

# AVR – Verificare functii de scriere LED si PF

The screenshot displays the AVR Studio IDE with the following components:

- Processor Window:** Shows system registers such as Program Counter (0x0000F0), Stack Pointer (0x04FB), X pointer (0x010E), Y pointer (0x01F4), Z pointer (0x00EE), Cycle Counter (1608318), Frequency (1.0000 MHz), Stop Watch (1608318.00 us), and SREG.
- Code Editor:** Displays the source code for `main.c`. The function `write_LED(key);` and `write_PF(key);` are circled in red. The code includes a loop for reading a key and writing to the LED and Port F.
- Watch Window:** A table showing the current state of variables:

Name	Value	Type	Location
key	0x0E, 'I'	char	R7
row	Not in scope		
col	Not in scope		
cod	Not in scope		
val	Not in scope		
- I/O View:** A tree view of hardware components. The `PORTC` register is highlighted in blue. Its bit status is shown as `0xEE`. Red arrows point from the bit patterns in `PORTC` to the labels "LED" and "PF out".
- Message Window:** Contains several error messages: "Setting breakpoint at address 0x01d4 failed in model: AvrDebugger.AvrDebugTriggerCollec".

# AVR – Verificare functie citire PF

The screenshot displays the Analog Devices VisualDSP++ interface for an ADSP-2181 simulation. The main window shows assembly code for a project named 'test\_ext'. A 'Streams' dialog box is open, showing active streams with source and destination targets. A red arrow points from the 'FILE [941.dat, Fractional, Rewind, Cr...' entry in the dialog to the assembly code line 'citire de la rx\_buf+2 (941 Hz)'. Another blue arrow points from the 'FILE [sin0.dat, Fractional, Rewind]' entry to the assembly code line 'scriere la tx\_buf+1 (sin "0")'. A third blue arrow points from the 'Data Memory I/O Port - DM[0x1102]' entry to the assembly code line 'scriere la tx\_buf+2 (sin "1")'. Other windows include a 'PF regs' window showing PFDATA 00 and FFTYPE F0, a 'sin1' window showing a frequency plot with peaks at approximately 1000 Hz and 1500 Hz, a 'Computational' window showing register values (AX0-AX1, AY0-AY1, MX0-MX1, NY0-NY1, MR2-MR1, SR1-SR0, SI, SE, AR, AF, MF, MR0, SB), and a 'DAG' window showing a data path graph with nodes I0-I7 and M0-M7.

**Assembly Code (test\_ext.asm):**

```

//mr=mx0*my0(rnd), ay0=dm(i0,m1); //{inmulteste, get Q2
mr=mx0*my0(rnd), ay0=dm(i0,m2); //{inmulteste, get Q2
sr=ashift mrl by 1 (hi); //{schimba 2.30 in 1.15
ar=sr1-ay0; //{Q1*COEF - Q2
ar=ar+ay1; //{Q1*COEF - Q2 + intru
dm(i0,m0)=ar;
dm(i0,m0)=mx0;
jump end;;

backs:
//{----- C A N D F A Z A F E E D
//{
skip_backs:
call feedforward;
call test_and_output;
call restart;
nop;

rts;

generator0:
citire de la rx_buf+2 (941 Hz)

ay0=dm(sum0);
si=dm(hertz0);
sr=ashift si by 3 (hi);
my0=0x4189;
    
```

**Streams Dialog (Active streams):**

Source Target (Device)	Destination Target (Device)
Data Memory I/O Port - DM[0x1101]	FILE [sin0.dat, Fractional, Rewind]
FILE [941.dat, Fractional, Rewind, Cr...	Data Memory I/O Port - DM[0x1012]
Data Memory I/O Port - DM[0x1102]	FILE [sin1.dat, Fractional, Rewind]

**PF regs:**

```

PFDATA 00
FFTYPE F0
    
```

**sin1 (Line Plot):**

Frequency (Hz) vs Amplitude. The plot shows a signal with two prominent peaks at approximately 1000 Hz and 1500 Hz.

**Computational:**

AX0 1170	AX1 0080	AR FFFF
AY0 0000	AY1 00AA	AF 1170
MX0 FFD9	MX1 1CCE	
NY0 4189	MY1 1170	MF 0001
MR2 00	MR1 06A3	MR0 BBDE
SR1 0031	SR0 0000	
SI 0310	SE uu	SB uu

**DAG:**

I0 0000	M0 0001	I0 0010
I1 0012	M1 0001	I1 0000
I2 0035	M2 3FFF	I2 0000
I3 0042	M3 0001	I3 0000
I4 uuuu	M4 0001	I4 0000
I5 1010	M5 0001	I5 0003
I6 1100	M6 uuuu	I6 0003
I7 3000	M7 uuuu	I7 0008

# AVR – verificare funzionare tastatura

The screenshot displays the AVR Studio environment during a debugging session. The main window shows the source code for a keyboard scan routine. The code iterates through a 4x4 matrix of keys, reading the state of PORTA and PORTB to determine which key is pressed. The watch window shows the current state of variables: key (0x00), row (0x03), col (0x02), and cod (0x0E). The I/O View window shows the state of the PORT registers: PORTA (0x0F), PORTB (0x00), PORTC (0xFF), and PORTD (0x00). Red annotations highlight the current state of the keyboard matrix and the corresponding I/O register values.

**Code Snippet:**

```

// line 0 - PA0, line 1 - PA1, line 2 - PA2, line 3 - PA3 - outputs
char scan[4]={0xFE,0xFD,0xFB, 0xF7};
char row,col;
char cod=0xFF;

for (row=0; row<4; row++)
{
  PORTA=scan[row];
  delay_us(1);
  // col 0 - PA4, col 1 - PA5, col 2 - PA6, col 3 - PA7 - inputs
  col=PINA>>4;
  if (col!=0x0F)
  {
    if (col==0x0E) col=0;
    if (col==0x0D) col=1;
    if (col==0x0B) col=2;
    if (col==0x07) col=3;
    cod=4*row+col;
    break;
  }
}
return cod;

void write_LED(char a)

```

**Watch Window:**

Name	Value	Type	Location
key	0x00	char	R7
row	0x03	char	R17
col	0x02	char	R16
cod	0x0E	char	R19

**I/O View Window:**

Name	Address	Value	Bits
PORTA	0x01 (0x21)	0x0F	00001111
PINA	0x00 (0x20)	0xB7	10110111
PORTB	0x02 (0x22)	0xF7	11110111
PORTB	0x04 (0x24)	0x00	00000000
PINB	0x03 (0x23)	0x00	00000000
PORTB	0x05 (0x25)	0x00	00000000
PORTC	0x07 (0x27)	0xFF	11111111
PINC	0x06 (0x26)	0x00	00000000
PORTC	0x08 (0x28)	0x00	00000000

**Keyboard Matrix Diagram:**

	0	1	2	3	
0	Q	W	E	R	PA0 0
1	A	S	D	F	PA1 1
2	Z	X	C	V	PA2 2
3	B	N	M	Com	PA3 3
	PA4	PA5	PA6	PA7	

Red annotations in the diagram indicate that the current state is row 3 (line 3) and column 2 (col 2), resulting in a key code of 0x0E (3\*4 + 2 = 14).

# DSP – Verificare functionare PF (input)

The screenshot displays the Analog Devices VisualDSP++ interface for a target ADSP-2181. The main assembly window shows the following code:

```

ax0=dm(Prog_Flag_Data);
ay0=0x000F;
ar=ax0 and ay0;
dm(PF_input)=ar;
// PF outputs 4-7
ar=dm(PF_output);
sr=lshift ar by 4 (hi);
ax0=sr1;
dm(Prog_Flag_Data)=ax0;

// test generator
// frecventa de "0"
call generator0;
dm(tx_buf+1)=ar;
// frecventa de "1"
call generator1;
dm(tx_buf+2)=ar;

// test receptor
call receptor;

rts;

init_receptor;
    
```

Key components and their values are highlighted with red circles and arrows:

- PF regs window:** Shows PFDATA: 05 and PFTYPE: FU.
- Computational window:**

AX0	0005	AX1	uuuu	AR	0005
AY0	000F	AY1	0089	AF	2F44
MX0	0000	MX1	1CCE		
MY0	23CE	MY1	2F44	MF	00E1
MR2	00	MR1	0000	MRO	8000
SR1	0000	SR0	0000		
SI	003F	SE	uu	SB	uu
- DAG window:**

I0	0011	M0	0000	I0	0010
I1	uuuu	M1	0001	I1	0000
I2	uuuu	M2	3FFF	I2	0000
I3	0042	M3	0001	I3	0000
I4	uuuu	M4	0001	I4	0000
I5	1010	M5	0001	I5	0003
I6	1100	M6	uuuu	I6	0003
I7	3000	M7	uuuu	I7	0008
- DM [Hexadecimal] windows:**
  - Left window shows memory addresses 000022 to 00002B with values like 0000, 0089, 00CC, 0100, 0100, 0100.
  - Right window shows PF\_input at address 000046 with value 0005, and PF\_output at address 000047 with value 000A.
- Disassembly window:** Shows instructions at addresses 0000A3 to 0000A8, including `dm(PF_input) = sr`, `ar = dm(PF_output)`, `sr = lshift ar h`, `ax0 = sr1`, and `dm(3fe5) = ax0`.
- Output Window:** Shows messages: "Hit breakpoint at Address 0xa0", "Loading: 'D:\sorin\project2\2020\_cod\v0\test\_s", "Load complete.", "Hit breakpoint at Address 0xa0", "Hit breakpoint at Address 0xa0".

# DSP – verificare functionare PF output

The screenshot displays the Analog Devices VisualDSP++ interface during a simulation of an ADSP-2181. The main window shows the assembly code for 'test\_ext.asm'. A red circle highlights the initialization of the 'ax0' register to the address of 'Prog\_Flag\_Data' and the assignment of 'PF\_output' to 'ar'. Another red circle highlights the 'call generator0' instruction. A third red circle highlights the 'PFDATA A0' register in the 'PF regs' window. A red arrow points from this register to the 'Computational' window, which shows the current state of registers: AX0 is 00A0, AX1 is uuuu, AR is 000A, AY0 is 000F, AY1 is 0089, AF is 2F44, MX0 is 0000, MX1 is 1CCE, MF is 00E1, MY0 is 23CE, MY1 is 2F44, MR0 is 8000, MR2 is 00, MR1 is 0000, SR0 is 0000, SR1 is 00A0, SR0 is 0000, SE is uu, and SB is uu. The 'Disassembly' window shows the instruction 'call generator0' at address 0000A8. The 'DM [Hexadecimal]' windows show memory contents, with 'PF\_input' at 000046 containing 0005 and 'PF\_output' at 000047 containing 000A. The 'Output Window' shows the execution flow, including a hit breakpoint at address 0xa0. The status bar at the bottom indicates the processor is 'Halted' at 'Line 543, Col 1'.

# DSP – Setare intreruperi

Analog Devices VisualDSP++ - [Target: ADSP-2181 Simulation] - [Project: test\_ext]

File Edit Session View Project Register Memory Debug Settings Tools Window Help

test\_ext Debug

Project: test\_ext.dpj

Project group: 1 project(s)

- test\_ext
  - Source Files
  - test\_ext.asm
  - Linker Files
  - Header Files

PF regs

```
PFDATA 00
PFTYPE F0
```

test\_ext.asm

```
//mr=mx0*my0(rnd), ay0=dm(i0.m1); //{inmulteste.get Q2
mr=mx0*my0(rnd), ay0=dm(i0.m2); //{inmulteste.get Q2
sr=ashift mrl by 1 (hi); //{schimba 2.30 in 1.15
ar=sr-ay0; //{Q1*COEF - Q2
ar=sr+ay1; //{Q1*COEF - Q2 + intra
dm(i0.m0)=ar; //{rezultatul = noul Q1
jump end;;

back:
//{----- C A N D F A Z A
skip_backs:
call feedforward;
call test_and_output;
call restart;
nop;
end:
rts;

generator0:
ay0=dm(sum0);
si=dm(hertz0);
sr=ashift si by 3 (hi);
my0=0x4189;
```

Interrupt Timing

New Interrupt

External interrupts: IRQ0

Interrupt Properties

Min cycles: 0 Max cycles: 0 Offset cycles: 0

Interrupts:

Name	Min	Max	Delay
IRQ2	1000	1000	0

DM [Hexadecimal]

```
outcode
[000022] 0008
in_sample
[000023] 0031
countN
[000024] FFFF
min_tone_level
[000025] 0100 0100 0100
[000028] 0100 0100 0100
[00002B] 0100 0100
mnsqr
```

Disassembly

```
[0000CB] jump end
[0000CC] skip_backs
[0000CC] call feedforward
[0000CD] call test_and_output
[0000CE] call restart
end
```

PF regs

```
AX0 1170
AY0 0000
MX0 FFD9
MY0 4189
MR2 00
SR1 0031
SI 0310
MX1 1CCE
MY1 1170
MR1 06A3
SR0 0000
SE uu
MF 0001
MR0 BBDE
SB uu
```

DAG

I0 0000	M0 0001	I0 0010
I1 0012	M1 0001	I1 0000
I2 0035	M2 3FFF	I2 0000
I3 0042	M3 0001	I3 0000
I4 uuuu	M4 0001	I4 0000
I5 1010	M5 0001	I5 0003
I6 1100	M6 uuuu	I6 0003
I7 3000	M7 uuuu	I7 0008

sin1

Output Window

Ready

Halted Line 606, Col 1 Tcl

1:31 PM 3/24/2020

# DSP – Setare fisiere in , out

The screenshot shows the Analog Devices VisualDSP++ interface for a project named 'test\_ext'. The main window displays assembly code for 'test\_ext.asm'. A 'Streams' dialog box is open, showing active streams with source and destination targets. A 'sin1' window shows a line plot of a sine wave. Other windows include 'PF regs', 'Computational', 'DAG', and 'DM [Hexadecimal]'.

**Assembly Code (test\_ext.asm):**

```

//mr=mx0*my0(rnd), ay0=dm(i0,m1); //{inmulteste, get Q2
mr=mx0*my0(rnd), ay0=dm(i0,m2); //{inmulteste, get Q2
sr=ashift mrl by 1 (hi);
//schimba 2.30 in 1.15
ar=sr1-ay0;
//Q1*COEF - Q2
//Q1*COEF - Q2 + intrinsec
dm(i0,m0)=ar;
ar=ar+ay1;
dm(i0,m0)=mx0;
jump end;

backs:
//{----- C A N D F A Z A F E E D
//{
skip_backs:
call feedforward;
call test_and_output;
call restart;
nop;

end:
rts;

generator0:
    citire de la rx_buf+2 (941 Hz)
ay0=dm(sum0);
si=dm(hertz0);
sr=ashift si by 3 (hi);
mx0=0x4189;
    
```

**Streams Dialog:**

Source Target (Device)	Destination Target (Device)
Data Memory I/O Port - DM[0x1101]	FILE [sin0.dat, Fractional, Rewind]
FILE [941.dat, Fractional, Rewind, Cir...	Data Memory I/O Port - DM[0x1012]
Data Memory I/O Port - DM[0x1102]	FILE [sin1.dat, Fractional, Rewind]

**sin1 Window:**

Line Plot showing Frequency (Hz) on the x-axis (0 to 4000) and amplitude on the y-axis (0\*10<sup>4</sup> to 3\*10<sup>4</sup>). The plot shows a sine wave with a peak amplitude of approximately 2.5\*10<sup>4</sup>.

**Computational Window:**

AX0	AX1	AR
1170	0080	FFFF
0000	00AA	AF 1170
FFD9	MX1	1CCE
4189	MY1	1170
00	MR1	06A3
0031	SR0	0000
0310	SE	uu
	SB	uu

**DAG Window:**

I0	M0	I0
0000	0001	0010
0012	M1	0001
0035	M2	3FFF
0042	M3	0001
uuuu	M4	0001
1010	M5	0001
1100	M6	uuuu
3000	M7	uuuu

**DM [Hexadecimal] Windows:**

Left DM [Hexadecimal]:

```

outcode
[000022] 0008
in_sample
B1
untN
FF
a_tone_level
00 0100 0100
00 0100 0100
00 0100
sqr
    
```

Right DM [Hexadecimal]:

```

ansqr
[00002D] 0005 000A 0025
[000030] 31E3 0003 0001
[000033] 0001 0001
bits
[000035] 0001 0002 0004
[000038] 0008 0010 0020
[00003B] 0040 0080
sin_coeff
[00003D] 3240 0053 AAC
[000040] 08B7 1CCE
sum0
[000042] A7F0
hertz0
[000043] 0302
sum1
[000044] 1170
hertz1
[000045] 05C5
PF_input
[000046] 0000
PF_output
[000047] 0000 uuuu uuuu
[00004A] uuuu uuuu uuuu
[00004D] uuuu uuuu uuuu
[000050] uuuu uuuu uuuu
[000053] uuuu uuuu uuuu
[000056] uuuu uuuu uuuu
[000059] uuuu uuuu uuuu
[00005C] uuuu uuuu uuuu
[00005F] uuuu uuuu uuuu
[000062] uuuu uuuu uuuu
    
```

# DSP – Verificare functionare generator

The screenshot displays the Analog Devices VisualDSP++ interface for a simulation of an ADSP-2181. The main window shows the assembly code for 'test\_ext.asm' with the following macros and comments:

```
#define scale -4
#define F0 770
#define F1 1477
#define MIN_LEVEL 0x100

/*
Frecventele sint: 697,770,852,941,1209,1336,1477,1633
in aceasta ordine a bitilor in variabila outcode: 0,1,2,3,4,5,6,7
*/
```

The 'Line Plot' window shows a frequency spectrum with two prominent peaks at 770 Hz and 1477 Hz. The x-axis is labeled 'Frequency (Hz)' and ranges from 0 to 4,000. The y-axis ranges from 0 to 3x10<sup>6</sup>.

The 'PF regs' window shows the following register values:

PFDATA	A0
PFTYPE	F0

The 'DAG' window shows the following data:

I0	0011	M0	0000	I0	0010
I1	uuuu	M1	0001	I1	0000
I2	uuuu	M2	3FFF	I2	0000
I3	0042	M3	0001	I3	0000
I4	uuuu	M4	0001	I4	0000
I5	1010	M5	0001	I5	0003
I6	1100	M6	uuuu	I6	0003
I7	3000	M7	uuuu	I7	0008

The console window shows the following output:

```
Loading: "D:\sorin\proiect2\2020_cod\v0\test
Load complete.
Hit breakpoint at Address 0xa0
Hit breakpoint at Address 0xa0
Hit breakpoint at Address 0xc
```



# DSP – Verificarea functionarii receptor

Analog Devices VisualDSP++ - [Target: ADSP-2181 Simulation] - [Project: test\_ext]

File Edit Session View Project Register Memory Debug Settings Tools Window Help

test\_ext Debug

Project: test\_ext.dpj

Project group: 1 project(s)

- test\_ext
  - Source Files
  - test\_ext.asm
  - Linker Files
  - Header Files

PF regs

PFDATA 00  
FFTYPE F0

test\_ext.asm

```

//mr=mx0*my0(rnd), ay0=dm(i0,m1); //{inmulteste, get Q2
mr=mx0*my0(rnd), ay0=dm(i0,m2); //{inmulteste, get Q2
sr=ashift mrl by 1 (hi); //{schimba 2.30 in 1.15
ar=sr1-ay0; //{Q1*COEF - Q2
ar=ar+ay1; //{Q1*COEF - Q2 + intra
dm(i0,m0)=ar; //{rezultatul = noul Q1
dm(i0,m0)=mx0; //{vechitul Q1 = noul Q2
jump end;

//----- CAND FAZA FEEDBACK ESTE GAT
skip_backs:
call feedforward;
call test_and_output;
call restart;
end:
nop;

rts;
generator0:
f0 f1 f2 f3 f4 f5 f6 f7
697 770 852 941 1209 1336 1477 1633

ay0=dm(sum0);
si=dm(hertz0);
sr=ashift si by 3 (hi);
mv0=0x4189;
    
```

DM [Hexadecimal]

```

[000022] 0008
[000023] 00CB
[000024] FFFF
[000025] 0100 0100 0100
[000028] 0100 0100 0100
[00002B] 0100 0100
    
```

Disassembly

```

[0000C7] jump end
[0000C8] call feedforward
[0000C9] call test_and_output
[0000CA] call restart
end
    
```

PF regs

mnsqr

```

[00002D] 0005 000A 0025
[000030] 31E3 0003 0001
[000033] 0001 0001
[000035] 0001 0002 0004
[000038] 0008 0010 0020
[00003B] 0040 0080
[00003D] 3240 0053 AAC3
[000040] 08B7 1CCE
[000042] 0000
[000043] 0302
[000044] 0000
[000045] 05C5
[000046] 0000
[000047] 0000 uuuu uuuu
[00004A] uuuu uuuu uuuu
[00004D] uuuu uuuu uuuu
[000050] uuuu uuuu uuuu
[000053] uuuu uuuu uuuu
[000056] uuuu uuuu uuuu
[000059] uuuu uuuu uuuu
[00005C] uuuu uuuu uuuu
[00005F] uuuu uuuu uuuu
[000062] uuuu uuuu uuuu
    
```

sin1

Computational

AX0 0001	AX1 0080	AR 0008
AY0 0100	AY1 0085	AF 0008
MX0 004F	MY1 0092	MF 0000
MR2 00	MR1 0001	MRO 574A
SR1 0000	SR0 0000	
SI 0CB4	SE uu	SB uu

DAG

I0 0000	M0 0001	L0 0010
I1 002D	M1 0001	L1 0000
I2 0035	M2 3FFF	L2 0000
I3 003D	M3 0000	L3 0000
I4 uuuu	M4 0001	L4 0000
I5 1010	M5 0001	L5 0003
I6 1100	M6 uuuu	L6 0003
I7 3000	M7 uuuu	L7 0008

Ready

Halted Line 606, Col 1 Tcl

1:13 PM 3/24/2020