Getting Started with DeZog and CSpect on Windows

> **Note**: This tutorial was written targeting DeZog v1.2.7 and CSpect V2.12.23

Prerequisites

This tutorial assumes the following:

- Visual Studio Code has been installed
- You have downloaded **CSpect** - available here.
- You are using **sjasmplus** as your assembler

Installing DeZog Components on Windows

CSpect Plugin

1. Download the CSpect plugin files from the following location: https://github.com/maziac/DeZogPlugin/releases/tag/v0.9.0

   There are two files that must be downloaded:

   - DeZogPlugin.dll
   - DeZogPlugin.dll.config

   > **Note**: It is likely your browser will complain about downloading a dll - override the warning and download as usual.

2. Open the location where you downloaded the files (typically you **Downloads** folder) and **right-click** the **DeZogPlugin.dll** to show the context menu and select **Properties**.

3. In the **Properties** dialog, find the **Security** section at the bottom of the dialog and select the **Unblock** checkbox, then click **OK**.

4. Repeat the previous two steps for the **DeZogPlugin.dll.config** file.

5. Copy the **DeZogPlugin.dll** and **DeZogPlugin.dll.config** files to the same location as the **CSpect.exe**.

Installing the DeZog Visual Studio Code Extension

1. Download the DeZog extension **dezog-1.2.7.vsix** from the following location: https://github.com/maziac/DeZog/releases/tag/v1.2.7

2. Open Visual Studio Code and view the command palette (**F1** or **Ctrl+Shift+P**) and enter **Extensions: Install from VSIX...**.

   The **Install from VSIX** dialog will open - navigate to the downloaded **dezog-1.2.7.vsix** file, select it, and click **Install**.

   > **Note**: You may be prompted to reload Visual Studio Code - do so.

You have now installed the required components. The next step is to configure a project to use DeZog.

Configuring a Project

> **Note**: A number of these configuration options depend on how you have laid out your project and the sub-folders. In this tutorial I will describe my environment and how that influences the options.

The configuration of my environment is inspired by the blog post Creating a Z80 Assembly Development Environment on Windows on the Spectrum Next web site.

My project layout is as follows:

- **MyCoolApp**
  - **.vscode** - contains my tasks.json and launch.json file
  - **project**
    - **bin** - contains **CSpect.exe**, **hdfmonkey.exe**, **sjasmpluse.exe** as well as the **DeZogPlugin.dll** and **DeZogPlugin.dll.config** files.
    - **data** - binary data files I include in my app
    - **etc** - nothing at all 😃
    - **src** - all of the **asm** files and the **lst** file created during compilation.
  - **sdcard** - contains the **cspect-next-2gb.img** SD card image and the **enNextZX.rom** and **enNxtmmc.rom** rom files used with CSpect.

Configuring DeZog

In order to configure Visual Studio Code to launch DeZog when you want to debug, you need to add a launch configuration to a **launch.json** file.

1. In the Visual Studio Code explorer pane, locate the **.vscode** folder in the root of your workspace.

   > **Note**: If it doesn't exist, just add the **.vscode** folder yourself.

2. If the **launch.json** file exists, open it, otherwise add a new file to the **.vscode** folder and name it **launch.json**.

3. The **DeZog** Author has documented the configuration to be added here. However, the entry I am using is shown below:

```
{
    "configurations": [
        {
            "type": "dezog",
            "request": "launch",
            "name": "DeZog",
            "remoteType": "cspect",
            "zsim": {
                "loadZxRom": true
            },
            "listFiles": [
                {
                    "path": "${workspaceRoot}\\project\\src\\Project.lst",
                    "asm": "sjasmplus",
                    "mainFile":
```

```
"${workspaceRoot}\\project\\src\\Project.asm"
                },

            ],
            "startAutomatically": false,
            "skipInterrupt": false,
            "history": {
                "reverseDebugInstructionCount": 10000,
                "codeCoverageEnabled": true
            },
            "commandsAfterLaunch": [
                //"-sprites",
                //"-patterns"
            ],
            "disassemblerArgs": {
                "esxdosRst": true
            },
            "rootFolder": "${workspaceFolder}",
            "topOfStack": "stack_top",
            "load": "project.nex",
            "smallValuesMaximum": 513,
            "tmpDir": ".tmp"
        }
    ]
}
```

Most descriptions taken from the Author's documentation - I note where I made a change:

- **type**: Specifies the dezog extension.
- **name**: The (human readable) name of DeZog as it appears in vscode.
- **remoteType**: Specifies the **CSpect** emulator **<== Changed**
- **listFiles**: Note I have specified the full path to the **Project.lst** and **Project.asm** files using `${workspaceRoot}` macro and also note the use of `\\` to escape the backslashes. **<== Changed**
- **startAutomatically**: If true the program is started directly after loading. If false the program stops after launch. (Default=true). Please note: If this is set to true and a .tap file is loaded it will stop at address 0x0000 as this is where ZEsarUX tape load emulation starts.
- **skipInterrupt**: Is passed to ZEsarUX at the start of the debug session. If true ZEsarUX does not break in interrupts (on manual break)
- **reverseDebugInstructionCount**: The number of lines you can step back during reverse debug. Use 0 to disable.
- **codeCoverageEnabled**: If enabled (default) code coverage information is displayed. I.e. allsource codes lines that have been executed are highlighted in green. You can clear the code coverage display with the command palette "dezog: Clear current code coverage decoration".

- **commandsAfterLaunch**: Here you can enter commands that are executed right after the launch and connection of the debugger. These commands are the same as you can enter in the debug console. E.g. you can use "-sprites" to show all sprites in case of a ZX Next program. See Debug Console.
- **disassemblerArgs**: Arguments that can be passed to the internal disassembler. At the moment the only option is "esxdosRst". If enabled the disassembler will disassemble "RST 8; defb N" correctly.
- **rootFolder**: Typically = workspaceFolder. All other file paths are relative to this path.
- **topOfStack**: This is an important parameter to make the callstack display convenient to use. Please add here the label of the top of the stack. Without this information DeZog does not know where the stack ends and may show useless/misleading/wrong information. In order to use this correctly first you need a label that indicates the top of your stack. Here is an example how this may look like:

  Your assembler file:

  ```
  stack_bottom:
      defs    STACK_SIZE*2, 0
  stack_top:
  ```

  In your launch.json:

  ```
  "topOfStack": "stack_top"
  ```

  > **Note**: topOfStack: instead of a label you can also use a fixed number.

- **load**: The .nex, .sna (or .tap) file to load. **<== Changed**
- **smallValuesMaximum**: DeZog format numbers (labels, constants) basically in 2 ways depending on their size: 'small values' and 'big values'. Small values are typically constants like the maximum number of something you defined in your asm file. Big values are typically addresses. Here you can give the boundary between these 2 groups. bigValues usually also show their contents, i.e. the value at the address along the address itself. Usually 512 is a good boundary value.
- **tmpDir**: A temporary directory used for files created during the debugging. At the moment this is only used to create the file for the disassembly if the PC reaches areas without any associated assembler listing.

4. Once the **launch.json** has been configured, you are nearly ready to debug!

Debugging a Project

> **Assumptions**:
>
> - My main assembly file is **Project.asm**.

> - The listing file **Project.lst** is generated in the same **src** folder as the **Project.asm** file.
> - I output **project.nex** and **project.map** to the root of my workspace.

In order to start debugging a session, I found the following:

1. Your app must be compiled and generate a listing - I have a Visual Studio Task configured that uses the following **sjasmplus** command-line:

   ```
   "command": "${workspaceRoot}\\project\\bin\\sjasmplus ${file} --zxnext=cspect --msg=all --fullpath --lst",
   ```

   > **Note**: I use the **Z80 Macro-Assembler** extension which provides syntax highlighting and problem matchers for sjasmplus and other assemblers - the `--fullpath` switch causes any compilation errors to include the full path to the source file, enabling the vscode problem list to show the errors and navigate directly to the file/line when clicked.

2. The output of the compilation (I use the **nex** format) must be copied to the SD Card Image:

   ```
   "command": "${workspaceRoot}\\project\\bin\\hdfmonkey put ${workspaceRoot}\\sdcard\\cspect-next-2gb.img project.nex",
   ```

3. CSpect must be already running so that DeZog can connect to it. I use the `break; nop; nop` pattern as the first line of my assembly code and then launch CSpect with the following command:

   ```
   "command": "${workspaceRoot}\\project\\bin\\CSpect.exe -w4 -r -s14 -tv -esc -brk -basickeys -zxnext -nextrom -map=${workspaceRoot}\\project.map -mmc=${workspaceRoot}\\sdcard\\cspect-next-2gb.img project.nex",
   ```

4. Press **F5** to launch the DeZog debugger and connect to CSpect.

Example Tasks Configuration

In my projects I have 3 tasks:

- **Compile Assembly** - mapped to the **Ctrl+Shift+B** shortcut, this compiles the currently focused file with sjasmplus
- **Update SDCard** - copies the ouput (**project.nex**) to the SD Card
- **Launch CSpect** - runs **Compile Assembly** and **Update SDCard** then launches CSpect in break mode

Here is my **tasks.json** in case anyone is interested:

```
{
    // See https://go.microsoft.com/fwlink/?LinkId=733558
    // for the documentation about the tasks.json format
    "version": "2.0.0",
```

```
    "tasks": [

        {
            "label": "Compile Assembly",
            "type": "shell",
            "command": "${workspaceRoot}\\project\\bin\\sjasmplus ${file} --
zxnext=cspect --msg=all --fullpath --lst",
            "group": {
                "kind": "build",
                "isDefault": true
            },
            "presentation": {
                "reveal": "always",
            },
            "problemMatcher": [
                "$errmatcher-sjasmplus"
            ]
        },
        {
            "label": "Update SDCard",
            "type": "shell",
            "command": "${workspaceRoot}\\project\\bin\\hdfmonkey put
${workspaceRoot}\\sdcard\\cspect-next-2gb.img project.nex",
            "presentation": {
                "reveal": "always",
            },
            "problemMatcher": [
                "$errmatcher-sjasmplus"
            ]
        },
        {
            "label": "Launch CSpect",
            "type": "shell",
            "command": "${workspaceRoot}\\project\\bin\\CSpect.exe -w4 -r -s14 -tv
-esc -brk -basickeys -zxnext -nextrom -map=${workspaceRoot}\\project.map -
mmc=${workspaceRoot}\\sdcard\\cspect-next-2gb.img project.nex",
            "dependsOrder": "sequence",
            "dependsOn": [
                "Compile Assembly",
                "Update SDCard"
            ],
            "problemMatcher": []
        }
    ]
}
```

References

- vscode vsix file: https://github.com/maziac/DeZog/releases/tag/v1.2.7
- CSpect plugin: https://github.com/maziac/DeZogPlugin/releases/tag/v0.9.0
- Usage documentation: https://github.com/maziac/DeZog/blob/develop/documentation/Usage.md