2. Unpublished articles

This section contains material that was submitted to *MICROpendium*, but was not published before it ceased publishing. Thanks to Bruce Harrison for supplying his "Art of Assembly" material.

2.1. The Art of Assembly. Part 77. Breaking New Ground

by Bruce Harrison

For the most part, this edition of The Art will deal with some details of ways of making use of the AMS memory, but we'll also get into some stuff dealing with the SCSI hard drive.

2.1.1. MIDI Album without Mini-Memory

While we were developing our AMS version of MIDI-Master, our friend Richard Bell, who was testing our efforts, asked what turned out to be an interesting question, "Why does Album still need the Mini-Memory when there's plenty of memory in the AMS?" The reason had to do with address mapping. The Album feature of MIDI-Master was set up to use memory in the >7000 block (nearly all of that block). The mapper chip in the AMS does not allow any of the AMS' pages to be mapped to addresses other than the >2000->3000 and >A000->F000 blocks. We know this is true because we did controlled experiments that proved this to be fact. Still, Richard Bell's question was one of those things that nags at the programmer's mind night and day. Why indeed?

2.1.2. The Sudden Inspiration

Yes, we know what Thomas Edison said about inspiration versus perspiration, and most of the time that applies, except that without the inspiration the perspiration is just sweat. The inspiration had to do with pages 0 and 1 of the AMS memory. Until now, to our knowledge, nobody who's programmed for the AMS card has made any use whatsoever of pages 0 and 1. Many don't even use pages 4 through 9 either, and that seems a waste. Our AMS version of MIDI-Master uses pages 2 and 3 for its main program code, and pages 4 through whatever for storing the music that it plays. That's reasonably good use of memory, but there were still 8192 bytes of perfectly good memory in pages 0 and 1, while we were having to plug in the Mini-Memory to run Album. It simply didn't make good sense. Finally the idea hit the brain: Use page 0 or 1 mapped as if it were some legal area in high memory. In the actual end product, that wound up being the use of page 1 mapped to the area >C000 through >CFFF.

2.1.3. Code that "Moves Itself"

This isn't strictly speaking true, but the idea came from that AMS Testing program that we helped Bob Carmany with. In that program there were sections of code that re-located themselves from one AMS page to another so that all pages of the AMS could be tested without destroying or losing the program's own code. In our Album case, we decided to use a small section of code outside the actual Album code to move Album.

To do that, we first AORG to location >BFB8, so there's enough room between there and >C000 to do a few things. First, the code starting at >BFB8 initializes the AMS memory. At that point the mapping registers are set so that page 0 is "mapped" to >0000, page 1 to >1000, etc. Pages 2 and 3 are then mapped to the area containing the main MIDI-Master code, and page >B and >C are mapped to >B000 and >C000, where the Album code resides. Next, the little section of "move it" code checks to see if an AMS has been found, and branches to a "sorry, no AMS found" message display in the main program if no AMS is present. So long as an AMS is there, the code proceeds to do its "move" operation. First, it sets page 1 to map at >A000. Next it moves everything from >C000 to the end of Album's code into page 1, now masquerading as >A000. Now with the Album's code all safely stashed in page 1, it re-maps page 1 to appear as >C000, then jumps to >C000 to start executing Album.

Confused? Please don't get confused yet, as this soup gets thicker with continued stirring. Album operates to do a few things on its own, such as cataloging a disk, but to actually load and play music, it uses routines in the main program part, which is sitting there in pages 2 and 3. Whenever that happens, the main program does a re-mapping of the High Memory addresses, so that >A000 thru >FFFF are actually pages 4 through 9 of the AMS. Thus while the main program's routine is executing, >C000 is actually page 6 (or higher) of the AMS card. None the less, the Album part is still kept in page 1, so it doesn't get overwritten when music data gets put into >C000.

Before Album branches into the main code, it sets a flag that's in the main program's data section. Each routine that gets called by Album has been modified so that before an exit from the routine, that flag is checked, and if it's set, a mapping is done on the AMS to set page 1 back to the >C000 address space. This way, control returns to Album, since page 1 is once again treated as >C000. Yes, it sounds complicated, but it really isn't, once you understand how the AMS card and its mapping registers work.

2.1.4. Workspace Registers

One potential problem when doing this kind of fooling around is the possibility of losing one's workspace registers. We "headed that off at the pass" by using two sets of registers, both of which are in Ram Pad. This way, our register spaces don't get affected by the mapping of pages in the AMS. The Album part uses >8300 thru >831F, and the main program uses >8320 thru >833F. Right at the start of the "move" section of Album, the workspace is set by LWPI >8320, so that the "move" addresses can be kept in registers and won't be affected by the move. Also, the code that does the re-mapping operations, except for one operation in the "move" section, is all kept in the main program's part, where pages 2 and 3 remain mapped to >2000 and >3000 throughout all phases of the program's running.

2.1.5. A Quick Review

MIDI-Master and its Album feature are contained in two E/A Option 5 program files. The first, MASEXA, has a header starting with >FFFF, so that once it's been loaded, the second program file MASEXB will also be loaded. MASEXA's header includes a load-point specified as >BFB8, and a length just a little under 4096 (>1000) bytes. Thus the loader will place this code into the high memory starting at >BFB8 and running through most of the >C000 area. At this point the AMS Mapper is inactivated, so this stuff actually goes into pages >B and >C of the AMS memory. MASEXB, which is nearly >2000 bytes, loads into >2000 and >3000, which correspond to pages 2 and 3 of the AMS memory. These pages are not re-mapped at any time during the program. After the code at >BFB8 does its "move" operation, the Album code resides in page 1 of the AMS, which is mapped to act as >C000 while Album itself is in control.

2.1.6. Fringe Benefits

By our using the >B000 and >C000 sections of high memory as the initial load area for Album, we not only made it possible to run Album without Mini-Memory, but also made it possible to load the two program files in many ways. There is even a LOAD program on the disk which will load both programs from Extended Basic. Both can also be loaded via Load Master by selecting MASEXA from its catalog listing. These two programs can also be loaded via Ramdisk menu, from Funnelweb, or from E/A Option 5 itself. Yes, there is a Santa Claus, and he's offered up an interesting gift in the AMS version of MIDI-Master.

2.1.7. Cataloging the SCSI drive

Our friend Lew King has inspired several of our AMS products since he first purchased his AMS card. This time, however, he's inspired a product that doesn't need the AMS.

Lew reported that he was to a degree happy with his SCSI, except that the cataloging program supplied with SCSI was written in Extended Basic, and was painfully slow in operation. Lew had used our AMS Slideshow with a sub-directory on his SCSI, so he knew that we had ready-made routines capable of reading the catalog of any sub-directory on his SCSI (or the root directory if desired). The cataloging part of AMS Slideshow is tailored so that it "filters" the file names, showing only those ending in _P, which are assumed to be TI-Artist picture files. Lew figured (correctly) that we could remove that file name filter and create a general-purpose cataloger for his SCSI.

It took only a few days to have an initial working version, and then some refinements were added to make it a more complete product. The thing Lew wanted most was speed, and this program gives him that. It reads a directory or sub-directory from the SCSI in about a second or two depending how many files are there. The list appears on screen with the first 22 files showing. FCTN-4, X, x, CTRL-X or FCTN-X "pages" through the list in the forward direction and FCTN-6, E, e, CTRL-E or FCTN-E pages backward.

Two features are important to Lew, and probably to anyone else with a SCSI. The initial prompt field allows up to a 40 character path name, so that the exact path desired can be cataloged right away. For example, if Lew wants to see what's in his UT sub-directory, he types SCS1.UT. at the prompt, then presses ENTER and very quickly gets the contents of SCS1.UT on screen. The second important feature is that by simply pressing P or p while the catalog is on screen, he gets a very rapid printout of that sub-directory sent to his printer.

2.1.8. More "Fringe Benes"

SCSICAT works on Lew's SCSI drive, but also will catalog any of my Horizon Ramdisks, any Floppy drive, and even the older Myarc Hard Drive systems. Thus it's sort of a general purpose quick cataloger.

2.1.9. Today's Sidebar

Not too big this time, just a couple of sample "snippets" for your amusement. First in the sidebar is the "move" part of the MIDI Album code. This uses subroutines in the main program code, and if no AMS is found, it branches to a place in the main code unconditionally. We've also shown the code from the main program that sets page 1 to act as >C000, just so you can make the connection.

There's also a small portion of source code from SCSICAT. In this case, we wanted you to see a small trick we used to speed up the process of getting the numeric variables from the VDP Buffer. By reading those directly from VDP into FAC and ARG, we speed up the process of dealing with doing the necessary floating point math operations. In the case of the numbers for capacity of the SCSI drive, we keep these in floating point format because they are generally too large to be converted to integer values. For types, sizes and record lengths of files, we convert to integers before doing anything else with them. The file records from the catalog get tucked away in memory as groups of 16 bytes each. That's ten for the name, (which we fill out to ten with spaces) then two bytes for each number after conversion to integer format.

2.1.10. Another Myarc Mystery

For some reason, files of the D/V 80 type that we have gotten from Geneve users do not have the usual 2 reported in the TYPE number from the catalog operation. Instead, their type is given as 18! On our Ramdisk menu's Show Directory function, these show up as H80 instead of d80. It took a while to figure out what to do with these, so they'd be properly identified as DIS/VAR 80. In the end it was quite easy, since 18 when expressed in Hex is >12. Thus if we simply ANDI the value to 7, that strips off the 1 hex digit, leaving just the 2, which our program translates to DIS/VAR for display. For those who want to play around with such things, we'll pass along that we found out the TYPE number for sub-directory names on SCSI drives. They show up as type 6, when you might have thought type numbers didn't go past 5. In SCSICAT, those get reported on the screen and printouts as SUB-DIR, with no sector size (always 2) or record size shown. We don't have a clue why those Geneve files show up as TYPE 18, but at least we found an easy way to deal with them. Load Master, by the way, is not confused by these files, as it examines the directory sector bit-by-bit and correctly identifies these that way. Load Master cannot, however, catalog SCSI or hard drives of any kind. It works on floppies and Ramdisk drives only.

That's it for this time. Hope you'll enjoy this rather heavy stuff.

```
* SIDEBAR 77
* SOME SNIPPETS FOR YOUR AMUSEMENT
* CODE BY BRUCE HARRISON
*
* PART ONE FROM NEW MIDI ALBUM
*
       DEF  ALSTRT
       AORG >BFB8
ALSTRT LWPI >8320     WORKSPACE IN RAM PAD
       BL   @AMSINI   INITIALIZE AMS MAP
       C    R1,@>401E IS AMS PRESENT?
       JEQ  ALSTR0    IF SO, PROCEED
       B    @NOAMS    ELSE BRANCH INTO MAIN PROGRAM
ALSTR0 SBO  0         TURN ON AMS CARD
       LI   R1,>100   1 IN LEFT BYTE R1
       MOVB R1,@>4014 SETS PAGE 1 TO >A000
       SBZ  0         CARD OFF
       SBO  1         MAPPER ON
       LI   R0,ALENT  START OF ALBUM CODE (>C000)
       LI   R1,>A000  START OF PAGE 1 (AS >A000)
       LI   R4,ALEND->C000  LENGTH OF STUFF TO MOVE
MOVALB MOV  *R0+,*R1+  MOVE A WORD OF ALBUM TO PAGE 1
       DECT R4        DEC COUNT BY 2
       JGT  MOVALB    IF >0, REPEAT
       SETO @ALBFLG   SET ALBUM FLAG IN MAIN PROGRAM
       BL   @SETP1C   SET PAGE 1 TO >C000
       JMP  ALENT     JUMP TO START OF ALBUM
ENDMOV EQU $
       AORG >C000     LOADS AT >C000 IN HIGH MEMORY
ALENT  B    @ALBNTR   BRANCH TO START OF CODE
*
* FOLLOWING ARE SUBROUTINES IN MAIN
* MIDI-MASTER THAT ARE USED BY THE ABOVE
*
* AMSINI - INITIALIZES MAPPING OF AMS CARD TO "NORMAL"
* MEANING PAGES ARE AT THEIR NAMED ADDRESSES
* E.G PAGE 2 IS >2000, 10 IS AT >A000, ETC
* MAPPER IS NOT TURNED ON
AMSINI LI   R12,>1E00    AMS CRU BASE
       SBO  0            TURN ON AMS
       LI   R1,>FEFF     -257 IN R1
       LI   R0,>4000     START OF MEMORY
AMSLP  AI   R1,>0101     ADD 1 PAGE (257)
       MOV  R1,*R0+      MOVE 2 BYTES TO MEM-MAPPER
       CI   R0,>4020     ALL DONE?
       JLT  AMSLP        NO, INIT MORE
       RT                RETURN
*
* ROUTINE SETP1C
* SETS AMS PAGE 1 TO >C000
```

```
*
SETP1C LI   R12,>1E00    AMS CARD CRU
       SBZ  1            TURN OFF MAPPER
       SBO  0            TURN ON CARD
       LI   R1,>100      1 IN LEFT BYTE R1
       MOVB R1,@>4018    PAGE TO >C000
       SBZ  0            TURN OFF CARD
       SBO  1            TURN ON MAPPER
       RT                RETURN
*
* PART TWO - SNIPPET FROM SCSICAT
*
* AT THIS STAGE, CATALOG FILE IS OPEN
* AND WE'RE READING THE DISKNAME RECORD (#0)
*
RDDNAM LI   R1,>0200    READ OPCODE
       BL   @DSOP3      SUBROUTINE PERFORMS READ
       JNE  RDNM1       JUMP IF NO ERROR
       B    @REDERR     ELSE REPORT ERROR
RDNM1  LI   R0,>1000    VDP BUFFER
       MOV  R0,R6       SAVE ADDRESS IN R6
       INCT R6          ADD 2 TO R6
       BLWP @VSBR       READ LENGTH OF DISK NAME
       MOVB R1,@DNAME   MOVE TO STORAGE
       MOVB R1,R2       AND TO R2
       SRL  R2,8        RIGHT JUSTIFY
       MOV  R2,R4       STASH IN R4
       JEQ  CLRROW      IF ZERO, SKIP AHEAD
       INC  R0          R0 POINTS AT NAME ITSELF
       LI   R1,DNAME+1  TEXT STORAGE
       BLWP @VMBR       READ THE DISK NAME
CLRROW CLR  R0          SCREEN ORIGIN
       LI   R2,40       40 CHAR ROW
       BL   @CLFLD      CLEAR TOP ROW OF SCREEN
       CLR  R5          R5=0
       LI   R1,DNAME    POINT AT STASHED DISK NAME
       BL   @DISSTR     DISPLAY THAT
       A    R4,R5       ADD SAVED LENGTH TO R5
NONAM  MOV  R6,R0       GET R6 BACK INTO R0
       A    R4,R0       ADD LENGTH OF DISK NAME
       AI   R0,9        SKIP OVER THE F.P TYPE NUMBER
       LI   R1,>835C    POINT AT F.P. ARGUMENT IN PAD
       LI   R2,8        EIGHT BYTES IN F.P. NUMBER
       BLWP @VMBR       READ TOTAL CAPACITY ON DRIVE
       AI   R0,9        AHEAD TO NEXT F.P. NUMBER
       MOV  R0,R6       SAVE R0 IN R6 AGAIN
       LI   R1,>834A    POINT AT F.P. ACCUMULATOR IN PAD
       BLWP @VMBR       READ EIGHT BYTES (SECTORS FREE)
       BLWP @XMLLNK     USE XMLLNK
       DATA >0700       SUBTRACT FAC FROM ARG (RESULT AT FAC)
       LI   R0,USDNUM   SECTORS USED STORAGE
PUTUSD MOVB *R1+,*R0+   COPY A BYTE FROM FAC TO USED
       DEC  R2          DEC COUNT
       JNE  PUTUSD      REPEAT IF NOT 0
       MOV  R5,R0       GET SCREEN LOCATION BACK FROM R5
       BL   @DISFPN     SUBROUTINE DISPLAYS FLOATING POINT # (SECTORS USED)
       INC  R0          POINT AHEAD ONE SPOT
       MOV  R0,R5       SAVE ADDR IN R5
       LI   R1,USDSTR   "USED"
       BL   @DISSTR     DISPLAY THAT
       A    R2,R5       ADD LENGTH
       MOV  R6,R0       GET FILE RECORD ADDRESS BACK
       LI   R2,8        EIGHT BYTES
       LI   R1,>834A    FLOATING POINT ACCUMULATOR
       BLWP @VMBR       SECTORS FREE NUMBER AGAIN
       LI   R1,FRENUM   AND OUR STORAGE FOR LATER
       BLWP @VMBR       READ INTO STORAGE
       MOV  R5,R0       SCREEN ADDRESS BACK IN R0
       BL   @DISFPN     DISPLAY FLOATING POINT #
       INC  R0          POINT AHEAD ONE
       LI   R1,FRESTR   "FREE"
       BL   @DISSTR     DISPLAY THAT
*
* THAT'S IT
```

2.2. The Art of Assembly. Part 78. Selective Cataloging

by Bruce Harrison

This time we're doing disk cataloging, but in a special way. The idea is simple enough, in that we want to read the catalog of a disk and report out only those files suitable for use with a particular program.

Our friend Lew King inspired the first example for this treatise, when he said that he'd used our Font Designer to create many special fonts for his 24-pin printer, but had caused himself a problem in using the program. That is, he had trouble remembering the names of the font files he has on his SCSI drive. Lew suggested we add a catalog function to that program, so he could pick one for loading right off the catalog screen.

2.2.1. Ending With a Dot

To make the idea as user-friendly as possible, we used the regular FILE NAME input routine that was already in the program, but added a new feature. If you know what file name you want, you simply type in the whole name and that file gets loaded. If you don't, you just enter a "path" name (e.g. DSK1.) and press ENTER. The program sees that the last character in the name you entered is a period, and catalogs that path for you. This works for floppy disks, ramdisks, and hard drives, including the SCSI type. In Lew's case, he could type SCS1.FONT. at the prompt, and the program catalogs the FONT sub-directory on his SCS1 drive.

When we take this directory, we first check the type of each entry read. If the type is not 2, we know this is not a Display/Variable file, so we don't need to include it in our list. Next we skip over to the record size, and check for a record size of 120. If that doesn't match, then this file can be rejected as not suitable for use with the Font Designer program. What we're left with is a list of only those files that are the correct type and record size for our program.

On the screen, Lew gets a list of file names in one to three columns on the screen, but only those that are of the D/V 120 type, which means they are Font files for use with our Font Designer program. The cataloging continues until the end of the Catalog file is found, which means we're through reading the catalog entries. That's when we proceed to put the selection cursor on the screen, and let the user select any file from that path for loading. By simply moving the cursor next to a file name and pressing ENTER, Lew gets his desired Font file to load. We've added a similar catalog function to the QDUMP program on that disk, which is just for downloading fonts to the printer without editing.

In the Sidebar (Part One) is a portion of the source code that shows how each catalog file record is read and how we select only the appropriate files. The name for each file (padded to 10 characters if necessary) gets tucked away in a temporary name location. There are three floating point numbers in the catalog record. The first is the file type, which we check for the value 2. If the type is not 2, we skip to the next record in the file without further checking. The second number is the file size in sectors, which we ignore, and the third is the max record size for the file, and this gets checked for the value 120. If the file type is 2 AND the max record size is 120, then the name gets put on the screen. Otherwise we just try the next file in the catalog.

2.2.2. Flushed with Success

Having gotten that to work correctly, we decided to tackle a somewhat more difficult case, in our AMS Video Titler program. This case was more complicated because the files that can be used by this program are of two kinds. Both kinds are of the type 5 (Program, aka Memory Image). The TI-Artist format pictures always have a file name ending in _P, and are always 25 sectors in size. The Harrison Drawing type picture files don't have the _P at the end of their names, but are always 43 sectors in size.

To accept both of these kinds while rejecting all other kinds of files, we use a process similar to what was done for the font files, but with a twist or two. First the name gets checked to see if the _P is at its end. If that's so, we set a flag for use later in the selection process. Next, we check for type, and if that's not 5, we reject the name. Finally, if the type is correct, we check that flag and compare the size in sectors to either 25 or 43. Those conversant with source code will notice that this code also checks for sector size 24 and 42. That's needed because on a SCSI drive, sector sizes are reported as one smaller than on other drive types. We think that's because the designers forgot to add one for the directory sector that's used for each file, but not included in the number contained in that sector. Which number is checked depends on the state of the flag. If the tests are all passed, the file name gets listed on the screen.

The code that does this is shown in Part two of the sidebar. It's similar in many respects to the code in part 1, but shows the checking of the name for _P endings and the checking for file size in sectors, which is important as a criterion in this program.

Again when all of the catalog has been read, the catalog file is closed and a flashing cursor is placed next to the first name on the screen. ENTER selects that file for loading, and it gets loaded into the current "frame" in the AMS memory.

2.2.3. Another Tough Case

The AMS slideshow program was originally designed for use with floppy drives. Its catalog function simply filters by the presence or absence of the _P on a file's name, meaning a TI-Artist format picture. Since each TI-Artist Picture file takes up 25 sectors, and since floppy disks generally don't have more than 1440 sectors total (DSDD), and since most TI-Artist pictures have an associated 25 sector color file (_C) on the same disk, there wouldn't be many cases where more than 28 picture files would be kept on one disk. The size of the catalog list was then arbitrarily set at 46.

Nobody objected to that until people with SCSI drives showed interest in the AMS Slideshow. On a SCSI drive, one can't have more than 127 files in a sub-directory, but that's the only limit. It was thus possible to have 63 color TI-Artist pictures or 127 black and white ones able to be read from a single directory. What, then, would we do with the file names once we were past 46? The answer was: Page 1 of the AMS. Page one of the AMS provides 4096 bytes of storage space If we find a 47th, then that whole screen gets stashed away in Page 1 (addressed as >A000 thru >AFFF). The screen then clears and there's room for another 46 names. If there's a 93rd name, this process repeats, storing the second screen starting at >A300, and continuing right up to the 127th name. The user can then switch back to previous screens by CTRL-E and forward to later screens by CTRL-X. The selection process works much the same as the older version, allowing selection of names until enough have been selected to completely fill the AMS Card's pages from 4 thru the end. In the case of 1 Meg AMS cards, that would be 84 slides in the sequence.

2.2.4. While We Were There

In the original Slideshow and the earlier AMS versions, if one wanted each slide to remain on screen for a particular amount of time, one could enter a time delay factor from 0.1 second through 300 seconds. When first we developed the AMS version, Jim Krych asked whether one could put a zero in for the time delay, so that slide changing would happen as fast as possible. At that time, the answer was no, because we'd put in some protection against time entries shorter than 0.1 second.

While we were revising the code, though, we looked again at how the timing was done, and determined that using zero as the delay time would do no harm. We therefore took out the lower limit protections, so one can now either leave the time entry field blank or put in a zero with the same result, and the slides will change as rapidly as possible.

2.2.5. Two More "Goodies"

After putting in the "zero time" feature, we thought it might be nice if the user could pause the show at any frame and then allow it to continue when he'd finished looking at that picture. There was already a keyboard scan included in the timing loop, but it looked only for FCTN-9 to stop the showing and return to the catalog list. By adding just a couple of lines of code, we set this up so that when in the timed mode, one can press and hold the space bar, and the current picture will remain until the space bar is released. While the space bar is down, time stops counting, and resumes upon its release. Thus if the time set were two seconds and the user pressed the space bar after one second, the picture would remain for its another second after the space bar was released.

The other minor annoyance we encountered was the business of "How many have I already selected?" This was especially difficult with more than one screenful of names, as we'd have to scan through the screens and count the selection marks. Not good! One can easily lose count and have to start over. We then went back to our bag of tricks and pulled out a small subroutine used in SCSICAT. By slightly modifying that routine, we were able to put the number of slides selected in the upper right corner of the screen. This number goes along when screens change, so that at any time while selecting, the user has the number selected readily available on the screen.

The number increments each time a selection is made, and decrements each time one is deleted, so it always tracks the current number. For our 256K card, that number won't go past 20, at which point selection is disabled because we've selected enough to totally fill the memory. For those with 512K capacity, the number will go to 41, and for those 1 meg users it will go to 84.

All of these upgrades are available for the sum of $1.00, including media and S&H. In the case of the two AMS programs, you must be a current owner of the program to get the upgrade, but the Font Designer is Public Domain, so it's not limited to current owners. To become a current owner of either AMS program (TITLER or SLIDESHOW) will cost you $5.00, including S&H.

```
* SIDEBAR 78
* SOME CODE FRAGMENTS
* TO ILLUSTRATE SELECTIVE CATALOGING
* BY BRUCE HARRISON
* PUBLIC DOMAIN
*
* PART ONE - FROM FONT DESIGNER
* STARTS AT THE INPUT FILE NAME SECTION
*
INFNT  BL   @CLS           CLEAR THE SCREEN
       LI   R0,40*10+9     ROW 11, COL 10
       LI   R1,EFNSTR      "ENTER FILE NAME"
       BL   @DISSTR        DISPLAY THAT
       LI   R0,40*12+1     ROW 13, COL 2
       LI   R1,IPABDT+9    OLD FILE NAME
       BL   @DISSTR        DISPLAY
INFBP  BL   @BEEP          SOUND BEEP
       BL   @ACCEPT        ACCEPT INPUT
       DATA 40*12+1        ROW 13, COL 2
       DATA 38             38 CHARS
       DATA 0              DON'T CLEAR FIELD
       DATA IPABDT+9       RESPONSE ADDRESS
       CI   R8,15          FCTN-9?
       JNE  INOF           JUMP IF NOT
       CLR  @IPABDT+8      CLEAR FILE NAME
       B    @MENU0         BACK TO MENU
INOF   MOV  R2,R2          NAME LENGTH 0?
       JEQ  INFBP          IF SO, TRY AGAIN
       DEC  R1             R1 POINTS AT IPABDT+9
       MOVB *R1,R3         NAME LENGTH TO R3
       SRL  R3,8           RIGHT JUST.
       A    R1,R3          ADD R1 SO R3 POINTS AT LAST CHAR IN NAME
       CB   *R3,@PERIOD    IS THAT A PERIOD?
       JNE  INOFA          IF NOT, JUMP AHEAD
       B    @INFCAT        ELSE CATALOG PATH
INOFA  LI   R9,IPABDT+9    TAKE INPUT NAME
       LI   R10,OPABDT+9   TO OUTPUT NAME
       BL   @MOVSTR        COPY THE STRING
       LI   R1,IPABDT      POINT AT INPUT PAB
       BL   @OPNF          OPEN THE FILE
       JNE  INP1           JUMP IF NO ERROR
* code omitted here that reads the
* font file
* Starting at INFCAT is the stuff that catalogs path
*
INFCAT MOV  R1,R9          COPY R1 TO R9
       LI   R10,DIRPAB+9   POINT AT DIRECTORY PAB NAME
       BL   @MOVSTR        COPY STRING
       BL   @CLS           CLEAR THE SCREEN
OPN0   LI   R1,DIRPAB      DIRECTORY PAB DATA
       CLR  R9             R9=0
       MOV  @ONE,@INSFLG   SET INSERT FLAG
       BL   @OPNF          OPEN CATALOG FILE
       JNE  AMB0           JUMP IF NO ERROR
       B    @INFNO         ELSE BRANCH TO ERROR TRAP
AMB0   LI   R13,2          R13=2
       LI   R14,14         R14=14
AMB1   BL   @REDCAT        READ A CATALOG RECORD
       JNE  AMB2           JUMP IF NO ERROR
       SRL  R0,8           RT. JUST R0
       CI   R0,5           IS THAT END OF FILE?
       JEQ  ALBCLS         IF SO, JUMP TO CLOSE FILE
       BL   @CLOSF         ELSE CLOSE ANYWAY
       B    @INPERR        BRANCH TO ERROR REPORT
AMB2   LI   R0,>1080       POINT AT LENGTH OF NAME BYTE
       BLWP @VSBR          READ THAT TO R1
       MOVB R1,R2          MOVE TO R2
       JEQ  ALBCLS         IF ZERO, WE'RE DONE
       SRL  R2,8           RT. JUSTIFY
       LI   R6,10          R6=10
       LI   R4,TEMFN       TEMPORARY FILE NAME
       INC  R0             ADD ONE TO R0
AMB2B  BLWP @VSBR          READ A BYTE FROM NAME
       MOVB R1,*R4+        PUT INTO TEMFN
       INC  R0             NEXT SPOT IN VDP
       DEC  R6             DEC R6
       JEQ  AMB2A          EXIT IF ZERO
       DEC  R2             DEC ACTUAL LENGTH
       JNE  AMB2B          READ NEXT IF NOT 0
AMB2BA MOVB @ANYKEY,*R4+   SPACE INTO TEMFN
       DEC  R6             DEC MAX COUNT
       JNE  AMB2BA         REPEAT IF NOT ZERO
AMB2A  LI   R6,3           THREE NUMERIC VALUES IN RECORD
REDNUM INC  R0             POINT PAST LENGTH BYTE (ALWAYS 8)
       LI   R1,>834A       POINT AT FAC IN RAM PAD
       LI   R2,8           8 BYTES TO GET
       BLWP @VMBR          READ F.P. NUMBER TO FAC
       BLWP @XMLLNK        USE XML ROUTINE
       DATA >1200          CONVERT F.P. TO INTEGER
       MOV  *R1,R5        MOVE INTEGER TO R5
       A    R2,R0         ADD 8 TO R0
       CI   R6,3          IS THIS THE FIRST PASS?
       JNE  CHKR61        IF NOT, JUMP
       ABS  R5            TAKE ABSOLUTE VALUE OF R5
       ANDI R5,7          LIMIT TO 7
       CI   R5,2          IS IT 2 (IF 2, THEN IT'S D/V TYPE)
       JNE  AMB1          IF NOT, SKIP THIS RECORD
CHKR61 CI   R6,1          IS THIS THE THIRD PASS?
       JNE  RDEC6         IF NOT, SKIP
       CI   R5,120        IS RECORD LENGTH 120?
       JNE  AMB1          IF NOT, SKIP RECORD
       INC  R9            R9 COUNTS FILES
       MOV  R0,R3         SAVE R0 IN R3
       MOV  R13,R0        COPY R13 TO R0
       LI   R1,TEMFN      POINT AT TEMP FILE NAME
       LI   R2,10         10 CHARACTERS
       BLWP @VMBW         WRITE TO SCREEN
       MOV  R3,R0         GET OLD R0 BACK
       AI   R13,40        MOVE DOWN ONE ROW
       CI   R13,960       PAST SCREEN BOTTOM?
       JLT  RDEC6         IF NOT, JUMP
       MOV  R14,R13       ELSE SET FOR NEXT SCREEN COLUMN
       AI   R14,12        ADD 12 FOR NEXT TIME
       CI   R14,40        CHECK 40
       JGT  ALBCLS        IF >, CLOSE FILE
RDEC6  DEC  R6            DEC NUMBER IN R6
       JNE  REDNUM        IF NOT 0, READ ANOTHER NUMERIC
       JMP  AMB1          ELSE READ NEXT RECORD
ALBCLS BL   @CLOSF        CLOSE THE FILE
       MOV  R9,@NFILES    SAVE R9 AS NUMBER OF FILES
       JNE  SELFIL        IF NOT 0, JUMP
       B    @NFLS         ELSE REPORT NO FILES FOUND
SELFIL LI   R3,1          START AT TOP OF SCREEN
       MOV  R3,R4         COPY TO R4
SELF0  AI   R4,12         ADD 12
SELF1  MOV  R3,R0         PUT R3 IN R0
       BL   @KEYBLN       KEY IN WITH BLINK
       BL   @KEYDLY       SHORT DELAY TO PREVENT RUNAWAY
       CI   R8,15         FCTN-9?
       JNE  SELCKX        IF NOT, JUMP
       B    @INFNT        ELSE BACK TO START
SELCKX CI   R8,'X'        CAPITAL X?
* from here on is code to allow user to
* select a file for loading
*
* END OF PART ONE
*
* PART TWO - FROM AMS TITLER
AMB0   LI   R13,2       ROW 1, COL 3
```

```
       LI   R14,14      R14=14 FOR LATER
AMB1   BL   @REDCAT     READ A RECORD
       JNE  AMB2        JUMP IF NO ERROR
       SRL  R0,8        RT. JUST R0
       CI   R0,5        END OF FILE?
       JEQ  ALBCLS      IF SO, CLOSE
       BL   @CLOSF      ELSE CLOSE ANYWAY
       LI   R1,RDRSTR   READ ERROR
       BL   @ERR40      REPORT
       B    @RELOAD     BRANCH BACK
AMB2   CLR  @PGNUM      PGNUM=0
       LI   R0,>0C80    POINT AT FILE BUFFER
       BLWP @VSBR       READ NAME LENGTH
       MOVB R1,R2       COPY TO R2
       JEQ  ALBCLS      IF ZERO, CLOSE FILE
       SRL  R2,8        RT. JUST
       LI   R6,10       MAX NAME LENGTH
       LI   R4,TEMSTR   TEMPORARY STRING
       MOV  R2,R5       SAVE LENGTH IN R5
       INC  R0          POINT AT 1ST CHAR
AMB2B  BLWP @VSBR       READ
       MOVB R1,*R4+     STASH
       INC  R0          NEXT CHAR
       DEC  R6          DEC 10 COUNT
       JEQ  AMB2A       JUMP IF ZERO
       DEC  R2          DEC LENGTH
       JNE  AMB2B       REPEAT IF NOT 0
AMB2BA MOVB @ANYKEY,*R4+ SPACE FILL
       DEC  R6          TIL END OF 10 CHARS
       JNE  AMB2BA
AMB2A  LI   R7,TEMSTR   POINT R7 AT TEMSTR
       DECT R5          DEC LENGTH BY 2
       JLT  AMB1        IF <0, JUMP
       A    R5,R7       POINT AHEAD TO NEXT-TO-LAST
       LI   R3,UNDP     POINT AT _P
       CB   *R7+,*R3+   CHECK _
       JNE  AMB2C       IF NOT, JUMP
       CB   *R7,*R3     CHECK 'P'
       JNE  AMB2C       IF NOT, JUMP
       SETO @PGNUM      SET PGNUM AS FLAG
AMB2C  LI   R6,2        TWO NUMBERS TO CHECK
       CLR  R7          R7=0
REDNUM INC  R0          POINT AT F.P. NUMBER
       LI   R1,>834A    AND AT FAC
       LI   R2,8        8 BYTES
       BLWP @VMBR       READ F.P. NUMBER TO FAC
       BLWP @XMLLNK     USE XML
       DATA >1200       CONVERT F.P. TO INTEGER
       MOV  *R1,R5      MOVE TO R5
       A    R2,R0       ADD 8 TO R0
       CI   R6,2         FIRST NUMBER?
       JNE  CHKR61       IF NOT, JUMP
       ABS  R5           ABSOLUTE VALUE R5
       ANDI R5,7         LIMIT TO 7
       CI   R5,5         IS THIS PROGRAM TYPE?
       JNE  AMB1         IF NOT, SKIP RECORD
CHKR61 CI   R6,1         SECOND NUMBER?
       JNE  RDEC6        IF NOT, SKIP
       MOV  @PGNUM,R1    FLAG SET?
       JEQ  CHK43        IF NOT, CHECK 43
       CI   R5,25        ELSE IS FILE LENGTH 25?
       JEQ  REDIN9       IF SO, JUMP
       CI   R5,24        CHECK LENGTH 24
       JEQ  REDIN9       IF SO, JUMP
       JMP  RDEC6        ELSE JUMP AHEAD
CHK43  CI   R5,43        LENGTH 43?
       JEQ  REDIN9       JUMP IF SO
       CI   R5,42        LENGTH 42?
       JNE  RDEC6        JUMP IF NOT
REDIN9 INC  R9           R9=R9+1
       MOV  R0,R3        SAVE R0 IN R3
       MOV  R13,R0       SCREEN POSITION FROM R13
       LI   R1,TEMSTR    FILE NAME
       LI   R2,10        10 CHARS
       BLWP @VMBW        WRITE TO SCREEN
       MOV  R3,R0        GET R0 BACK
       AI   R13,40       ADD ONE ROW
       CI   R13,960      END OF SCREEN?
       JLT  RDEC6        JUMP IF LESS
       MOV  R14,R13      COPY R14 TO R13
       AI   R14,12       ADD 12 FOR NEXT TIME
       CI   R14,40       PAST RIGHT EDGE?
       JGT  ALBCLS       CLOSE FILE IF SO
RDEC6  MOV  R7,R7        R7=0
       JNE  AMB1         JUMP IF NOT
       DEC  R6           DECREMENT R6
       JNE  REDNUM       IF NOT 0, REPEAT
       JMP  AMB1         ELSE NEXT RECORD
```

2.3. The Art of Assembly. Part 79. Playing It In

by Bruce Harrison

Back when we started on the revisions of MIDI-Master, there was a nagging void in all the improvements we wanted to make. Most "sequencer" programs, including the Cakewalk that we use on our PC, allow the user to get music into the system by simply playing that music on the MIDI instrument while its MIDI OUT port is connected to the computer's MIDI IN. Alas, that was not the case with MIDI-Master.

Mike Maksimik, the original author of MIDI-Master, had tried some experiments toward having the Play-In feature, but with less than acceptable results. After we completed both the AMS and non-AMS versions of the "new" MIDI-Master, it was very clear that Play-In could not be fitted into the structure of those programs, mainly because memory simply would not allow enough extra code, and still allow the entire high memory to be used as music data storage.

2.3.1. A Separate Program

Thus we started with the idea that this capability would have to be made available as a separate program, not part of the MIDI-Master per se, but able to be used ultimately with MIDI-Master. Through some selective dis-assembly and modification of code supplied by Mike, we were able to make the core of a Play-In program and got that core to work fairly quickly.

2.3.2. Borrowing a Concept

The problem facing us, once we were able to receive bytes through the MIDI port, was how to store those bytes in the TI's high memory. Many ideas were considered and rejected for various reasons, mainly because they lacked the necessary efficiency of memory use to be practical on the TI. As part of the "thinking through" process, we tried playing some music into Cakewalk on our ancient PC, and looked at what Cakewalk did with the data. Eureka! There on the PC screen was a concept just waiting to be "borrowed" for the TI.

Cakewalk keeps a metronome running while music is being played in. The metronome keeps track of musical time. When events (pressing or release of keys) are recorded, what goes into memory is the time of the event by measure, beat, and "tick" numbers. To make this simple, let's consider that we're working in 4/4 time, four beats per measure, and a quarter note gets one beat. In Cakewalk, a quarter note gets 120 "ticks" of internal time, while in MIDI-Master a quarter note is 48 "ticks". No matter this difference, the concept of putting an event into memory was important.

On the TI, our internal metronome counts ticks, beats, and measures. In the 4/4 case, 48 ticks constitute a beat, four beats make a measure. Each "tick" actually represents a large number of passes through a delay loop. Typically that "delay factor" is a number between 250 and 1000. When a note event happens, five bytes get written to the high memory. These are the current Measure, Beat, and Tick of the metronome, plus the note value and its "velocity" value. When a note key is struck, the "velocity" byte is always a number between 1 and 127, and when that key is released, the velocity value is 0. Note that nothing gets put into memory between note events. Thus the playing of a note, including its release, requires just ten bytes to get stored in memory.

In the program, before we actually start taking bytes from the MIDI Instrument, we provide the user with a number of prompts so that the parameters can be set. This starts with the DELAY FACTOR, with a default entry of 750 in place. Next comes the TIME SIGNATURE, with 4/4 as a default. Both can be changed easily by just typing in a different choice and pressing ENTER. The third prompt concerns the METRONOME. This defaults to ON, which is what we recommend. The metronome works by sending short bursts of sound via the TI Sound Chip. The starting beat of each measure is accented to make it easier to track measures by sound. Finally there's a prompt for ECHO SERVICE, with N as the default answer. In almost all cases N for NO is the correct answer. Some Yamaha model keyboards will require this service in one operating mode, but generally it is not needed. If your setup needs it, however, it's there by just pressing Y or y at the prompt. If the TI system being used for Play-In has the AMS card, that fact will be known to the program. Also known will be the number of 24K "groups" of memory available in that AMS card. If the AMS is present, then, there will be one final prompt, for the memory group to be used as high memory. This always defaults to the next group that doesn't already contain some music.

2.3.3. Gotta Be Quick

A good player on a keyboard can generate a lot of notes in very little time. These get sent out the MIDI port at a blazing 31,200 baud rate. To keep up with bytes coming into our RS-232 port at such a rate requires a whole different approach to using the RS-232 port. Using a concept borrowed from MIDI-Master, we activate the port by direct actions on the CRU lines, thus avoiding the delays that would be involved in using the RS-232 as a file device. The program "speaks to" the UART in the RS-232 card by what amounts to a direct line, so that bytes can be sucked into a workspace register in very little time. The loop that accepts bytes from the UART is a very tight and fast one. It uses mainly registers, which are in the >8300 memory block (Ram Pad) so that all register actions happen as fast as possible. There are no BLs or BLWPs in this loop, just "straight line" code. The time counts for Measure, Beat, and Ticks are kept in the left bytes of registers, etc.

In the early stages of development, we tried putting a BLWP @KSCAN in the loop so that the user could stop input with a keystroke, but that took far too much time, and notes being played got lost. The current source, then, runs as fast as we could manage, and seems to catch all the notes even with a skilled pianist doing the playing. Our main testers for the program have been Richard and Valeda Bell, of Staten Island. Their advice and encouragement have contributed in a very large way to the development of the program. Encouragement has also come from Mike Maksimik, without whose original MIDI-Master source code none of this would be possible.

Today's sidebar is the portion of code that handles the playing in of notes and storing them in memory. The program resides in low memory, so all of the note events are stored in high memory. You'll see that this code has been made to operate very quickly, with almost everything governed by the values in registers. The workspace is at >8300, just to make all these registers quicker for access.

2.3.4. Once It's in the Memory . . .

Having solved the major problem of taking bytes from the MIDI keyboard into the memory in an organized and efficient manner, we had two other problems to solve. First was "How do we know when the player has stopped playing?" After a lot of thought, we decided that if two complete measures passed by without any note events (note struck or note released) we would simply stop and return to the program's menu. We reasoned that there are almost zero musical works in which the instrument is at rest for two complete measures. (There is a case in the very comical Hoffnung Festival records where a mythical piece has two measures of silence, and "The first measure of silence is in 4/4 time, while the second is in 8/8 time.") We don't think anyone will be playing in mythical pieces from Hoffnung, so the two measure silence criterion has stayed.

So now there's music in the memory as played in by the family musician. What can we do with it? The first and most obvious thing is to simply play the music back through the instrument. For that, we've got Option 2 on our menu, Play Back. Since we might want to hear the music at a faster or slower tempo than what was played in, there's a prompt for the DELAY FACTOR, with a default the same as the play-in already in place. Once the prompt has been answered, the computer starts a count of measures, beats, and ticks running, but without making sound. Instead, it keeps comparing its present "time count" to the next event stored in memory. When the time matches, that event gets sent out to the MIDI instrument. Thus each event goes out at the relative time of its storage, and play of notes is paced just like the original playing. The relative timing is accurate to the nearest tick, which is 1/48th of a quarter note duration. That's the same "resolution" as used in MIDI-Master.

2.3.5. Saving Your Work

There's always the possibility that you might not want to keep your TI system up and running 24 hours a day just to keep that little minuet in memory, so we added two more menu choices to the list. SAVE IMAGE takes the music from memory and puts it out to a disk file just as it sits in memory. The one exception is the six bytes from >A000 through >A005. The actual music stores starting at >A006. The bytes before that are filled up with first the delay factor in use, then the beat length in ticks, and finally the number of beats per measure. This means that the output file will store those "vital stats" about this piece, so that when it's later brought back in, the tempo and time signature will be set correctly for playing.

The number and size of files created by SAVE depends on how much music was played in. Only that memory which was filled with music will be saved to one or more files. This can range from just one file of two sector size through a whole series of files of 33 sector size. Because we've used a whole new way of storing what's played in memory, the files saved by this program in its own memory-image format are NOT compatible with any of the versions of MIDI-Master, nor are files saved by MIDI-Master compatible with this program. The ones saved by this program can of course be re-loaded by this program and played back by it, and can also be used to create source files (see below) which ARE compatible with the updated versions of MIDI-Master.

2.3.6. Loading Saved Works

Item 4 from the menu is for loading a previously saved file or series of files. As with saving, just type in the name of the first file in the series, and the program will keep loading until it finds a non-existing member of the series, then it will return to the menu.

2.3.7. The De-Compiler

Item 5 from the menu, CREATE SNF, is the "payoff" for having played some music into this program. This option takes the music in memory and creates from it one or more D/V 80 files in SNF source format. These files are designed for use with the updated versions of MIDI-Master, either version 2.5Z or 2.5A. If a work is too long to make the source file a length that's "editable" on the TI, the output file will be split into a series of such files, each of which is of a size that can be edited with the E/A Editor, with the TI-Writer Editor or with either of the Funnelweb Editors. After editing, these files can be re-combined using our TOOL2UT program into a master file for the MIDI-Master compiler.

The idea here was to allow the user, having gotten a piece into the "system" through play-in, to be able to revise and refine his work. In these editable source files, such things as note durations and rests can be changed as desired. Mistakes made in the playing can be corrected, patch changes can be introduced to change instruments, and so on. In other words, Play-In gives the user a solid and consistent starting point for making a really finished musical work available to MIDI-Master. With the new versions of MIDI-Master, there are numerous possible additions that can be made. For example, with the "Voice Volume" feature, the playing volume of each track in the composition can be tailored just as desired by the musician. Since the music has been played in by a musician, you might want to leave the numeric note and rest durations just as they were, since this will leave the finished work with a more "human" quality than it would have if "perfected" by edits.

2.3.8. Loop Closed

This addition to the MIDI-Master "family" of programs closes the loop. You can now start with music that's played in, make SNF source files from that, edit those files to your own satisfaction, and then compile these SNF files through the new editions of MIDI-Master. Play-In is available to all MIDI-Master owners for the paltry sum of $5.00, including S&H.

See you again in two months.

```
* SIDEBAR 79
* SNIPPET FROM PLAY-IN
* CODE BY BRUCE HARRISON
* PUBLIC DOMAIN
*
* FIRST, THE SETUP SUBROUTINE IS USED
* TO SET THE RS232 FOR MIDI BAUD RATE
*
GOSET  BL   @SETUP          SET UP RS232
*********************
* THE NEXT PART SETS UP THE INTERNAL
* SOUND CHIP FOR THE METRONOME SOUNDS
*********************
       LI   R2,BEEP         OUR BEEP SOUNDS
       LI   R4,6            SIX BYTES
SNDBEP MOVB *R2+,@>8400     SEND TO SOUND CHIP
       DEC  R4              DEC COUNT
       JNE  SNDBEP          REPEAT IF NOT 0
*********************
* NEXT PART SETS UP INITIAL CONDITIONS
* FOR PLAYING IN MUSIC
*********************
       MOV  @DELAY,R13      DELAY FACTOR IN R13
       LI  R15,>0100        LEFT BYTE R15=1
       SETO R0              R0=>FFFF
*********************
* R5 SET TO -4 TO SERVE AS A COUNTDOWN
* IF NOTHING GETS PLAYED IN
* R0 WILL BECOME 0 ONCE SOMETHING'S PLAYED
*********************
       LI   R5,-4           R5=-4
       CLR  R1              R1=TOGGLE FLAG
       LI   R2,>B090        CTRL & CHAN1
       MOV @BEATL,R9        BEAT LENGTH IN HIGH BYTE R9
       LI  R3,BUFFER        POINT AT >A006
       CLR R6               MEASURE = 0
       MOV @CARD,R12        CARD CRU ADDR IN R12
       SBO 7                TURN ON LED
       MOV @UART,R12        UART CRU ADDR IN R12
********************
* FROM HERE ON, THE ACTIVITY LIGHT ON
* THE RS-232 STAYS ON, AND R12 HAS THE
* CRU ADDRESS FOR THE UART CHIP ON THAT CARD
********************
       MOV @ECHFLG,R11      R11 SIGNALS ECHO
NMEAS  CLR R7               BEAT 0
       CLR R4               R4=0
       MOV R3,@OLD3         SAVE MEMORY POINTER
*******************
* AT THE START OF EACH BEAT, THE SOUND CHIP
* IS GIVEN A VOLUME BYTE SO THAT IT WILL
* SOUND AS A METRONOME FOR ONE TICK DURATION
* THE START BEAT OF EACH MEASURE IS LOUDER
* THAN THE OTHER BEATS. (WHEN R7=0)
*******************
NBEAT  MOV R7,R7            IS R7=0?
       JNE  BEAT            JUMP IF NOT
       MOVB @LOUD,@>8400    LOUD VOLUME TO SOUND CHIP
       JMP   CLRTIC         THEN JUMP
BEAT   MOVB @SOFT+1,@>8400  SOFT VOLUME TO SOUND CHIP
CLRTIC CLR R8               TICKS=0
*******************
* AFTER A BEAT HAS RUN ONE TICK,
* WHICH IS 1/48TH OF A QUARTER NOTE,
* THE SOUND CHIP IS SILENCED.
*******************
NTICK  CI  R8,>0100         TICKS=1?
       JNE LOAD10           JUMP IF NOT
       MOVB @SILENT,@>8400  SHUT UP METRONOME
       MOVB @SILENT+1,@>8400 BOTH GENERATORS
LOAD10 MOV  R13,R10         DELAY COUNT TO R10
*******************
* CODE HERE DETERMINES IF THE MIDI INSTRUMENT
* HAS SENT A BYTE, AND ACTS ACCORDINGLY
*******************
GETBYT TB  >15              TEST BIT FOR BYTE READY
       JNE NOBYTE           JUMP IF NO BYTE
       STCR R4,8            TAKE 8 BITS INTO R4
       SBZ  >12             SHUT OFF INPUT
       CB   R4,R2           LEFT BYTE R4=>B0?
       JEQ  CTRLPI          IF SO, INPUT PEDAL
       MOV  R4,R4           CHECK R4
*******************
* AT THIS POINT IF THE BYTE JUST TAKEN
* FROM THE INSTRUMENT IS EITHER A NOTE OR A VOLUME,
* R4 WILL BE EITHER 0 OR A POSITIVE NUMBER
* IF IT'S A NEGATIVE OTHER THAN >B0, WE'LL REJECT IT
*******************
       JLT  GETBYT          IF NEGATIVE, IGNORE BYTE
       MOV  R0,R0           IS R0=0?
       JEQ  MOV1            IF SO, JUMP
       CLR  R0              SET R0=0
       CLR  R6              SET MEASURE=0
MOV1   MOV  R1,R1           CHECK R1
       JNE  MOVR4           JUMP IF NOT 0
******************
* A NOTE EVENT COMES IN AS TWO BYTES, ONE FOR THE
* NOTE VALUE AND ONE FOR THE VELOCITY (AKA VOLUME)
* R1 STARTS OUT 0, SO THE TIMING GETS SENT TO
* MEMORY BEFORE THE NOTE.  AFTER THAT HAPPENS,
* R1 IS INVERTED SO THAT THE VOLUME BYTE WILL
* GO INTO MEMORY JUST AFTER THE NOTE IT'S SUPPOSED
* TO ACCOMPANY.  AGAIN R1 GETS INVERTED AFTER THE
* VOLUME IS STORED, SO IT'S ZERO FOR THE NEXT NOTE
******************
       MOVB R6,*R3+         MEASURE TO MEMORY
       MOVB R7,*R3+         BEAT TO MEMORY
       MOVB R8,*R3+         TICK TO MEMORY
MOVR4  MOVB R4,*R3+         INPUT BYTE TO MEMORY
       CI   R3,>FFDE        MEMORY FULL?
       JGT  MEMER           IF SO, ERROR
******************
* INVERT R1 MEANS THAT R1 TOGGLES BETWEEN
* BEING ZERO AND BEING >FFFF (ALL BITS ON)
******************
       INV  R1              INVERT ALL BITS IN R1
       MOV  R11,R11         ECHO REQUIRED?
       JEQ  NOBYTE          IF R11=0, NO ECHO
******************
* FOR MOST KEYBOARDS, THIS NEXT SECTION WILL BE
* SKIPPED BECAUSE R11 WILL BE ZERO.
* FOR CERTAIN YAMAHA MODELS IN CERTAIN OPERATING
* MODES, R11 WILL BE NON-ZERO, SO THE BYTE JUST
* RECEIVED WILL BE SENT BACK OUT TO THE KEYBOARD
*******************
       SBO  16              SET BIT FOR OUTPUT
ECLP1  TB   22              CHECK AVAILABLE
       JNE  ECLP1           REPEAT IF NOT
       LDCR R4,8            SEND 8 BITS TO OUTPUT
       SBZ  16              RESET OUTPUT BIT
NOBYTE DEC  R10             DECREMENT DELAY COUNT
       JNE  GETBYT          REPEAT IF NOT 0
*******************
* WHEN A DELAY FACTOR CYCLE IS DONE, THE
* NEXT SECTION WILL INCREMENT THE TICK, BEAT, AND MEASURE
* COUNTS AS APPROPRIATE.
*******************
INCTIC AB   R15,R8          ADD 1 TO TICKS
       C    R8,R9           EQUAL 1 BEAT?
       JEQ  INCR7           IF SO, JUMP
       JMP  NTICK           NEXT TICK
INCR7  AB   R15,R7          ADD 1 TO BEAT
       CLR  R8              TICKS=0
BPM1   CI   R7,>0400        = BEATS PER MEASURE?
       JEQ  INCR6           JUMP IF SO
       JMP  NBEAT           NEXT BEAT
INCR6  AB   R15,R6          ADD 1 TO MEASURE
       C    R3,@OLD3        HAS POINTER CHANGED?
       JNE  LEDIN           IF YES, JUMP
******************
```

```
* IF R3=OLD3, A MEASURE HAS PASSED WITHOUT ANY
* INPUT FROM THE KEYBOARD.  AT STARTUP, R5 WAS SET
* AT -4, SO R5 WILL BECOME POSITIVE IF WE'VE JUST
* STARTED AND FIVE MEASURES HAVE PASSED WITHOUT INPUT.
* IF THERE HAS BEEN INPUT, CONTROL WILL JUMP TO
* LEDIN, AND R5 WILL BE SET TO -1
* AFTER THAT, TWO MEASURES OF SILENCE WILL
* CAUSE R5 TO BE POSITIVE AND WE'LL EXIT
******************
       INC  R5              R5=R5+1
       JGT  EXIT            EXIT IF >0
       JMP  NMEAS           ELSE NEW MEASURE
LEDIN  SETO R5              R5=-1
       JMP  NMEAS           NEW MEASURE
*****************
* THE SECTION BELOW TAKES A CONTROL ACTION
* (USUALLY SUSTAIN PEDAL ACTION) INTO MEMORY
* THAT CONSISTS OF A >B0, >40, >7F STRING
* AFTER THAT STRING, IF A NOTE EVENT COMES IN
* DURING OUR TIMEOUT, IT WILL BE SENT STARTING
* WITH A >90 BYTE, WHICH WILL ALSO BE STORED
* AND WILL CAUSE EXIT FROM THIS LOOP
* IF TIMEOUT HAPPENS (R1 BECOMES 0) THEN THE
* >90 WILL BE SENT ANYWAY AND WE'LL RETURN TO
* THE NORMAL LOOP ABOVE.
*****************
CTRLPI SWPB R2              R2=>90B0
       MOV  R0,R0           R0=0?
       JEQ  CTRL0           JUMP IF YES
       CLR  R0              SET R0 TO 0
       CLR  R6              SET MEASURE TO 0
CTRL0  MOVB R6,*R3+         MEASURE TO MEMORY
       MOVB R7,*R3+         BEAT TO MEMORY
       MOVB R8,*R3+         TICKS TO MEMORY
       MOVB R4,*R3+         >B0 BYTE TO MEMORY
       LI   R1,50           TIMEOUT COUNT IN R1
CTRL1  TB   >15             BYTE READY?
       JNE  CTRL2           IF NOT, REPEAT
       STCR R4,8            BYTE TO R4
       SBZ  >12             RESET BIT >12
       MOVB R4,*R3+         BYTE TO MEMORY
       CB   R4,R2           WAS THAT >90?
       JNE  CTRL1           IF NOT, GET NEXT BYTE
CTR0A  SWPB R2              ELSE SWAP R2 TO >B090
       CLR  R1              R1=0
       JMP  GETBYT          BACK FOR NOTE BYTE
CTRL2  DEC  R1              DEC TIMEOUT COUNT
       JNE  CTRL1           REPEAT IF NOT 0
       MOVB R2,*R3+         PUT >90 IN MEMORY
       JMP  CTR0A           THEN JUMP
```

```
******************
* WE GET TO EXIT IF EITHER OF TWO THINGS HAVE
* HAPPENED.  IF WE STARTED THE LOOP AND NOTHING
* CAME IN FROM THE KEYBOARD FOR FIVE MEASURES,
* WE EXIT.  IF SOMETHING WAS PLAYED IN AND THEN
* TWO COMPLETE MEASURES HAVE SEEN NO INPUT, WE
* EXIT.
******************
EXIT   MOV  R3,@ENDMUS      SAVE ENDING POINTER
       MOV  @CARD,R12       CARD CRU ADDR IN R12
       SBZ  7               SHUT OFF LED
       MOVB @SILENT,@>8400  SILENCE METRONOME
       MOVB @SILENT+1,@>8400 BOTH GENERATORS
       B    @MENU           BACK TO MENU
```

2.4. The Art of Assembly. Part 80. What's a NewLine?

by Bruce Harrison

Is this number 80? How can that be? How can he keep on cranking these out? Well, besides being a pretty "cranky" old guy, it helps if the Assembly guy keeps writing new Assembly programs on his TI. That way, there's always new ground to cover in this column.

2.4.1. The PC Connection Problem

The subject of today's column starts with a small change in our use of the PC computer. We have for a long time been "holdouts" from the world of Internet. That changed forever when our friend Lew King gave a presentation at the 1998 Lima M.U.G. Conference. Lew used a TI computer and Term80 to connect to the Internet live during his lecture. Two things about this impressed my partner Dolores. First, just how easy it could be to get into various websites, and second, how even with a text-only connection one could reach so much nifty information. We talk to Lew on the phone frequently, but seeing his demo of internet access was a revelation. After the demo, we talked to Lew at length. Like us, Lew also has a PC at home, and he assured us that what he did from the TI could also be done very easily from our "ancient" Tandy 1000 SX PC.

Lew even found us a modem at a "Ham Fest" somewhere in his state (PA) and shipped it to us. He also provided a program called TELIX, made by a now-defunct Canadian company, that would allow our old PC to talk to the world as a VT-100 Terminal. Thus we're now able to get into that other world right from our living room. We can get to many resources, including the catalogs of various libraries, the articles in newspapers, scientific journals, and so on.

2.4.2. Meanwhile, there's Cakewalk (TM)

Dolores is our family musician. She's the one who did all that Assembly Language music for the sound chip, and who's done numerous pieces on MIDI-Master. She helped at every stage in the re-vamping of MIDI-Master by testing and suggesting improvements. Still, when she wants to program some MIDI music quickly, she uses the Cakewalk program on our Tandy 1000 SX. That program is far easier to use, and allows direct entry of music into a file that can be played instantly without having to be compiled. Hmmm. . . Maybe someday we'll have "son of MIDI-Master" with that capability on the TI. Hmmm... In some cases, she's done a musical work first on the PC using Cakewalk, then did the same work over again on MIDI-Master. That's a very hard way to do things. Starting completely over again on the TI is really not an acceptable way of doing things.

With Cakewalk, we have a little PC program called CAKE2ASC. That takes one of the .WRK files made with Cakewalk and converts it into an ASCII format, or in other words a human-readable file format. That being the case, she asked, why could we not take those ASCII files over an RS-232 connection into the TI and then have the TI convert them into SNF source files for MIDI-Master. The resulting files might need some editing to really sound right when compiled by MIDI-Master, but that would be lots quicker and easier than starting over from scratch.

2.4.3. Pre-AMS ways

Some years back, we developed a little system called Smart Connect, to transfer text files between our PC and TI in both directions. Others, most notably Dr. Charles Good, came up with other ways of getting text files from PC to TI. He used Funnelweb's LOAD FILE by simply using a file name like RS232 etc. Still, there's always the problem that many of the files found on PCs are much too large to fit into memory on the TI. In our Smart Connect system, we used Basic on the PC to do the sending and receiving, and allowed the user at the TI to stop now and then and change file names. That worked, but it isn't really a sound solution.

So it was concluded that we needed something new to handle the rather large files created by CAKE2ASC. Using that, even for fairly short musical works, creates files in the 100 Kbyte range. Having done four programs already that use the AMS card to handle large amounts of data, the answer to handling the ASCII files from Cakewalk was obvious. Write a program that would take files of enormous size in through the RS-232 and store all those file records in the many pages of AMS memory.

Having had this brainstorm led within a few days to a program for our own use only, just to take those Cakewalk ASCII files over the 25-wire ribbon cable between our PC and TI. With the 256K AMS available, it was easy to take a 100 Kbyte file over the line. Once it was all in memory, our little program would write it out to disk in pieces of about 12 Kbytes each, so that each file on the TI side would be editable via the Funnelweb editors or TI-Writer. This worked well after we removed a couple of bugs. For the converted Cakewalk files, that was all we needed.

2.4.4. Once That was Working. . .

As so often happens in these things we develop just for our own use, it occurred to us that with some slight changes it could be made into a more General Purpose tool, suitable for use by Lew King and perhaps others in the TI Community who have both TI and PC computers. Here's where that "NewLine" question comes into play.

When one captures or downloads text files through the Internet or from a BBS, the files usually contain what are called NewLine characters. These are no mystery, they're just Line Feeds without a Carriage Return. Putting in Line Feeds [CHR$(10)] makes it easier to send these files through various communication paths where Carriage Returns should not be sent except to signal the end of a record.

In editor programs like Funnelweb, however, we want a carriage return [CHR$(13)] as the end of a line and at the end of a file record. In some cases, one finds files in which there are both Carriage Returns and Line Feeds used together at the end of a line. When the file is being brought in from the RS-232, we simply take it as it comes, as a D/V 80 input. When we're writing it out to TI disk files, however, we run each input string through a "filter" process before it goes to the disk file.

2.4.5. The Filter Rules

Each character in each string gets examined to see if it's a Carriage Return or Line Feed. If it's neither of those, it goes out unchanged. If it's a carriage return, it gets changed to a space. If it's a Line Feed, (aka NewLine) several things happen. First, the Line Feed gets changed to a Carriage Return. Next, the current record up to and including this CR gets sent out as a record to the output file, then the remainder of the input string goes back to resume the filter process. If there are more LFs within the rest of the string, this process repeats until all of the input string has been examined and sent out. If there are no Line Feeds [as in those converted Cakewalk files] then each record (80 characters or less) gets sent to the disk file as-is, that is a record with its original length.

2.4.6. The Editing Process

Having this filter in the program makes the final editing process much easier. Let's say, for example, that what would normally be one line in a text file winds up split between two records in the output file, and that there was a NewLine at the end of the second part. In Funnelweb, we can correct this very easily by putting the cursor at the start of the first of those two lines and pressing CTRL-2. This will re-join to two lines into one if possible, ending its work at the CR which was a LF in the original file. If the split happened in the middle of a word, Funnelweb will leave a space in the middle of that word, which you'll have to delete with FCTN-1. If a line ended up with leading spaces in front of it, those leading spaces can be removed by just pressing CTRL-2 with the cursor on that line. In short, having CRs where those LFs used to be makes cleaning up the file with Funnelweb's Text Edit much easier.

Today's Sidebar

The sidebar today is a portion of the source code from the output part of the program. It shows the filtering process and the means used to write out part of an input record when we've found that LF character. It's fully annotated, so we probably don't need to discuss it in detail here.

2.4.7. The Disk is Available

The disk that has this program is available either from me or from Lima for only $1.00. It's Public Domain, so you may also find a friend who can make you a copy or get it from your User Group's library. It's SS/SD format, so even those with only single sided drives can use it. As usual the disk contains complete instructions, a LOAD program to run it from XB if needed, and an XB program to print the instructions for you.

Note: Cakewalk is a registered trademark of Twelve Tone Systems, Inc.

```
* SIDEBAR 80
* PART OF OUR TRANSFER PROGRAM
* SOURCE CODE - NOT A COMPLETE PROGRAM
*
* THIS PART SAVES THE STRINGS FROM MEMORY
* INTO A SERIES OF DISK FILES
* IT USES NUMEROUS SUBROUTINES (NOT SHOWN)
* COMPLETE SOURCE IS ON THE TRANSFER P.D. DISK
*
* PUBLIC DOMAIN
* CODE BY Bruce Harrison
*
SAVE   MOV  @AMSFLG,R4   AMS IN USE?
       JEQ  SAVEN        IF NOT, JUMP
*
* IF AMS IS IN USE, THE STUFF BELOW
* RECORDS THE HIGHEST GROUP NUMBER THAT
* HAS CONTENT, THEN RESETS TO GROUP 1
* TO START THE OUTPUT FILES.
*
       MOV  @CURGRP,@HIGRP HIGHEST USED = CURRENT GROUP
       LI   R12,1        GROUP 1 (PAGES 4 THRU 9)
       BLWP @SETGRP      SET TO THAT GROUP
SAVEN  BL   @CLS         CLEAR SCREEN
       LI   R0,10*32+4   ROW 11, COL 5
       LI   R1,SPMSTR    SAVE NAME PROMPT
       BL   @DISSTR      DISPLAY
*
* THE SECTION STARTING AT SAV2 ACCEPTS FIRST OUTPUT
* FILE NAME FOR THE SERIES
*
SAV2   LI   R0,12*32+1   ROW 13, COL 2
       LI   R1,OUTPAB+9  OUTPUT FILE NAME STRING
       BL   @DISSTR      DISPLAY
       BL   @ACCEPT      ACCEPT FILE NAME
       DATA 12*32+1,30,0,OUTPAB+9
       MOV  R2,R2        NAME LENGTH 0?
       JEQ  SAV2         IF SO, TRY AGAIN
       CI   R8,15        FCTN-9 PRESSED?
       JEQ  EXIT2        IF SO, EXIT PROGRAM
       BL   @CLOSE       CLOSE INPUT FILE
       LI   R0,PABLOC    PAB VDP ADDR IN R0
       BL   @DSROP2      OPEN OUTPUT FILE
       JNE  LDR9         JUMP IF NO ERROR
       LI   R0,22*32+2   ROW 23, COL 3
       LI   R1,FNOSTR    "FILE DID NOT OPEN"
       BL   @DISSTR      DISPLAY
       BL   @KEYHNK      HONK & AWAIT KEYPRESS
       LI   R2,32        32 BYTES
       BL   @CLFLD       ERASE ERROR MESSAGE LINE
       JMP  SAV2          TRY AGAIN
LDR9   MOV  @WREFLG,R4    HAS THERE BEEN A WRITING ERROR?
       JEQ  LDR9A         IF NOT, JUMP
       MOV  @SAVR3,R9     GET OLD VALUE R9 BACK
       CLR  @WREFLG       CLEAR THE FLAG
       MOV  R9,R6         STASH R9 IN R6
       JMP  SAV6          JUMP AHEAD
LDR9A  LI   R9,>A000      START OF HIGH MEM
LDR9B  MOV  R9,R6         STASH R9 IN R6
       CLR  @PTTFLG       NOT PART TWO
SAV6   MOV  @PTTFLG,R4    IN PART 2?
       JNE  SAV6A         JUMP IF SO
SAV61  CI   R9,>D000      HALFWAY THROUGH HI MEM?
       JLT  SAV6A         SKIP IF LESS
       JMP  INCSAV        ELSE START NEW FILE
SAV6A  MOVB *R9+,R2       STRING LENGTH TO R2
       CB   R2,@NOKEY     IS THAT >FF?
       JEQ  NMSAVE        IF SO, JUMP
SAV6B  LI   R0,PABLOC+5   POINT AT PAB VDP ADDR +5
       MOVB R2,R1         LENGTH IN R1
       BLWP @VSBW         WRITE TO PAB+5 IN VDP
       SRL  R2,8          RT JUST R2
       MOV  R9,R1         MOVE R9 POINTER TO R1
       MOV  R9,R3         AND TO R3
       MOV  R2,R4         COPY LENGTH TO R4
*
* THE SECTION STARTING AT CMPLF IS THE
* "FILTER" THAT CHECKS FOR CR AND LF AND
* TAKES ACTION IF FOUND
*
CMPLF  CB   *R3,@LFBYTE   LINE FEED?
       JNE  CMPCR         IF NOT, JUMP
       MOVB @ENTERV,*R3   PUT CR THERE
       B    @PTREC        THEN WRITE PARTIAL STRING AS RECORD
CMPCR  CB   *R3,@ENTERV   CARRIAGE RETURN?
       JNE  SVINC3        ELSE JUMP
       MOVB @ANYKEY,*R3   PUT SPACE THERE
SVINC3 INC  R3            NEXT BYTE OF STRING
       DEC  R4            DEC LENGTH
       JGT  CMPLF         RPT IF >0
SAV7A  LI   R0,P2BUF      VDP OUTPUT BUFFER
       BLWP @VMBW         WRITE STRING THERE
SAV7   A    R2,R9         ADD LENGTH TO POINTER
SAV8   LI   R0,PABLOC     0TH BYTE IN PAB
       MOVB @WRITEF,R1    WRITE RECORD OPCODE
       BLWP @VSBW         WRITE TO VDP
       BL   @DSROP4       PERFORM DSR OPERATION
       JNE  SAV6          IF NO ERROR, REPEAT
WRTERR BL   @CLOSE        CLOSE FILE
       LI   R1,SVESTR     "SAVE ERROR"
```

```
SHWWRE LI   R0,22*32+2   ROW 23, COL 3
       BL   @DISSTR      DISPLAY
       BL   @KEYHNK      SEND "HONK", AWAIT KEY
       LI   R2,32        32 BYTES
       BL   @CLFLD       CLEAR AWAY MESSAGE
       MOV  R6,@SAVR3    STASH AWAY R6
       SETO @WREFLG      SET WRITE ERROR FLAG
       B    @SAV2        BACK TO FILE NAME
NMSAVE MOV  @AMSFLG,R4   AMS IN USE?
       JEQ  QUTSAV       IF NOT, JUMP
       MOV  @CURGRP,R12  CURRENT GROUP IN R12
       INC  R12          NEXT GROUP
       C    R12,@MAXGRP  CHECK MAXIMUM
       JGT  QUTSAV       IF GREATER, OUT OF HERE
       C    R12,@HIGRP   CHECK HIGHEST GROUP USED
       JGT  QUTSAV       IF GREATER, OUT OF HERE
       BLWP @SETGRP      SET NEW GROUP
       LI   R9,>A000     POINTER TO START
INCSAV MOV  R9,R6        STASH IN R6
       BL   @CLOSE       CLOSE CURRENT FILE
       BL   @INCNAM      INCREMENT NAME
       INV  @PTTFLG      INVERT PART TWO FLAG
       BL   @DSROP2      OPEN NEW FILE
       JNE  GOSAV6       JUMP IF NO ERROR   G
       BL   @CLOSE       CLOSE EVEN IF NOT OPEN
       LI   R1,FNOSTR    FILE DIDN'T OPEN
       JMP  SHWWRE       ISSUE ERROR
GOSAV6 JMP  SAV6         BACK TO SAV6
QUTSAV BL   @CLOSE       CLOSE FILE
       B    @MENU        BACK TO START
*
* SECTION STARTING AT PTREC TAKES THE
* PART OF THE CURRENT RECORD UP TO AND
* INCLUDING THE NEW LINE (NOW A CR)
* OUT AS A SEPARATE DISK RECORD
*
PTREC  INC  R3           PAST THE LF (NOW CR)
       DEC  R4           DEC COUNT
       JGT  PTREC1       IF >0, JUMP
*
* IF R4 HAS BECOME 0 HERE, THE CR IS AT THE END
* OF THE INPUT FILE RECORD, SO WE CAN PROCEED
* TO SEND AND THEN MOVE ON TO NEXT RECORD IN
* THE MEMORY
*
       B    @SAV7A       ELSE WRITE RECORD
PTREC1 MOV  R3,R7        R3 TO R7
       S    R9,R7        SUBTRACT R9 - R7=LENGTH
       MOV  R7,R2        COPY R7 TO R2
       LI   R0,PABLOC+5  PAB LENGTH BYTE
       SWPB R7           LEN IN HIGH BYTE
       MOVB R7,R1        COPY TO R1
       BLWP @VSBW        WRITE LENGTH TO PAB
       LI   R0,P2BUF     OUTPUT BUFFER
       MOV  R9,R1        R1=START OF CONTENT
       BLWP @VMBW        WRITE TO VDP BUFFER
       LI   R0,PABLOC    0TH BYTE OF PAB
       MOVB @WRITEF,R1   WRITE OPCODE
       BLWP @VSBW        WRITE THAT
       BL   @DSROP4      OUTPUT TO FILE
       JNE  PTREC2       JUMP IF NO ERROR
       B    @WRTERR      ELSE TO ERROR REPORT
PTREC2 MOV  R3,R9        R3 TO R9 POINTER
       MOV  R4,R2        R2=REMAINING LENGTH
       SWPB R2           IN LEFT BYTE R2
       B    @SAV6B       BACK TO MAIN PROCESS
* HERE, THE PROCESS RE-ENTERS THE MAIN
* PART TO PROCESS THE REST OF THE INPUT
* RECORD
*
* THAT'S IT
```

2.5. The Art of Assembly. Part 81. Strange Happenings

by Bruce Harrison

Now that we're "connected" to the rest of the world via our beloved Tandy 1000 SX PC, and even have an E-mail address, we can experience things that never happened to us before. If you don't know what a Tandy 1000 SX is, perhaps a little story will make it clear.

In a phone conversation with our friend John Bull, we described our PC as being a "stone age" model. John asked if that meant it was a 286. 286? Try 8088! I guess it says something about the PC business that John would think my expression "stone age" would mean a 286. Later in the same phone call, John agreed with my putting the 286 in the "bronze age". Of course when we bought that Tandy 1000 SX brand new, it was the "latest and greatest" PC. In less than a year it was becoming obsolete.

Had an E-mail from Stephen Shaw (received over that same "stone age" Tandy) telling how his three year old PC is too old now, and can't run any of the newly-released software. Luckily for us, we've still got all that "stone age" software for our PC, and it still does what we want it to do. Won't run Windows (TM) (R) of course, and won't support Netscape, or run PC99, but it's still a useful and reliable tool. Most of the time it is, anyway.

2.5.1. But we Digress. . .

Now that we're "connected", we get to experience all the good things of Internet access, and some of the bad things, too. We can now capture text from the Internet onto our PC's hard drive by the ton. This creates two problems. First, what are we to do with it all. Some of it can be read on the screen using the TYPE with MORE command, but that gets tiresome. We can print it out, but then where do we store all that paper? In just a few days we're already suffering some information overload.

Then there are those "Frames". Many of today's web sites seem to consist of them, and these sites always remind us that we can't handle them. In many such cases, we're offered a download of a new browser. We know better than to try taking one of those offers. For one thing, there are only about 8 megabytes left on our 10 megabyte hard drive, and that's most likely not near enough for any browser we're offered. Also, of course, the preferred browser will expect us to be running under some version of Windows. No way, Jose!

Still, even as a VT100 terminal emulation, there are some good deals out there. We found a site with thousands of MIDI files available for download, and took some of those. Many of them worked okay with Cakewalk (TM) and played pretty nicely on our new Casio keyboard. But then there were some that sent "mystery" control codes to the Casio, and after a while these "locked up" our keyboard. We don't know why. Getting the MIDI keyboard unlocked required removing its battery power, letting it sit overnight, then sending it some music from MIDI-Master on the TI. Even then, Lory found some channels not responding. She wound up sending a "reset all controllers" code to each and every channel. That worked perfectly, and after that we could again use it with Cakewalk on the PC, provided we didn't play any of those downloaded MIDI files.

2.5.2. And Then Came E-mail

Shortly after getting equipped to "browse" via a service provided by the Maryland Public Library system, we found our way to HOTMAIL, established our E-Mail hookup, and opened yet another Pandora's box. E-mail now arrives from various people on a regular basis. Seems every day there are new messages in our in-box that must be dealt with. Most, fortunately, are of a nature that they can be read once or twice and then deleted. But what of the others? Stephen Shaw sent what he'd previously done by regular mail. Just three pages or so, but not something I'd throw away. HOTMAIL's help screens are not the most helpful. If I want to download Stephen's letter to my own hard drive, I find the menus on HOTMAIL give no indication how to do that. The only time "download" is mentioned anywhere on any of their displays is to offer me a browser! Of course since HOTMAIL is a Microsoft service, guess whose browser that would be, and guess what operating system it would need. Bill Gates rules the world. Actually there is one thing I can download from Hotmail. Michael Becker sent an e-mail with another file attached. In that case, Hotmail offers the option of downloading the attached file. Took me three tries to get it right, but I did manage to download that file.

In many such cases, I consult with Lew King. By ordinary voice phone, that is. Lew can usually help out very quickly that way, and we can reach a mutual understanding of what we're talking about much more easily when we're talking, as opposed to writing E-Mail back and forth. Still, the phone bill isn't affected by E-Mail, but those half hour conversations can add up. By the way, if you can help, that E-mail address is: rottencat13@hotmail.com. Seems rottencat was already taken, so we added 13 (a lucky number?) to have our own unique address. This probably proves that we're not the only people with E-mail who have a rotten cat. My partner, Dolores P. Werths, has her very own E-Mail address: Lorysmandolin@hotmail.com.

2.5.3. The List Server Saga

Thank you Tom Wills for providing the TI99 list server. This is a valuable service to the community. We, however, had some trouble getting started with it. In fact, as we write this, we're still not subscribed. On our first attempt we put a hyphen between TI and 99. WRONG! It has to be just TI99. We've tried twice more, but all we've gotten is return e-mail from majordomo@TheRiver.com. Each time this let us know that we were not subscribed. Since we didn't get anything from TI99 we could sort of figure that out.

On our fourth try we got everything right, and got subscribed on the List Server. Like the Internet itself, this is a mixed blessing. Yes, some valuable stuff shows up there, but one has to read a lot of less valuable stuff to sort it out.

But then that's life. One of our U.S. Senators here in Maryland has lots of stuff on her web site, including her favorite recipe for crab cakes. (Maryland is for Crabs - why else would I live here?) There's one other complaint while I'm being a Maryland crab. Some people don't answer their E-Mail. Of course some don't answer regular paper mail either. Mainly the non-answers have been people outside the TI realm. The TI people have been very good about responding, sometimes with more text than necessary, but at least they respond. Of course it's easier if I get their addresses right on the outgoing.

It's very easy working from a printed source to mistake a 1 (number) for an l (lowercase L). Such mistakes cause much grief, and of course the two characters are just as hard to tell apart on the computer screen as in print. (Maybe even more so.)

2.5.4. The Other Penalty

As this is written, it's been about a week since we started getting into the Internet, and less than that since we've had e-mail. Your author has not written a single line of source code in that period. There's a project "in work" for yet another new program for our beloved TI, but now that we're busy "browsing" web sites and sending/receiving e-mail, not one lousy line of source has made it into the project. Maybe on Saturday night. Today happens to be a Friday, and we don't want to miss this week's re-run of Sabrina, The Teenage Witch. We don't watch it for Sabrina, but for Salem the cat, who has all the best lines. We've never had a cat that could talk, but have had cats that were truly as "wiseacre" as Salem.

Still, on the internet we can visit the Fermi Laboratory and learn about fundamental particles of matter, particle accelerators, methods for detecting sub-atomic particles, the relationships between astronomy and physics, etc. etc. etc. Our boy Jean-Guy can visit a website that's all about Spawn, and spend hours reading up on his favorite action hero.

No, there's no sidebar this time. Authors who haven't done any source code don't have Sidebars. Shame on me! Next time I PROMISE. . .

2.6. The Art of Assembly. Part 82. Click on Icon Again

by Bruce Harrison

It's been an exciting few weeks since we wrote Part 81 of the series. During that time, we bought a brand-new PC, of the "modern" variety, with Cyrix 233 processor, 32 MEGABYTES! 4.3 GIGABYTES Hard Drive, 32X CD-ROM drive, etc. Came with Windows 98 and a whole bunch of other stuff pre-loaded on the Hard Drive.

With the new machine, we've gotten connected to America Online, got a new e-mail address, new and better capabilities for sending and receiving e-mail, and a whole new outlook on the world of the PC in its present evolution. Of course being the pessimist that I am, I'm quite certain this model will be rendered obsolete soon after the credit card charges are paid off, but for the present, it's wonderful!

Before we bought this machine, I had tried computers with the GUI concept twice before. Once on a MacIntosh at the office, and once on a Windows PC at the library. They mystified me. I was sure it would take me a long time to learn to use Windows 98. WRONG! It took me very little time to get used to the "click on icon" idea, and not much longer to become at ease with the new computer. Yes, there are things I don't like. It takes quite a while to boot up, but of course the TI is kind of a "spoiler" in that respect. The second thing I don't like is having to go through a "shutdown" procedure before turning it off. With the TI I can just power it down any time. With the Tandy 1000, I did have to park the hard drive, but I used a batch file for that, so it was just typing the four letters PARK <ENTER> to get ready for the OFF position. The new one CAN just be powered down, but it remembers that you did that awful thing, and on the next powerup it chastises you for that evil act and checks the hard drive for "damage". That makes boot-up take longer, so the machine gets even with me for the time I saved by just turning it off.

Don't get me wrong, though, I'm starting to really love the new machine. Among other things, we could visit the MUG98 web site and see all those very nice pictures. We could download a selected subset of them and enjoy them any time. With a little help from AOL, we were able to create two web site of our very own:

```
http://members.aol.com/rottencat1/homepage.html
http://members.aol.com/flatpickin/loryspage.html
```

The first of these is called The Assembly Guru, named for your author. Among other things it has links to pages full of various kinds of software product descriptions, a "bio" of your author, and a page of recipes for various very unhealthy dishes. Thanks to our friend Dr. Charles Good, there's a picture of your author available at that website, as a "click" off of the Brief Bio page. The picture was taken by Charlie with his digital camera at the MUG 98 in Lima, put on his MUG98 page, downloaded to the hard drive on our new PC, then uploaded to our space on AOL to be available with our web site. The subject of the picture is just as awful looking after all those down and up loadings, but the picture itself is terrific.

The second of those web sites is by my partner, Dolores P. Werths. This deals mostly with her exploits as a musician in various venues, including Renaissance Faires (& Festivals) and numerous other gigs. The irony of all of the above is that I'm writing this column on the trusty old TI, using Funnelweb, while our boy Marcel uses the new PC to play "Atomic Bomberman" in the other room. I too, though, have been known to play more than a few games of solitaire on the new machine, but I try not to do that when Marcel's home, because he always kibitzes over my shoulder, seeing moves I've missed. I still lose the game when he does that, it just takes longer to get really stuck.

So what does all this have to do with Assembly Language on the TI? Not much except that if my "fans" have noticed a slight decline in my production of new programs (all the way to zero) they can blame the new PC and the really excellent game of solitaire that was "bundled" with it. By The Way, nobody bothers to "bundle" any programming languages with the PCs now. Back when we bought our "stone age" Tandy, GW Basic came with it as a matter of course, and Assemblers were easy to find. Not now. I suppose if one went to the right store, one could buy some kind of C++++++ compiler for about $150.00, but there is no more free lunch for programming. (Yes, I know that Peter Drucker said there's no such thing as a free lunch, and once again he's been proven correct.) (Peter Drucker is ALWAYS correct!)

It's been a couple of weeks since the part above was written, during which again no TI source code was written except to make a minor correction in a program we produced some time back. We know we promised some kind of sidebar with source code this time, but have broken that promise. The power of the new PC is just too much of a temptation. Finally we can look at all those "Frames" websites and see what's there. Sometimes they're a bit of a let-down, particularly those that produce what I call "lists of lists". In those, each choice made from a list produces yet another list, and maybe five or six such levels go by without producing any "meat" on the screen. On my own web site, I've tried very hard to avoid that pitfall. There's some "meat" on every page, but there are also "Details" links so that those interested in a particular subject can dig deeper, but those just passing through won't have to suffer all the details. My partner's page is structured that way, too.

The other thing we've done is to keep checking out the web site on that old Tandy 1000 SX, so that we'll know that it looks okay for those without the modern PC, and even those using TERM80 on a TI will be able to extract all the information from the site. (All but the picture of yours truly, but that's no great loss anyway. On our Tandy, we get offered the choice of downloading that picture, but since we already have it on the new computer there's really no need. Maybe Lew King could download it onto his SCSI, but then it's a .jpg file, so what exactly he'd be able to do with it from there is questionable. The JPG format is now the standard for photographs on the PC. Our daughter Karen works at a large photofinishing plant, and she can have our regular photos put on 3 1/2 inch (HD) disks in JPG format. They render pretty nicely on our new PC, and can of course be uploaded to the web sites for others to enjoy, but there's not much hope of having them rendered at all on the TI. Maybe some ambitious Geneve programmer will come up with a "JPG Show" program for that machine. There, now a challenge has been issued!

2.7. Art Of Assembly. Part 83. The End of an Era

By Bruce Harrison

This is IT! The END! No More! If you're hurting for something to read, there are 82 previous parts to keep you entertained. By the time this is published, it will be old news, because the folks who attended MUG 99 spread the word around that Harrison has "thrown in the towel" for good. In today's column, we'll try to explain why.

2.7.1. Third Party Hardware

Those three words summarize the reason this particular Assembly programmer won't be writing any more programs. As more and more new things get developed to "enhance" the TI-99/4A computer, it gets more and more difficult to test any software and have reasonable confidence that it will work on the systems owned by the clients. At this point I consider it quite impossible. That's because there are simply too many kinds of hardware setups other than the one you're writing and testing on. No matter what you're trying to do, chances are that somebody out there has a "combination of ingredients" in his system that will defeat your program.

2.7.2. My First Experience

Many years ago, while I was still perfecting the "Assembly Music" programs that made Harrison Software a "household name" in the TI community, I made the trip to a TI Faire in Ottawa. At the time, I had absolutely no third party hardware in my own system. To save on lugging the system (and explaining it to customs officials) I decided to borrow a system from the Ottawa UG to do my table demos.

Looked fine until it was turned on, and then there was no familiar color bar display, but some kind of menu instead. This was my first time seeing a system with the Horizon Ramdisk in use. Since my "Assembly Music" programs ran in Extended Basic, I chose that, and discovered one way that such things can wreck one's software. In the startup of my XB code I wanted to have a copyright notice come up only on the first showing of the program's main menu. To do that I checked to see if there were no definition for character 143, which is normally undefined when one enters XB. Unfortunately, the Ramdisk's own menu defines that character for its own purposes, so my program skipped its copyright notice, and also skipped the steps that loaded the "defs" file from my disk. As a result, my programs would simply lock up the computer.

The system's owner looked at what was happening, turned off the power and then removed his Ramdisk card. After that, everything worked fine with my disks on his system, users were happy, etc. But this was a harbinger of doom! I didn't see it at the time, but the end was already in sight. I was able later to re-work the main menu part of my Assembly music products so they could c Ramdisk equipped systems.

2.7.3. The Big "G"

At about that same time, Lou Phillips was starting to make and sell the Geneve 9640 in quantity. Enthusiasts sprang up and put cash in Lou's pockets for the new "improved" capability. In theory, the Geneve (in its GPL Mode) would run any existing TI-99/4A software, so the person buying it would not lose the ability to use software he already owned, and would be able to use new software still being written for the original machine. That theory never quite worked out.

My own first experience with the Geneve came when I sold an Assembly Music disk to a guy named Don West. Don had both a TI and a Geneve at his house, so he was able to do some comparing. As it turned out, my disk actually worked on both machines! The only problem was that the speed of playing was very different. On the Geneve, the music played about twice as fast as on the TI. Not good.

Eventually, we were able to develop a "self calibration" method for the music's pacing, so that a few of our Assembly Music disks could actually be used on either the TI or the Geneve, and would play at the same pace on either one. That did not last long, however, as another improvement we introduced for the TI, which involved using a DSRLNK in our assembly code, made the newer products completely incompatible with Geneve, and so they remained.

The claimed compatibility of the GPL Mode with TI software was always more than a little "iffy", and remains so to this day. One never knows what little trick that works perfectly on the TI will cause total lockup on the Geneve. For those who don't own the Geneve, and therefore can't test anything on it, this is particularly annoying. There's always a "Geneve Surprise" waiting to happen every time you send out new TI Software. (See "The Final Straw", below)

2.7.4. Other Fun Hardware

Of course not only Lou Phillips' Myarc made third party hardware. There are also Horizon (Ramdisks and P-Gram) and Cor-Comp (various things) to consider. One of my all-time favorites involved the Cor-Comp Mini System, in which the 32K etc. plugged into the side of the console instead of having to use a P- Box. Sent off some package or other to a gentleman in Maine. Soon heard those dreaded words "didn't work". There was, as I recall, nothing exotic being done by the software, no tricks outside of the usual Assembly stuff, but the program got just so far on his Cor-Comp mini system and then froze up so that only the on-off switch had any effect. Never did find out why that happened, but had to refund his money.

The Horizon Ramdisk usually causes no trouble, except in cases like trying to read or write sectors, and then only if the CRU for the Ramdisk is set above >1100. It can, however, cause other problems, as a recent (Feb 1999) spate of messages on the TI List Server showed. Seems the original suspected culprit for a problem with TI-Artist was the AMS Card. Turned out instead to be the Horizon Ramdisk. At the 1998 Chicago Faire, we were given a disk from Europe that included a new Disk Manager program called DM2K. This was intended to work with floppies, hard drives, and SCSI drives. Back at home, I tried it with my system's floppy drives and Ramdisk. It would catalog any of them just fine, but would not copy files to or from the Ramdisk. As it happens, my Ramdisk card is at CRU >1200. Another program for the "doesn't work with" file.

We have not even mentioned such things of beauty as the Myarc HFDC, which can cause a simple boot-tracking routine to lock the system up, the SCSI card, on which boot tracking does not work, and the infamous Rocky Mountain Memory card (32K) which could not properly run our Assembly Music products. And now there are still more surprises just waiting for us, in that whole slew of cards that Michael Becker and his friends are building. So far, my only experience with those had to do with the AMS Emulation that's included in the three-card package. I had to modify my AMS packages so they would work on both the SW99ers SAMS card and the German emulation. A simple enough fix was devised with help from Michael, and tested okay on both AMS systems, but as a result all of those packages are now TOTALLY incompatible with Geneve. Is that a surprise?

2.7.5. The Helpless Community

There was a time, not too long ago, when most of the people using the TI-99/4A computer were conversant at the very least with TI Extended Basic. If they encountered a problem with a program in that language, they would at the very least have a look at the listed program to see whether it could easily be fixed. Not any more! Recently I was sent an entire package that was written in Extended Basic to help with a problem that took ten minutes or so to find in the XB code, and just a few seconds to correct. Testing the fix took a lot longer, because it involved printing several pages of graphics through XB (painfully slow).

The message is, TI users don't try to help themselves any more. They're becoming just as helpless as the PC owning "Windows Wizards" who never understand anything beyond the point and click operation. Of course in the case of the PC, keeping users from knowing what's going on is the whole purpose of Windows, but it used to be different for TI users. Just a few years back, any TI owner faced with a problem in an XB program would, at the very least, LIST the program and examine it for anything obvious that might explain the problem. In the case mentioned above, the problem had to do with what happened in the year 2000, and right there in the XB code was an IF-THEN with the expression IF (Y= 2000). Really obvious if one just takes a look! Is it too much to expect that a user would be able to figure out that Y might mean Year, and that an exception being made when Y=2000 might have something to do with the observed problem?

2.7.6. The Final Straw

Over all the years that I've been writing Assembly programs, never had I written one that used the TI Speech Synthesizer Module. I'd always considered that just a bit too "Toys R Us" for my liking. I changed my mind about that just enough to make a program for playing Bingo on the TI that would use the Speech module (if available) to say the letter-number combinations as they came up. Worked just beautifully on the TI-99/4A here at home, and just beautifully on the system provided for my lecture at Chicago (also a TI-99/4A). I was happy with the results until I got my copy of the Jan-Feb 1999 MICROpendium. There in Charlie Good's MICROreviews column were the "final straw" words: "Unfortunately the speech of TI Bingo doesn't work on my Geneve. I have a Rave speech card in my Geneve. . ."

There it was, a double third-party whammy! Not only is the computer in question a Geneve, but the speech device is also a third-party item. Finished! No More! My Bingo has been discontinued, removed from the web site, no longer offered for sale, etc. No more software for the TI-99/4A, period. In Navy terminology, we have gone to the engine order telegraph and rung up Finished With Engines.