Assembly Language Programming with Zilog Developer Studio II

Before beginning a new project for ZDS II, use Windows Explorer to create a folder for the project. In this folder place a copy of the `initADL.asm` file (see below) to create an ADL mode program, or a copy of the `initZ80.asm` file to create a Z80 mode program. Also, the SEGMENT directive in each of the files in the project MUST BE REPLACED with:

```
.ASSUME ADL = 1                           .ASSUME ADL = 0
DEFINE  ADL_CODE,SPACE=RAM      or        DEFINE  Z80_CODE,SPACE=RAM
SEGMENT ADL_CODE                          SEGMENT Z80_CODE
```

Then open ZDS II and select `New Project` from the File menu. In the dialog box that appears, enter the Project Type, CPU, and Build Type:

Project Type:	`Assembly Only`
CPU:	`eZ80F92`
Build Type:	`Executable`

And then click on the Browse button `[...]` to the right of the `Project Name:` field.

A new dialog box, `Select Project Name`, will appear. Use the `Look in:` drop-down list box here to find the directory that you want the project to appear in and double-click on it. For example, I might select:

```
C:\Users\shoem\OneDrive\Agon\myProject
```

as the project directory.

Next, in the `File name:` field, enter the name for your project (it's best to pick the same name as the project directory, e.g. `myProject`). Click `Select` to return to the `New Project` dialog box. In the `New Project` dialog box, click `Continue`.

The `New Project Wizard` dialog box will now appear. Pick `eZ80F92_99C0873_Flash` as the Target Name and `Simulator` as the Debug Tool. Then click `Next>>`.

The `New Project Wizard` dialog box now changes to show the `Target Memory Configuration`. In the field labelled RAM, enter `040000-0BFFFF` for an ADL mode program or `000000-007FFF` for a Z80 mode program. Then click `Finish`.

ZDS II has now created a new project in the `..\Agon\myProject` Directory, which will contain the project file `myProject.zdsproj`. You will now be back in the main window of the ZDS Developer Studio. In the Toolbar above the main window, BE SURE TO SELECT `Release` instead of Debug.

Now go to the menu bar in the main window and select `Project-->Add Files`… to add any source files that you want into the project. It's best to have all your source files in the project directory.

Click on the '+' sign next to the `Assembly Only Project Files` at the top of the left sidebar to see the files you've added to the project. Double-clicking on any file opens it in a window for viewing/editing.

Finally, to build the project, select `Build-->Build` on the menu bar, or just press the F7 key. The output window at the bottom of the screen will show if the build succeeded. If there are errors, they will be displayed along with the line number where the error occurred. To find the line containing the error, you'll want to have line numbers appear in the open file's window. To do this, select `Tools-->Options`. In the Options dialog box select the `Editor` tab and then click on `Advanced Editor Options`. Click the check box for `Display Line Number Margin` and click `OK`.
The Hex file (and other files) produced by the build command will appear in the Releases subdirectory of the project directory.

```
;     << initADL.asm >>
;     R.L. Shoemaker  14-Feb-24
; This is the initialization code needed for programs running as MOS commands
; in 24-bit ADL mode.
      .assume ADL = 1
      .define  ADL_CODE,SPACE=RAM
      SEGMENT ADL_CODE
      .extern _main
      ORG 040000h

      JP    _start      ;jump to _start
      ALIGN 64          ;the 5-byte MOS header must be located at byte 64
      DB    "MOS"       ;flag for MOS - to confirm this is a valid MOS command
      DB    00          ;MOS header version 0
      DB    01          ;flag for run mode (0: Z80, 1: ADL)
_start:
      PUSH AF           ;save all registers
      PUSH BC
      PUSH DE
      PUSH IX
      PUSH IY           ;we must preserve IY for MOS
      CALL _main        ;execute the main C program.
_exit:
      POP   IY          ;restore all registers
      POP   IX
      POP   DE
      POP  BC
      POP   AF
      LD    HL,0000     ;load the MOS API return code ( = 0) if no errors
      RET

;============================================================================
;     << initZ80.asm >>
;     R.L. Shoemaker 15-Feb-24
; This is the initialization code needed for programs running as MOS commands
; in 16-bit Z80 mode.
; In Z80 mode, the 16-bit restart vectors RST N must be rerouted to the
; 24-bit restart vectors located at physical address 000000. This is done
; by using RST.LIS N, which causes the 16-bit address in the RST N instruction
; to be expanded to a 24-bit address using MBASE for the additional 8 bits.
      .assume ADL = 0        ;set the program to run in Z80 mode
      .define Z80_CODE,SPACE=RAM
      SEGMENT Z80_CODE
      .extern _main
      .ORG 0000         ;the restart vector table for RST0 - RST38 must
                        ;   be located starting at address 0000
```

```
     JP  _start ;jump around vectors and MOS header
     DS  5
RST_08:              ;execute a MOS command with A = command number
     RST.LIS 08h
     RET
     DS  5
RST_10:
     RST.LIS 10h     ;output a byte to the screen via the ESP2 VDU handler
     RET             ;   with A = byte to display
     DS  5
RST_18:              ;output a string to the screen via the ESP2 VDU handler
     RST.LIS 18h     ;  with HL = pointer to string, BC = number of bytes in
     RET             ;  the string or 0 if string delimited,
     DS  5           ;  A = string delimiter (commonly 00)
RST_20: DS  8
RST_28: DS  8
RST_30: DS  8
RST_38: DS  8        ;NMI interrupt vector (not currently used by AGON)

     ALIGN 64        ;the 5-byte MOS header must be located at byte 64 (40h)
     DB   "MOS"      ;signature for MOS - to confirm this is a valid MOS command
     DB   00         ;MOS header version 0
     DB   00         ;flag for run mode (0: Z80, 1: ADL)
_start:
     PUSH.LIL IY     ;save IY as a 24-bit value.
     LD   IY,0000
     ADD  IY,SP      ;also save SPS as a 24-bit value
     PUSH.LIL IY
     LD   SP,7FFFh   ;set SP to 7FFFh (the top of the MOS command user area)
     PUSH AF         ;save the rest of the registers
     PUSH.LIL BC
     PUSH.LIL DE
     PUSH.LIL IX
     CALL _main      ;execute the main program code
_exit:
     POP.LIL IX      ;restore all registers except SP and IY
     POP.LIL DE
     POP.LIL BC
     POP  AF
     POP.LIL IY      ;restore the saved SPS
     LD   SP,IY
     POP.LIL IY      ;restore IY
     LD   HL,0000    ;load the MOS API return code ( = 0 if no errors)
     RET.L           ;return to MOS
```

Next, on the menu bar in the main window open the Project Settings dialog box (`Project-->Settings`) and select `Use Existing`. Then click on the Browse button `[...]` to the right of the `Existing` field. The name of your `linkcmd` file will appear in the `Select Linker Command File` pop-up window. Double-click on the name, and the pop-up window will disappear. Click `OK` back in in the `Project Settings` window and you are done setting up your project.

To do a transfer, boot up the AgonLight. It boots into BBC BASIC, so type

```
*BYE
```

to exit into the MOS. Next, to transfer a Hex file into the Agon using hexload and have it run as an MOS command, enter

```
hexload uart1 57600 mos/filename.bin
```

and the program will sign on and wait for a file to be sent from the PC.
On the PC, open a Command Prompt and go to the directory containing the Hex file, then enter

```
python send.py filename.hex COM3 57600
```

The send.py script will show that the file is being sent from the PC, and the Agon will show that the file was received.

Before building the project, you need to edit the ReleaseADL.linkcmd file. In the last section of this file, replace myProgram with the name you want for your executable Hex file, and list all the object modules in your project. For example, if you have two files in your project, myProgram.asm and initADL.asm and want to create an executable named myProgram.hex, enter:

```
    " myProgram "= \
     ".\initADL.obj", \
     ".\ myProgram.obj", \
```

as the last part of the file. Save the linkcmd file in your project folder and rename it as myProject.linkcmd.

For C programs, move a copy of the Release.linkcmd file into your project directory, and edit it to change the project name to the name of your project. Then open the Project Settings dialog box (Project-->Settings), select Custom for the Link Configuration and enter Release.linkcmd in the Use Existing dialog box.