GEOS PROGRAMMING INFO:

This small article is not intended to teach 8-bit Machine Language but
to get started with ML and GEOS/Wheels programming, mainly for programming
your own applications.

My best Teacher for learning the basics of GEOS/Wheels code:
Werner Weicht whom resides in Denmark and is the supporter for the European MP3 upgrade for GEOS. Mr Weicht has many years of GEOS/Wheels MP3 programming experience and I cannot recommend a more patient teacher.
My thanks goes out to Mr Werner Weicht.

Getting started what is needed:

Basic setup:

C=REU(not required for GEOS but better)
C=64
C=128
C=128DCR
2 floppy drives
C= 1351 mouse

GEOS V2.0 (good stable but not a lot of RAM to work with)
IN this case the programmer has to pack code to conserve memory in the app.
Geo Programmer
Geo Programmer user manual
Other possible ML books

Advanced hardware setup:

C= 17XX REU's (need some sort of RAM expansion to boot Wheels OS)
CMD V1 or V2 Super CPU (FROM 1MB TO 16MB ramcard).
CMD HD
CMD Ramlink with Paralell cable
1351 mouse

Wheels 64/ Wheels 128 GEOS update (handles RAM better)
Have all of the latest Wheels updates installed, WHY! Because some Wheels apps
have check routines to check for these updates, if the updates are not installed the
application will not fun or may not work correctly, VERY IMPORTANT.

Wheels still too only has so much Ram to work with, so the programmer has to keep in mind to
keep all routines small and simple, packing code may still be necessary if this is a long app.

Concept
Concept+

Geo Programmer and Concept all use Geo Write to create code and editing code, very simple.
NOTE: Concept and Concept+ do not work with the original Geo Programmer: Geo Debugger, both
are incompatible.

When creating code it is best to start with one routine, sort that routine from the rest
in its own single page, or routines. This keeps things organized to where all routines are
easily found and then debugging is easier.

In ML or GEOS code you will see semicolons and information after semicolon, these are simply
Comments that explain what routines do etc.

Comments do not use up RAM they are simply ignored when assembled, Comments explain code and help
new programmers understand (sometimes) how the code works.

In GEOS/Wheels you start with the Header file which contains all file information, including
the small Icon you click on to start any GEOS/Wheels application.
These at first are hard to figure out the "EXACT SIZE" the guy that helped me with the basics
made me a GeoPaint template. That template helps stay within the size the Icon is supposed
to be. If you go to big your Icon will look funny. If too small it will work but will be very small.
So for now try to use the Icon template.
The Icon can be created right in GeoPaint, then cut and saved, then you copy and paste that
right into the Header file, (the code section where the icon belongs) which in my example
says: Paste Icon here which are in Comments. If you do this that is all is needed and it should work
if you follow these guidelines.

The actual code:

After the Header file is the main code or however the programmer sets these code files up.
I have found it is best to work in small goals or steps in the code, then assemble the code.
Believe me you will get errors in programming.
NOTE: these are not the errors you get in BASIC programming these are OPP code errors
which then requires knowing or having ML programming knowledge to debug and fix
programming :
Errors, bugs, typos etc. For example from one of my fun debugging experiences (funny), was a
typo but a typo for a label in GEOS code, for example:

DoThis:

```
                LDA  more code etc                        :  this does this
```

Are you paying attention, note the Colon, this is supposed to be a Semi colon very easy to slip up and
type this, which can make you pull your hair out, what does this do.

The assembler thinks, :this does this is a simple label (only an example here), then there is no code
for it. Heh because the programmer slips up and makes this silly typo. :-P Not fun if youre not paying
much attention, but soon you will laugh at yourself and see your simple error. Other typos can occur
which can also cause annoying bugs, so be careful when typing. But sometimes these things happen so
you have to "think" and look in your code, "even" in your comments.

So be careful when typing, but I do suppose this could happen to anyone, but if this does occur then
always look at your comments: example:

;my comment But make sure you do type the semi colon and not the Colon, can really drive you nuts when the
assembler says this label exists and you do not see it, hmmm just watch the typos and if they occur look in your
comments and look for a colon, can save time and frustration and being confused, as to where did it find this
mysterious label. It took me sometime to figure this out and it was just a simple comment that turned out
to be interpreted as a label. Yeah and hours later your hair is being pulled out and the gnashing of the teeth.   :P

GEOS Macro's

On your original Geo Programmer disk are:
GEOS Macro's these are hard to explain but they tell the assembler how to assemble etc, mainly the main Macro is:

geosMac these are the "main" Geos macros that tells the assembler what GEOS
routines are being used in any application code. These are needed in any application so the
assembler knows how to "recognize" All GEOS routines. Without this the assembler has no clue
and will give you lots of errors, WHY? Simple it does not yet know about the GEOS main routines
and is totally dumb without them. In my own experience it took me awhile to think about this and
then by looking at others code I finally figured this out, very simple but one does not think about this,
but this simple macro is what is needed at the start of any application.

NOTE: in all Geos Macros and routines you will see all caps, mixed caps, small caps in routines,
GEOS/Wheels routines are formatted in this manner so it is very important the programmer type all
macro files, and routines the exact way you see those routines, if not this will either create errors or hard to
find bugs in your code, or whatever your coding wont work at all.

NOTE: GEOS/wheels code uses these routines and is its own unique routines, therefore standard ML code
will not work, because not starting code with GEOS specific routines. At times you can use HEX values
but you need to start with GEOS routines or the rest will not work etc. (to my knowledge).

Starting new routines:

it is always best to format and start new routines with the following code:

```
.noglbl
.noeqin
.if   Pass1
```

```
.include    geosSym ;Also a necessary standard GEOS macro
.include    geosMac
.endif
.eqin
.glbl
```

Starting app code:

The programmer simply starts with the apps first label which gets the app ready for coding, so
you can start with something like:

```
;*******************************
;    My application Main
;    Coded by: "your name"
;    Date
;    others authors code names
;*******************************

ProgStart:
```

Setting up screen and menu

```
;setup graphical screen

            LoadB     dispBufferOn,#(ST_WR_FORE|ST_WR_BACK)
            LoadW     r0,#ClearScreen         ;clear screen
            jsr       GraphicsString

            jsr       CkVersion               ;Checks for Wheels
Ver.4.0: If not exit if not this version
        LoadW         r0,#MenuTable           ;start of menu
                      lda    #0
            jsr       DoMenu                  ;bring up menu window

            rts                               ;return to GEOS main loop

ClrScrn:
```

```
              .byte NEWPATTERN,5              ;GEOS main screen pattern
(choose your own)
              .byte MOVEPENTO                 ;move pen to draw screen
              .word        0                  ;position where pen is
moved to
              .byte        0
              .byte RECTANGLETO               ;draw screen REGTANGLE
              .word        319                ;screen size: Horizontal
and Vertical
              .byte        199
              .byte        NULL

;check Wheels version number:
CKVersion:
              lda          $c00f              ;Wheels kernal version =
$C00F
              cmp          #$41               ;is this kernal version 4.1
or higher?
              bcc          90$                ;branch if not this version!
              lda          $904f              ;Wheels driver version =
904F

              cmp          #$52               ;are the driver's 5.2 or
higher?
              bcc          90$                ;branch if not this version!
              rts                             ;this version of Wheels will
work, now return to main loop.

90$
              LoadW        r0,#badBox         ;not a good box, hes a bad
boy!  :-P
              jsr          DoDlgBox           ;tell user they need Wheels
OS only
              jmp          EnterDesktop       ;exit back to Desktop
wrong version. :-(
                                              ;Time to update your
version of Wheels.
```

```
badBox:
            .byte       DEF_DB_POS              ;good example of coding a
Dialogue box
            .byte       DBTXTSTR,TXT_LN_X,TXT_LN_2_Y     ;Geesh
            .word       bad1Txt
            .byte       DBTXTSTR,TXT_LN_X,TXT_LN_3_Y     ;more
coordinates I guess
            .word       bad2Txt
            .byte       DBSYSOPV                ;exit DB with a Mouse
Click: a clickie little mouse. :P
            .byte       0

bad1Txt:
            .byte       BOLDON,"This software requires",0
bad2Txt:
            .byte       BOLDON,"Wheel's V4.2 or higher",0
;Good example for checking for Wheels version in your own Wheels
apps!
```

Now for the Menu code:

The menu here was built with the app:
Geo Beaver. This app spits out Hex code,
$0000, $009d etc. This app takes a little getting used to but helps in the designing of Menu's, Dialogue boxes, Icons etc.
www.zimmers.net (Bo Zimmerman)

The below menu is one way but other ways you can lay out GEOS menus, but coded in this manner, so experiment. (Horizontal only)...

```
MenuTable:
            .byte       $00,$0e                 ;menu top and bottom
            .word       $0000,$009d             ;menu left and right
            .byte       HORIZONTAL|$03 ;VERTICAL Menu|#items : The
vertical bar is a GEOS parameter
```

```
                                                ;find all GEOS conventions in
the GeoProgrammer manual.
          .word      MnuTxt00             ;menu text pointer
          .byte      MENU_ACTION          ;menu action
          .word      quitprg              ;pointer to EXIT application.

          .word      MnuTxt01             ;menu text pointer
          .byte      MENU_ACTION          ;menu action
          .word      DlgBox               ;pointer to--> ABOUT DB:

          .word      MnuTxt02             ;menu text pointer
          .byte      MENU_ACTION          ;menu action
          .word      OpenS                ;pointer to Open DB

MnuTxt00: .byte      27,"Exit App",0      ;Exit text parms etc
MnuTxt01: .byte      27,"About App",0     ;About text parms etc
MnuTxt02: .byte      27,"Open App",0      ;Open File text parms etc

quitprg:
          jmp        EnterDeskTop         ;Exits app back to Desktop.
                                          ;This routine is very
important, if you dont use this routine
                                          ;or forget this routine, you
are stuck, only way to exit is to
                                          ;Reset Commodore and
reboot GEOS or Wheels, so think
                                          ;about what youre doing, or
paint yourself into a corner.
```

Creating Dialogue Boxes:

```
DlgBox:
          jsr        ReDoMenu                  ;go back to first
menu
          LoadW      r0,#DlgDat                ;text pointer to
Dialogue box
          jsr        DoDlgBox                  ;start DB
          rts                                  ;return to GEOS main
loop
;Dialogue box created with GeoBeaver app
```

```
DlgDat:
             .byte      $81                        ;size of flag, and
shadow pattern
             .byte      $0b,$07,$0a
             .word      DlgT00
             .byte      $0b,$05,$14
             .word      DlgT01
             .byte      $0b,$05,$20
             .word      DlgT02
             .byte      $0b,$05,$29
             .word      DlgT03
             .byte      $01,$0b,$3a                ; OK button
             .byte      $00                        ;end of DB table

DlgT00:
             .byte      "AppName  Version: By Author",27,0
DlgT01:
             .byte      "Other programmers",27,0
DlgT02:
             .byte      "Copyright Dates",27,0
DlgT03:
             .byte      "Description on apps use",27,0
```

At this time I have not done anything with Icons so one of these days I will learn how to
do this.

For now these are the basics of programming GEOS or Wheels apps, I assume
for just 8-bit routines for GEOS or wheels.

Someone that could help Wheels programmers wishing to program for 16 bit code
email Mr Werner Weicht and I bet he could help, if you can ever get him to land for even
a day. Hee hee :-P

Enjoy this information and by no means have fun with coding for either GEOS or Wheels
in the Millenium we need more assembly language and GEOS/Wheels programmers.
Fire up GEOS or smoke your tires with Wheels and lets do some programming.

I myself would like to start up a new GEOS/Wheels either forum or email list for:

GEOS/Wheels support, using, help etc.
GEOS/Wheels programming, etc.

Any interested parties please contact me at the following places:

Address:

Terry L. Raymond
P.O. Box 173
Pavillion, WY 82523

EMAIL:
traymond20@gmail.com

I hope you enjoy this information on GEOS/Wheels programming
have a nice day.

If you have any programming questions comments feel free to email me at the above address.

Thank you.
TERRY RAYMOND