# SHIFT838 Newsletter

SEPTEMBER 19, 2015 VOLUME 1, NO 9

This newsletter is dedicated to the ongoing support for the Texas Instruments TI-99/4A and Myarc Geneve 9640 user community and is published by SHIFT838.

First I wanted to inform everyone that the '**Floppy Days Podcast**' by Randy Kindig that was supposed to happen in August was postponed and we are shooting for September 23, 2015.

These are just a few of the links available for '**Floppy Days Podcast**':

http://floppydays.libsyn.com/

https://www.facebook.com/floppydayspodcast

https://twitter.com/floppydays

Once Randy has finished it I am sure he will post a link to it on AtariAge as I will on my site. Be sure to look for it. There are going to be some topics highlighted specifically for the TI-99/4A.

In this edition I was able to score some time to pose a few questions with ***Danny Lousberg*** *otherwise known as* '***The Mole***', developer of the upcoming new and exciting game port of '**Alex Kidd'** for the TI-99/4A. Please read on, I am sure you will enjoy.

If you have not read up on his Alex Kidd port please do. It can be found on Atariage.com at: http://atariage.com/forums/topic/226068-alex-kidd-port/

Thanks to all that have subscribed!

## Interview

**Chris:** How long have you been involved in the TI?

**The Mole:** My dad bought a TI when I was 4 years old, which was in 1982. I very distinctly remember playing TI Invaders for the first time for half an hour or so before going to bed. It was super exciting and I couldn't sleep, so early the next morning I snuck back downstairs to find my parents playing, trying to beat each other's high scores. They'd played all through the night. A few years later, when I was 10, we got an apartment by the sea (where my mom's side of the family lives) where we spent most of our vacations. But if you know a bit about the weather in Belgium, you know that for the better part of the year you end up being stuck indoors. I played my TI games for hours on end there, and when I finally got fed up with Parsec, Carwars, Invaders and a handful of Extended Basic games I started

messing about with some type-in magazine games. You know, tediously typing in those long listings, changing graphics here and there, adding lives or speeding up bullets, adding my name to the title screen. That's when my dad gave me the Extended Basic reference manual and taught me how to code my first BASIC programs.

**Chris:** What specifically drew you to the TI-99/4A?

**The Mole:** It was basically what we had for a very long period of time. I was quite jealous of my friends with their C64's though, the quality of games on that system was just miles ahead of what we had on the TI. We did get a Sega Master System and Genesis eventually, so while I kept using the TI for programming, I was able to satiate my gaming needs elsewhere on more powerful systems.

**Chris:** Can you highlight a few other programs or projects you have been involved in for the TI?

**The Mole:** Not that much really, I only got back into the TI a couple of years ago by discovering ti99sim and the GCC port by Insomnia (I've been a Linux user for the longest time, and used to work as a C developer, so that's really where my skills and interests lay). My first public project was probably my attempt at doing a raytracer for the TI, of which you can still find some videos on my YouTube channel. In the end I got it to work, but it was too slow for an action game, so I left it at that. In between Alex Kidd coding sessions I have used that codebase to test my build environment for the F18A GPU though. I might just come back to it for an F18A exclusive game, who knows... I also released (an early version of) the source code for the PC-side tool that creates the graphics data for Alex's scrolling levels. Oh, and I did the splash screen graphics for Magellan's more recent versions.

**Chris:** Can you explain to the users what do you think 'YOUR' favorite feature is of the TI currently?

**The Mole:** This is a hard one to answer, nothing really sticks out. It's the complete package and a healthy helping of nostalgia that makes me love the system. It's funny really, if you would've asked me what I dislike about the TI, I could probably list any number of things... yet although it's my favorite system ever it's hard for me to single out what exactly I love about it.

**Chris:** Do you mainly exercise your skills within a TI Emulator or actual real hardware and why?

**The Mole:** Well, I use a cross-compiler, so I do everything on my Mac. I do have the real deal right in front of me (check the attached photo of my messy desk) and will ever so often check my work on that, but the development cycle is just so much easier with emulation. I would not have been able to make much progress if I actually had to code on the TI directly and I'm not one of those people who actually uses the TI as a computer for day-to-day use.

**Chris:** Why did you choose Alex Kidd to port to the TI-99/4A?

**The Mole:** I've always been someone who roots for the underdog, so when all my friends got Nintendo's, I got a Master System. The game that happened to come with the console was Alex Kidd in Miracle World and I absolutely loved it. It had a depth and playability that kept me coming back, even to this day where I will pick up the game and just play a few levels. I always wanted to make a side-scrolling platform game for the TI, since I feel the system is severely lacking in that area and it was probably the most popular genre in the generation of game consoles and home computers that came after the TI. Lastly, I figured my first game would be better if I could piggy back on the design of an existing game, instead of having to figure everything out from scratch.

**Chris:** I believe the game is coded completely in C.  in your opinion what do you think the response from other coders are going to be by showing that the TI-99 can utilize C very extensively, specifically for the gaming area?

**The Mole:** All the game logic is done in C, with a few small bits of inline assembly here and there to talk to the hardware. I actually spent a few hours replacing the majority of assembly code with C code, for maintainability reasons. I truly believe that Insomnia's GCC port creates code that is so close to pure assembly in terms of speed for everyone but the most experienced assembly coders that it is an ideal development environment for action games. I hope that it will inspire other coders who are perhaps a bit daunted by assembler to try out coding something in C. Much like BASIC, C is a language that is easy to understand and very structured if you pay a bit of attention and keep to a few simple rules. It's infinitely faster though, so for some of the people that are looking to take their code one step further it is an excellent choice. And for those that are new to C, I would recommend using Tursi's very nice libti99 which provides the equivalent to a lot of BASIC's CALL statements (not work-alike's though, you'll need to learn how to use them) so you don't have to worry about talking to the hardware yourself.

**Chris:** With your port of Alex Kidd, are you adhering to the original gameplay or have you changed certain aspects to your liking while keeping the feel of the original game?

**The Mole:** I spent quite some time trying to get the physics as close as possible to the original. I think this is what I find lacking in most early platform games, like for instance miner2049'er, Jungle Hunt or even Donkey Kong: the character just doesn't control very well, with fixed jumping arcs and instant acceleration/deceleration.  On the other hand, I had to make a fairly large number of concessions to make the game work on the TI, specifically in level design and features.

> **Chris Follow Up:** If you have changed things, can you elaborate on a couple that you have done and why?
>
> **The Mole:** There's a large number of things that will be different for technical reasons:
>
> - For instance, there's just no more room for patterns to make the distinction between large and small money bags or to add the special item boxes that some levels have. So I stuck with one size of bag and I will hide special items in the regular 'star' boxes.
> - The original game has vertical scrolling levels as well as horizontal scrolling levels. I'd need to extend my graphics engine quite a bit to

support that on the TI and I'd rather get something out first before going overboard with special features.
- I'm not sure I'll include the iconic Janken matches. I never felt throwing in a couple of rock-paper-scissors matches was a good fit for an action game. I haven't decided on this one yet.
- The shop that you find in the original game will not be part of the levels, but instead show up before you start the level. Purely for technical reasons, since the shop would eat up way too many patterns and not look very good given the limitations in colors imposed by the scrolling algorithm.
- There's a story in the original, which is told by characters you meet at the end of certain levels. This will be replaced by separate story screens, again due to lack of patterns.

Given that I'm already forced to make a number of changes for technical reasons, I decided to not make a straight-up port, but to create a new game in the same universe, with mostly the same type of gameplay but with new and original levels. The current levels in the demo are ripped from the original game, but it will be mostly original levels in the final version.

**Chris:** What are some of the main hurdles that you have had to overcome for this port?

**The Mole:** The game has a relatively dynamic environment, so it consumes a lot of RAM. Originally, the game was supposed to run from disk but I ran out of memory. I spent a lot of time devising compression methods that would be speedy enough to decompress/update/recompress the map data in real time, or would allow manipulation of the map data without decompressing/recompressing, but they all turned out to be too slow to keep the game running at 60fps. I'm sure a more talented coder than me would be able to pull it off, but as things were going the whole thing nearly brought me to the point of abandoning (or at least greatly simplifying) the project. I left it for what it was for a good 8 months or so, before concluding that a cartridge release was the right approach. It will still need the 32k memory expansion, but all code runs from cartridge leaving most of that 32k available for game data.

**Chris:** I know there has been new games that are specifically designed to utilize the new F18A video upgrade for enhanced graphics and will not run on the standard TI-99/4A without the F18A video upgrade.  Is your Alex Kidd port work on a standard TI-99/4A or does it require F18A enhanced graphics?

**The Mole:** The game will run on a standard TI with 32k memory expansion, but users that have an F18A installed will see greatly improved graphics. The game will automatically detect which VDP you have and select the best rendering path accordingly. This means that there will be one release for all systems. Rasmus has done some wonderful things on the F18A, and that has really inspired me to treat it like a first-class citizen.

**Chris:** Your Map screen you came up with is quite different from the original (I personally think its better, less cluttered up), but why did you steer this way?

**The Mole:** I always felt that the more stylized over world maps like you see in Mario or Zelda were a better match for that generation's graphics capabilities, and most of the graphics in the original Alex Kidd were very cartoonish and stylized, except for

the map screen. I never really understood why they did this and figured I'd take a stab at designing my own map from scratch. Doing graphics is really something that I tend to do when I feel like I need to take a break from coding, but still want to work on the project. It's a way to clear my mind when I'm facing a nasty bug or particularly daunting piece of new functionality. I very much enjoyed doing the backgrounds for the F18A version recently.

**Chris:** How much improvement do you think has been done to your game since you have been open to user feedback on the AtariAge forum?

**The Mole:** That's difficult to say. I'm lucky in the sense that I've been able to stand on the shoulders of giants like Rasmus, Tursi and many others for some of the more tricky technical things. One example of where I got a lot of really useful feedback was on the map design, where multiple people actually stepped up and contributed bits and pieces that ended up in the final design. I've always felt that was a great showcase for how well our little community can work together.

I can tell you one thing though, if it weren't for the many positive reactions on the board I probably wouldn't have stuck with the project for so long. Typically in my projects, I tend to get to the proof of concept phase and then lose interest, but the continued support and kind words from the community really help in keeping my motivation up.

**Chris:** The music is amazing, was the music coded by you and worked out by trial and error or was it an easy port?

**The Mole:** All credit for the music and sound effects goes to Tursi's amazing SPF music library. I don't think enough people realize what an impact Tursi's work on this has had, but a good deal of the new demos and games you've seen recently make use of his excellent tools. They've allowed me to basically download the original games' music files in VGM format from www.smspower.org and convert them to a compressed format that is directly usable from C and assembly using Tursi's highly optimized playback library.

**Chris:** Can you briefly explain why this game is now going to have to be a physical 512k cartridge for owners of real hardware.

**The Mole:** Like I mentioned before, I need a lot of RAM to store level data while the game is running, with the actual amount depending on the length of the level. On average you're looking at 3k for the music, 12k for the nametable and collision data and a couple of K for other miscellaneous stuff. I also need about 24k for code, so the 32k didn't cut it. I thought about requiring a SuperCart, but that would've only given me an extra 8k, which might have worked but still would've been very tight. Another option was targeting (S)AMS equipped systems, but I wanted to target as wide an audience as possible. So I decided to run from cartridge and drop the disk requirement. The reason I'm targeting the 512k carts (and not anything smaller) is because I need quite a bit of storage space for the level data for the two different VDP chips. In the original disk-based project I had planned on doing two separate downloadable releases, but for a cartridge release you really need to make sure you have just one SKU for all versions.

**Chris:** Will you be producing the physical cartridges and selling them with manuals, etc.  Or are you going to just provide the code for the user to burn to an EPROM on their own cartridge board or both?

**The Mole:** I'm currently planning on doing a physical release with real boxes, manuals and the works first, and releasing the final cartridge image for those that want to burn their own once I've got enough pre-orders to cover the setup costs. I really hope that I can match the sort of quality that you would see with a collectorvision release, for instance.

**Chris:** I know the game is not completely finished but if you had to estimate how many hours do you think you have invested into this project and how much longer do you think it will take to wrap it up?

**The Mole:** I've been working on this on and off for short periods of time over the past two years or so. It's hard to put a number on it, but we're certainly looking at several weeks if not months' worth of full-time coding. I don't consider myself a bad coder by any stretch of the imagination, but when I see the speed at which people like sometimes99er or Rasmus can knock out games I have to admit that I'm nowhere near as prolific as those guys. I see light at the end of the tunnel as far as coding is concerned, but I still have a lot of work on creating and designing levels and graphics. When I recently did the design for the forest level, I probably spent the equivalent of about 3 8-hour working days on that. New levels will go faster as I optimize my workflow, but I want to have at least 10 more levels before I think I have a game worth releasing.

**Chris:** Do you have an estimate release date for us to look forward to?

**The Mole:** I plan on releasing a playable demo sometime this month. It will have three levels and is mainly meant to capture all remaining bugs and feedback from the community. I also hope it'll tide people over while I work on the final content of the game. With a bit of luck I can have everything wrapped up in time to put one of the cartridges under my own Christmas tree.

**Chris:** Have you had a chance to think of what you see on the horizon for your next project and if so can you elaborate?

**The Mole:** You're not rid of me yet, there are a number of things that I would really like to do. I'd like to make a really good pseudo-3D racing game in the style of Pole Position, but with a much smoother rendering engine. One of my all-time favorites in the genre is Pit Stop II on the C64. This might be an F18A/v99x8 only game though, I'm not sure the original VDP would be able to pull it off.

I'm also eager to do a port of Activision's Ghostbusters game. That's another favorite of mine for which almost every other home computer and console of the day got a port, and there's really nothing there that the TI couldn't do. This is the current frontrunner for my next project.

Lastly, I hope that I can continue to evolve the Alex Kidd engine with things like horizontal/vertical scrolling, slopes, etc. and use it for more/other games. Eventually I hope other people will be able to pick up the tools and design their own games. It's not for the faint of heart and it'll be a bit more complex than tools like Magellan, but

I hope to make it easy enough so that motivated non-coders can play around with it as well.

## September Highlights

- Check out this month's Atariage "TI-99 Hi Score Contest" game "Miner2049er". The winner takes home a pristine TI-99/4A Football cartridge and manual. Last month's winner, globeron, scored over 150,000 points in Munchman!!
- Remember to keep tabs on Rasmus's new game in development, "Bouncy." Looks to be equal in visual brilliance to his multiple previous games.
- Vorticon is currently developing a new game in TurboForth entitled JetPac. The development thread can be viewed in the AtariAge TI-99/4A Development forum.
- Jedimatt42 has been working on a USB Keyboard Adapter utilizing Adruino style components. He has already made significant progress. Check out his thread on AtariAge in the TI-99/4A Development forum.

## Software

**9640 Menu System for the TI-99/4A!**

Looks like Tim T. (Insane Multitasker) is slamming another one out of the park for us! Tim has managed to find the original source code to the 80-column Geneve version of BOOT. This program has not yet been released to the public as of yet and Tim is working to get it to that point. I have been lucky enough to get a copy for review.

You can read the original thread on Atariage.com at:

http://atariage.com/forums/topic/241745-80-column-menu/

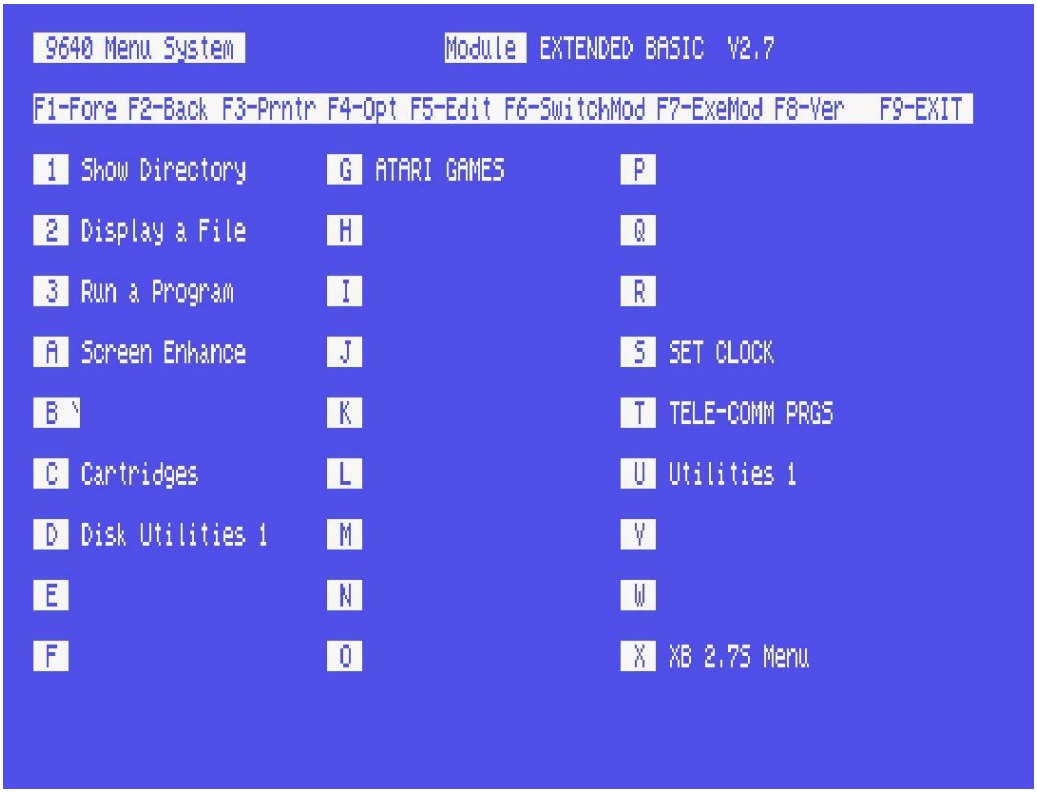
This menu system will definitely satisfy F18A and 9938 users at the same time.

The new BOOT gives true 80 columns so you can have quite a few selections (A through X) and you can also chain the menu systems by saving as executables to be configured on the menu as selections to give you an unlimited amount of menu options. This is known as chaining and can be configured using the 'F4' option on the menu.

This new menu systems supports multiple clock cards. This program has also been tested with the MESS system with the HSGPL card mounted. Just be sure to update your version of MESS to 0.153, which has addressed several bugs in order to run properly.

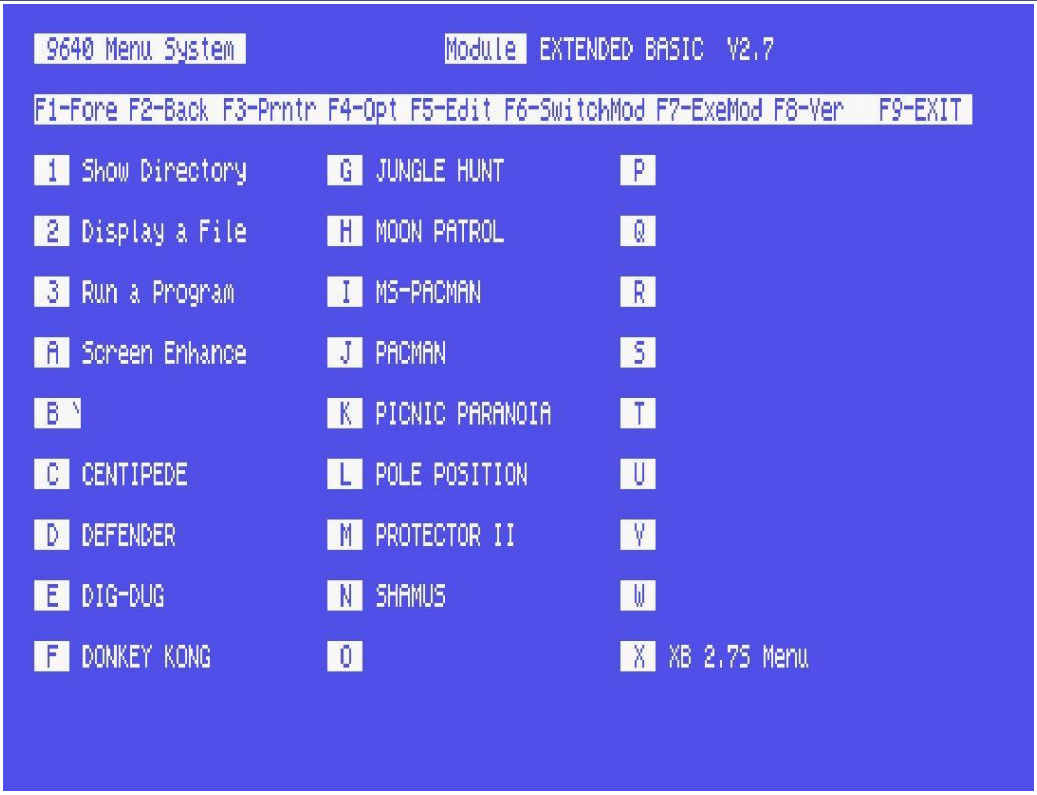
In order to do chain correctly it takes a little planning on your part. When I laid out my menu chain I setup the chain menu as I wanted, for example called 'Disk Utilities'. After I saved it I would then hit F4 to save as an executable. This saves a file on the floppy drive you specific as 'USEROPT1'. You will then need to rename that file via a disk manager to the menu name you want. Keep repeating this for every chain menu.

Once you have all of your chain menus you can then edit the menu one more time and call all your chain menus for the key you want pressed. I have taken a few screenshots below. I have configured a handful to provide examples.




The above screen is my main menu screen for the new BOOT.


Above is an example of a Disk Utilities Menu.




Of course we are all familiar with the Atari Games!

I have also configured a TeleComm menu for all of my telecommunications programs. I am really just testing the chaining options and as Tim has said this gives you unlimited options for menus. I am sure I could no way configure that many. Most of us could probably get away with one screen menu for all of our programs we typically run or at least add menus for ATARI GAMES, GAMES, and UTILITIES. I think since there are so many Atari games for the TI they should have their own menu! I invested many hours playing those games through the nights when I was young!

In my opinion this menu system is by far the slickest that has been created for the TI-99/4A. I always wanted a menu system like the one on the Geneve 9640 and now it's going to be a reality for TI-99/4A users with 80 column capabilities.

Also a quote from "Ω" about the program:

"***This program will become THE MENU of choice for every TI'er using an F18A***".

## Calling All GAMERS!

Owen Brand (***Opry99er***) has started a TI Gaming competition on AtariAge where a TI-99/4A game is chosen every month and TI'ers can compete to see who can get the highest score. At the months end the person with the highest scores receives some type of prize.

If you want to read the message thread in its entirety and possibly participate in the friendly completion then click below:

http://atariage.com/forums/topic/241547-official-ti-994a-hi-score-competition/page-1

Last month's game was : **MunchMan**
Winner was: **globeron** with a score of **153,840**

All I have to say is DAMN, he has to have some blisters!

This month's game is : **Miner 2049er**

**GOOD LUCK!**

I never really played Miner 2049er much as a kid but I recently started to see how far I can get. This is a lot harder than MunchMan! I can get to level 3 and a bit over 9000 points, but I have yet to figure out how to get past this level.

To help us out, Sometimes99er has hacked the original .bin file for use with an emulator to help us practice. As this new file never takes any lives away when you die! Remember this is only to be used for practice and Sometimes99er has changed the font of the score to make sure. You can find this file in the original post at the link above.

## Coding

I have had more than a few inquiries to start some type of assembly language tutorial in the newsletter. I have reached out to a few individuals that have experience in programming with TMS9900 assembly language. Lee Stewart was more than willing to help out for this first tutorial. The tutorials are not going to be extensive in nature as this would cause the newsletter to be extremely long. So we are aiming to some fairly short tutorials just to show some basic operations. I hope that this will be the first of many tutorials to help out the TI Users.

**Assembly Tutorial #1 - Screen Color by Lee Stewart**

In our first tutorial, we will be showing how to change '**Screen Color**' via TMS9900 assembly programming, but let's start by describing some of the areas this tutorial will cover.

The TMS9900 CPU uses 3 internal and 16 general-purpose, RAM-based, workspace registers. The Assembly Language Code (ALC) programmer can directly modify the workspace registers. S/he can also reserve and switch to any workspace location desired. More about TMS9900 registers can be found at:

https://en.wikipedia.org/wiki/Texas_Instruments_TMS9900

Workspace registers save time and instruction space and some instructions will only work with registers. Most CPUs have all of their registers on the CPU chip. The TMS9900 processor has only three 16-bit, on-chip registers: PC (Program Counter—contains the 16-bit address of the next instruction to execute), WP (Workspace Pointer—contains the CPU address of the first register of the current, 16-register,

32-byte workspace) and ST (STatus—contains the status result of the last instruction affecting status—not all do).

Registers are referenced by number: 0 – 15. We can represent numbers in ALC by naming assembly-time constants with the EQU directive that make it easier for us to make sense of our program. That is why we usually refer to register #0 as R0, etc. Of course, we can use any name we want for a given constant:

```
R0    EQU  0
R1    EQU  1
.
.
.
R15   EQU  15
```

Actually, we do not need to write the above EQUates into our program because we can direct the Assembler to do it for us.

There are 5 kinds of addressing used by the TMS9900:

1. Workspace Register,
2. Workspace Register Indirect,
3. Symbolic Memory,
4. Indexed Memory and
5. Workspace Register Indirect Auto-increment

Four of the above use registers and only two use memory addresses directly. Our little SCREENCOLOR program uses only two, workspace-register and symbolic-memory addressing. There are many examples of workspace-register addressing in our program. Symbolic-memory addressing uses '@' in front of the address value or symbol. The only example of symbolic-memory addressing we use in our program is @VWTR.

One more thing before we get to the program: Hexadecimal (base 16) numbers are represented in TMS9900 ALC prefixed with '>', e.g., >01F0, which is 496 decimal (base 10).

Below is the code used to change the screen color (comments included):

```
*===
SCREENCOLOR=========================================
* This program changes the screen to TEXT mode without clearing
* the screen.  The text from the Editor/Assembler "Load and Run"
* screen remains displayed but is skewed because the screen is 8
* characters wider.
*=============================================
      REF  VWTR         * reference E/A utility
      DEF  START        * declare Start label of our program so E/A loader
*                       can find program entry point

* Program entry point

START  LI   R4,>F485      * load R4 with pattern for color changes
```

```
*                          rotating bit pattern begins with 1111010010000101
*: LI = Load Immediate value into register.  The immediate value must be known
*:   at assembly time, so it can also be a label and involve a calculation.

     LI   R1,>0700      * load R1 with # of VDP register for screen color in MSB
     LI   R0,>01F0      * load R0 with TEXT mode setting for VDP register# 1
*                         VWTR needs R0 with MSB=VR#=(>01) and
LSB=setting=(>F0)
     BLWP @VWTR         * change to TEXT mode

*:   BLWP = Branch and Load Workspace Pointer.  Uses the workspace pointed to by
*:   the source address/register; saves the current PC, WP and ST in the new R13,
*:   R14 and R15; and branches to the address pointed to by the address 2 bytes
*:   after the source address/register.
*:   VWTR points to the WP and address of the E/A utility for loading values
*:   into VDP registers.

* Infinite color change loop
NEXT   LI   R5,>6000      * load delay counter
     MOV  R4,R0           * copy R4 (contains changing FG-BG color byte in LSB) to
                          * R0

*: MOV = MOVe (copy) contents of source address/register to destination
*: address/register

     MOVB R1,R0           * copy MSB (>07) from R1 to MSB of R0 to change screen
                          * colors

*: MOVB = MOVe Byte from MSB of source address/register to MSB of destination
*:   address/register

     BLWP @VWTR         * change FG and BG screen color (FG only affects TEXT
                          * mode)
*                           VWTR needs R0 with MSB=VR#7 and LSB=colors
*                           colors are left nybble=FG; right nybble=BG

* Delay loop
DELAY  NOP                * a do-nothing operation
     DEC  R5              * DECrement delay count by 1
     JNE  DELAY           * if R5<>0, continue delay loop

*: JNE = Jump to label if status of last instruction is Not Equal to 0

   SRC  R4,1                * shift R4 circularly right 1 bit to get different colors in
                            * LSB
*: SRC = Shift Right Circular the value in the register by the number of bits
*: indicated
*:   by the immediate value after the comma.

   JMP  NEXT                * loop for next go-round

*: JMP = Jump unconditionally to label

     END  START             *END of program; start automatically at START when
```

```
                    *program loaded

*== NOTES ============================================
*==
*== Labels start in first column.  If '*' is in column one, entire line is a comment.
*==
*== First text after spaces on non-comment line is an instruction word.
*==
*== After next space are one or more operands separated by commas with no
*== spaces (except
*== within TEXT quotes).
*==
*== After next space, remainder of line is comment.
*==
*== ASM994A:   ';' starts a comment anywhere on a line.
*==
*==          Labels may be longer than 6 characters
*==
*==          Program may be in lower case.
```

Regarding the values >0700 and >01F0 used early in the program, these have to do with the ALC function VWTR provided by the E/A cartridge. The name of the function stands for "VDP RAM Write Register". That function requires a value in R0 before it is called. The MSB (Most Significant Byte [left byte]) of the value is the VDP register number (0 – 7) and the LSB (Least Significant Byte [right byte]) is the value to be written to that register—the registers are byte registers. >0700 has 7 in the MSB in preparation for getting it into the MSB of R0 later. We are preparing to write to VR#07, the screen-color register, a value for foreground (FG) color of text-mode characters and screen color ( also, background [BG] color of text-mode characters). The FG and BG colors are 4-bits (1 nybble or ½ byte) each, e.g., >F4 is white (>F) on dark blue (4).

The value >01F0 is stashed in R0 prior to calling VWTR to write the value >F0 (LSB) to VR#01 (MSB) to change to text mode. Bit >10 of VR#01 is the text-mode-setting bit. See §21.1 of the E/A Manual for more information.

I hope this helps a lot of the 'Assembly Beginners'. If anyone wants a certain assembly tutorial covered, please e-mail me directly. I cannot guarantee that it can be done in a newsletter as we are gearing this for beginners and covering somewhat very small assembly programs just to provide a basic understanding.


## Resources

**Contact information**
To contact me please feel free to visit my website and click on the 'Contact' tab.

http://shift838.wix.com/shift838

**Newsletter Topics**
If you would like to participate in the writing of this newsletter or provide any topics for this newsletter please contact me via my web site.

## Sites

There are a few of sites that I think should get their own list below. These are for the TI Hall of Fame and TI-99ers Unsung website. Please visit these below sites as both have great information.

http://www.ti99hof.org/index.html

http://www.ti99ers.org/unsung/

Also the below site has a list of all the TI-99ers that have passed. Please be sure to check them out.

http://ti99ers.org/modules/Inspire/remember.htm

Below resources are just a handful of sites that support the TI-99/4A and/or Geneve 9640 computers. It is in no way a full list. This section will be included in all future newsletters. If there is a site that you think should be mentioned then please contact me.

Web sites / FTP Sites

http://www.99er.net

http://www.ninerpedia.org/

ftp://ftp.whtech.com

http://shift838.wix.com/shift838

http://www.ti99-geek.nl/

http://www.mainbyte.com

http://www.atariage.com

http://www.harmlesslion.com

http://www.ti99iuc.it

http://www.turboforth.net

http://www.ninerpedia.org/

Yahoo List Groups:

https://groups.yahoo.com/neo/groups/TI99-4A/info

https://groups.yahoo.com/neo/groups/TI994A/info

https://groups.yahoo.com/neo/groups/Geneve9640/info

https://groups.yahoo.com/neo/groups/turboforth/info

## Active BBS'

### HeatWave BBS

Access: Dial-Up and Telnet
System: Geneve 9640
Software: S&T BBS Software
Location: Arizona
Content: TI and Geneve file libraries, message bases, door games and e-mail.
Telnet to: www.heatwavebbs.com port 9640  Dialup : **602-955-4491 @ 8-N-1**

### The Hidden Reef

Access: Dial-Up
System: TI-99/4a Modified
Software: S&T BBS Software
Location: New York
Content: TI and Geneve file libraries, message bases, door games and e-mail.
Dialup : **718-448-9402 @ 8-N-1**

### The Keep

Access: HTTP  and Telnet
System: Pentium 4 running Windows 2000
Software: Worldgroup BBS Software (up to 256 user connections)
Location: Tigard, Oregon
Content: TI and Geneve file libraries, message bases, door games, multi-user and multiplayer games and e-mail.
Telnet : www.thekeep.net port **23**  Web browser to http://www.thekeep.net

The Keep has TI File libraries, Message bases, e-mail, door games, multi-user and multiplayer games.  The keep also has a modem line connected for anyone that would like to contact The Hidden Reef BBS from the internet through The Keep.

Simply telnet to www.thekeep.net on port 23, login to The KEEP and then type /**GO DIALOUT** at the main menu, then D1 to dial out to The Hidden Reef.  It's that simple.

## Vendors

SHIFT838 – Provides used TI equipment as acquired.  Check with me often.  A lot of the items need rehoming from other TI Users.

Arcade Shopper – Provides old and new TI equipment, upgrades and new runs of PCBs at www.arcadeshopper.com

## Repair Centers

### Richard Bell

Repairs available on limited basis, please contact Richard at swim4home@verizon.net for wait-time before sending any repairs

### Tim

Myarc-related hardware repairs on a limited, as-available basis.  Contact Tim at insane_m@hotmail.com for wait times or to request service.