SAN DIEGO COMPUTER SOCIETY

TI-SIG NEWSLETTER

APRIL 1989

NOTES FROM WOODY'S DESK--

DO NOT FORGET THAT THE ANNUAL MEETING FOR THE ELECTION OF OFFICERS WILL BE HELD IN MAY!! THE COMING MEETING WILL BE ONE OF THE MOST IMPORTANT ONES HELD SINCE THE SIG WAS FORMED. PLEASE ATTEND AND LET YOUR VOICE BE HEARD AS TO THE FUTURE OF THE SIG. NO OFFICERS---NO TI-SIG!

AS MANY OF YOU ARE AWARE, John Johnson will not hold any office in the coming year. He has expressed a desire to become active in the DIGSIG. We all know his reasons and wish him well. We will miss him.

IN THIS MONTH'S ISSUE OF the newsletter, I have used two of Tony McGovern's fine articles. I felt that I should get them out to you while we still have a newsletter (just in case we no longer put one out). Starting on page 2 is an article, "CAUTION: DEATH BY CLEANING", from the Journal of the San Diego Computer Society. I believe that you will find it well worth reading. Then, as filler material, you will find two short programs by me. They are not world shaking, but the one on displaying large numbers was made to help a non-member who needed something like it for his work. The second program is just a filler.....if you have a CorComp disk controller it may suggest some tricks you can do to fool your friends. The display on the screen when you call it back from Expansion memory is extremely fast.

NEXT MONTH I HAVE a nice program called "FIELDSORT" that can sort Display Variable 80 files provided the files follow a specified format. Roland Anderson and I slightly modified a program by Art Byers from the February 1989 newsletter of the Northern NJ 99er's Users Group. I would have used it this time, but it took too many pages.

SOME OF YOU MAY NOT know that we use the copying machine at the Recreation Center for making our newsletter. We can make about 62 newsletters without running over our free allotment. I sure try to keep within the free range...call me cheap! So next month gets the rest of the material I have prepared. Anyway it will put me a little bit ahead for a change.

==========

REPORT FOR THE SDCS: MAR. 21ST MEETING

President J.D.^Johnson called meeting to order at 7:00 pm. There were 12 persons present.

Woody Wilson was asked to give a report on the TI-FEST WEST. It appeared that due to lack of publicity in the newspapers, many of the local area TI owners were unaware of the FEST. As a result, attendance was not as great as expected and the vendors could not be reimbursed fully for their rooms.
However, a partial repayment was made of around $50.00. The March issue of the newsletter carried an article by Woody on the FEST.

J.D.^Johnson asked for volunteers for 1989-1990 officers. He pointed out that if there were none, he would have to notify the SDCS that the SIG would be closing out.

Woody reported on Jim Peterson"s three diskS of programs called "NUTS AND BOLTS". He pointed out that these were in mergeable format and cover just about "everything under the sun" and only cost $15 per disk with over 100 subprograms on each disk.

Woody then presented the program for the evening--TIBASE, Version 2.01. A review of this program is too long to include in this report since the presentation took over an hour just to show part of the capabilities of TIBASE.

A lady visitor asked for help in resolving a problem she was having with an Axiom Interface that was connected to a Sanyo Daisy wheel printer. The printer had stopped working after it was moved. Suggestions were made by the members and help was offered to the lady if she was unable to resolve the problem using the ideas presented.

Some of Jim Peterson"s games; TK-WRITER; and Ray Kazmer"s 1989 ST. Valentine's Day Card were also demoed.

The meeting closed at 9:30 pm.

Next meeting: 7:00 pm, May 16th, 1989

THIS MONTH WE CONTINUE with Part 5 of Tony McGovern's tutorials on squeezing assembly language programs. This part was just received on March 29, 1989.

ASSEMBLY SQUEEZING Part 5

Tony McGovern -- Funnelweb Farm

Very often individual bits in a byte or word are used to flag some condition. The 9900 status register does this all the time and the various conditional jump instructions operate on various combinations of these. Now if you are writing your own code you usually end up using a whole word to store each Boolean flag that in principle needs only one bit, just because it costs more to pack the flags one bit at a time than it saves. There are some intermediate cases however. Mostly you will find these when dealing with system routines and the GPL status byte at >837C. Remember that in GPL the poor old 9900 is weighed down by the task of having to emulate in software an imaginary 8 bit processor. Another situation is exemplified by the directory (SD and QD) routines in Funnelweb. Here the full directory information for up to 127 files must be held in memory (VDP in this case) and yet compete as little as possible for machine resources with other programs. To do this each directory entry is encoded into 14 bytes, which means a certain amount of bitpicking to unravel each entry for display. With 127 entries allowed for, there is scope for the trade-off with the extra bit-picking code. No doubt you will have your own favorite problem.

Suppose you want to check the equals bit in the GPL status register, say after a keyscan or DSRlink call. Your options are a little bit confined here as most of the bitpicking operations assume one or both operands are in a register. You have here to concentrate attention on a particular bit. One way is to set all other bits to zero and then examine the byte (or word or register) to see if it is null. The most general instruction for this is SZCB (or less commonly SZC). You also have to have a mask byte as data somewhere

```
SZCB @BDF,@GPLST
JNE  NEWKEY
```

where BDF is the label of a byte with only bit 2 zero, >DF being the appropriate value. This takes 4 words plus a data byte somewhere. An interesting variation on this that can save space if you you have your own BLWP @KSCAN called from several places is to set this flag before leaving the routine.

```
KSCAN  DATA KWKSP,KSCN1

KSCN1  LIMI 2
       LIMI 0
       LWPI GPLWS
       MOV  R11,@KWKSP+22
       BL   @>E
       LWPI KWKSP
       MOV  R11,@GPLWS+22
       SZCB @BDF,@GPLST
       STST R15
       RTWP
```

The LIMI instructions are to enable interrupts every keyscan and may be omitted if this is not desired. The next few instructions call the console keyscan routine. Saving and restoring R11 of the GPLWS is usually advisable, but not always necessary. The SZCB sets the CPU status bit as before and STST reads the status into R15 just so RTWP can put it back into the CPU status again. Now on every call to KSCAN it need only be followed by a JEQ or JNE to test the GPL status for a repeated key as in

```
BLWP @KSCAN
JNE NEWKEY
```

If this level of byte saving is not called for, a small saving can be made by doing something like

```
MOVB @GPLST,R0
SLA  R0,3
JOC  NEWKEY
```

As each bit is shifted out it is placed in the Carry status bit, and can be tested with a JOC instruction. This saves a word over checking the bit with

```
MOVB @GPLST,R0
ANDI R0,>2000
JNE  NEWKEY
```

but this last code has the advantage that it does an implicit CLR of the register if the jump is not taken. It all very much depends on the details just what is best to do.

THE FOLLOWING ARTICLE appeared in the February 1989 issue of PERSONAL SYSTEMS, from the San Diego Computer Society.

CAUTION: DEATH BY CLEANING

Contributed by Murray Underwood, a mem-

(Continued on page 3)

(Continued from page 2)
ber of the Saint Louis User Group and a professor of chemical engineering at Washington University.

A recent safety bulletin from the Naval Weapons Center, China Lake, California, warns of the hazard of using flammable glass cleaners on computer display screens. An employee sprayed his screen with cleaner, presumably a commercial cleaner that contains propyl alcohol. When he went to wipe it off, a static spark from his finger set the computer on fire.

The boss apparently doubted his story and checked the procedure on other personal computers. Two out of eight caught fire. The Naval Weapons Center now recommends using the spray BEFORE the machine is turned on (to avoid static buildup) and grounding oneself before touching the wet screen.

A better recommendation, in my opinion, would be to clean the screen with a cloth or tissue dampened with water and a drop of dish washing detergent. Not only does it avoid the flammability problem, it does a better job.

The detergent leaves an invisible film that is electrically conductive. This helps drain off static charge and minimizes the attraction for dust.

==========

HERE IS ANOTHER EXCELLENT article from Tony. This time he discusses:

DSR-LiNKs and the RS232

Tony McGovern - Feb 89

Perhaps the feature of the TI 99/4a that has most contributed to long life after being sent to the orphanage is its very general scheme for adding peripheral devices to the system, in a way that is independent of the internal details of any particular device. All devices are addressed in a uniform fashion as part of the file management system, and each is responsible for its own actions with its own Device Service Routine (DSR), usually in ROM, paged in for the particular card. Peripherals are called by name and the only physical preconfiguring necessary is to set the CRU base address on the peripheral card to avoid conflicts in paging the DSR where TI hadn't set a specific address. All addressing is by device name and physical location in the PE Box or even CRU assignment is not relevant to programs. This system compares more than favorably in sophistication with that on most home/personal computers even to the present day

Most programs engage a particular peripheral device by calling a standard type of interfacing program called a Link to Device Service Routine (DSRLNK) which accepts the calling data in a universal specified form and handles the details of accessing the device. The data is set up in a Peripheral Access Block (PAB) which in the 99/4a is in VDP RAM with a pointer at a standard location in CPU PAD RAM. This is similar to File Control Blocks in other systems, but TI's name indicates the thoroughness of its implementation in the 99/4a. No fussing about loading "device drivers" - in the 99/4a system if a device is present it brings along its own driver. MDOS on the Geneve seems a real step back into living pre-history in this regard. Some functions are addressed directly at the hardware level in the console, video display, sound and speech, the first two for reasons of speed and the last one for sake of cheapness. There is a price paid for all this easy access by name to peripherals, and that is speed of access because the devices are not preconfigured to standard channels, and are searched for on each and every access via a DSRLNK. This can be speeded up on subsequent accesses to the same device, and going even further it is possible to address the peripheral device hardware directly if adequate performance is not otherwise obtainable for special tasks. This is the equivalent of providing your own device driver. Given the RAM limitations in the 99/4a it is something a programmer does only if there is a real need for it. For a while after the 99/4a was orphaned this was no real problem as there was very little alternative hardware, but it is becoming more of a problem for software writers to keep track of the variety of PE-Box cards for each function.

The point of this article is not to go into the details of of PABs and the file management system which have all been treated extensively elsewhere, not least in the E/A manual. What I do want to talk about is the DSRLNK itself, which is a subject TI never did go into in detail. As far as I know the only release acknowledged by TI was in the commented code in the FORTH source code release. Even there the commenting is not entirely accurate. DSRLNKs come in a variety of forms in utility programs, essential variations usually being in the degree of error handling in the DSRLNK itself, and in how much internal information is saved outside the routine. The console ROM contains a
(Continued on page 4)

(Continued from page 3)
DSRLNK (and does a GSRLNK for searching GROMs too) but this was not made readily accessible though it has been done. TI's own well known utility programs TI-Writer and E/A that run in memory expansion all have their own DSRLNKs, as do the E/A utilities and Mini-mem, and none use the console code.

In fact I don't even intend to go into detailed code very much if at all, but let's see what is going on. First any DSRLNK clears the Status <=>'s bit which is the prime error flag on return. Since DSRLNKs are usually BLWP routines this is done by fiddling the appropriate bit in R15 which then is loaded into the status register on leaving via RTWP. The PAB in VDP RAM is then located by the pointer stored in PAD at >8356 (PAD is at >8300 in all 99/4a's sold though TI's DSR specifications allow for it to be anywhere, and locate it from the GPL workspace at PAD+>E0). This points to the length byte of the device name in the PAB and is all stuff the calling program, be it Basic or an application program, has already set up. The name is fetched to the PAD, typically at FAC (PAD+>4A) and its length checked on the way. The device name before the first decimal point can be 7 characters long at most or the DSRLNK will return an error, though some TI-written DSRLNKs allow 8. Extra name length after the device name is handled by the DSR itself, as filename or software switch or whatever.

The next step is to search for this device name in all the DSRs. This is usually done with the GPL workspace set for the speed of being on the 16-bit bus. The DSR ROMs are turned on one at a time starting at >1000 in a standard DSRLNK and each searched for the name in the linked list of names. See the Technical Manual or other sources if you want to follow up the details. The initial pointer to this list is at >4008, an initial offset from >4000 set by the DATA 8 specified in the E/A manual to follow BLWP @DSRLNK calls. The other useful data item to follow the call is DATA 10 which starts the search in the list of subprograms, for instance the sector read/write routines in a disk DSR. Not all DSR calls in programs follow this format precisely. The TI-Writer Editor's DSRLNK assumes file type access only and the calls have no following DATA item. Funnelweb has followed this pattern for better or for worse - it saves some words in the main program not to have a predictable data item repeated several times. These DSRLNKs retain reasonably full generality and do a search over the whole DSR range. Simplified versions have been published which assume a particular CRU base and omit the search over DSRs. These forms are not suitable for general use. Extensions to handle multiple RAMdisks were covered in a recent article in the HV99 News and elsewhere by now.

If the device name is not found the next DSR is searched until the end is reached or the device found. When the device is found the GPL workspace is set if it is not already in use and the code entry address placed in R9 of the GPL workspace and the DSR entered by a BL *R9. TI specification for exit from a DSR is that it step over the next word in the calling code with a INCT R11 or equivalent before returning, though no explanation is offered as to why this should be done differently from the power-up or interrupt routines. This word in the DSRLNK is usually a JMP to resume the search in the next ROM. More of this in a little while.

Standard E/A pattern DSRLNKs save some of the information gained along the way. In the E/A Utilities this is done in UTLTAB as several entries with standard names of which the following are the most important.

SAVCRU - the CRU base address of the DSR ROM where the device name was found.

SAVENT - actual entry address of the DSR

There is also SAVVER implied for version number of the DSR. If you know that you are entering the same DSR repeatedly then it is possible to speed things up by entering directly and bypassing the DSR search. The E/A 3 Load & Run does this to speed reading of DF/80 Object files. This is a trick that doesn't seem to be used all that often in other software but one example is the LINEHUNTER program in the Funnelweb package which gets some of its speed by using this technique for both master and copy files.

SAVVER doesn't seem to be used by TI in the way implied by the E/A or other TI documents. This came up a few days ago in the form of a query from Geoff Trott down in Wollongong as to why there was sometimes trouble in accessing a second RS232 card in a system. This was something I had never gone into before as we have had no particular interest in communications and so in the details of RS232. So I dug out my copy of Colin Hinson's excellent detailed exegesis of the RS232 DSR ROM and there it all was staring me in the face. In
(Continued on page 5)

(Continued from page 4)
retrospect the question that needed to be answered was how two RS232 cards with exactly the same DSR ROM are usable at different CRU addresses, which is all that I understand the modifications to the second card to involve. Why then don't the device names such as RS232/3 or PIO/2 which are supposed to address the second card cause the first card to respond ? The horrible unpleasant fact is that the RS232 card uses an undocumented addition to the DSRLNK. R1 of the GPL workspace is used as a counter of the number of times a DSR of the correct name has been entered. The RS232 second card device name entries check to see if R1 contains 2 and if not exit immediately without an INCT R11 so that the search is continued. This explains the CLR R1 before the ROM search and the INC R1 before DSR entry that you will find in standard DSRLNKs. I had noticed these as seemingly useless instructions early in the development of Funnelweb. When they were removed the disk drives still worked fine but it was found a little while later that the RS232 didn't work anymore. Since this isn't a well behaved device anyway, I just put it down to an idiosyncrasy of the card and restored the code in the DSRLNK. Now I know why it is there.

This may be a new discovery down on Funnelweb Farm but it must have been known to to a number of people out there, not least those who rewrite DSRs. It may have long since been discussed on BBSs or even in UG Newsletters, but as this is the first we have seen of it here then it is probably new to most of our HV99 Newsletter readers. Now that I've written that someone will come along and point out where it has been staring me in the face all along ! I presume that someone in authority at TI insisted, when the possibility of having 2 RS232 cards in the PE-box was raised, that it be done without changing the ROM in the second card. Minor ROM changes between the two are clearly the best way to do it, but more expensive for the manufacturer. So this extra was added to DSRLNKs and has been copied ever since. In the TI-Forth source code it is commented as related to Version found, and saved at SAVVER. Nowhere is any confusion between this and the DSR ROM version stored as the second byte of the header resolved, and no TI documentation of any kind that I have seen discusses the matter and how it is used in the RS232. Perhaps they were all too embarrassed by the violation of the essential elegance of the 99/4a system design to commit it to paper.

==========

NOTES FROM WOODY'S DESK

Have you ever wondered if it would be possible to display 14 digits in a math problem WITHOUT having the computer put the answer in scientific notation? The Extended BASIC book on page 39 says we can display the entire number with a USING clause in a PRINT or DISPLAY statement. Here is a simple little program in Ext. BASIC to illustrate how we can do it. Please note that if you use signed numbers (+ or -) a maximum of 13 digits can be displayed. While the program is written for multiplication, it can easily be adapted to cover any type of math problem.

```
100 ! DISPLAYING LARGE NUMBE
RS BY WOODY WILSON, SAN DIEG
O TI-SIG. APRIL 1989
110 CALL CLEAR
120 DISPLAY AT(3,1):"IF THE
PRODUCT OF THE TWO   NUMBERS
 EXCEEDS 14 DIGITS,  THE RES
ULT WILL BE SHOWN AS A SERIE
S OF ASTERISKS."
130 INPUT "FIRST NUMBER? ":A
140 INPUT "SECOND NUMBER? ":
B
150 X=A*B
160 IMAGE THE "IMAGE" VALUE
IS EQUAL  TO  ##############
170 PRINT
180 PRINT USING 160:X
190 PRINT
200 PRINT "THE ""PRINT"" VAL
UE IS EQUAL  TO ";X
210 PRINT
220 PRINT "PRESS ""Q"" TO QU
IT, ANY OTHER KEY TO TRY ANO
THER"
230 CALL KEY(3,K,S):: IF S<1
 THEN 230
240 IF K=81 THEN END ELSE 11
0
```

==========

DO YOU HAVE A CorCOMP disk controller? If you have one, have you ever tried any of the TOOL SHED subprograms? Here is a short program that can illustrate some of the power of the utility:

```
90 ! CorComp Disk Controller Tool Shed
Subprogram utilities. Sample program by
Woody Wilson, San Diego TI-SIG, Apr. 89.
100 CALL CLEAR
110 CALL INIT :: DELETE "LD-
CMDS"
120 CALL HCHAR(1,1,42,768)
130 CALL LINK("MOVEM")(2,0,4
0960,768)
140 CALL CLEAR
150 DISPLAY AT(12,3):"YES, Y
OU SAW STARS, BUT NOW YOU DO
 NOT"
160 DISPLAY AT(17,3):"I WILL
 BRING THEM BACK"
170 CALL KEY(0,K,S):: IF S<1
 THEN 170
180 CALL CLEAR
190 CALL LINK("MOVEM")(3,409
60,0,768)
200 GOTO 200
```

SAN DIEGO COMPUTER SOCIETY

TI-SIG

NEWSLETTER APRIL 1989

Meetings: 3rd Tuesday of each month at 7 P.M.
Crafts Room, North Park Rec. Center, 4044 Idaho Street, San Diego

1988/89 OFFICERS

President: John Johnson 279-9740	Vice Pres: Woody Wilson 264-6515
Treasurer: Woody Wilson 264-6515	Secretary: Lutz Winkler 277-4437

TI-SIG
San Diego Computer Society
P.O. Box 5263
San Diego, CA 92105

DALLAS TI-HOME COMPUTER GROUP
P.O. BOX 29863
DALLAS TX 75229