


Decode firmware (dmcu.img file)

 Administrator: C:\Windows\system32\cmd.exe

```
C:\MCU>mtcdmcutool.exe -d dmcu.img decoded.img
```

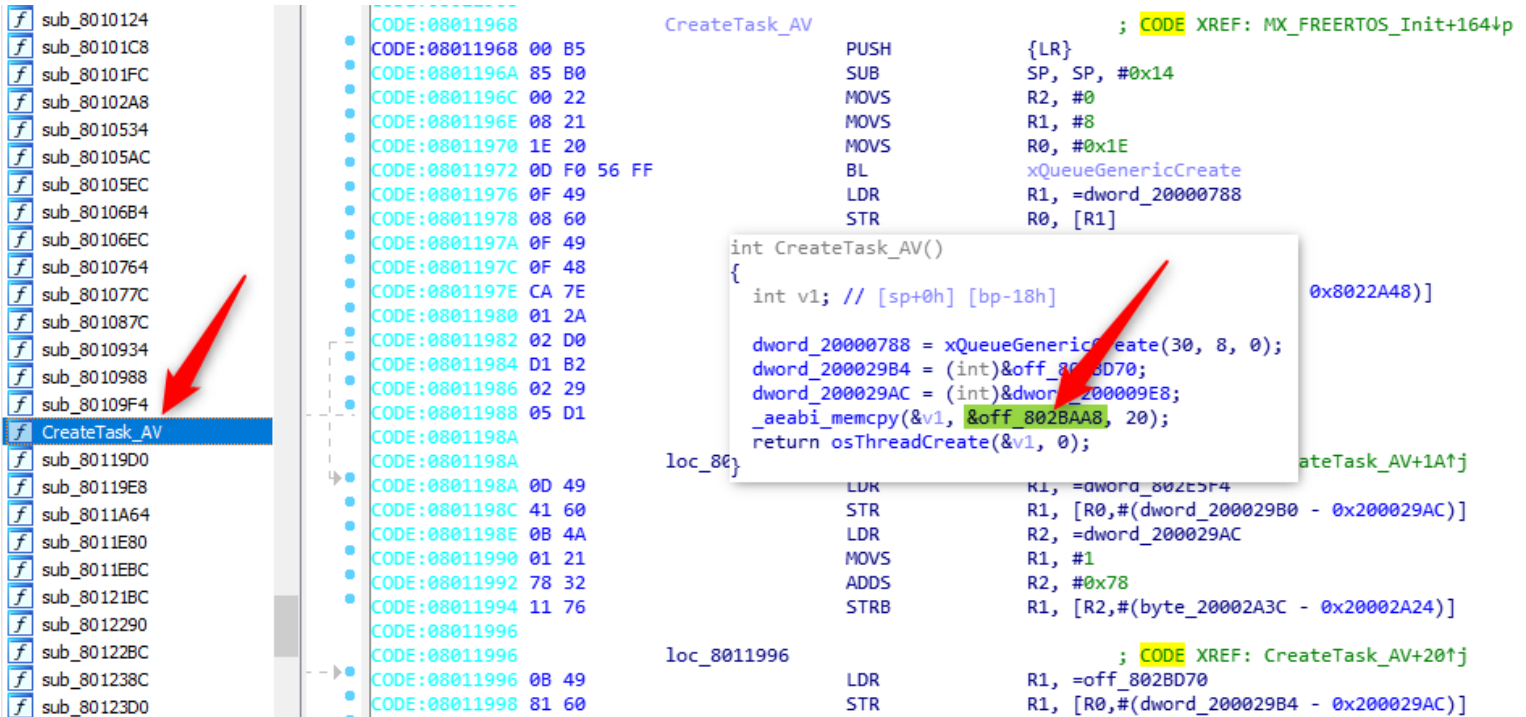
```
mtcdmcutool: a tool to decode/encode/info MTCD MCU firmware images for RK3xxx/PX3/PX5 headunits.  
v.1.2, (c) 2018, Wadzio
```

```
Done!
```

```
C:\MCU>
```

Load decoded file into IDA Pro (Use useful additions files from [XDA Thread](#))

Find subroutine *CreateTask_AV* and go to offset (marked below, green & red arrow)




```
sub_8010124
sub_80101C8
sub_80101FC
sub_80102A8
sub_8010534
sub_80105AC
sub_80105EC
sub_80106B4
sub_80106EC
sub_8010764
sub_801077C
sub_801087C
sub_8010934
sub_8010988
sub_80109F4
CreateTask_AV
sub_80119D0
sub_80119E8
sub_8011A64
sub_8011E80
sub_8011EBC
sub_80121BC
sub_8012290
sub_80122BC
sub_801238C
sub_80123D0
```

```
CODE:08011968      CreateTask_AV      ; CODE XREF: MX_FREERTOS_Init+1644p
CODE:08011968 00 B5      PUSH      {LR}
CODE:0801196A 85 B0      SUB       SP, SP, #0x14
CODE:0801196C 00 22      MOVS      R2, #0
CODE:0801196E 08 21      MOVS      R1, #8
CODE:08011970 1E 20      MOVS      R0, #0x1E
CODE:08011972 0D F0 56 FF  BL      xQueueGenericCreate
CODE:08011976 0F 49      LDR       R1, =dword_20000788
CODE:08011978 08 60      STR       R0, [R1]
CODE:0801197A 0F 49
CODE:0801197C 0F 48
CODE:0801197E CA 7E
CODE:08011980 01 2A
CODE:08011982 02 D0
CODE:08011984 D1 B2
CODE:08011986 02 29
CODE:08011988 05 D1
CODE:0801198A      loc_801198A      int CreateTask_AV()
CODE:0801198A      {
CODE:0801198A      int v1; // [sp+0h] [bp-18h] 0x8022A48]]
CODE:0801198A      dword_20000788 = xQueueGenericCreate(30, 8, 0);
CODE:0801198A      dword_200029B4 = (int)&off_802BD70;
CODE:0801198A      dword_200029AC = (int)&dword_200009E8;
CODE:0801198A      _aeabi_memcpy(&v1, &off_802BAA8, 20);
CODE:0801198A      return osThreadCreate(&v1, 0);
CODE:0801198A      }
CODE:0801198A 0D 49      LDR       R1, =dword_802E5F4
CODE:0801198C 41 60      STR       R1, [R0, # (dword_200029B0 - 0x200029AC)]
CODE:0801198E 0B 4A      LDR       R2, =dword_200029AC
CODE:08011990 01 21      MOVS      R1, #1
CODE:08011992 78 32      ADDS      R2, #0x78
CODE:08011994 11 76      STRB      R1, [R2, # (byte_20002A3C - 0x20002A24)]
CODE:08011996      loc_8011996      ; CODE XREF: CreateTask_AV+20↑j
CODE:08011996 0B 49      LDR       R1, =off_802BD70
CODE:08011998 81 60      STR       R1, [R0, # (dword_200029B4 - 0x200029AC)]
```

If you cannot find *CreateTask_AV* go to [APPENDIX](#) at the end of this guide

This will be “our entry point”. Go to offset marked below (red arrow)

CODE:0802BA88 01 0D 0E 00+	DCD 0xD0D010E, 0xE0E00, 0xE0E
CODE:0802BAA8 EC B1 03 08 off_802BAA8	DCD aAvTask ; DATA XREF: CreateTask_AV+38↑o
CODE:0802BAA8	; CODE:off_80119CC↑o
CODE:0802BAA8	; "AV_TASK"
CODE:0802BAAC 69 0A 01 08	DCD loc_8010A68+1
CODE:0802BAB0 01 00 00 00+	DCD 1, 0
CODE:0802BAB8 C0 00 00 00	DCD 0xC0
CODE:0802BABC 01 02 03 05+dword_802BABC	DCD 0x5030201, 0x29282006, 0x302C2B2A, 0x51474441, 0x755754
CODE:0802BABC 06 20 28 29+	; DATA XREF: sub_8004F64+1C↑o
CODE:0802BABC 2A 2B 2C 30+	; CODE:off_8004FB0↑o
CODE:0802BAD0 00 00 00 00 dword_802BAD0	DCD 0 ; DATA XREF: CODE:08010C18↑o
CODE:0802BAD0	; CODE:off_8010E8C↑o
CODE:0802BAD4 5D FC 00 08	DCD sub_800FC5C+1
CODE:0802BAD8 21 FB 00 08	DCD sub_800FB20+1
CODE:0802BADC 0D FA 00 08	DCD sub_800FA0C+1
CODE:0802BAE0 25 FD 00 08	DCD sub_800FD24+1
CODE:0802BAE4 75 FB 00 08	DCD sub_800FB74+1
CODE:0802BAE8 71 F9 00 08	DCD sub_800F970+1
CODE:0802BAEC 71 FB 00 08	DCD locret_800FB70+1
CODE:0802BAF0 00 00 00 00	DCD 0
CODE:0802BAF4 02 01 02 03+dword_802BAF4	DCD 0x3020102, 0x39040003, 0x100A7006, 0x20101B0F, 0x40126211
CODE:0802BAF4 03 00 04 39+	; DATA XREF: sub_8017060:loc_80170C2↑o



Make proc and decompile

CODE:08010A68	sub_8010A68		; DATA XREF: CODE:0802BAAC↓o
CODE:08010A68			
CODE:08010A68	var_58	void sub_8010A68()	
CODE:08010A68	var_57	{	
CODE:08010A68	var_50	unsigned int v0; // r6	
CODE:08010A68	var_4F	char *v1; // r1	
CODE:08010A68	var_4E	int v2; // r0	
CODE:08010A68	var_4D	int v3; // r0	
CODE:08010A68	var_4C	int v4; // r0	
CODE:08010A68	var_44	int v5; // r2	
CODE:08010A68	var_40	void (__fastcall *v6)(signed int); // r3	
CODE:08010A68	var_28	signed int v7; // r0	
CODE:08010A68	var_24	int v8; // r0	
CODE:08010A68	var_20	int v9; // r0	
CODE:08010A68	var_1C	signed int v10; // r0	
CODE:08010A68	var_18	signed int v11; // r0	
CODE:08010A68	var_14	char v12; // r0	
CODE:08010A68	var_10	signed int v13; // r2	
CODE:08010A68	var_C	char *v14; // r0	
CODE:08010A68	var_8	unsigned __int8 *v15; // r1	
CODE:08010A68		char v16; // r0	
CODE:08010A68 96 B0		char v17; // r0	
CODE:08010A6A 32 20		char *v18; // r1	
CODE:08010A6C 0A F0 46 F8		char v19; // r0	
CODE:08010A70 F7 49		int v20; // r0	
CODE:08010A72 0A 20		int v21; // r0	
CODE:08010A74 C8 71		int v22; // r0	x20002A24)]
CODE:08010A76 00 22		int v23; // r0	
CODE:08010A78 08 72		signed int v24; // r1	x20002A24)]
CODE:08010A7A 48 72		_BOOL1 v25; // cf	x20002A24)]
CODE:08010A7C 8A 71		char *v26; // r2	x20002A24)]
CODE:08010A7E 0C 46		const char *v27; // r1	
CODE:08010A80 4A 71		STRB	R2, [R1, #(byte_20002A29 - 0x20002A24)]
CODE:08010A82 58 3C		SUBS	R4, #0x58
CODE:08010A84 62 76		STRB	R2, [R4, #(byte_200029E5 - 0x200029CC)]
CODE:08010A86 16 46		MOV	R6, R2
CODE:08010A88			
CODE:08010A88	loc_8010A88		; CODE XREF: sub_8010A68+32↓j
CODE:08010A88 F1 49		LDR	R1, =byte_20002A24

Find characteristic points (yellow) and important subroutine (red arrow)

```
    if ( !sub_800DD94() )
    {
        SendDbgStrToArmTask("audio dsp adau1701 found %d", v71);
        dword_200029B0 = (int)&off_802E5D0;
        byte_20002A3C = 3;
        goto LABEL_30;
    }
    osDelay(50);
    ++v71;
}
while ( v71 < 0x14u );
if ( !byte_20002A3C )
{
    SendDbgStrToArmTask("audio bd37534");
    dword_200029B0 = (int)&dword_802BAD0;
    byte_20002A3C = 2;
}
}
LABEL_30:
sub_80105EC();
if ( (unsigned __int8)byte_200029DA << 28 )
{
    byte_200029D3 = byte_200029DA;
    byte_200029D4 = 50;
}
if ( (void **)dword_200029B0 == &off_802E5D0 )
{
    byte_20002A2B = 10;
    byte_20002A2C = 10;
    byte_20002A2D = 10;
}
v2 = sub_8012908();
sub_80139D8(v2);
if ( (void **)dword_200029B0 == &off_802E5D0 )
{
    (*(void (__fastcall **)(signed int))(dword_200029B0 + 28))(230);
    (*(void (__fastcall **)(signed int))(dword_200029B0 + 28))(238);
    (*(void (__fastcall **)(signed int))(dword_200029B0 + 28))(190);
    (*(void (__fastcall **)(signed int))(dword_200029B0 + 28))(226);
    (*(void (__fastcall **)(signed int))(dword_200029B0 + 28))(227);
    (*(void (__fastcall **)(signed int))(dword_200029B0 + 28))(228);
    (*(void (__fastcall **)(signed int))(dword_200029B0 + 28))(235);
}
else
{
    v3 = sub_8010534();
    v4 = sub_801287C(v3);
    v5 = (unsigned __int8)v79[22];
    if ( v4 )
    {
        v6 = *(void (__fastcall **)(signed int))(dword_200029B0 + 24);
    }
}
```

Function must be changed from left to right

```
int sub_8010534()
{
    signed int v0; // r1
    signed int v1; // r0
    int (__fastcall *v2)(signed int, signed int); // r4

    if ( (void **)dword_200029B0 == &off_802E5D0 )
        return (*(int (__fastcall **)(signed int))(dword_200029B0 + 28))(190);
    if ( sub_80128DC() )
    {
        v2 = *(int (__fastcall **)(signed int, signed int))(dword_200029B0 + 20);
        v0 = 10;
        v1 = 10;
    }
    else
    {
        v0 = (unsigned __int8)byte_20002A2C;
        v1 = (unsigned __int8)byte_20002A2B;
        v2 = *(int (__fastcall **)(signed int, signed int))(dword_200029B0 + 20);
    }
    return v2(v1, v0);
}
```



```
int sub_8010534()
{
    signed int v0; // r1
    signed int v1; // r0
    void (__fastcall *v2)(signed int, signed int); // r4

    if ( sub_80128DC(dword_200029B0) )
    {
        v2 = *(void (__fastcall **)(signed int, signed int))(dword_200029B0 + 20);
        v0 = 10;
        v1 = 10;
    }
    else
    {
        v0 = (unsigned __int8)byte_20002A2C;
        v1 = (unsigned __int8)byte_20002A2B;
        v2 = *(void (__fastcall **)(signed int, signed int))(dword_200029B0 + 20);
    }
    v2(v1, v0);
    return (*(int (__fastcall **)(signed int))(dword_200029B0 + 28))(190);
}
```

Patching

CODE:08010534 ; sub_8010534

CODE:08010534

CODE:08010534

CODE:08010534 var_18 = -0x18

CODE:08010534 F8 B5 PUSH {R3-R7,LR}

CODE:08010536 1A 4F LDR R7, =dword_2000

CODE:08010538 1A 49 LDR R1, =off_802E5D

CODE:0801053A 3D 46 MOV R5, R7

CODE:0801053C 78 68 LDR R0, [R7, #dword

CODE:0801053E 1A 4E LDR R6, =byte_8022A

CODE:08010540 60 35 ADDS R5, #0x60

CODE:08010542 88 42 CMP R0, R1

CODE:08010544 1E D0 BEQ loc_8010584

CODE:08010546 02 F0 C9 F9 BL sub_80128DC

CODE:0801054A 2C 46 MOV R4, R5

CODE:0801054C 18 34 ADDS R4, #0x18

CODE:0801054E 00 28 CMP R0, #0

CODE:08010550 05 D1 BNE loc_801055E

CODE:08010552 B0 79 LDRB R0, [R6, #byte

CODE:08010554 00 28 CMP R0, #0

CODE:08010556 0B D0 BEQ loc_8010570

CODE:08010558 A8 7D LDRB R0, [R5, #byte

CODE:0801055A 00 28 CMP R0, #0

CODE:0801055C 0B D0 BEQ loc_8010570

CODE:0801055E loc_801055E

CODE:0801055E 60 79 LDRB R0, [R4, #byte_20002A29 - 0x20002A24]

CODE:08010560 00 90 STR R0, [SP, #0x18+var_18]

CODE:08010562 70 60 LDR R0, [R7, #dword_200029B0 - 0x200029AC]

KEYPATCH:: Patcher

Endian: Little Endian

Address: CODE:08010544

Original: BEQ loc_8010584

Encode: 1E D0

Size: 2

Assembly: MOV R0, R0

Fixup: MOV R0, R0

Encode: 00 46

Size: 2

☒ Save original instructions in IDA comment

Patch Cancel

CODE:08010534 ; sub_8010534

CODE:08010534

CODE:08010534

CODE:08010534 var_18 = -0x18

CODE:08010534 F8 B5 PUSH {R3-R7,LR}

CODE:08010536 1A 4F LDR R7, =dword_200029AC

CODE:08010538 1A 49 LDR R1, =off_802E5D0

CODE:0801053A 3D 46 MOV R5, R7

CODE:0801053C 78 68 LDR R0, [R7, #dword_200029B0 - 0x200029AC]

CODE:0801053E 1A 4E LDR R6, =byte_8022A68

CODE:08010540 60 35 ADDS R5, #0x60

CODE:08010542 88 42 CMP R0, R1

CODE:08010544 00 46 MOV R0, R0 ; Keypatch modified this from: ; BEQ loc_8010584

CODE:08010546 02 F0 C9 F9 BL sub_80128DC

CODE:0801054A 2C 46 MOV R4, R5

CODE:0801054C 18 34 ADDS R4, #0x18

CODE:0801054E 00 28 CMP R0, #0

CODE:08010550 05 D1 BNE loc_801055E

CODE:08010552 B0 79 LDRB R0, [R6, #byte_8022A6E - 0x8022A68]

CODE:08010554 00 28 CMP R0, #0

CODE:08010556 0B D0 BEQ loc_8010570

CODE:08010558 A8 7D LDRB R0, [R5, #byte_20002A22 - 0x20002A0C]

CODE:0801055A 00 28 CMP R0, #0

CODE:0801055C 0B D0 BEQ loc_8010570

CODE:0801055E loc_801055E

CODE:0801055E 60 79 LDRB R0, [R4, #byte_20002A29 - 0x20002A24]

CODE:08010560 00 90 STR R0, [SP, #0x18+var_18]

CODE:08010562 70 60 LDR R0, [R7, #dword_200029B0 - 0x200029AC]

```

CODE:08010570
CODE:08010570
CODE:08010570
CODE:08010570 60 79
CODE:08010572 00 90
CODE:08010574 A3 79
CODE:08010576 62 7A
CODE:08010578 21 7A
CODE:0801057A E0 79
CODE:0801057C 7C 68
CODE:0801057E 64 69
CODE:08010580
CODE:08010580
CODE:08010580 A0 47
CODE:08010582 F8 BD
CODE:08010584
CODE:08010584 A9 7D
CODE:08010586 00 29
CODE:08010588 05 D0
CODE:0801058A B1 79
CODE:0801058C 00 29
CODE:0801058E 02 D0
CODE:08010590 C1 69
CODE:08010592 BF 20
CODE:08010594 01 E0
CODE:08010596
CODE:08010596
CODE:08010596
CODE:08010596 C1 69

```

KEYPATCH:: Patcher

Endian Little Endian

Address CODE:08010582

Original POP {R3-R7,PC}

- Encode F8 BD

- Size 2

Assembly LDR R0,[R7,#4]

- Fixup LDR R0,[R7,#4]

- Encode 78 68

- Size 2

☒ Save original instructions in IDA comment

Patch

Cancel

```

CODE:08010570
CODE:08010570
CODE:08010570
CODE:08010570 60 79
CODE:08010572 00 90
CODE:08010574 A3 79
CODE:08010576 62 7A
CODE:08010578 21 7A
CODE:0801057A E0 79
CODE:0801057C 7C 68
CODE:0801057E 64 69
CODE:08010580
CODE:08010580
CODE:08010580 A0 47
CODE:08010582 78 68
CODE:08010584
CODE:08010584 A9 7D
CODE:08010586 00 29
CODE:08010588 05 D0
CODE:0801058A B1 79
CODE:0801058C 00 29
CODE:0801058E 02 D0
CODE:08010590 C1 69
CODE:08010592 BF 20
CODE:08010594 01 E0
CODE:08010596
CODE:08010596
CODE:08010596
CODE:08010596 C1 69

```

```

loc_8010570
; CODE XREF: sub_8010534+22↑j
; KEYPATCH:: Patcher
LDRB R0, [R4,#(
STR R0, [SP,#0
LDRB R3, [R4,#(
LDRB R2, [R4,#(
LDRB R1, [R4,#(
LDRB R0, [R4,#(
LDR R4, [R7,#(
LDR R4, [R4,#0
loc_8010580
;
BLX R4
POP {R3-R7,PC}
; -----
LDRB R1, [R5,#(
CMP R1, #0
BEQ loc_801059
LDRB R1, [R6,#(
CMP R1, #0
BEQ loc_801059
LDR R1, [R0,#0
MOVSW R0, #0xBF
B loc_801059
loc_8010596
; CODE XREF: sub_8010534+54↑j
; sub_8010534+5A↑j
LDR R1, [R0,#0x1C]

```

KEYPATCH:: Patcher

Endian Little Endian

Address CODE:08010582

Original POP {R3-R7,PC}

- Encode F8 BD

- Size 2

Assembly LDR R0,[R7,#4]

- Fixup LDR R0,[R7,#4]

- Encode 78 68

- Size 2

☒ Save original instructions in IDA comment

Patch

Cancel

```

CODE:08010570
CODE:08010570
CODE:08010570
CODE:08010570 60 79
CODE:08010572 00 90
CODE:08010574 A3 79
CODE:08010576 62 7A
CODE:08010578 21 7A
CODE:0801057A E0 79
CODE:0801057C 7C 68
CODE:0801057E 64 69
CODE:08010580
CODE:08010580
CODE:08010580 A0 47
CODE:08010582 78 68
CODE:08010584
CODE:08010584 A9 7D
CODE:08010586 00 29
CODE:08010588 05 D0
CODE:0801058A B1 79
CODE:0801058C 00 29
CODE:0801058E 02 D0
CODE:08010590 C1 69
CODE:08010592 BF 20
CODE:08010594 01 E0
CODE:08010596
CODE:08010596
CODE:08010596
CODE:08010596 C1 69

```

```

loc_8010570
; CODE XREF: sub_8010534+22↑j
; sub_8010534+28↑j
LDRB R0, [R4,#(byte_20002A29 - 0x20002A24)]
STR R0, [SP,#0x18+var_18]
LDRB R3, [R4,#(byte_20002A2A - 0x20002A24)]
LDRB R2, [R4,#(byte_20002A2D - 0x20002A24)]
LDRB R1, [R4,#(byte_20002A2C - 0x20002A24)]
LDRB R0, [R4,#(byte_20002A2B - 0x20002A24)]
LDR R4, [R7,#(dword_200029B0 - 0x200029AC)]
LDR R4, [R4,#0x14]
loc_8010580
; CODE XREF: sub_8010534+3A↑j
BLX R4
LDR R0, [R7,#(dword_200029B0 - 0x200029AC)] ; Key:
; POP {R3-R7,PC}
LDRB R1, [R5,#(byte_20002A22 - 0x20002A0C)]
CMP R1, #0
BEQ loc_8010596
LDRB R1, [R6,#(byte_8022A6E - 0x8022A68)]
CMP R1, #0
BEQ loc_8010596
LDR R1, [R0,#0x1C]
MOVSW R0, #0xBF
B loc_801059A
loc_8010596
; CODE XREF: sub_8010534+54↑j
; sub_8010534+5A↑j
LDR R1, [R0,#0x1C]

```

KEYPATCH:: Patcher

Endian Little Endian

Address CODE:08010582

Original POP {R3-R7,PC}

- Encode F8 BD

- Size 2

Assembly LDR R0,[R7,#4]

- Fixup LDR R0,[R7,#4]

- Encode 78 68

- Size 2

☒ Save original instructions in IDA comment

Patch

Cancel

```

CODE:08010570
CODE:08010570
CODE:08010570
CODE:08010570 60 79
CODE:08010572 00 90
CODE:08010574 A3 79
CODE:08010576 62 7A
CODE:08010578 21 7A
CODE:0801057A E0 79
CODE:0801057C 7C 68
CODE:0801057E 64 69
CODE:08010580
CODE:08010580
CODE:08010580 A0 47
CODE:08010582 78 68
CODE:08010584
CODE:08010584 A9 7D
CODE:08010586 00 29
CODE:08010588 05 D0
CODE:0801058A B1 79
CODE:0801058C 00 29
CODE:0801058E 02 D0
CODE:08010590 C1 69
CODE:08010592 BF 20
CODE:08010594 01 E0
CODE:08010596
CODE:08010596
CODE:08010596
CODE:08010596 C1 69

```


Go back to “our entry point” and go to next subroutine

CODE:0802BA88	01 0D 0E 00+	DCD 0xD0D010E, 0xE0E00, 0xE0E	
CODE:0802BAA8	EC B1 03 08 off_802BAA8	DCD aAvTask	; DATA XREF: CreateTask_AV+38↑o
CODE:0802BAA8			; CODE:off_80119CC↑o
CODE:0802BAA8			; "AV_TASK"
CODE:0802BAAC	69 0A 01 08	DCD loc_8010A68+1	
CODE:0802BAB0	01 00 00 00+	DCD 1, 0	
CODE:0802BAB8	C0 00 00 00	DCD 0xC0	
CODE:0802BABC	01 02 03 05+dword_802BABC	DCD 0x5030201, 0x29282006, 0x302C2B2A, 0x51474441, 0x755754	
CODE:0802BABC	06 20 28 29+		; DATA XREF: sub_8004F64+1C↑o
CODE:0802BABC	2A 2B 2C 30+		; CODE:off_8004FB0↑o
CODE:0802BAD0	00 00 00 00 dword_802BAD0	DCD 0	; DATA XREF: CODE:08010C18↑o
CODE:0802BAD0			; CODE:off_8010E8C↑o
CODE:0802BAD4	5D FC 00 08	DCD sub_800FC5C+1	
CODE:0802BAD8	21 FB 00 08	DCD sub_800FB20+1	
CODE:0802BADC	0D FA 00 08	DCD sub_800FA0C+1	
CODE:0802BAE0	25 FD 00 08	DCD sub_800FD24+1	
CODE:0802BAE4	75 FB 00 08	DCD sub_800FB74+1	
CODE:0802BAE8	71 F9 00 08	DCD sub_800F970+1	
CODE:0802BAEC	71 FB 00 08	DCD locret_800FB70+1	
CODE:0802BAF0	00 00 00 00	DCD 0	
CODE:0802BAF4	02 01 02 03+dword_802BAF4	DCD 0x3020102, 0x39040003, 0x100A7006, 0x20101B0F, 0x40126211	
CODE:0802BAF4	03 00 04 39+		; DATA XREF: sub_8017060:loc_80170C2↑o

Which must to be changed from left one to right one

```
int __fastcall sub_800FB74(unsigned int a1, unsigned int a2)
{
    unsigned __int8 v5; // r0
    int v6; // r7
    unsigned __int8 v7; // r0
    int v8; // r5
    unsigned __int8 v9; // r0
    int v10; // r4
    int v11; // r5
    unsigned __int8 v12; // r1
    char v14; // [sp+10h] [bp-18h]

    v14 = a4;
    if ( a1 < 0xA )
        v5 = 20 - (signed int)(12 * (10 - a1)) / 10;
    else
        v5 = (signed int)(12 * (a1 - 10)) / 10 + 20;
    v6 = v5;
    if ( a2 < 0xA )
        v7 = 20 - (signed int)(12 * (10 - a2)) / 10;
    else
        v7 = (signed int)(12 * (a2 - 10)) / 10 + 20;
    v8 = v7;
    if ( a3 < 0xA )
        v9 = 20 - (signed int)(12 * (10 - a3)) / 10;
    else
        v9 = (signed int)(12 * (a3 - 10)) / 10 + 20;
    v10 = byte_2000083E[v8];
    v11 = byte_2000083E[v9];
    sub_8004F64(15, byte_2000083E[v6]);
    sub_8004F64(16, v10);
    sub_8004F64(17, v11);
    byte_20000834 = v14;
    sub_8019FA4();
    v12 = -123;
    if ( a5 <= 0x14 )
        v12 = -123 - a5;
    byte_2000083D = v12;
    return sub_8004F64(10, v12);
}
```

```
int __fastcall sub_800FB74(unsigned int a1, unsigned int a2,
{
    int v5; // ST10_4
    int v6; // r4
    int v7; // r5
    int v8; // r2
    unsigned int v9; // r3

    v5 = a4;
    v6 = byte_2000083E[a2];
    v7 = byte_2000083E[a3];
    sub_8004F64(15, byte_2000083E[a1]);
    sub_8004F64(16, v6);
    sub_8004F64(17, v7);
    byte_20000834 = v5;
    sub_8019FA4(v5, (unsigned int)&byte_20000834, v8, v9);
    byte_2000083D = -128 - a5;
    return sub_8004F64(10, (unsigned __int8)(-128 - a5));
}
```


Patching

```
CODE:0800FB74      sub_800FB74      ; DATA XREF: CODE:0802BAE4+0
CODE:0800FB74
CODE:0800FB74      var_18      = -0x18
CODE:0800FB74      arg_0      = 0
CODE:0800FB74
CODE:0800FB74      FF B5      PUSH      {R0-R7,LR}
CODE:0800FB76      0A 26      MOVNS    R6, #0xA
CODE:0800FB78      81 B0      SUB      SP, SP, #4
CODE:0800FB7A      14 46      MOV      R4, R2
CODE:0800FB7C      0D 46      MOV      R5, R1
CODE:0800FB7E      0A 28      CMP      R0, #0xA
CODE:0800FB80      07 D3      BCC      loc_800FB92
CODE:0800FB82      0C 21      MOVNS    R1, #0xC
CODE:0800FB84      0A 38      SUBS     R0, #0xA
CODE:0800FB86      48 43      MULS     R0, R1
CODE:0800FB88      0A 21      MOVNS    R1, #0xA
CODE:0800FB8A      F4 F7 EF FA BL      __aeabi_idivmod
CODE:0800FB8E      14 30      ADDS     R0, #0x14
CODE:0800FB90      07 E0      B        loc_800FBA2
CODE:0800FB92      loc_800FB92      ; CODE XREF: sub_800FB74+C↑j
CODE:0800FB92      30 1A      SUBS     R0, R6, R0
CODE:0800FB94      0C 21      MOVNS    R1, #0xC
CODE:0800FB96      48 43      MULS     R0, R1
CODE:0800FB98      0A 21      MOVNS    R1, #0xA
CODE:0800FB9A      F4 F7 E7 FA BL      __aeabi_idivmod
CODE:0800FB9E      14 21      MOVNS    R1, #0x14
CODE:0800FBA0      08 1A      SUBS     R0, R1, R0
CODE:0800FBA2      loc_800FBA2      ; CODE XREF: sub_800FB74+1C↑j
CODE:0800FBA2
```

```
CODE:0800FC2E      0A F0 B9 F9      BL      sub_8019FA4
CODE:0800FC32      08 48      LDR      R0, =byte_2000083E
CODE:0800FC34      0A 9A      LDR      R2, [SP,#0x28+arg_0]
CODE:0800FC36      85 21      MOVNS    R1, #0x85
CODE:0800FC38      40 1F      SUBS     R0, R0, #5
CODE:0800FC3A      14 2A      CMP      R2, #0x14
CODE:0800FC3C      00 D8      BHI      loc_800FC40
CODE:0800FC3E      89 1A      SUBS     R1, R1, R2
CODE:0800FC40      loc_800FC40      ; CODE XREF: sub_800FB74+C8↑j
CODE:0800FC40      01 71      STRB     R1, [R0,#(byte_2000083D - 0x20000839)]
CODE:0800FC42      C9 B2      UXTB     R1, R1
CODE:0800FC44      0A 20      MOVNS    R0, #0xA
CODE:0800FC46      F5 F7 8D F9      BL      sub_8004F64
CODE:0800FC4A      05 B0      ADD      SP, SP, #0x14
CODE:0800FC4C      F0 BD      POP      {R4-R7,PC}
CODE:0800FC4C      ; End of function sub_800FB74
-----
```

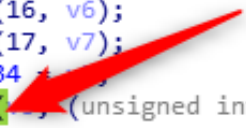
```
CODE:0800FB74      sub_800FB74      ; DATA XREF: CODE:0802BAE4+0
CODE:0800FB74
CODE:0800FB74      var_18      = -0x18
CODE:0800FB74      arg_0      = 0
CODE:0800FB74
CODE:0800FB74      FF B5      PUSH      {R0-R7,LR}
CODE:0800FB76      07 46      MOV      R7, R0 ; Keypatch modified this from:
CODE:0800FB78      81 B0      SUB      SP, SP, #4 ; MOVNS R6, #0xA
CODE:0800FB7A      13 46      MOV      R3, R2 ; Keypatch modified this from:
CODE:0800FB7C      0D 46      MOV      R5, R1 ; MOV R4, R2
CODE:0800FB7E      0A 28      CMP      R0, #0xA
CODE:0800FB80      38 E0      B        loc_800FBF4 ; Keypatch modified this from:
CODE:0800FB82      ; BCC loc_800FB92
CODE:0800FB82      0C 21      MOVNS    R1, #0xC
CODE:0800FB84      0A 38      SUBS     R0, #0xA
CODE:0800FB86      48 43      MULS     R0, R1
CODE:0800FB88      0A 21      MOVNS    R1, #0xA
CODE:0800FB8A      F4 F7 EF FA BL      __aeabi_idivmod
CODE:0800FB8E      14 30      ADDS     R0, #0x14
CODE:0800FB90      07 E0      B        loc_800FBA2
CODE:0800FB92      30 1A      SUBS     R0, R6, R0
CODE:0800FB94      0C 21      MOVNS    R1, #0xC
CODE:0800FB96      48 43      MULS     R0, R1
CODE:0800FB98      0A 21      MOVNS    R1, #0xA
CODE:0800FB9A      F4 F7 E7 FA BL      __aeabi_idivmod
CODE:0800FB9E      14 21      MOVNS    R1, #0x14
CODE:0800FBA0      08 1A      SUBS     R0, R1, R0
CODE:0800FBA2      loc_800FBA2      ; CODE XREF: sub_800FB74+1C↑j
CODE:0800FBA2
```

```
CODE:0800FC2E      0A F0 B9 F9      BL      sub_8019FA4
CODE:0800FC32      08 48      LDR      R0, =byte_2000083E
CODE:0800FC34      0A 9A      LDR      R2, [SP,#0x28+arg_0]
CODE:0800FC36      80 21      MOVNS    R1, #0x80 ; Keypatch modified this from:
CODE:0800FC38      40 1F      SUBS     R0, R0, #5 ; MOVNS R1, #0x85
CODE:0800FC3A      0A 46      MOV      R0, R0 ; Keypatch modified this from:
CODE:0800FC3C      00 46      MOV      R0, R0 ; Keypatch modified this from:
CODE:0800FC3E      89 1A      SUBS     R1, R1, R2 ; BHI loc_800FC40
CODE:0800FC40      01 71      STRB     R1, [R0,#(byte_2000083D - 0x20000839)]
CODE:0800FC42      C9 B2      UXTB     R1, R1
CODE:0800FC44      0A 20      MOVNS    R0, #0xA
CODE:0800FC46      F5 F7 8D F9      BL      sub_8004F64
CODE:0800FC4A      05 B0      ADD      SP, SP, #0x14
CODE:0800FC4C      F0 BD      POP      {R4-R7,PC}
CODE:0800FC4C      ; End of function sub_800FB74
```

In patched above subroutine go to the next subroutine

```
int __fastcall sub_800FB74(unsigned int a1, unsigned int a2, unsigned int a3, int a4, unsigned int a5)
{
    int v5; // ST10_4
    int v6; // r4
    int v7; // r5
    int v8; // r2
    unsigned int v9; // r3

    v5 = a4;
    v6 = byte_2000083E[a2];
    v7 = byte_2000083E[a3];
    sub_8004F64(15, byte_2000083E[a1]);
    sub_8004F64(16, v6);
    sub_8004F64(17, v7);
    byte_20000834 = v5;
    sub_8019FA4(-1, (unsigned int)&byte_20000834, v8, v9);
    byte_2000083D = -128 - a5;
    return sub_8004F64(10, (unsigned __int8)(-128 - a5));
}
```



Function must be changed from left to right

```
int sub_8019FA4()
{
    unsigned int v0; // r1
    unsigned int v1; // r3
    int v2; // r0
    int v3; // r1
    signed int v4; // r0

    v0 = 0;
    v1 = 55;
    if ( *((_BYTE *)((*((_DWORD *)off_20000624 + 2) + 31) )
        v1 = 50; LoudOn
    if ( byte_20000834 && (unsigned __int8)byte_20000835 != 255 )
    {
        if ( (unsigned __int8)byte_20000836 < v1 )
        {
            v0 = (((v1 - (unsigned __int8)byte_20000836) >> 31) + v1 - (unsigned __int8)byte_20000836) << 23 >> 24;
            if ( v0 > 0xE )
                v0 = 14;
        }
        v2 = (unsigned __int8)byte_20000835 - v0;
        if ( v2 < 113 )
            LOBYTE(v2) = 113;
        byte_20005855 = v2;
    }
    else
    {
        byte_20005855 = byte_20000835; Volume
        if ( !byte_20000834 )
        {
            sub_8004F64(5, (unsigned __int8)byte_20000835);
            v3 = 0;
            v4 = 18;
            return sub_8004F64(v4, v3);
        }
    }
    sub_8004F64(18, v0);
    v3 = (unsigned __int8)byte_20005855;
    v4 = 5;
    return sub_8004F64(v4, v3);
}
```



```
int __fastcall sub_8019FA4(int a1, unsigned int a2, int a3, unsigned int a4)
{
    byte_20005855 = byte_20000835;
    return sub_8004F64(5, (unsigned __int8)byte_20000835);
}
```

Patching

```
CODE:08019FA4      sub_8019FA4      ; CODE XREF: sub_800FB74+BA↑p
CODE:08019FA4      ; sub_800FD24+12E↑p
CODE:08019FA4  70 B5      PUSH      {R4-R6,LR}
CODE:08019FA6  1A 48      LDR        R0, =off_20000624
CODE:08019FA8  00 21      MOVW      R1, #0
CODE:08019FAA  00 68      LDR        R0, [R0]
CODE:08019FAC  37 23      MOVW      R3, #0x37
CODE:08019FAE  80 68      LDR        R0, [R0,#010]
CODE:08019FB0  C0 7F      LDRB       R0, [R0,#0x1F]
CODE:08019FB2  00 28      CMP        R0, #0
CODE:08019FB4  00 D0      BEQ        loc_8019FB8
CODE:08019FB6  32 23      MOVW      R3, #0x32
CODE:08019FB8
CODE:08019FB8      loc_8019FB8      ; CODE XREF: sub_8019FA4+10↑j
CODE:08019FB8  16 4A      LDR        R2, =byte_20000834
CODE:08019FBA  17 4C      LDR        R4, =byte_20005B50
CODE:08019FBC  15 78      LDRB       R5, [R2]
CODE:08019FBE  10 46      MOV        R0, R2
CODE:08019FC0  40 78      LDRB       R0, [R0,#(byte_20000835 - 0x20000834)]
CODE:08019FC2  00 2D      CMP        R5, #0
CODE:08019FC4  12 D0      BEQ        loc_8019FEC
CODE:08019FC6  FF 28      CMP        R0, #0xFF
CODE:08019FC8  10 D0      BEQ        loc_8019FEC
CODE:08019FCA  92 78      LDRB       R2, [R2,#(byte_20000836 - 0x20000834)]
CODE:08019FCC  9A 42      CMP        R2, R3
CODE:08019FCE  07 D2      BCS        loc_8019FE0
CODE:08019FD0  99 1A      SUBS       R1, R3, R2
CODE:08019FD2  CA 0F      LSRW      R2, R1, #0x1F
CODE:08019FD4  51 18      ADDS       R1, R2, R1
CODE:08019FD6  C9 05      LSLS      R1, R1, #0x17
CODE:08019FD8  09 0E      LSRW      R1, R1, #0x18
CODE:08019FDA  0E 29      CMP        R1, #0xE
CODE:08019FDC  00 D9      BLS        loc_8019FE0
CODE:08019FDE  0E 21      MOVW      R1, #0xE
```

```
CODE:08019FA4      sub_8019FA4      ; CODE XREF: sub_800FB74+BA↑p
CODE:08019FA4      ; sub_800FD24+12E↑p
CODE:08019FA4  70 B5      PUSH      {R4-R6,LR}
CODE:08019FA6  07 E0      B          loc_8019FB8 ; Keypatch modified this from:
CODE:08019FA6      ; LDR R0, =off_20000624
CODE:08019FA8      ; -----
CODE:08019FA8  00 21      MOVW      R1, #0
CODE:08019FAA  00 68      LDR        R0, [R0]
CODE:08019FAC  37 23      MOVW      R3, #0x37
CODE:08019FAE  80 68      LDR        R0, [R0,#010]
CODE:08019FB0  C0 7F      LDRB       R0, [R0,#0x1F]
CODE:08019FB2  00 28      CMP        R0, #0
CODE:08019FB4  00 D0      BEQ        loc_8019FB8
CODE:08019FB6  32 23      MOVW      R3, #0x32
CODE:08019FB8      loc_8019FB8      ; CODE XREF: sub_8019FA4+2↑j
CODE:08019FB8      ; sub_8019FA4+10↑j
CODE:08019FB8  16 4A      LDR        R2, =byte_20000834
CODE:08019FBA  17 4C      LDR        R4, =byte_20005B50
CODE:08019FBC  15 78      LDRB       R5, [R2]
CODE:08019FBE  10 46      MOV        R0, R2
CODE:08019FC0  40 78      LDRB       R0, [R0,#(byte_20000835 - 0x20000834)]
CODE:08019FC2  00 2D      CMP        R5, #0
CODE:08019FC4  12 E0      B          loc_8019FEC ; Keypatch modified this from:
CODE:08019FC4      ; BEQ loc_8019FEC
CODE:08019FC6      ; -----
CODE:08019FC6  FF 28      CMP        R0, #0xFF
CODE:08019FC8  10 D0      BEQ        loc_8019FEC
CODE:08019FCA  92 78      LDRB       R2, [R2,#(byte_20000836 - 0x20000834)]
CODE:08019FCC  9A 42      CMP        R2, R3
CODE:08019FCE  07 D2      BCS        loc_8019FE0
CODE:08019FD0  99 1A      SUBS       R1, R3, R2
CODE:08019FD2  CA 0F      LSRW      R2, R1, #0x1F
CODE:08019FD4  51 18      ADDS       R1, R2, R1
CODE:08019FD6  C9 05      LSLS      R1, R1, #0x17
CODE:08019FD8  09 0E      LSRW      R1, R1, #0x18
CODE:08019FDA  0E 29      CMP        R1, #0xE
CODE:08019FDC  00 D9      BLS        loc_8019FE0
CODE:08019FDE  0E 21      MOVW      R1, #0xE
```

```

CODE:08019FE8      loc_8019FE8      ; CODE XREF: sub_8019FA4+40↑j
CODE:08019FE8 60 71      STRB         R0, [R4,#(byte_20005B55 - 0x20005B50)]
CODE:08019FEA 02 E0      B            loc_8019FF2
CODE:08019FEC      ; -----
CODE:08019FEC      loc_8019FEC      ; CODE XREF: sub_8019FA4+20↑j
CODE:08019FEC      ; sub_8019FA4+24↑j
CODE:08019FEC 60 71      STRB         R0, [R4,#(byte_20005B55 - 0x20005B50)]
CODE:08019FEE 00 2D      CMP         R5, #0
CODE:08019FF0 05 D0      BEQ         loc_8019FFE
CODE:08019FF2      loc_8019FF2      ; CODE XREF: sub_8019FA4+46↑j
CODE:08019FF2 12 20      MOV         R0, #0x12
CODE:08019FF4 EA F7 B6 FF  BL         sub_8004F64
CODE:08019FF8 61 79      LDRB        R1, [R4,#(byte_20005B55 - 0x20005B50)]
CODE:08019FFA 05 20      MOV         R0, #5
CODE:08019FFC 05 E0      B            loc_801A00A
CODE:08019FFE      ; -----
CODE:08019FFE      loc_8019FFE      ; CODE XREF: sub_8019FA4+4C↑j
CODE:08019FFE 01 46      MOV         R1, R0
CODE:0801A000 05 20      MOV         R0, #5
CODE:0801A002 EA F7 AF FF  BL         sub_8004F64
CODE:0801A006 00 21      MOV         R1, #0
CODE:0801A008 12 20      MOV         R0, #0x12
CODE:0801A00A      loc_801A00A      ; CODE XREF: sub_8019FA4+58↑j
CODE:0801A00A EA F7 AB FF  BL         sub_8004F64
CODE:0801A00E 70 BD      POP         {R4-R6,PC}
CODE:0801A00E      ; End of function sub_8019FA4

```

```

CODE:08019FE8      loc_8019FE8      ; CODE XREF: sub_8019FA4+40↑j
CODE:08019FE8 60 71      STRB         R0, [R4,#(byte_20005B55 - 0x20005B50)]
CODE:08019FEA 02 E0      B            loc_8019FF2
CODE:08019FEC      ; -----
CODE:08019FEC      loc_8019FEC      ; CODE XREF: sub_8019FA4+20↑j
CODE:08019FEC      ; sub_8019FA4+24↑j
CODE:08019FEC 60 71      STRB         R0, [R4,#(byte_20005B55 - 0x20005B50)]
CODE:08019FEE 00 2D      CMP         R5, #0
CODE:08019FF0 05 E0      B            loc_8019FFE ; Keypatch modified this from:
CODE:08019FF0      ; BEQ loc_8019FFE
CODE:08019FF2      ; -----
CODE:08019FF2      loc_8019FF2      ; CODE XREF: sub_8019FA4+46↑j
CODE:08019FF2 12 20      MOV         R0, #0x12
CODE:08019FF4 EA F7 B6 FF  BL         sub_8004F64
CODE:08019FF8 61 79      LDRB        R1, [R4,#(byte_20005B55 - 0x20005B50)]
CODE:08019FFA 05 20      MOV         R0, #5
CODE:08019FFC 05 E0      B            loc_801A00A
CODE:08019FFE      ; -----
CODE:08019FFE      loc_8019FFE      ; CODE XREF: sub_8019FA4+4C↑j
CODE:08019FFE 01 46      MOV         R1, R0
CODE:0801A000 05 20      MOV         R0, #5
CODE:0801A002 EA F7 AF FF  BL         sub_8004F64
CODE:0801A006 00 46      MOV         R0, R0 ; Keypatch modified this from:
CODE:0801A006      ; MOV R1, #0
CODE:0801A008 00 46      MOV         R0, R0 ; Keypatch modified this from:
CODE:0801A008      ; MOV R0, #0x12
CODE:0801A00A      loc_801A00A      ; CODE XREF: sub_8019FA4+58↑j
CODE:0801A00A 00 46      MOV         R0, R0 ; Keypatch modified this from:
CODE:0801A00A      ; BL sub_8004F64
CODE:0801A00C 00 46      MOV         R0, R0 ; Keypatch modified this from:
CODE:0801A00C      ; DCB 0xAB, 0xFF
CODE:0801A00E 70 BD      POP         {R4-R6,PC}
CODE:0801A00E      ; End of function sub_8019FA4

```

Go back to “our entry point” and change sequence of bytes

```
CODE:0802BAA8 EC B1 03 08 off_802BAA8
CODE:0802BAA8
CODE:0802BAA8
CODE:0802BAAC 69 0A 01 08
CODE:0802BAB0 01 00 00 00+
CODE:0802BAB8 C0 00 00 00
CODE:0802BABC 01 02 03 05+dword_802BABC
CODE:0802BABC 06 20 28 29+
CODE:0802BABC 2A 2B 2C 30+
CODE:0802BAD0 00 00 00 00 dword_802BAD0
CODE:0802BAD0
CODE:0802BAD4 5D FC 00 08
CODE:0802BAD8 21 FB 00 08
CODE:0802BADC 0D FA 00 08
CODE:0802BAE0 25 FD 00 08
CODE:0802BAE4 75 FB 00 08
CODE:0802BAE8 71 F9 00 08
CODE:0802BAEC 71 FB 00 08
CODE:0802BAF0 00 00 00 00
CODE:0802BAF4 02 01 02 03+dword_802BAF4
```

```
DCD aAvTask ; DATA XREF: CreateTask_AV+38fo
; CODE:off_80119CCfo
; "AV_TASK"
DCD sub_8010A68+1
DCD 1, 0
DCD 0xC0
DCD 0x5030201, 0x29282006, 0x302C2B2A, 0x51474441, 0x75F3E4
DCD 0
DCD sub_800FC5C+1
DCD sub_800FB20+1
DCD sub_800FA0C+1
DCD sub_800FD24+1
DCD sub_800FB74+1
DCD sub_800F970+1
DCD locret_800FB70+1
DCD 0
DCD 0x3020102, 0x39040003, 0x100A7006, 0x20101B0F, 0x40126211
```

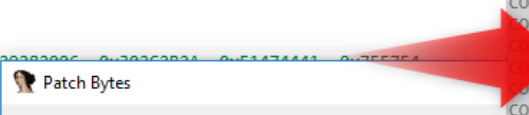
```
CODE:0802BAA8 EC B1 03 08 off_802BAA8
CODE:0802BAA8
CODE:0802BAA8
CODE:0802BAAC 69 0A 01 08
CODE:0802BAB0 01 00 00 00+
CODE:0802BAB8 C0 00 00 00
CODE:0802BABC 01 02 03 05+dword_802BABC
CODE:0802BABC 06 20 28 29+
CODE:0802BABC 2A 2B 2C 30+
CODE:0802BAD0 00 00 00 00 dword_802BAD0
CODE:0802BAD0
CODE:0802BAD4 5D FC 00 08
CODE:0802BAD8 21 FB 00 08
CODE:0802BADC 0D FA 00 08
CODE:0802BAE0 25 FD 00 08
CODE:0802BAE4 75 FB 00 08
CODE:0802BAE8 71 F9 00 08
CODE:0802BAEC 01 D0 03 08
CODE:0802BAF0 00 00 00 00
CODE:0802BAF4 02 01 02 03+dword_802BAF4
```

```
DCD aAvTask ; DATA XREF: CreateTask_AV
; CODE:off_80119CCfo
; "AV_TASK"
DCD sub_8010A68+1
DCD 1, 0
DCD 0xC0
DCD 0x5030201, 0x29282006, 0x302C2B2A, 0x51474441,
; DATA XREF: sub_8004F64+1
; CODE:off_8004FB0fo
; DATA XREF: sub_8010A68+1
; sub_8010A68:off_8010E8Cfo
DCD sub_800FC5C+1
DCD sub_800FB20+1
DCD sub_800FA0C+1
DCD sub_800FD24+1
DCD sub_800FB74+1
DCD sub_800F970+1
DCD 0x803D001
DCD 0
DCD 0x3020102, 0x39040003, 0x100A7006, 0x20101B0F,
```

Patch Bytes

Address: 0x802BAEC
File offset: 0x27AEC
Original value: 71 FB 00 08 00 00 00 02 01 02 03 00 04 39
Values: 01 D0 03 08 00 00 00 02 01 02 03 00 04 39

OK Cancel Help



Save patched file

Apply patches to input file ✕

Start EA:

End EA:

Input file: ...

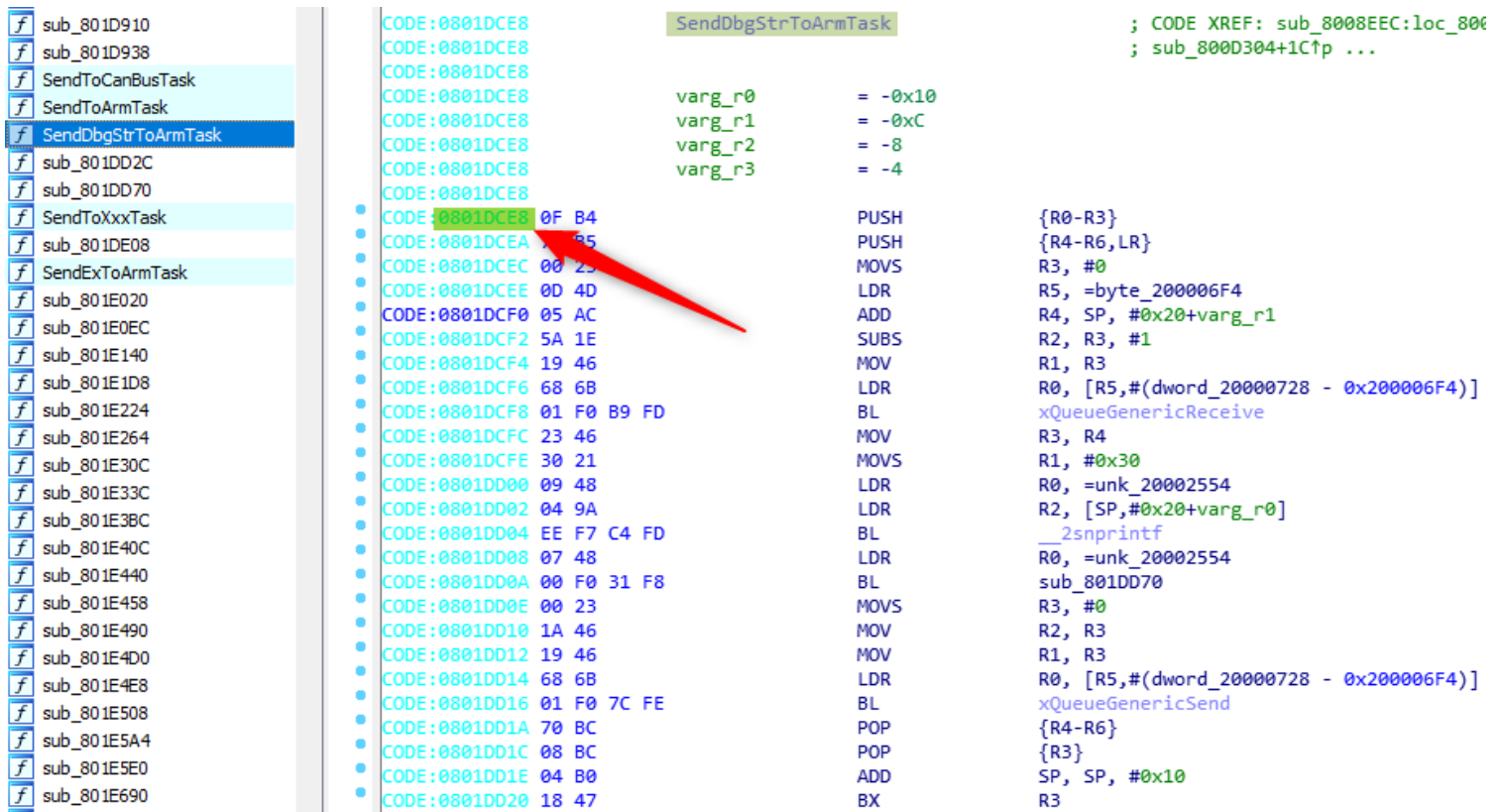
Backup file: ...

☒ Create backup
☐ Restore original bytes

OK Cancel Help

Open SoundPatched project in Atollic TrueStudio and open file *Flash.ld*


Find subroutine *SendDbgStrToArmTask*



The screenshot displays the Atollic TrueStudio interface. On the left, a list of subroutines is shown, with *SendDbgStrToArmTask* selected. The main window shows the assembly code for this subroutine, starting at address *CODE:0801DCE8*. A red arrow points to the address *0x0801DCE8* in the code list. The code includes several instructions, including *PUSH*, *MOV*, *LDR*, *BL*, and *BX*. The subroutine is labeled *SendDbgStrToArmTask* and has a cross-reference to *sub_8008EEC:loc_800* and *sub_800D304+1C↑p ...*.

```
CODE:0801DCE8      SendDbgStrToArmTask      ; CODE XREF: sub_8008EEC:loc_800
CODE:0801DCE8      ; sub_800D304+1C↑p ...
CODE:0801DCE8      varg_r0          = -0x10
CODE:0801DCE8      varg_r1          = -0xC
CODE:0801DCE8      varg_r2          = -8
CODE:0801DCE8      varg_r3          = -4
CODE:0801DCE8      CODE:0801DCE8 0F B4      PUSH      {R0-R3}
CODE:0801DCE8      CODE:0801DCE8 0F B5      PUSH      {R4-R6,LR}
CODE:0801DCE8      CODE:0801DCE8 00 23      MOV      R3, #0
CODE:0801DCE8      CODE:0801DCE8 0D 4D      LDR      R5, =byte_200006F4
CODE:0801DCE8      CODE:0801DCE8 05 AC      ADD      R4, SP, #0x20+varg_r1
CODE:0801DCE8      CODE:0801DCE8 5A 1E      SUBS     R2, R3, #1
CODE:0801DCE8      CODE:0801DCE8 19 46      MOV      R1, R3
CODE:0801DCE8      CODE:0801DCE8 68 6B      LDR      R0, [R5, #(dword_20000728 - 0x200006F4)]
CODE:0801DCE8      CODE:0801DCE8 01 F0 B9 FD      BL      xQueueGenericReceive
CODE:0801DCE8      CODE:0801DCE8 23 46      MOV      R3, R4
CODE:0801DCE8      CODE:0801DCE8 30 21      MOVS     R1, #0x30
CODE:0801DCE8      CODE:0801DCE8 09 48      LDR      R0, =unk_20002554
CODE:0801DCE8      CODE:0801DCE8 04 9A      LDR      R2, [SP, #0x20+varg_r0]
CODE:0801DCE8      CODE:0801DCE8 EE F7 C4 FD      BL      __2snprintf
CODE:0801DCE8      CODE:0801DCE8 07 48      LDR      R0, =unk_20002554
CODE:0801DCE8      CODE:0801DCE8 00 F0 31 F8      BL      sub_801DD70
CODE:0801DCE8      CODE:0801DCE8 00 23      MOVS     R3, #0
CODE:0801DCE8      CODE:0801DCE8 1A 46      MOV      R2, R3
CODE:0801DCE8      CODE:0801DCE8 19 46      MOV      R1, R3
CODE:0801DCE8      CODE:0801DCE8 01 F0 7C FE      BL      R0, [R5, #(dword_20000728 - 0x200006F4)]
CODE:0801DCE8      CODE:0801DCE8 70 BC      POP      {R4-R6}
CODE:0801DCE8      CODE:0801DCE8 08 BC      POP      {R3}
CODE:0801DCE8      CODE:0801DCE8 04 B0      ADD      SP, SP, #0x10
CODE:0801DCE8      CODE:0801DCE8 18 47      BX
```

Copy address of *SendDbgStrToArmTask* to right place into *Flash.ld*



The screenshot shows the *Flash.ld* file in Atollic TrueStudio. The file contains several memory addresses and their corresponding values. The address *0x0801DCE8* is highlighted in green, and a red arrow points to it. The file is structured as follows:

```
*Flash.ld
1 sendToBD37xx      = 0x08004F64;
2 sendToBD37xxAll   = 0x0800FCAC;
3 sendDbgStrToArmTask = 0x0801DCE8;
4
5 avDspEq           = 0x200029E6;
6 bd37xxRegisters   = 0x20005B50;
7
8 modCode           = 0x0803D000;
9 modData           = 0x20007000;
10
11
12 /*****
13 PROVIDE( sendToBD37xx      = sendToBD37xx
```


Go back to “our entry point” and go to subroutine

```

CODE:0802BA88 01 0D 0E 00+      DCD 0xD0D010E, 0xE0E00, 0xE0E
CODE:0802BAA8 EC B1 03 08 off_802BAA8 DCD aAvTask ; DATA XREF: CreateTask_AV+38fo
CODE:0802BAA8 ; CODE:off_80119CCfo
CODE:0802BAA8 ; "AV_TASK"
CODE:0802BAAC 69 0A 01 08      DCD loc_8010A68+1
CODE:0802BAB0 01 00 00 00+      DCD 1, 0
CODE:0802BAB8 C0 00 00 00      DCD 0xC0
CODE:0802BABC 01 02 03 05+dword_802BABC DCD 0x5030201, 0x29282006, 0x302C2B2A, 0x51474441, 0x755754
CODE:0802BABC 06 20 28 29+      ; DATA XREF: sub_8004F64+1Cfo
CODE:0802BABC 2A 2B 2C 30+      ; CODE:off_8004FB0fo
CODE:0802BAD0 00 00 00 00 dword_802BAD0 DCD 0 ; DATA XREF: CODE:08010C18fo
CODE:0802BAD0 ; CODE:off_8010E8Cfo
CODE:0802BAD4 5D FC 00 08      DCD sub_800FC5C+1
CODE:0802BAD8 21 FB 00 08      DCD sub_800FB20+1
CODE:0802BADC 0D FA 00 08      DCD sub_800FA0C+1
CODE:0802BAE0 25 FD 00 08      DCD sub_800FD24+1
CODE:0802BAE4 75 FE 00 08      DCD sub_800FB74+1
CODE:0802BAE8 71 F9 00 08      DCD sub_800F970+1
CODE:0802BAEC 71 FB 00 08      DCD locret_800FB70+1
CODE:0802BAF0 00 00 00 00      DCD 0
CODE:0802BAF4 02 01 02 03+dword_802BAF4 DCD 0x3020102, 0x39040003, 0x100A7006, 0x20101B0F, 0x40126211
CODE:0802BAF4 03 00 04 39+      ; DATA XREF: sub_8017060:loc_80170C2fo

```

Decompile and copy address to right place into *Flash.ld*

```

char *sub_800FC5C()
{
    char *result; // r0

    byte_20005B50[0] = -9;
    byte_20005B51 = -125;
    byte_20005B52 = -119;
    byte_20005B5B = -1;
    byte_20000835 = -1;
    byte_20005B55 = -1;
    byte_20005B5C = 0;
    byte_20005B5D = 18;
    byte_20005B5E = 32;
    byte_20005B5A = -1;
    byte_20005B62 = 0;
    sub_800FCAC();
    result = &byte_200029CC;
    byte_200029DF = 1;
    return result;
}

```

```

*Flash.ld
1 sendToBD37xx = 0x08004F64;
2 sendToBD37xxAll = 0x0800FCAC;
3 sendDbgStrToArmTas 0x0801DCE8;
4
5 avDspEq = 0x200029E6;
6 bd37xxRegisters = 0x20005B50;
7
8 modCode = 0x0803D000;
9 modData = 0x20007000;
10
11
12 /*****
13 PROVIDE( sendToBD37xx = sendToBD37xx

```

From the same function go to the next subroutine

```
char *sub_800FC5C()
{
    char *result; // r0

    byte_20005B50[0] = -9;
    byte_20005B51 = -125;
    byte_20005B52 = -119;
    byte_20005B58 = -1;
    byte_20000835 = -1;
    byte_20005B55 = -1;
    byte_20005B5C = 0;
    byte_20005B5D = 18;
    byte_20005B5E = 32;
    byte_20005B5A = -1;
    byte_20005B62 = 0;
    sub_800FCAC();
    result = &byte_200029CC;
    byte_200029DF = 1;
    return result;
}
```

Copy next address to right place into *Flash.ld*

```
int sub_800FCAC()
{
    unsigned int v0; // r4
    int result; // r0

    v0 = 0;
    do
    {
        result = sub_8004F64(v0, (unsigned __int8)byte_20005B50[v0]);
        v0 = (unsigned __int8)(v0 + 1);
    }
    while ( v0 < 0x13 );
    return result;
}
```

```
*Flash.ld
1 sendToBD37xx = 0x08004F64;
2 sendToBD37xxAll = 0x0800FCAC;
3 sendDbgStrToArmTask = 0x0801DCE8;
4
5 avDspEq = 0x200029E6;
6 bd37xxRegisters = 0x20005B50;
7
8 modCode = 0x0803D000;
9 modData = 0x20007000;
10
11
12 /*****
13 PROVIDE( sendToBD37xx = sendToBD37xx
```

From the same function copy another address to right place into *Flash.ld*

```
int sub_800FCAC()
{
    unsigned int v0; // r4
    int result; // r0

    v0 = 0;
    do
    {
        result = sub_8004F64(v0, (unsigned __int8)byte_20005B50[v0]);
        v0 = (unsigned __int8)(v0 + 1);
    }
    while ( v0 < 0x13 );
    return result;
}
```


```
*Flash.ld
1 sendToBD37xx = 0x08004F64;
2 sendToBD37xxAll = 0x0800FCAC;
3 sendDbgStrToArmTask = 0x0801DCE8;
4
5 avDspEq = 0x200029E6;
6 bd37xxRegisters = 0x20005B50;
7
8 modCode = 0x0803D000;
9 modData = 0x20007000;
10
11
12 /*****
13 PROVIDE( sendToBD37xx = sendToBD37xx
```

Go back to “our entry point” and go to subroutine

```

CODE:0802BA88 01 0D 0E 00+      DCD 0xD0D010E, 0xE0E00, 0xE0E
CODE:0802BAA8 EC B1 03 08 off_802BAA8 DCD aAvTask ; DATA XREF: CreateTask_AV+38fo
CODE:0802BAA8 ; CODE:off_80119CCfo
CODE:0802BAA8 ; "AV_TASK"
CODE:0802BAAC 69 0A 01 08      DCD loc_8010A68+1
CODE:0802BAB0 01 00 00 00+      DCD 1, 0
CODE:0802BAB8 C0 00 00 00      DCD 0xC0
CODE:0802BABC 01 02 03 05+dword_802BABC DCD 0x5030201, 0x29282006, 0x302C2B2A, 0x51474441, 0x755754
CODE:0802BABC 06 20 28 29+      ; DATA XREF: sub_8004F64+1Cfo
CODE:0802BABC 2A 2B 2C 30+      ; CODE:off_8004FB0fo
CODE:0802BAD0 00 00 00 00 dword_802BAD0 DCD 0 ; DATA XREF: CODE:08010C18fo
CODE:0802BAD0 ; CODE:off_8010E8Cfo
CODE:0802BAD4 5D FC 00 08      DCD sub_800FC5C+1
CODE:0802BAD8 21 FB 00 08      DCD sub_800FB20+1
CODE:0802BADC 0D FA 00 08      DCD sub_800FA0C+1
CODE:0802BAE0 25 FD 00 08      DCD sub_800FD24+1
CODE:0802BAE4 75 FE 00 08      DCD sub_800FB74+1
CODE:0802BAE8 71 F9 00 08      DCD sub_800F970+1
CODE:0802BAEC 71 FB 00 08      DCD locret_800FB70+1
CODE:0802BAF0 00 00 00 00      DCD 0
CODE:0802BAF4 02 01 02 03+dword_802BAF4 DCD 0x3020102, 0x39040003, 0x100A7006, 0x20101B0F, 0x40126211
CODE:0802BAF4 03 00 04 39+      ; DATA XREF: sub_8017060:loc_80170C2fo

```



Almost at the end of this function find characteristic points (yellow) and variable (red)

```

v21 = (int)&dword_200029AC;
if ( byte_200029C4 != 1 )
    goto LABEL_357;
v62 = 0;
LABEL_351:
    byte_200029C4 = v62;
    goto LABEL_356;
case 24:
    goto LABEL_356;
case 26:
    v46 = 10;
    goto LABEL_276;
case 27:
    v46 = 15;
LABEL_276:
    byte_200029E5 = v46;
    aeabi_memcpy(&v78[v75], &v76, 5);
    if ( v75 == (unsigned __int8)byte_200029E5 - 5 )
LABEL_326:
        sub_8010534();
        goto LABEL_357;
case 28:
    if ( v74 <= 3u )
        aeabi_memcpy(v88, &v75, v74);
    v47 = *(void (__fastcall **)(signed int))(dword_200029B0 + 28);
    v48 = 226;
    goto LABEL_306;

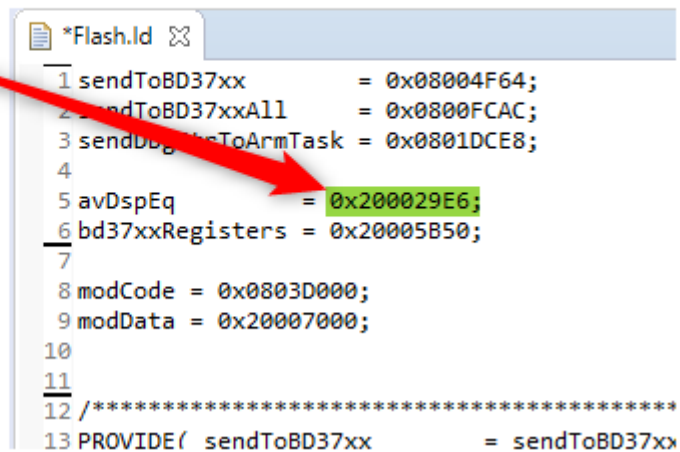
```

Go to the beginning of this function and find the assignation to this variable

```
byte_200029DB = 0;  
byte_200029C6 = 0;  
v79 = &byte_20002A0C;  
byte_20002A21 = 2;  
v78 = &byte_200029E6;  
v88 = &byte_200029F5;  
v87 = &byte_200029F8;  
v84 = &byte_200029FB;  
v83 = &byte_200029FD;  
v82 = &byte_20002A14;  
v81 = &byte_20002A09;  
v80 = &byte_200029FF;  
while ( 1 )  
{
```

Copy address to right place into *Flash.ld*

```
byte_200029DB = 0;  
byte_200029C6 = 0;  
v79 = &byte_20002A0C;  
byte_20002A21 = 2;  
v78 = &byte_200029E6;  
v88 = &byte_200029F5;  
v87 = &byte_200029F8;  
v84 = &byte_200029FB;  
v83 = &byte_200029FD;  
v82 = &byte_20002A14;  
v81 = &byte_20002A09;  
v80 = &byte_200029FF;  
while ( 1 )  
{
```



```
*Flash.ld  
1 sendToBD37xx = 0x08004F64;  
2 sendToBD37xxAll = 0x0800FCAC;  
3 sendDebugToArmTask = 0x0801DCE8;  
4  
5 avDspEq = 0x200029E6;  
6 bd37xxRegisters = 0x20005B50;  
7  
8 modCode = 0x0803D000;  
9 modData = 0x20007000;  
10  
11  
12 /*****  
13 PROVIDE( sendToBD37xx = sendToBD37xx
```

Build project in Atollic TrueStudio

23:17:44 **** Rebuild of configuration Release for project SoundPatch ****

Info: Internal Builder is used for build

arm-atollic-eabi-gcc -c -mthumb -mcpu=cortex-m0 -I../src -Wa,--warn -x assembler-with-
arm-atollic-eabi-gcc -c ../src\MTCDMod.c -mthumb -mcpu=cortex-m0 -std=gnu11 -I../src
arm-atollic-eabi-gcc -c ../src\MTCD.c -mthumb -mcpu=cortex-m0 -std=gnu11 -I../src -O0
arm-atollic-eabi-gcc -o SoundPatch.elf src\MTCD.o src\MTCDMod.o src\Startup.o -mthumb
C:\EDA\TrueSTUDIO for STM32 9.0.1\ide\jre\bin\java -jar C:\EDA\TrueSTUDIO for STM32 9

Generate build reports...

Converting build output to binary

Output sent to: SoundPatch.binary

Converting build output to binary done

Print size information

text	data	bss	dec	hex	filename
233776	0	49	233825	39161	SoundPatch.elf

Print size information done

Generate listing file

Output sent to: SoundPatch.list

Generate listing file done

Generate build reports done

23:17:46 Build Finished (took 1s.661ms)

Open output file: *SoundPatched.binary* and previously patched mcu firmware in binary editor

And copy bytes from file offset 0x039000

HxD - [SoundPatch.binary]

File Edit Search View Analysis Extras Windows ?

16 hex

SoundPatch.binary decodedGS_288_WithProc.img

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
00038FE0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00038FF0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00039000	80	B5	84	B0	00	AF	78	60	7B	68	BE	2B	00	D0	70	E0°Žx`{hI+.Đpf
00039010	00	23	FB	60	69	E0	0B	23	FB	18	38	49	FA	68	8A	18#ũ`if.ũ.8IũhŠ.
00039020	12	78	1A	70	36	4A	FB	68	D3	18	1B	78	0A	2B	59	D0x.p6JũhÓ...x.+YĐ
00039030	33	4A	FB	68	D3	18	1A	78	32	49	FB	68	CB	18	1B	783JũhÓ...x2IũhĚ...x
00039040	9A	42	4F	D0	2E	4A	FB	68	D3	18	19	78	2D	4A	FB	68šBOĐ.JũhÓ...x-Jũh
00039050	D3	18	0A	1C	1A	70	0B	23	FB	18	1B	78	00	2B	22	D0Ó...p.ũ...x.+"Đ
00039060	0B	23	FB	18	1B	78	01	2B	1D	D0	0B	23	FB	18	1B	78#ũ...x.+.Đ.#ũ...x
00039070	02	2B	18	D0	0B	23	FB	18	1B	78	04	2B	13	D0	0B	23+.Đ.#ũ...x.+.Đ.#
00039080	FB	18	1B	78	12	2B	0E	D0	0B	23	FB	18	1B	78	0C	2Bũ...x.+.Đ.#ũ...x.+
00039090	09	D0	0B	23	FB	18	1B	78	0D	2B	04	D0	0B	23	FB	18Đ.#ũ...x.+.Đ.#ũ.
000390A0	1B	78	0E	2B	1E	D1	0B	23	FB	18	19	78	15	4A	FB	68x.+.Ń.#ũ...x.Jũh
000390B0	D3	18	1B	78	1A	00	14	4B	18	00	E0	F7	B9	FC	0B	23Ó...x...K...f÷aũ.#
000390C0	FB	18	1B	78	0F	49	FA	68	8A	18	11	78	0F	4A	D1	54ũ...x.IũhŠ...x.JŃT
000390D0	0B	23	FB	18	18	78	0B	4A	FB	68	D3	18	1B	78	19	00#ũ...x.JũhÓ...x..
000390E0	C7	F7	40	FF	FB	68	01	33	FB	60	FB	68	0E	2B	92	DDÇ÷@`ũh.3ũ`ũh.+`Ý
000390F0	00	E0	C0	46	BD	46	04	B0	80	BD	C0	46	10	D1	03	08fŔF`F.°e`ŔF.Ń..
00039100	AE	29	00	20	00	70	00	20	20	D1	03	08	18	5B	00	20ć). .p. Ń...P[.
00039110	00	01	02	03	04	05	06	07	08	09	0B	0C	0D	0E	12	00
00039120	41	76	45	71	4D	6F	64	3A	20	25	64	20	25	64	00	00AvEqMod: %d %d..

Offset: 39000 Overwrite

HxD - [decodedGS_288_WithProc.img]

File Edit Search View Analysis Extras Windows ?

16 hex

SoundPatch.binary decodedGS_288_WithProc.img

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
00038FE0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00038FF0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00039000	80	B5	84	B0	00	AF	78	60	7B	68	BE	2B	00	D0	70	E0°Žx`{hI+.Đpf
00039010	00	23	FB	60	69	E0	0B	23	FB	18	38	49	FA	68	8A	18#ũ`if.ũ.8IũhŠ.
00039020	12	78	1A	70	36	4A	FB	68	D3	18	1B	78	0A	2B	59	D0x.p6JũhÓ...x.+YĐ
00039030	33	4A	FB	68	D3	18	1A	78	32	49	FB	68	CB	18	1B	783JũhÓ...x2IũhĚ...x
00039040	9A	42	4F	D0	2E	4A	FB	68	D3	18	19	78	2D	4A	FB	68šBOĐ.JũhÓ...x-Jũh
00039050	D3	18	0A	1C	1A	70	0B	23	FB	18	1B	78	00	2B	22	D0Ó...p.ũ...x.+"Đ
00039060	0B	23	FB	18	1B	78	01	2B	1D	D0	0B	23	FB	18	1B	78#ũ...x.+.Đ.#ũ...x
00039070	02	2B	18	D0	0B	23	FB	18	1B	78	04	2B	13	D0	0B	23+.Đ.#ũ...x.+.Đ.#
00039080	FB	18	1B	78	12	2B	0E	D0	0B	23	FB	18	1B	78	0C	2Bũ...x.+.Đ.#ũ...x.+
00039090	09	D0	0B	23	FB	18	1B	78	0D	2B	04	D0	0B	23	FB	18Đ.#ũ...x.+.Đ.#ũ.
000390A0	1B	78	0E	2B	1E	D1	0B	23	FB	18	19	78	15	4A	FB	68x.+.Ń.#ũ...x.Jũh
000390B0	D3	18	1B	78	1A	00	14	4B	18	00	E0	F7	15	FE	0B	23Ó...x...K...f÷.ç.#
000390C0	FB	18	1B	78	0F	49	FA	68	8A	18	11	78	0F	4A	D1	54ũ...x.IũhŠ...x.JŃT
000390D0	0B	23	FB	18	18	78	0B	4A	FB	68	D3	18	1B	78	19	00#ũ...x.JũhÓ...x..
000390E0	C7	F7	40	FF	FB	68	01	33	FB	60	FB	68	0E	2B	92	DDÇ÷@`ũh.3ũ`ũh.+`Ý
000390F0	00	E0	C0	46	BD	46	04	B0	80	BD	C0	46	10	D1	03	08fŔF`F.°e`ŔF.Ń..
00039100	E6	29	00	20	00	70	00	20	20	D1	03	08	50	5B	00	20ć). .p. Ń...P[.
00039110	00	01	02	03	04	05	06	07	08	09	0B	0C	0D	0E	12	00
00039120	41	76	45	71	4D	6F	64	3A	20	25	64	20	25	64	00	00AvEqMod: %d %d..

Offset: 39000 Overwrite

Find firmware version and mark as 'bd' (BD375xx)

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
00036F40	20	4D	61	6E	75	61	6C	00	20	20	30	64	62	00	00	00	Manual. 0db...
00036F50	2D	31	32	64	62	00	00	00	20	2D	33	64	62	00	00	00	-12db... -3db...
00036F60	2D	31	35	64	62	00	00	00	20	2D	36	64	62	00	00	00	-15db... -6db...
00036F70	20	2D	39	64	62	00	00	00	4E	6F	72	6D	61	6C	00	00	-9db...Normal...
00036F80	53	77	61	70	00	00	00	00	48	69	64	65	00	00	00	00	Swap....Hide....
00036F90	54	79	70	65	31	00	00	00	33	33	33	33	00	00	00	00	Type1...3333....
00036FA0	31	32	36	00	47	45	5F	53	48	49	00	00	47	53	00	00	126.GE_SHI..GS..
00036FB0	30	30	30	30	00	00	00	00	43	41	52	2D	4B	49	54	00	0000....CAR-KIT.
00036FC0	56	32	2E	38	38	62	64	00	54	56	5F	31	33	20	43	4D	V2.88bd.TV_13 CM
00036FD0	4D	42	28	2E	2E	2E	29	00	54	56	5F	30	34	20	51	53	MB(...).TV_04 QS
00036FE0	44	2D	4D	54	2D	53	37	37	28	51	53	44	29	00	00	00	D-MT-S77(QSD)...
00036FF0	54	56	5F	30	31	20	49	53	44	42	2D	54	28	52	69	73	TV_01 ISDB-T(Ris
00037000	68	74	61	29	00	00	00	00	54	56	5F	30	32	20	44	54	hta)....TV_02 DT
00037010	56	2D	54	28	4C	6F	6E	74	61	63	29	00	54	56	5F	30	V-T(Lontac).TV_0
00037020	33	20	49	53	44	42	2D	54	28	49	6E	66	6F	73	70	61	3 ISDB-T(Infospa
00037030	63	65	29	00	41	55	54	4F	44	41	42	20	32	2E	30	00	ce).AUTODAB 2.0.
00037040	54	56	5F	31	39	20	41	55	48	4B	41	20	48	52	2D	36	TV_19 AUHKA HR-6
00037050	30	30	00	00	54	56	5F	31	34	20	43	44	54	2D	36	45	00..TV_14 CDT-6E
00037060	50	4E	32	32	2D	53	44	30	30	00	00	00	54	56	5F	31	PN22-SD00...TV_1
00037070	35	00	00	00	54	56	5F	31	37	20	44	54	56	2D	48	44	5...TV_17 DTV-HD
00037080	2D	35	35	38	00	00	00	00	54	56	5F	30	35	20	42	4C	-558....TV 05 BL


Check, check, check, check.....

You must be sure is everything is ok, otherwise you can brick you unit

Open patched mcu firmware in IDA

Go to “our entry point” then go to subroutine (compiled in TrueStudio)

```
CODE:0802BA88          DCD 0xD0D010E, 0xE0E00, 0xE0E
CODE:0802BAA8 off_802BAA8 DCD aAvTask          ; DATA XREF: CreateTask_AV+38↑o
CODE:0802BAA8          ; CODE:off_80119CC↑o
CODE:0802BAA8          ; "AV_TASK"
CODE:0802BAAC          DCD loc_8010A68+1
CODE:0802BAB0          DCD 1, 0
CODE:0802BAB8          DCD 0xC0
CODE:0802BABC dword_802BABC DCD 0x5030201, 0x29282006, 0x302C2B2A, 0x51474441, 0x755754
CODE:0802BABC          ; DATA XREF: sub_8004F64+1C↑o
CODE:0802BABC          ; CODE:off_8004FB0↑o
CODE:0802BAD0 dword_802BAD0 DCD 0          ; DATA XREF: CODE:08010C18↑o
CODE:0802BAD0          ; CODE:off_8010E8C↑o
CODE:0802BAD4          DCD sub_800FC5C+1
CODE:0802BAD8          DCD sub_800FB20+1
CODE:0802BADC          DCD sub_800FA0C+1
CODE:0802BAE0          DCD sub_800FD24+1
CODE:0802BAE4          DCD sub_800FB74+1
CODE:0802BAE8          DCD sub_800F970+1
CODE:0802BAEC          DCD sub_803D000+1
CODE:0802BAF0          DCD 0
CODE:0802BAF4 dword_802BAF4 DCD 0x3020102, 0x39040003, 0x100A7006, 0x20101B0F, 0x40126211
```



Subroutine should look like this:

```
int __fastcall sub_803D000(int result)
{
    unsigned __int8 v1; // [sp+Bh] [bp+Bh]
    signed int i; // [sp+Ch] [bp+Ch]

    if ( result == 190 )
    {
        for ( i = 0; i <= 14; ++i )
        {
            v1 = *((_BYTE *)dword_803D110 + i);
            if ( byte_200029E6[i] != 10 && (unsigned __int8)byte_20007000[i] != byte_20007000[i] )
            {
                byte_20007000[i] = byte_200029E6[i];
                if ( !v1 || v1 == 1 || v1 == 2 || v1 == 4 || v1 == 18 || v1 == 12 || v1 == 13 || v1 == 14 )
                {
                    SendDbgStrToArmTask("AvEqMod: %d %d", v1, byte_20007000[i]);
                    byte_20005B50[v1] = byte_20007000[i];
                    result = sub_8004F64(v1, byte_20007000[i]);
                }
            }
        }
    }
    return result;
}
```



And check:

1. Is this point to *SendDbgStrToArmTask*
2. Is this point to subroutine you noted down in [Figure 1](#)
3. Is this is the same reference to you noted down in [Figure 2](#)
4. Is this is the same reference to you noted down in [Figure 3](#)


Also check previously patched subroutines

[Figure 4](#)

[Figure 5](#)

[Figure 6](#)

If you sure is everything is correct encode patched firmware

 C:\Windows\system32\cmd.exe

```
c:\MCU>mtcdmcutool.exe -e decoded.img dmcu.img
```

```
mtcdmcutool: a tool to decode/encode/info MTCD MCU firmware images for RK3xxx/PX3/PX5 headunits.  
v.1.2, (c) 2018, Wadzio
```

```
Done!
```

```
c:\MCU>
```


DONE

APPENDIX

If you cannot find *CreateTask_AV*:

Find text *AV_TASK* then go to XReference:

CODE:0803B1D0	aSync	DCB "SYNC",0	; DATA XREF: RAM:200007C4↓o
CODE:0803B1D5		DCB 0	
CODE:0803B1D6		DCB 0	
CODE:0803B1D7		DCB 0	
CODE:0803B1D8	aUsbipod	DCB "USBIPOD",0	; DATA XREF: RAM:200007C8↓o
CODE:0803B1E0	aIpod	DCB "IPOD",0	; DATA XREF: RAM:200007B4↓o
CODE:0803B1E5		DCB 0, 0, 0	
CODE:0803B1E8	aDvd	DCB "DVD",0	; DATA XREF: RAM:200007A4↓o
CODE:0803B1EC	aAvTask	DCB "AV_TASK",0	; DATA XREF: CODE:off_802BAA8↑o
CODE:0803B1F4	aRadio	DCB "RADIO",0	; DATA XREF: RAM:200007AC↓o
CODE:0803B1FA		ALIGN 4	
CODE:0803B1FC	aDvr	DCB "DVR",0	; DATA XREF: RAM:200007B8↓o
CODE:0803B200	aSys	DCB "SYS",0	; DATA XREF: RAM:200007A0↓o
CODE:0803B204	aBt	DCB "BT",0	; DATA XREF: RAM:off_2000079C↓o



OR


Find subroutine *main*

```
int __cdecl __noreturn main(int argc, const char **argv, const char **envp)
{
    unsigned int v3; // r0
    int v4; // r0
    int v5; // r0
    int v6; // r0
    int v7; // r0
    int v8; // r0
    int v9; // r0
    int v10; // r0
    int v11; // r0
    int v12; // r0
    int v13; // r0
    const char **v14; // [sp+0h] [bp-10h]
    int v15; // [sp+4h] [bp-Ch]

    v14 = envp;
    v3 = 0;
    do
    {
        dword_20000000[v3] = *(&off_8004000 + v3);
        ++v3;
    }
    while ( v3 < 0x30 );
    SYSCFG_COMP = 4 * ((unsigned int)SYSCFG_COMP >> 2);
    SYSCFG_COMP |= 3u;
    v4 = HAL_Init(&SYSCFG_COMP, SYSCFG_COMP);
```

At the end *main* find subroutine *MX_FREERTOS_Init* and go to it

```
dword_20000CB8 = (int)&USART5;  
dword_20000CBC = 115200;  
dword_20000CCC = 12;  
dword_20000D30 = 0;  
dword_20000D34 = 0;  
dword_20000D38 = 0;  
dword_20000D40 = 0;  
dword_20000D44 = 0;  
dword_20000D48 = 0;  
dword_20000D4C = 0;  
dword_20000D28 = (int)&USART6;  
dword_20000D2C = 115200;  
dword_20000D3C = 12;  
v13 = MX_FREERTOS_Init();  
osKernelStart(v13);  
while ( 1 )  
    ;  
}
```



In that function third from end should be *CreateAV_Task*

```
if ( !dword_200029A8 )  
{  
    if ( (unsigned __int8)byte_200029A4 < 2u )  
        byte_200029A4 = 2;  
    dword_200029A8 = v4;  
    WriteBufToIntEEPROM(6);  
}  
v6 = sub_8017560();  
v7 = sub_8016FFC(v6);  
v8 = CreateTasks_REV_CANBUS_UART_ARM(v7);  
v9 = CreateTask_ADC(v8);  
CreateTask_RADIO(v9);  
v10 = sub_8011968();  
v11 = CreateTask_DVD(v10);  
return CreateTask_MISC(v11);  
}
```

