 15 years in the future the optical disk system chosen has been obsoleted, but of course the images can be transferred. It might require some time, and it might require the old equipment to achieve it. But as the information is digital and on disk, the process will be almost automatic. Any vendor hoping to sell new equipment in place of old isn't likely to attract your order unless he can transfer the data. Mind you, don't forget the very first premise in this piece, "Why am I keeping this anyway ?".

All in all, I'm not impressed by the need for standards for imaging systems. Standards will eventually come, and many early users of imaging systems will end up as 'non–standard'. What I say is, "I can't see why it should matter.".

That completes the process of getting the image onto the disk. The job of getting it back again is the reverse. Notice how I've skipped the business of finding the correct image out of the thousands on the disk ? Just be patient. The last chapter is about the retrieval software, I haven't finished with the hardware yet.

As I said, the job of getting the image back is basically the reverse of putting it there in the first place. The compressed image is read back from the disk and placed in the computer memory. It's then fed to the compressor which works in reverse and expands the image back to it's original shape and size. Note that it usually takes a bit longer to expand an image than to compress it, but not much longer – a second or so. The complete expanded image resides in the computer memory exactly as if it had just been scanned in. Just to climb on to my soap–box again in my campaign against the murder of the English language in technical subjects. The opposite of compression in this sense is expansion and not de–compression. De–compression is something which happens to aircraft cabins or cylinders of compressed gas. De–compression is the wrong word to use in this context.

Anyway, that's all for this issue. The final gripping installment is next time!