

## [Tutorial] Making ASM File.

---

### [Tutorial] Making ASM File.

by **Shadow Player** on 28 Jan 2009 23:18

**[align=center]ASM File Making Tutorial[/align]**

#### 1.- The Patch:

```
;W580 SW-R8BE001
;Play a sound when opening/closing slide.
;In tpa/preset/system/sound put files slideropen.mp3 and sliderclose.mp3
;v 1.3
;Corrected detection of isAudioPlayerBook
;(c) John North
;(e) HierOS (correct mismatch)

452add9c: 26D0 24D0
452adde8: 2720CCE02820CAE0 004B18474109C045

45c00940: 0000000000000000 10B4114B98470028
45c00948: 0000 11D1
;45c00948: 0000 C046
45c0094A: 0000000000000000 1048114B9847
45c00950: 00000000000000000000000000000000 00280CD10E480F4B9847002807D10EA1
45c00960: 000000000000000000 3F2C00D014A10648
45c00968: 00 32
45c00969: 0000000000000000 22064B984710BC
45c00970: 00000000000000000000000000000000 2720402C00D12820004B184787DF2A45
45c00980: 00000000000000000000000000000000 FC9C7145354800450555FD44554EF344
45c00990: 00000000000000000000000000000000 09AD0F4599DC2A4573006C0069006400
45c009a0: 00000000000000000000000000000000 650072006F00700065006E002E006D00
45c009b0: 00000000000000000000000000000000 700033000000FFFF73006C0069006400
45c009c0: 00000000000000000000000000000000 6500720063006C006F00730065002E00
45c009d0: 000000000000000000 6D00700033000000
```

- **The comments.** Every comment in a patch must be preceded by a semicolon. You can write comments in any part of the patch.

- **The hooks and code modifications.** These are replacements done inside the firmware's code body, that means that it replaces bytes in actual functions that our phones use.

- **The Patch Body.** This is new code, written at the end of our phone's firmware, in an empty zone of space.

[hr][hr]

#### 2.- Understanding:

- **The comments** do not have any functionality in the patch more than let us know who made the patch, what it does and any other additional info possibly needed.

- **The hooks and code modifications** change existing firmware code to modify a piece

of code for the patch-maker's need. Sometimes the patch-maker needs to add more code to a patch (Add another function to an existing one), this new additional code then will be written in the empty space at the end of the firmware. That new chunk of code is called **The Patch Body**. To jump to the patch body the patch-maker has to write a so called 'hook' somewhere in the firmware's code that basically redirects some function's execution to the patch body, executes it and then returns to the original function, all this, without breaking the routine's logical order.

- **The Patch Body** is an additional piece of code that patch-makers write at the end of the firmware, in an empty space, that adds some function to the firmware's code. To jump to this piece of code we need a 'hook' that jumps execution from inside the firmware's code to this new piece of additional code. After the patch body, or some part of it, is executed; execution obviously has to jump back to the firmware's original code.

[hr][hr]

### 3.- Assembly Basics:

An important thing to understand is the co-relation between an instruction and the byte space that it uses.

Every byte in our firmwares has a specific meaning and reason of existence, they are classified in **Data** and **Code**. The logical order of execution tells the processor what bytes to understand as code, and whose as data.  
Every code instruction in our firmware uses space (in bytes).

ARM Processors (The ones we use) have two 'Execution Modes' (You don't need to understand that term): 16-bit (called THUMB) and 32-bit (called ARM) execution. What this means is that some (the most) code of our firmware is written in 16-bits (2 bytes) instructions and some other code in 32-bit (4 bytes) instructions. We will not touch the ARM Execution Mode, since we barely use it, and will continue with THUMB Execution Mode.

In THUMB (16-bit) Execution Mode every Assembly Instruction uses 2 bytes of space, so per each 2 bytes of THUMB code, the CPU executes an instruction.

[hr][hr]

### 4.- Interpretation:

What we do in our VKP Patches is to replace some of these bytes to change a function of our phone. All values are written in Hex.

452add9c: 26D0 24D0

This peace of our patch, replaces 26D0 for 24D0 at the offset (address/location) 452ADD9C in our firmware.

26D0 are 2 bytes, and if we disassemble them, they make an instruction:

Code: [Select all](#)

ROM:452ADD9C 26 D0                      BEQ              loc\_452ADDEC

ROM:452ADD9C -> The offset

26D0 -> The bytes  
BEQ -> The instruction  
loc\_452ADDEC -> The parameter

And when we apply the patch, 26D0 is changed to 24D0, making:

Code: [Select all](#)

```
ROM:452ADD9C 24 D0                      BEQ      loc_452ADDE8
```

The only thing that changed was the instruction's parameter.  
As you can have noticed, the parameter is obviously an offset: 452ADDE8.

[hr][hr]

#### 4.- The Process:

- Open IDA and load the corresponding firmware.
- Apply the patch using the 'Apply VKP Patch' IDC script.
- Menu > Options > General > 'Number of opcode bytes:' set to 4. (These are the bytes corresponding to each instruction, displayed on the left).
- Jump to the first offset of the patch: 452add9c. (Hotkey: G)
- Disassemble (Hotkey: C)
- Select the line corresponding to the bytes shown in the patch:

```
452add9c: 26D0 24D0
```

This:

Code: [Select all](#)

```
ROM:452ADD9C                      ;  
-----  
ROM:452ADD9C 24 D0                      BEQ      loc_452ADDE8
```

- Press Alt + F10 to create your first .asm file. Save it somewhere.  
(or go to File -> Produce file -> Create ASM file, and give a name to the \*.asm file)

You will get this in your file:

Code: [Select all](#)

```
;  
; +-----+  
; |   This file is generated by The Interactive Disassembler (IDA)   |  
; |   Copyright (c) 2007 by DataRescue sa/nv, <ida@datarescue.com>   |  
; |   Licensed to: Mach EDV Dienstleistungen, Jan Mach, 1 user, adv, 11/2007 |  
; +-----+  
;  
;  
;  
; -----
```

```
BEQ    loc_452ADDE8
```

- Clean it out:

Code: [Select all](#)

```
BEQ    0x452ADDE8
```

- And add this:

Code: [Select all](#)

```
org 0x452ADD9C
BEQ    0x452ADDE8
```

- Jump to the second offset: 452adde8.
- Look in the patch what lines to select.

```
452adde8: 2720CCE02820CAE0 004B18474109C045
```

- Disassemble if it's not disassembled already.
- Select all this:

Code: [Select all](#)

```
ROM:452ADDE8          loc_452ADDE8                      ; CODE XREF:
ROM:452ADD9Cj
ROM:452ADDE8 00 4B          LDR    R3, off_452ADDEC
ROM:452ADDEA 18 47          BX      R3
ROM:452ADDEA          ;
-----
ROM:452ADDEC 41 09 C0 45 off_452ADDEC    DCD    loc_45C00940+1    ; DATA XREF:
ROM:loc_452ADDE8r
```

- Press Alt + F10 and make your second assembly file.  
(or go to File -> Produce file -> Create ASM file, and give a name to the \*.asm file)
- Clean it out and add the same as before, it should now look like this:

Code: [Select all](#)

```
org 0x452ADDE8
    LDR    R3, off_452ADDEC
    BX     R3

off_452ADDEC    DCD    0x45C00940+1
```

- Jump to the third offset: 45c00940

Note: This offset is at the end of the firmware, in the empty zone, it's the beginning of the Patch Body.

Note2: All offsets after this one are in the patch body, you can jump to any of them and note that they are all inside this function that John North (The Author) wrote.

- Disassemble the whole patch body if it's not already disassembled.
- Select the whole Patch Body and make your .asm (Alt + F10).

All this:

Code: [Select all](#)

```

ROM:45C00940          loc_45C00940                      ; CODE XREF:
ROM:452ADDEAj
ROM:45C00940                      ; DATA XREF:
ROM:off_452ADDEC0
ROM:45C00940 10 B4          PUSH    {R4}
ROM:45C00942 11 4B          LDR     R3, off_45C00988
ROM:45C00944 98 47          BLX     R3
ROM:45C00946 00 28          CMP     R0, #0
ROM:45C00948 C0 46          NOP
ROM:45C0094A 10 48          LDR     R0, off_45C0098C
ROM:45C0094C 11 4B          LDR     R3, off_45C00994
ROM:45C0094E 98 47          BLX     R3
ROM:45C00950 00 28          CMP     R0, #0
ROM:45C00952 0C D1          BNE     loc_45C0096E
ROM:45C00954 0E 48          LDR     R0, off_45C00990
ROM:45C00956 0F 4B          LDR     R3, off_45C00994
ROM:45C00958 98 47          BLX     R3
ROM:45C0095A 00 28          CMP     R0, #0
ROM:45C0095C 07 D1          BNE     loc_45C0096E
ROM:45C0095E 0E A1          ADR     R1, aSlideropen_mp3 ;
"slideropen.mp3"
ROM:45C00960 3F 2C          CMP     R4, #0x3F
ROM:45C00962 00 D0          BEQ     loc_45C00966
ROM:45C00964 14 A1          ADR     R1, aSliderclose_mp ;
"sliderclose.mp3"
ROM:45C00966
ROM:45C00966          loc_45C00966                      ; CODE XREF:
ROM:45C00962j
ROM:45C00966 06 48          LDR     R0, off_45C00980
ROM:45C00968 32 22          MOVS    R2, #0x32
ROM:45C0096A 06 4B          LDR     R3, off_45C00984
ROM:45C0096C 98 47          BLX     R3
ROM:45C0096E
ROM:45C0096E          loc_45C0096E                      ; CODE XREF:
ROM:45C00952j
ROM:45C0096E                      ; ROM:45C0095Cj
ROM:45C0096E 10 BC          POP     {R4}
ROM:45C00970 27 20          MOVS    R0, #0x27
ROM:45C00972 40 2C          CMP     R4, #0x40
ROM:45C00974 00 D1          BNE     loc_45C00978
ROM:45C00976 28 20          MOVS    R0, #0x28
ROM:45C00978
ROM:45C00978          loc_45C00978                      ; CODE XREF:
ROM:45C00974j
ROM:45C00978 00 4B          LDR     R3, off_45C0097C
ROM:45C0097A 18 47          BX      R3
ROM:45C0097A          ;
-----
ROM:45C0097C 87 DF 2A 45 off_45C0097C  DCD loc_452ADF86+1      ; DATA XREF:
ROM:loc_45C00978r

```

```

ROM:45C00980 FC 9C 71 45 off_45C00980 DCD aTpaPresetSyste ; DATA XREF:
ROM:loc_45C00966r
ROM:45C00980 ;
"/tpa/preset/system/sound"
ROM:45C00984 35 48 00 45 off_45C00984 DCD loc_45004834+1 ; DATA XREF:
ROM:45C0096Ar
ROM:45C00988 05 55 FD 44 off_45C00988 DCD loc_44FD5504+1 ; DATA XREF:
ROM:45C00942r
ROM:45C0098C 55 4E F3 44 off_45C0098C DCD unk_44F34E55 ; DATA XREF:
ROM:45C0094Ar
ROM:45C00990 09 AD 0F 45 off_45C00990 DCD unk_450FAD09 ; DATA XREF:
ROM:45C00954r
ROM:45C00994 99 DC 2A 45 off_45C00994 DCD loc_452ADC98+1 ; DATA XREF:
ROM:45C0094Cr
ROM:45C00994 ; ROM:45C00956r
ROM:45C00998 73 00 6C 00+aSlideropen_mp3 unicode 0, <slideropen.mp3>,0
ROM:45C00998 69 00 64 00+ ; DATA XREF:
ROM:45C0095Eo
ROM:45C009B6 FF DCB 0xFF
ROM:45C009B7 FF DCB 0xFF
ROM:45C009B8 73 00 6C 00+aSliderclose_mp unicode 0, <sliderclose.mp3>,0
ROM:45C009B8 69 00 64 00+ ; DATA XREF:
ROM:45C00964o

```

In the asm file should look like this:

Code: [Select all](#)

```

org 0x45C00940
loc_45C00940:
    PUSH    {R4}
    LDR     R3, off_45C00988
    BLX     R3
    CMP     R0, 0
    NOP
    LDR     R0, off_45C0098C
    LDR     R3, off_45C00994
    BLX     R3
    CMP     R0, 0
    BNE     loc_45C0096E
    LDR     R0, off_45C00990
    LDR     R3, off_45C00994
    BLX     R3
    CMP     R0, 0
    BNE     loc_45C0096E
    ADR     R1, aSlideropen_mp3
    CMP     R4, 0x3F
    BEQ     loc_45C00966
    ADR     R1, aSliderclose_mp

loc_45C00966:
    LDR     R0, off_45C00980
    MOVS    R2, 0x32
    LDR     R3, off_45C00984
    BLX     R3

loc_45C0096E:

```

```

        POP     {R4}
        MOVS    R0, 0x27
        CMP     R4, 0x40
        BNE     loc_45C00978
        MOVS    R0, 0x28

loc_45C00978:
        LDR     R3, off_45C0097C
        BX      R3

align 4

off_45C0097C    DCD 0x452ADF86+1
off_45C00980    DCD 0x45719CFC
off_45C00984    DCD 0x45004834+1
off_45C00988    DCD 0x44FD5504+1
off_45C0098C    DCD 0x44F34E55
off_45C00990    DCD 0x450FAD09
off_45C00994    DCD 0x452ADC98+1

aSlideropen_mp3    du "slideropen.mp3",0

align 4

aSliderclose_mp    du "sliderclose.mp3",0

```

Note: Add 'align 4' before every data block or string.

- Join all asm files you made and add 'include "x.inc"' at the beginning:

Code: [Select all](#)

```

include "x.inc"

org 0x452ADD9C
        BEQ     0x452ADDE8

org 0x452ADDE8
        LDR     R3, off_452ADDEC
        BX      R3

off_452ADDEC    DCD 0x45C00940+1

org 0x45C00940
loc_45C00940:
        PUSH    {R4}
        LDR     R3, off_45C00988
        BLX     R3
        CMP     R0, 0
        NOP
        LDR     R0, off_45C0098C
        LDR     R3, off_45C00994
        BLX     R3
        CMP     R0, 0
        BNE     loc_45C0096E

```

```

        LDR    R0, off_45C00990
        LDR    R3, off_45C00994
        BLX    R3
        CMP    R0, 0
        BNE    loc_45C0096E
        ADR    R1, aSlideropen_mp3
        CMP    R4, 0x3F
        BEQ    loc_45C00966
        ADR    R1, aSliderclose_mp

loc_45C00966:
        LDR    R0, off_45C00980
        MOVS    R2, 0x32
        LDR    R3, off_45C00984
        BLX    R3

loc_45C0096E:
        POP     {R4}
        MOVS    R0, 0x27
        CMP    R4, 0x40
        BNE    loc_45C00978
        MOVS    R0, 0x28

loc_45C00978:
        LDR    R3, off_45C0097C
        BX     R3

align 4

off_45C0097C    DCD 0x452ADF86+1
off_45C00980    DCD 0x45719CFC
off_45C00984    DCD 0x45004834+1
off_45C00988    DCD 0x44FD5504+1
off_45C0098C    DCD 0x44F34E55
off_45C00990    DCD 0x450FAD09
off_45C00994    DCD 0x452ADC98+1

aSlideropen_mp3    du "slideropen.mp3",0

align 4

aSliderclose_mp    du "sliderclose.mp3",0

```

- Organize like this, adding the EQUs:

Code: [Select all](#)

```

include "x.inc"

branch    equ    0x452ADDE8
hook      equ    0x45C00940+1
address1  equ    0x452ADF86+1
address2  equ    0x45719CFC
address3  equ    0x45004834+1
address4  equ    0x44FD5504+1
address5  equ    0x44F34E55
address6  equ    0x450FAD09

```



```

address7    equ    0x452ADC98+1

org 0x452ADD9C
    BEQ    branch

org 0x452ADDE8
    LDR    R3, off_452ADDEC
    BX     R3

off_452ADDEC    DCD hook

org 0x45C00940
loc_45C00940:
    PUSH    {R4}
    LDR     R3, off_45C00988
    BLX     R3
    CMP     R0, 0
    NOP
    LDR     R0, off_45C0098C
    LDR     R3, off_45C00994
    BLX     R3
    CMP     R0, 0
    BNE     loc_45C0096E
    LDR     R0, off_45C00990
    LDR     R3, off_45C00994
    BLX     R3
    CMP     R0, 0
    BNE     loc_45C0096E
    ADR     R1, aSlideropen_mp3
    CMP     R4, 0x3F
    BEQ     loc_45C00966
    ADR     R1, aSliderclose_mp

loc_45C00966:
    LDR     R0, off_45C00980
    MOVS     R2, 0x32
    LDR     R3, off_45C00984
    BLX     R3

loc_45C0096E:
    POP     {R4}
    MOVS     R0, 0x27
    CMP     R4, 0x40
    BNE     loc_45C00978
    MOVS     R0, 0x28

loc_45C00978:
    LDR     R3, off_45C0097C
    BX     R3

align 4

off_45C0097C    DCD address1
off_45C00980    DCD address2
off_45C00984    DCD address3
off_45C00988    DCD address4
off_45C0098C    DCD address5

```

```
off_45C00990    DCD address6
off_45C00994    DCD address7

aSlideropen_mp3    du "slideropen.mp3",0

align 4

aSliderclose_mp    du "sliderclose.mp3",0
```

Note: You can choose any name for your EQUs.

[hr][hr]

This tutorial comes to it's end here because the next step is to port the offsets you organized at the beginning (The ones after 'equ').

**For Porting and Compiling ASM files,**

**[url=http://www.se-developers.net/showthread.php?tid=297]click here[/url]**

I hope this tutorial helps anybody interested in learning and porting patches for his/her phone.

**(c) Shadow Player**

<http://www.se-developers.net/>