Mass Users of the Ninety-nine and Computer Hobbyists

JANUARY 1995 Monthly Newsletter Version 14.01

WE'RE STILL ALIVE IN 95

HAPPY NEW YEAR !!!!!

M.U.N.C.H.
C/O J. W. COX
905 EGDEBROOK DRIVE
BOYLSTON, MASS. 01505

NEXT MEETING: TUESDAY, JANUARY 10th.

POSTMASTER: Forwarding and Address Correction Requested.

FIRST CLASS!!

R.A. BISHOP
16 FRENCH AVE
NORTHCOTE 3070.
VICTORIA AUSTRALIA

NEXT MEETING TUESDAY, Jan. 10, 1995 7:00 PM. Happy New Year!!!!!!!!

MUNCH OFFICERS AND NUMBERS (all in 508 area unless noted)

PRESIDENT	W. C. Wyman	865-1213		
VICE-PRESIDENT	Open		MUNCH DUES:	
TREAS./EDITOR/CLK.	Jim Cox	869-2704	New Membership	$25.00
DEMO LEADER	Jack Sughrue	476-7630	Renewal	$15.00
Asst. Demo Leader	Lou Holmes	617-965-3584	Newsletter Sub.	$13.00
LIBRARIAN	Walt Nowak	413-436-7675		
Advanced Programmer	Dan Rogers	248-5502		

****

DECEMBER MEETING. The December meeting had eight members in attendance. We demoed the DOM of Sam's Games. We also spent some time going over Walter's Ram disk, which was giving him some problems. Chris won the raffle for the second month in a row.

JANUARY MEETING. It's '95 and we're still alive will be our slogan for the year. I am sure Jack will have something of interest for us and we will demo the DOM of John Doone games. John was a member in the early 80's and did some very inventive programming for the time.

RAFFLE. Every month we have a raffle to help defer the rental cost of our meeting hall. A typical raffle will have programs, blank disks, books, bumper stickers and all sorts of odds and ends of interest to the T.I. user. This month we have some Tandy Model 4 computers.

REPRINTS. Reprints are permitted as long as credit is given to M.U.N.C.H.

ARTICLES. I am always looking for articles for this newsletter, anything which interest you will probably interest other members of the T.I. community, so please share your ideas and opinions with all of us.

DISK LIBRARY. The disk library is at all meetings. We have copies of all disks in the library and they are available to members for just $1.00 for each disk unless otherwise specified. You can order them through the mail, please add $1.00 for the first disk and $.40 for each additional disk ordered to cover postage and handling.

DISK OF THE MONTH. This month's DOM #140 is a game disk with programs from John Doone. Included are Amortization, Kismet, Cannibal, Checkers, Check Print, Metric, Mastermind, Pulse, Reading, Seastate, Secretary, Stats and Tablut.

ADVENTURE II. This is our fund-raiser for 1994/95. The cost to members is $4.00 add $2.00 for first class postage. The regular price is $6.95 plus postage. This is a two DSSD disk set, archived. There is also a special on The Adventure Compendium and Adventure II for members it is $8.00 plus $3.00 for first class postage.

FOR SALE. Former member Dennis Lavoie has two complete systems and lots of cartridges and software for sale. He wants to sell everything together. If you are interested contact Jim Cox at the number above and he will give you more information.

the LF2 set of composite numbers. You can eliminate them by ignoring them. You ignore them by not generating or using them. But you need them in experimental analysis of odd numbers. They are "there" when you need them.

Any prime number is a cofactor of every prime number less than itself in generating the limitless set of composite numbers of which that prime is the least factor. But no prime in the successive counting line can become least factor of its own limitless set of composite numbers except with its own square as a point of beginning. In proceeding systematically, beginning with 2 and omitting all limitless sets of composite numbers, no composite number will be left behind.

Thus, in omitting 2x2=4 and the LF2 set of even numbers, only the non-composite numbers, 1,2,3 remain. Omitting 3x3=9 and the LF3 set of composite numbers, 1,2,3,5,7 are left behind.

Proceeding to the LF5 set of composites, their omission adds 11,13,17,19,23 to the numbers left behind. This is one way of generating primes. They all fall through the "holes" left by systematically omitting successive limitless sets of composite numbers.

A key to another way is in the smallest composite number that is the product of two different primes: the smallest and only even prime, 2, and the smallest odd prime, 3. The product is 6. It has long been known that if a number P is prime then either P-1 or P+1 must be divisible, without remainder, by 6. This is essential but not sufficient to prove primality.

There are odd composite numbers that pass this and other tests. 97 is prime and 96 is divisible by 6 but 91 is not prime and 90 is also divisible by 6. But primes can be generated by finding a single factor, if any, for every Nx6 plus or minus 1 to get rid of composites.

Our NUMRES program is based on still another way that will be fully explained in another session. For now we suggest that you start playing with "6". The sparks of experiment may ignite the flame of exploration and discovery.

Try typing to screen: A=1/6 ENTER B=A/9 ENTER PRINT A:B:1/54 ENTER. You will find that the reciprocal of 6 must be divided by 9 to get a meaningful repetend on the repunit level. But division by 6 of any repunit number leaves a remainder. So you look to the reciprocal of the reciprocal which is 54 for some clue to follow.

At this point you may realize that, as a multiple of 6, 54 flags the probable primality of either 53 or 55. So you call for a rep number of 52 ones and divide it by 53 to take a look at its 50 digit repetend. You start the factoring and let it go through 11, 79, 101, 521, 821... waste no more time. File it with data for later reference.

Take another look at the repetend of 53. You will find that it contains four cycles of a smaller repetend: 20964360587, with the quickly found factor, 79, quotient: 265371653. Chop it off there. I let it go through the 4247 loops to prove the quotient prime.

By using the decimal factoring mode of the NUMRES program you can quickly obtain and compare the factors of 185, 185185, 185185185, and 185185185185. What are you trying to do? You are trying to get the "feel of the matrix", looking for clues or signs of where and what to explore.

Random shots: 185=5x37. 6x37=222 (2xR3). 6x185=11110 (R5-1). Eventually it will all begin to take shape. Our repunit rocket probe of ones, cuts right through the cycles of repetends that contain the code we are trying to crack.

```
11123456789009876543211 1
1                       1
1 PLAYING WITH NUMBERS  1
0        No 11          0
9  By Meredith Beyers   9
9                       9
99987654321001234567899 9
```

"THE STRONG LAW of SMALL NUMBERS"

The title is in quotes because I am quoting it. I first saw it twelve years ago in Martin Gardner's section of the SCIENTIFIC AMERICAN. Dr. Gardner reported that the title was that of an unpublished paper by Richard Kenneth Guy, a mathematician at the University of Calgary. A quote from the paper: "Two of the most important elements in mathematical research are asking the right questions and recognizing patterns."

The subtitle of Dr. Gardner's commentary was "Patterns in primes are a clue to the strong law of small numbers". But, as Dr. Gardner himself pointed out in his commentary, patterns in primes provide doubtful grounds for general conclusions.

I felt we should add an "s" to "law" to make it plural. The LAWS of small numbers should be applicable without limits to all the larger numbers they create. Over-emphasis on "Primes" has created an aura of "Chaos" that clouds our perspective in viewing a universe of generated composite numbers.

More than two thousand years ago the Greek astronomer, Eratosthenes, laid out a grid of the first 100 small numbers and pointed out the fact that if we crossed out all the composite numbers, the "holes" would be prime.

Early 20th century teachers (including my own) told us about this "Sieve" of Eratosthenes, but warned us that it would be of little use in discovering large prime numbers because it would be impossible for us to create arrays of the required magnitudes. None of us anticipated the creation of computers that would be able to extrapolate the "Sieve" of Eratosthenes without creating huge arrays of numerals or "crossing out" anything.

Now we should be able to start with "the strong LAWS of small numbers" and generate an array of ever larger and larger prime numbers by simply omitting the composite numbers that Eratosthenes would have "crossed out".

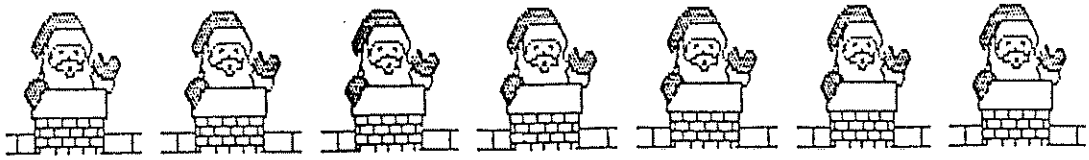
So we begin with the small decimal numerals we memorized in childhood. Our aim is to search for "laws" within the limits of the standardized decimal system of our computers and test them within the greater finite limits of our devised hectokilorad positional notation system. In this way we may justify limitless extrapolation.

We find the beginning of numerical law in the rules of single-digit decimal numerals and the power hierarchy of positional notation systems. Zero (0) and one (1) are the only symbols in common with all systems. They are both place keepers in multidigit numerals, the "0" cancelling the power value of the position, and "1" preserving it.

All by itself, "0" can serve only as a point of beginning on scales, rulers and measuring tapes. Without a numerical companion it is forbidden to play any part in arithmetical operations. "1" is the universal Counter, the unit of addition or subtraction. The limitless array of Counting Numbers is generated by uniform incrementation, beginning with 1+1=2, +1=3.....

Thus far we have "rules". Here we discover the beginning of "LAW". 2 is the first "even" numeral and the only even prime number. 3 is the first and smallest odd numeral, odd prime and cofactor of 2 in generating the limitless set of even numbers with 2 as least factor.

The set may be generated by uniform incrementation, step 2, but the first composite number, 2+2=4, the square of 2, is considered to be the beginning of



memory. If the file happens to be called LOAD, you're in luck, since XB looks for a file called LOAD on the disk and automatically loads it from the main menu. (Although you do have to press 2, to get XB.)

If the file is a DIS/VAR 163, it's most likely an XB merge file.

If the file is a DIS/FIX 80, it's most likely an A/L program.

If the file is a DIS/VAR 80 file, it probably is a text file for TI Writer or any of the numerous word processors programs available to TI users - again almost too numerous to list. Look at the file If the file is a DIS/FIX 128, it is possibly an archived file which will need aprogram that can un-arc the information into separate useable files.(This is yet another story.) We are also looking at the possibility that it might even be FORTH. (Ditto.)

You are quite likely somewhat confused by now and there is still more. Quickly looking at graphics programs, we can have some of the following goodies to deal with:

If the file is a 25 sector _C or _P, we are dealing with the colour and pattern portions of TI ARTIST.

If the file is any sector length DIS VAR 80 with _F as the ending, we probably have a character font.

If the file is an 18 sector DIS VAR 80 _S, it most likely is a TI ARTIST Slide.

If the file is a DIS VAR 80 _I, take it to be an Instance.

If the file is a DIS FIX 12 _V, take it to be a Vector. TI ARTIST PLUS

If the file is a DIS FIX 254 _M, take it to be a Movie. TI ARTIST PLUS

If you happen to be looking at TIPS (Texas Instruments Print Shop) then we have some more files:

IF53 is a Picture (TXT)
IF16 is name text (XXX)
DV250 is spooled graphic
IV254 is a TIPS Font.

I'm sure if I tried I could probably go on ad infinitum, as file types tend to go on ad infinitum. Just because you can't get a file to load the first time, don't give up on it. I would hesitate to guess how many pages one could fill of different file types available in the TI world. If all else fails, pick up the telephone and call a club member, or bring it to the club meeting. (I know it's a little tough on you out-of-towner people, BUT Vancouver and the surrounding area really is a beautiful place to come to (This sounds like the commercial), but you see that not all needs to be lost because the file won't load. THAT'S IT.

Finally, there is a seventh form for program files. This form is created and loded by the "EASY BUG" of the Mini Memory cartidge. It can only be written to cassette and is memory image, but it is slightly different from the E/A memory image files. The Easy Bug memory image program can only consist of one segment. The header for Easy Bug format consists of only two words, as follows:

1) This word is the CPU memory address at which the memory image is to be loaded.

2) This word is the length of the memory data, NOT including the four header bytes.

Okay, now that we have gone through all that, let's try to sort a bit of it out. Again, I am back to reading articles from the various newletters that we exchange with other TI computer clubs throughout Canada and the States. A lot of thanks goes to Earl Raguse for information I have used as well as Tom Freeman and I can't guess how many others have put their little bit of info in as well. As Earl Raguse writes - "A partial list of some newsletters which have published good file info are: the UGOC ROM, LA99ers TopIcs, Long Island 99er, St Louis Computer Bridge, Greater Akron 99er, Columbus's Spirit of 99, Cleveland Area UG newsletter, Birmingham BUG, etc, etc, etc.....>>>>"

Other sources of information on files are indeed the XBASIC and EDITOR/ASSEMBLER manuals.

Down to the nitty gritty. BASIC means TI Basic, XB means Extended Basic and A/L means Assembly Language.

If the file is 25 sectors, you could have files that relate to any number of graphic programs that are on the market - too numerous to list. A hint would be if the filename is followed by _C or _P. A hint to the nature of the files on a disk can also be gleaned from the DISKname (presuming the disk has a name) like ARTISTPGMS, indicating the disk might have something to do with one of those numerous graphic programs.

If the file is less than 33 sectors, try BASIC, then XB, then A/L

If the file is 33 sectors, then A/L

If the file is 34 sectors, then it is probably a GRAM-U-LATOR or GRAM KRACKER files. They tend to end in numbers, from 1 to 7. If you don't have a GRAM device, you will NOT get them to load or work or do anything using any of the BASIC, XB or A/L approaches.

If the file is over 34 sectors; try BASIC, then XB. You may need to do a CALL FILES NEW OLD DSKn.Filename RUN. See the XB manual for CALL FILES if you are unfamiliar with it. Also could be a FORTHSAVE file, which requires the Forth kernel. (Another story unto itself.)

If the file is 52 sectors, you are possibly looking at a Tunnels of Dome file.

If the file is 54 sectors, you are possibly looking at a Scott Adams Adventure file.

If the file is listed as a "Program" but still won't RUN, don't erase it immediately as a lost cause - it could be data for another program on the disk, which will not run if the erased file is not on the disk. WHO said this was going to be easy??

Now we'll go from the "PROGRAM" file to some of the other files that can appear on your disk.

If the file is an INT/VAR 254, it's probably XB - you'll need 32K

The "memory image" format of assembly language program is the most compact and the fastest loading. It can be stored on cassette or disk. It is identified as "PROGRAM" in the disk catalog (just like a Basic or XBasic program.) Memory image programs can be loaded by option 5 on the E/A cartridge menu or by option 3 on the TI Writer menu. It should be noted that there is one small difference between how the E/A calls a memory image program and how TI Writer does it. TI Writer blanks the screen just before calling the program and E/A does not. This means the program being loaded must turn the screen back on or nothing will show. Memory image programs are created by a utility program, like the one called "SAVE" that is provided on the E/A disk.

There is a limit on the size of an Assembler memory image file of >2000 bytes. (The reason for this limit is that the various loaders use the 16K VDP memory to transfer the data during the Device Service Routine [DSR], and only that number of bytes have been allocated in the transfer buffer.) However, the E/A and TI Writer cartridges will load multiple memory files to make up a program of any length. They use the convention that the name of the second and following files is obtained by incrementing the last digit or letter of the previous file name. For example, the TI Writer editor consists of two memory fields, "EDITA1" and "EDITA2".

(As a matter of interest....the Adventure, Tunnels of Doom, Personal Record Keeping, Statistics, and Personal Report Generator cartridges all use memory image files to store data. The fact that memory image files can be saved or loaded with single I/O operation makes them attractive for such uses.)

Now, let's take a closer look at memory image format. Assembler memory image files have a three word "header" followed by the program data to be placed in memory. this three word header is not "loaded", but rather directs the loading process. The three words are used as follows:

1) The first word is a "flag". If it is "FFFF", then this flag is NOT the last in a multi-file program. If it contains "0000", then it IS the last file.

2) This word contains the length of the file in bytes (approx >2000), including the six byte header.

3) The third word is the CPU memory address where the memory image is to be loaded.

Execution of a memory image program always begins at the first byte of the segment loaded.

OKAY, enough already (Actually there is one little bit left after this.) But first, let's take a look at what we have. Basically, if you can't get a "PROGRAM" file to run with the XBASIC cartridge in place, you are going to have to turn off the unit, and install the Editor/Assembler cartridge in its place. Restart the computer. Get E/A on screen and try Option 5 - especially if this PROGRAM is 33 sectors and happens to have another PROGRAM file (of almost any size up to 33) following that has the last letter or digit incremented by 1. I'm going to put a list at the end of all this that will hopefully do something for you. As always, if all else fails - PICK UP THE PHONE AND TALK TO ANOTHER COMPUTER CLUB MEMEBER - he/she might have the answer and save you lots of grief and possibly stop you from striking severely upon the computer.

VAR 254). The usual "SAVE" and "OLD" commands are used to store and load these programs. The third form of storing programs used by XB is the "merge format" as a "DISPLAY VARIABLE 163" file. (Again, it could show as a D/V 163 or DIS/VAR 163). This form is created when the "MERGE" option is specified with the "SAVE" command. (If you are not familiar with the merge format, see pages 122 and 123 of the TI Extended Basic manual). The beauty of merge format is that when it is loaded, it does not necessarily overwrite the program in memory. The "MERGE" command does just that - it merges the new program (or program segment) with the program in memory according to the line numbers. (A line from a file being "MERGEd" that has the same number as one already in memory will overwrite the old line).

So let's quickly review what just transpired (Is that like sweat??). You've catalogued a disk and found 3 files that are listed as "PROGRAM", 2 files listed as "INT VAR 254" and 1 file showing up as "DIS VAR 163". The INT VAR 254 files can be loaded via "OLD DSKn.Filename" and then RUN. With any degree of luck you will probably be looking at a new game, or whatever doing its thing on your TI99/4A. Doing the same with the "PROGRAM" files could get you the same results and if the program runs - GO FOR IT. You can load the DIS VAR 163 file into the computer and edit it to see if it is an update to one of the other files - or it might not even be related to any of them, but was on that particular disk.

If you type in wrong filenames, you WILL get an error message on the screen. If you typed everything in perfectly and your "PROGRAM" file still gives you an error message, do NOT dispair. There are other possibilities as to what that "PROGRAM" file is. [Nobody said the computer would be perfect.] There's more to come.

Now, we get to the "GOOD STUFF", Assembly Language programs. There are three formats for assembly language programs: (1) tagged object, (2) compressed tagged object, and (3) memory image.

Tagged object code files are stored in "DISPLAY FIXED 80" files on disk only. All program data are in hexadecimal code so that they can be edited by the E/A editor. Tagged object code can be loaded by "CALL LOAD" in XBasic, by option 3 ("Load and Run") on the E/A menu, by option 1 on the MiniMemory module menu (as the man said - that's a real tongue twister!!!), or by "CALL LOAD" in TI Basic when either the Editor/Assembler or MiniMemory cartridge is plugged in. The programs can have "absolute addresses" or be "relocatable". A program with an "absolute address" is always loaded into the same place in memory. A relocatable program can be loaded into any place in memory. A tagged object program can have references to other programs or subroutines. The loader will resolve these "external references", except for the XB loader.

Compressed tagged object code is very nearly the same as tagged object code except that the program data is stored as bytes rather than as hexidecimal digits. Compressed tagged object code loads faster than regular tagged object code, as you would expect. The XB loader cannot load compressed object code.

Tagged object code, in either regular or compressed form (compressed if the "C" option was chosen while assembling) can be produced by the Assembler when it "assembles" source code.

PROGRAM, PROGRAM?, PROGRAM!

by Dean Hancock

Well it must be something. It's on my disk. But what is it? How many times have you asked yourself that question - especially after you carefully typed in the correct filename for a program only to have the computer make some rude noises at you and put a meaningless message on the screen. The question of what different filenames and their cryptic types will NOT be completely solved here but I will attempt to shed some light on it. The reason I decided to type up some information on files and file types is because somewhere along the way at one of our Thursday night meetings, someone asked the question about filenames, types and how do you load them. Of course, numbers of answers were given to various questions and more than likely the answers were forgotten (mostly due to the fact that there was almost too much data to consume in one evening and if you didn't write all of the information down you forgot it by the time you got home [or misplaced the paper you wrote it on]). Now let it be known that 99.9% of the information that follows is NOT mine. I have found articles written by Earl Raguese in the LA 99er TopIcs newsletters plus some information again from LA 99er TopIcs by Tom Freeman that was in the Ozark 99 User Group Newsletter and sent to LA by Steve Langguth. To make matters even more interesting one of the articles is a compilation of THREE other articles appearing in other newsletters. One article was written by R.A. Green of the Ottawa 99/4 Users Group. It originally appeared in the May 1986 New Jersey Users Group newsletter and was reprinted in the September 1986 newsletter of the Kansas City TI 99/4A Users Group. Another article was written by Jerome Trinkl and was somewhat more detailed. It appeared first in the April 1986 newsletter of the Atlanta 99/4A Computer Users Group and was reprinted in the Greater Akron (Ohio) 99ers September 1986 newsletter. And the more other TI club newsletters I read, the more times I see it appearing in their newsletters in various writeups. Are we confused yet? So one can see that the articles have been around. BUT NOT in OUR newsletter, until NOW.

There are seven (7) different ways to store programs in the TI 99/4A. I'll try to list the program files and what to expect. The biggest problem we have is which cartridge do we need in the console. Let's try and stick with the Extended Basic and the Editor/Assembler cartridges. There are numerous offshoots of both those cartridges but let's stick with the stock TI ones, since most everyone has those (I hope).
Most everyone is familiar with the form used by TI Basic to store programs on cassette or disk. It is identified as "PROGRAM" when a disk is catalogued. It is created or stored by the Basic "SAVE" command and loaded into the computer by the Basic "OLD" command. This is the only way that TI Basic uses to store programs.

Extended Basic can, and usually does, use the same format as console Basic to store programs. There are, however, two other forms that Extended Basic can use to store programs on a disk (but not on cassette). If you have the 32K Memory Expansion (or almost any setup that gives you 32K memory), you can write an XB program that is too large to store in the usual format. XB will store these large programs in an "INTERNAL VARIABLE 254" file. (Depending on what method you use to read a disk, the file will show up as I/V 254 or INT

CECURE®
ELECTRONICS INC.

CURRENT PRICES FOR REPAIR / EXCHANGE of TI HOME COMPUTER EQUIPMENT.

PLEASE NOTE THE FOLLOWING CONDITIONS:

Your equipment will be repaired or exchanged in 5 business days or less. Equipment must be in a repairable condition with all components intact or we reserve the right to return item/items not repaired. The 6 month LIMITED WARRANTY is still in effect. Wisconsin ship addresses must pay 5% tax on REPAIR / EXCHANGE. All Shipping, Handling and Insurance charges are per item and for U.S. addresses only. CANADA and OVERSEAS contact CECURE ELECTRONICS INC. for Shipping, Handling, and Insurance costs. All shipments to CECURE ELECTRONICS INC. MUST be PRE-PAID and insured as we WILL NOT accept C.O.D. shipments or be responsible for lost or damaged shipments. To **EXPEDITE** your repair please CALL for a RA # prior to shipping your equipment. PRICES shown are **CASH DISCOUNTED PRICES.** Check or Money Order Must be with unit sent in. DO NOT SEND CASH! Sorry, but **NO C.O.D. ORDERS.**

MODEL		PRICE		S/H/I		TOTAL
TI 99/4A COMPUTER		45.00	+	6.00	=	51.00
PHP 1200 EXP BOX	*	45.00	+	12.00	=	57.00
PHP 1220 RS232		33.00	+	6.00	=	39.00
PHP 1240 DISK CONTROLLER		44.00	+	6.00	=	50.00
PHP 1250 B DRIVE INTERNAL	*	19.00	+	6.00	=	25.00
PHP 1260 32K MEMORY EXP		44.00	+	6.00	=	50.00
PHP 1270 P CODE		33.00	+	6.00	=	39.00
GENERAL ACCESSORIES						
PHP 1500 SPEECH SYN	*	20.00	+	3.00	=	23.00
PHP 1600 MODEM		33.00	+	6.00	=	39.00
PHP 1850 A DRIVE EXTERNAL	*	39.00	+	8.00	=	47.00
PHP 2500 EPSON PRINTER	*	64.00	+	12.00	=	76.00
PHP 2700 TAPE RECORDER		17.50	+	6.00	=	23.50
PHP 4100 PANASONIC MONITOR		100.00	+	12.00	=	112.00
PHP 2100 MODULATOR	*	10.95	+	3.00	=	13.95
PHP 1100 JOYSTICKS		9.75	+	3.00	=	12.75
AC 9500 TRANSFORMER		12.95	+	3.00	=	15.95
BOX CAR EXPANSION ACCESSORIES		- Limited repair & exchange				
PHP 1400 RS232		52.50	+	6.00	=	58.50
PHP 1800 DISK CONTROLLER		60.00	+	6.00	=	66.00
PHP 1900 OID PRINTER		70.00	+	6.00	=	76.00
PHP 2200 32K MEMORY EXP		70.00	+	6.00	=	76.00
PHP 2400 P CODE		70.00	+	6.00	=	76.00

SHIPPING ADDRESS FOR TI REPAIR / EXCHANGE ONLY

CECURE ELECTRONICS INC.
TI HOME COMPUTER SERVICE CENTER
SOUTH 74 WEST 17000 JANESVILLE ROAD
P.O. BOX 22
MUSKEGO WI 53150-0022

VOICE: 414 679-4343
FAX: 414 679-3736

* NEW REDUCED PRICE

REV. 1 PRICES AS OF 9-1-94

TEXAS INSTRUMENTS
ONLY AUTHORIZED REPAIR AND SERVICE CENTER FOR THE HOME COMPUTER