# CBPM3 Status

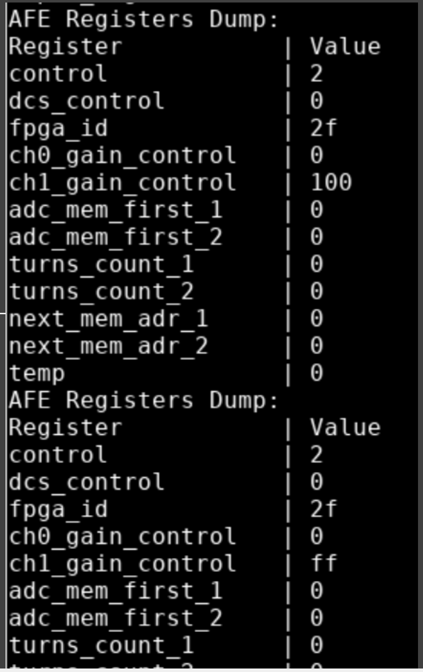
JUNE 7TH 2024

## Overview

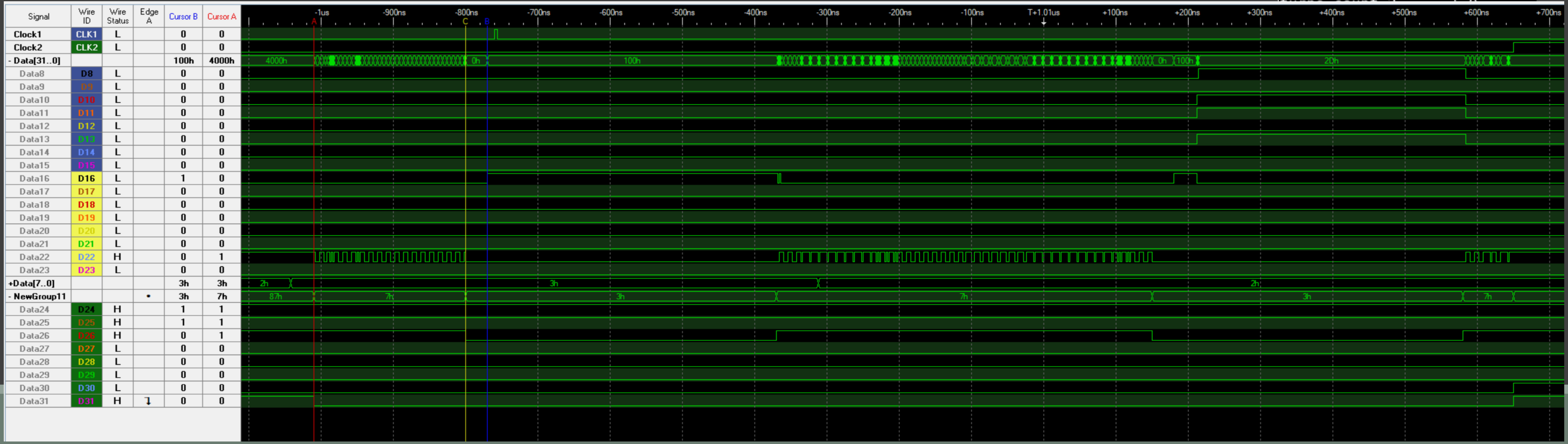
- Update on Register Access
  - Problem discussion
  - Current solution
- Hardware Changes
- Next Target

# Register Access Issue

- Some registers not reading correctly
- Replicated on a second prototype, but slightly different
- Hardware debugging showed transaction info was fine, all the way to the processor

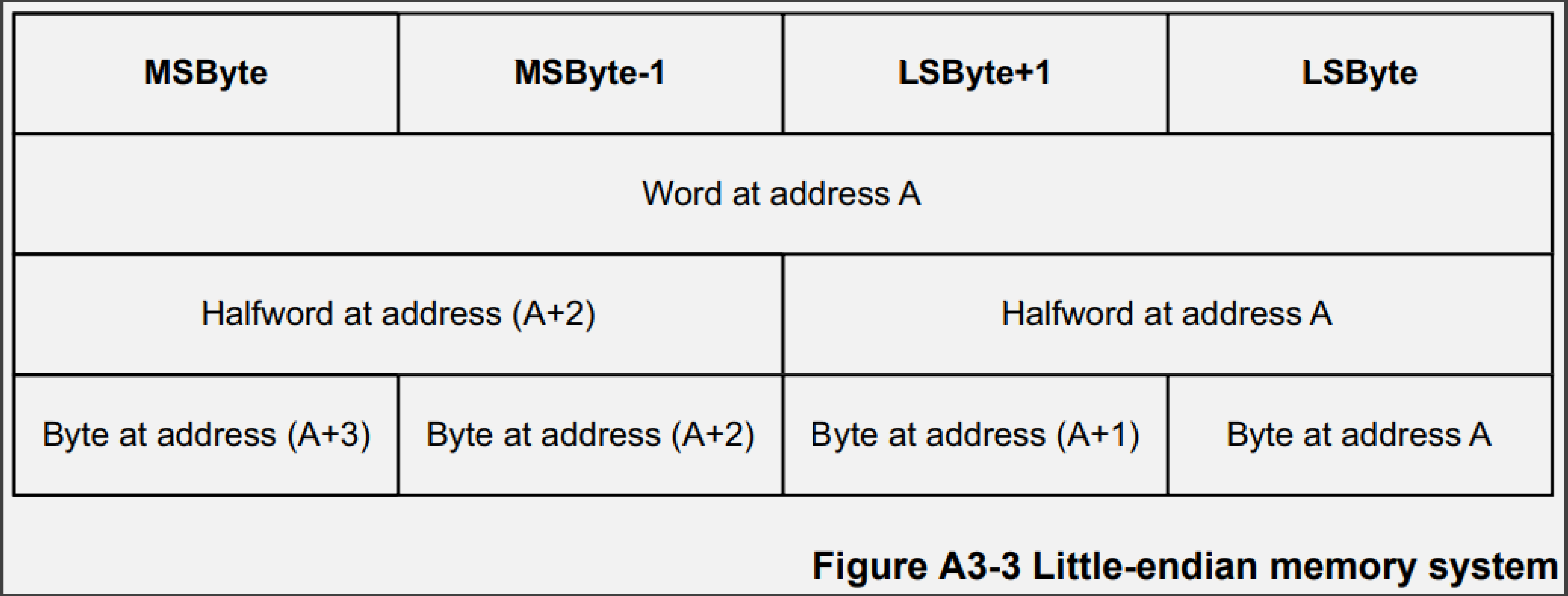


```
AFE Registers Dump:
Register           | Value
control            | 2
dcs_control        | 0
fpga_id            | 2f
ch0_gain_control   | 0
ch1_gain_control   | 100
adc_mem_first_1    | 0
adc_mem_first_2    | 0
turns_count_1      | 0
turns_count_2      | 0
next_mem_adr_1     | 0
next_mem_adr_2     | 0
temp               | 0
AFE Registers Dump:
Register           | Value
control            | 2
dcs_control        | 0
fpga_id            | 2f
ch0_gain_control   | 0
ch1_gain_control   | ff
adc_mem_first_1    | 0
adc_mem_first_2    | 0
turns_count_1      | 0
```



| Signal | Wire ID | Wire Status | Edge A | Cursor B | Cursor A |
|---|---|---|---|---|---|
| Clock1 | CLK1 | L | | 0 | 0 |
| Clock2 | CLK2 | L | | 0 | 0 |
| - Data[31..0] | | | | 100h | 4000h |
| Data8 | D8 | L | | 0 | 0 |
| Data9 | D9 | L | | 0 | 0 |
| Data10 | D10 | L | | 0 | 0 |
| Data11 | D11 | L | | 0 | 0 |
| Data12 | D12 | L | | 0 | 0 |
| Data13 | D13 | L | | 0 | 0 |
| Data14 | D14 | L | | 0 | 0 |
| Data15 | D15 | L | | 0 | 0 |
| Data16 | D16 | L | | 1 | 0 |
| Data17 | D17 | L | | 0 | 0 |
| Data18 | D18 | L | | 0 | 0 |
| Data19 | D19 | L | | 0 | 0 |
| Data20 | D20 | L | | 0 | 0 |
| Data21 | D21 | L | | 0 | 0 |
| Data22 | D22 | H | | 0 | 1 |
| Data23 | D23 | L | | 0 | 0 |
| +Data[7..0] | | | | 3h | 3h |
| - NewGroup11 | | | • | 3h | 7h |
| Data24 | D24 | H | | 1 | 1 |
| Data25 | D25 | H | | 1 | 1 |
| Data26 | D26 | H | | 0 | 1 |
| Data27 | D27 | L | | 0 | 0 |
| Data28 | D28 | L | | 0 | 0 |
| Data29 | D29 | L | | 0 | 0 |
| Data30 | D30 | L | | 0 | 0 |
| Data31 | D31 | H | ↧ | 0 | 0 |

# ARM Memory Standard



| MSByte | MSByte-1 | LSByte+1 | LSByte |
|---|---|---|---|
| Word at address A | | | |
| Halfword at address (A+2) | | Halfword at address A | |
| Byte at address (A+3) | Byte at address (A+2) | Byte at address (A+1) | Byte at address A |

**Figure A3-3 Little-endian memory system**

ARMv7-M Architecture Reference Manual, pg 67

# ARM Memory VS BPM Memory

ARM:

Any given address -> 8 bits

Prefers 32 bit data

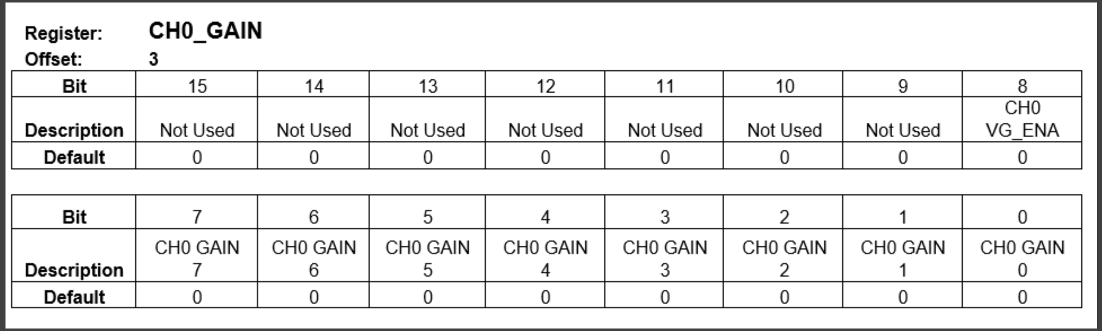
BPM:

Any given address -> 16 bits

Prefers 16 bit data

We handle this by dropping the LSB of the ARM address when translating to BPM

0x40400006 (AFE0 Gain 0)
drop upper bits -> 0b0000_0000_0110
shift >> 0b0000_0000_0011 which matches ->



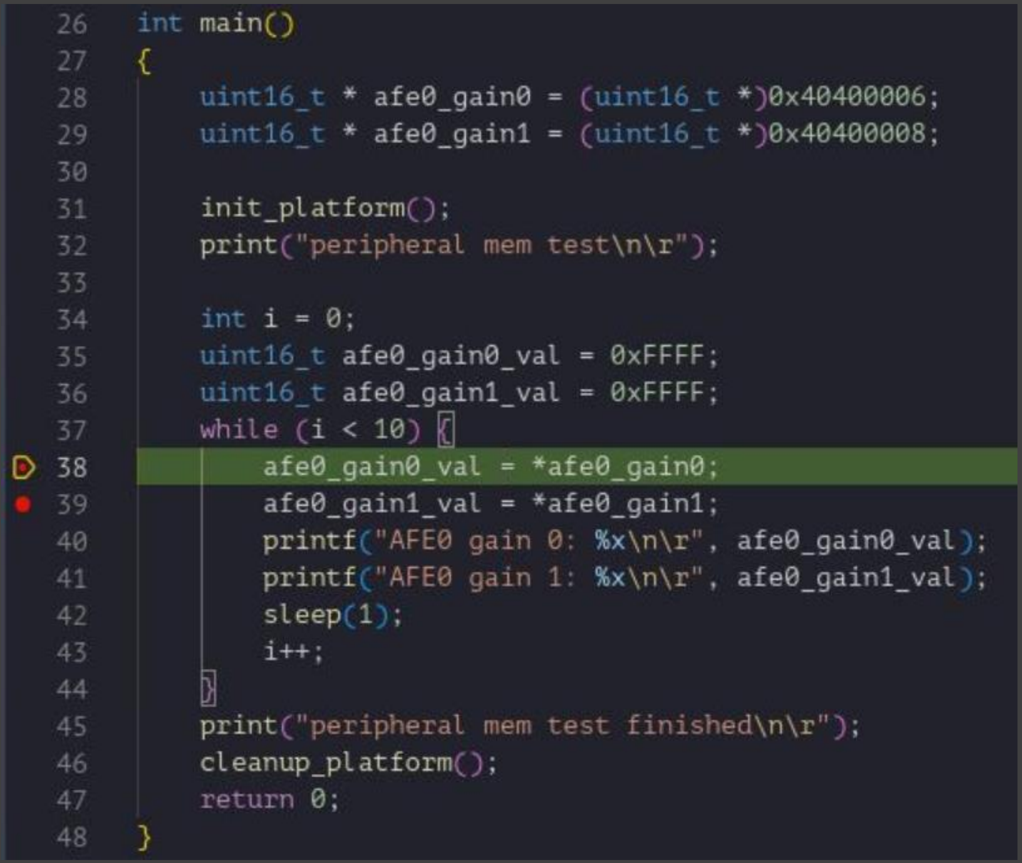
**Register:** **CH0_GAIN**
**Offset:** 3

| **Bit** | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| **Description** | Not Used | Not Used | Not Used | Not Used | Not Used | Not Used | Not Used | CH0 VG_ENA |
| **Default** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

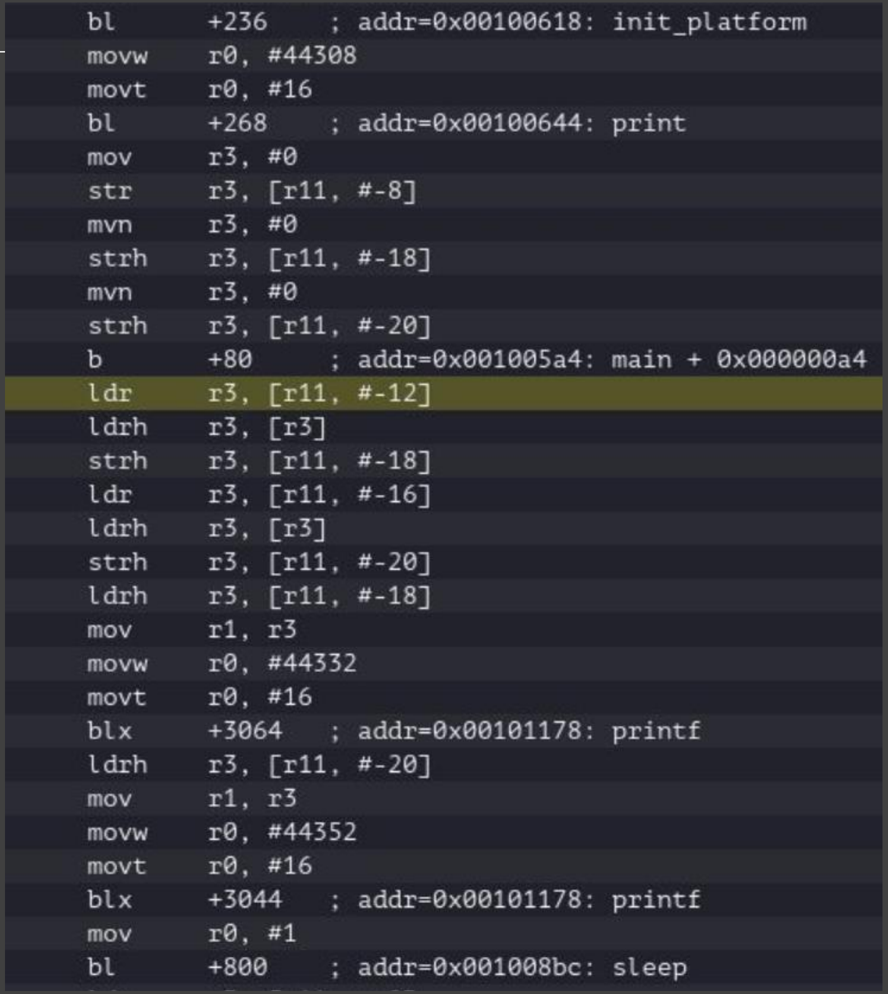
| **Bit** | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| **Description** | CH0 GAIN 7 | CH0 GAIN 6 | CH0 GAIN 5 | CH0 GAIN 4 | CH0 GAIN 3 | CH0 GAIN 2 | CH0 GAIN 1 | CH0 GAIN 0 |
| **Default** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

# ARM Non-word Accesses

- ARM has instructions for smaller accesses

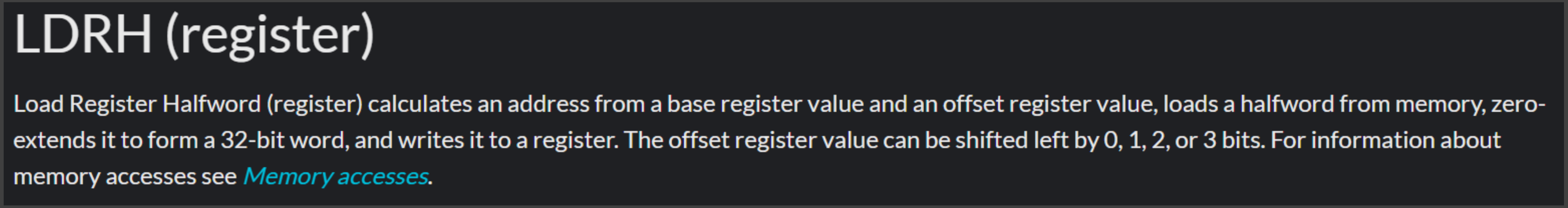


```c
int main()
{
    uint16_t * afe0_gain0 = (uint16_t *)0x40400006;
    uint16_t * afe0_gain1 = (uint16_t *)0x40400008;

    init_platform();
    print("peripheral mem test\n\r");

    int i = 0;
    uint16_t afe0_gain0_val = 0xFFFF;
    uint16_t afe0_gain1_val = 0xFFFF;
    while (i < 10) {
        afe0_gain0_val = *afe0_gain0;
        afe0_gain1_val = *afe0_gain1;
        printf("AFE0 gain 0: %x\n\r", afe0_gain0_val);
        printf("AFE0 gain 1: %x\n\r", afe0_gain1_val);
        sleep(1);
        i++;
    }
    print("peripheral mem test finished\n\r");
    cleanup_platform();
    return 0;
}
```



```
bl      +236    ; addr=0x00100618: init_platform
movw    r0, #44308
movt    r0, #16
bl      +268    ; addr=0x00100644: print
mov     r3, #0
str     r3, [r11, #-8]
mvn     r3, #0
strh    r3, [r11, #-18]
mvn     r3, #0
strh    r3, [r11, #-20]
b       +80     ; addr=0x001005a4: main + 0x000000a4
ldr     r3, [r11, #-12]
ldrh    r3, [r3]
strh    r3, [r11, #-18]
ldr     r3, [r11, #-16]
ldrh    r3, [r3]
strh    r3, [r11, #-20]
ldrh    r3, [r11, #-18]
mov     r1, r3
movw    r0, #44332
movt    r0, #16
blx     +3064   ; addr=0x00101178: printf
ldrh    r3, [r11, #-20]
mov     r1, r3
movw    r0, #44352
movt    r0, #16
blx     +3044   ; addr=0x00101178: printf
mov     r0, #1
bl      +800    ; addr=0x001008bc: sleep
```

## ARM LDRH



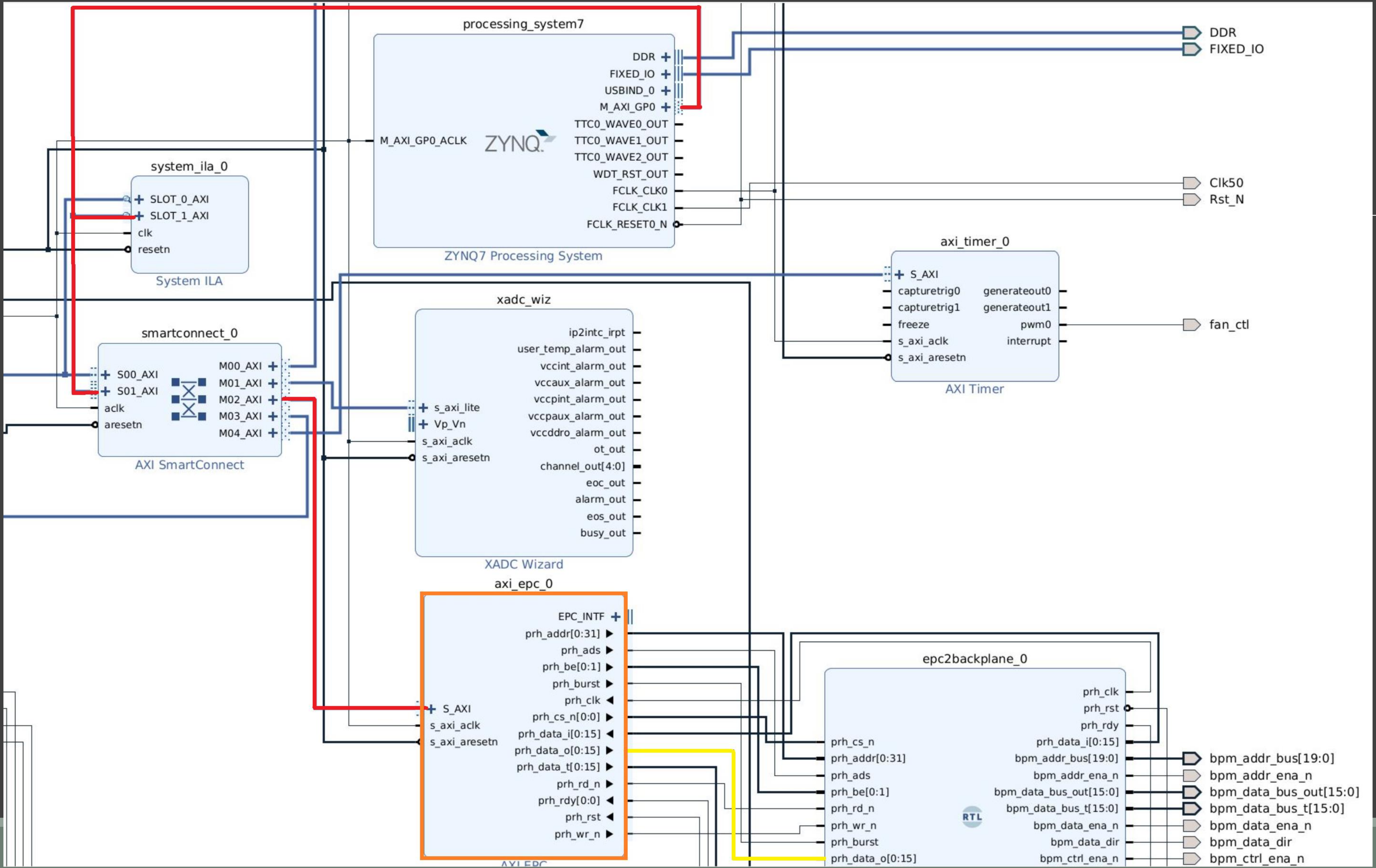
LDRH (register)

Load Register Halfword (register) calculates an address from a base register value and an offset register value, loads a halfword from memory, zero-extends it to form a 32-bit word, and writes it to a register. The offset register value can be shifted left by 0, 1, 2, or 3 bits. For information about memory accesses see *Memory accesses*.

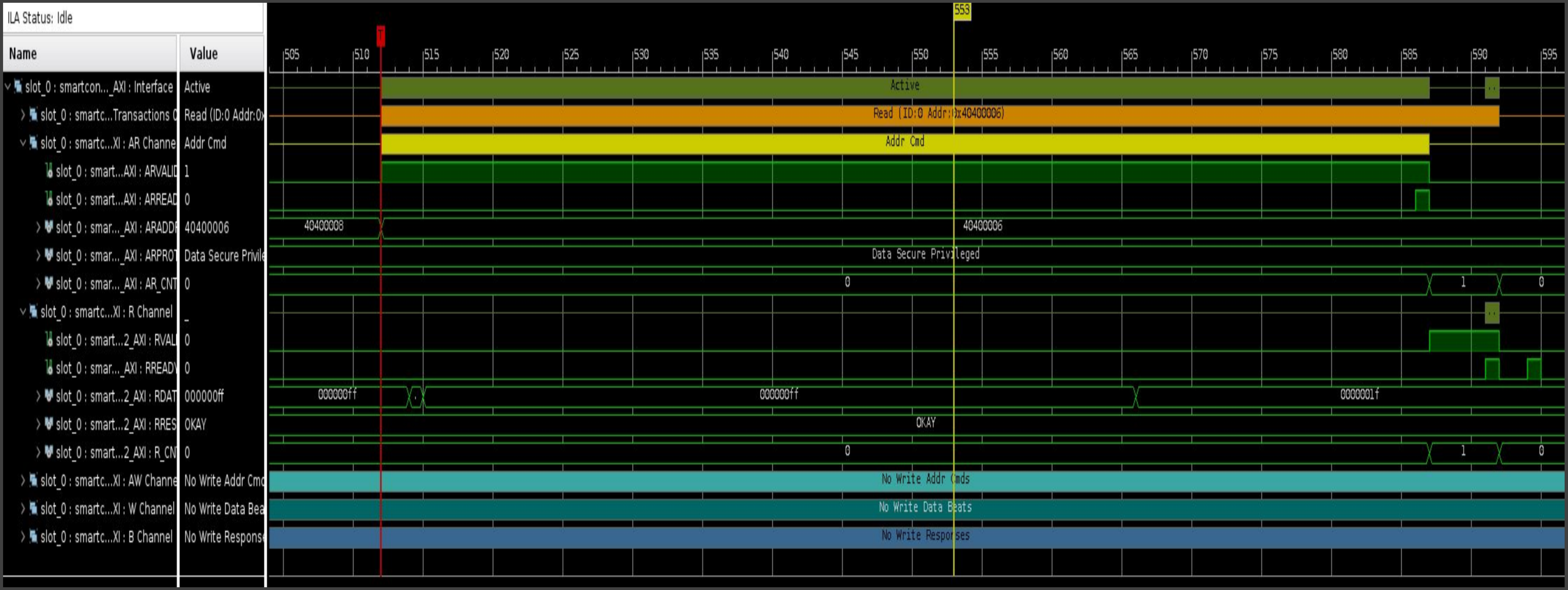
Note the “loads halfword from memory” is ambiguous

In reality* it’s still expecting 32 bit data, which is then shifted into position and zero-extended

https://developer.arm.com/documentation/ddi0406/c/Application-Level-Architecture/Instruction-Details/Alphabetical-list-of-instructions/LDRH--register-?lang=en


processing_system7
DDR
FIXED_IO
USBIND_0
M_AXI_GP0
M_AXI_GP0_ACLK
ZYNQ
TTC0_WAVE0_OUT
TTC0_WAVE1_OUT
TTC0_WAVE2_OUT
WDT_RST_OUT
FCLK_CLK0
FCLK_CLK1
FCLK_RESET0_N
ZYNQ7 Processing System
DDR
FIXED_IO
Clk50
Rst_N
system_ila_0
SLOT_0_AXI
SLOT_1_AXI
clk
resetn
System ILA
axi_timer_0
S_AXI
capturetrig0
capturetrig1
freeze
s_axi_aclk
s_axi_aresetn
generateout0
generateout1
pwm0
interrupt
AXI Timer
fan_ctl
smartconnect_0
S00_AXI
S01_AXI
aclk
aresetn
M00_AXI
M01_AXI
M02_AXI
M03_AXI
M04_AXI
AXI SmartConnect
xadc_wiz
s_axi_lite
Vp_Vn
s_axi_aclk
s_axi_aresetn
ip2intc_irpt
user_temp_alarm_out
vccint_alarm_out
vccaux_alarm_out
vccpint_alarm_out
vccpaux_alarm_out
vccddro_alarm_out
ot_out
channel_out[4:0]
eoc_out
alarm_out
eos_out
busy_out
XADC Wizard
axi_epc_0
EPC_INTF
prh_addr[0:31]
prh_ads
prh_be[0:1]
prh_burst
prh_clk
prh_cs_n[0:0]
prh_data_i[0:15]
prh_data_o[0:15]
prh_data_t[0:15]
prh_rd_n
prh_rdy[0:0]
prh_rst
prh_wr_n
S_AXI
s_axi_aclk
s_axi_aresetn
AXI EPC
epc2backplane_0
prh_clk
prh_rst
prh_rdy
prh_data_i[0:15]
bpm_addr_bus[19:0]
bpm_addr_ena_n
bpm_data_bus_out[15:0]
bpm_data_bus_t[15:0]
bpm_data_ena_n
bpm_data_dir
bpm_ctrl_ena_n
prh_cs_n
prh_addr[0:31]
prh_ads
prh_be[0:1]
prh_rd_n
prh_wr_n
prh_burst
prh_data_o[0:15]
RTL
bpm_addr_bus[19:0]
bpm_addr_ena_n
bpm_data_bus_out[15:0]
bpm_data_bus_t[15:0]
bpm_data_ena_n
bpm_data_dir
bpm_ctrl_ena_n

# AXI Unaligned Access

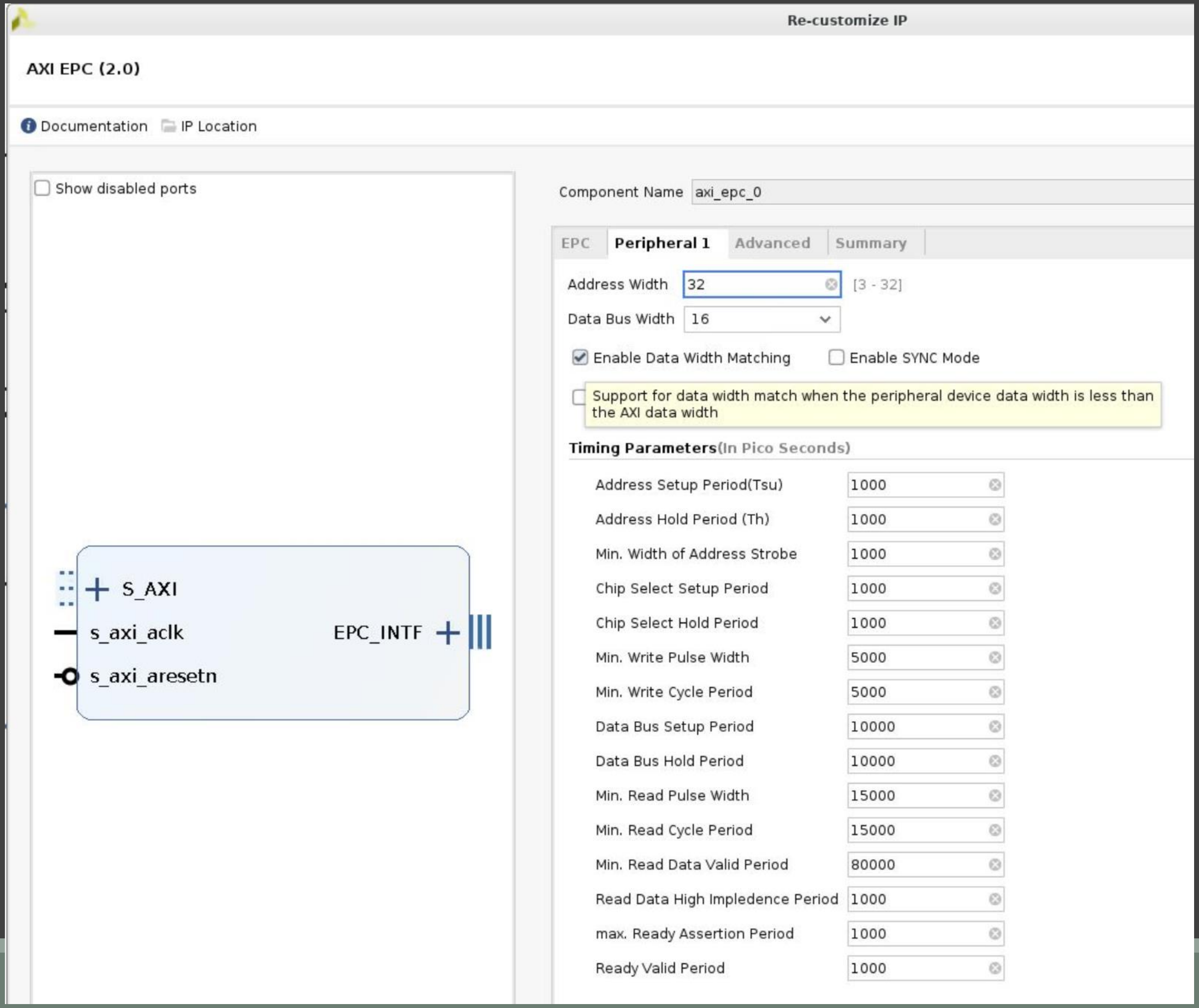


ILA Status: Idle

| Name | Value |
|---|---|
| slot_0 : smartcon..._AXI : Interface | Active |
| slot_0 : smartc...Transactions | Read (ID:0 Addr:0x |
| slot_0 : smartc...XI : AR Channel | Addr Cmd |
| slot_0 : smart...AXI : ARVALID | 1 |
| slot_0 : smart...AXI : ARREADY | 0 |
| slot_0 : smar..._AXI : ARADDR | 40400006 |
| slot_0 : smar..._AXI : ARPROT | Data Secure Privile |
| slot_0 : smar..._AXI : AR_CNT | 0 |
| slot_0 : smartc...XI : R Channel | _ |
| slot_0 : smart...2_AXI : RVALID | 0 |
| slot_0 : smar..._AXI : RREADY | 0 |
| slot_0 : smart...2_AXI : RDATA | 000000ff |
| slot_0 : smart...2_AXI : RRESP | OKAY |
| slot_0 : smart...2_AXI : R_CNT | 0 |
| slot_0 : smartc...XI : AW Channel | No Write Addr Cmd |
| slot_0 : smartc...XI : W Channel | No Write Data Bea |
| slot_0 : smartc...XI : B Channel | No Write Respons |

# External Peripheral Controller

- Adapter between AXI and our logic
- Supports "Data Width Matching"



Component Name axi_epc_0

Address Width 32 [3 - 32]

Data Bus Width 16

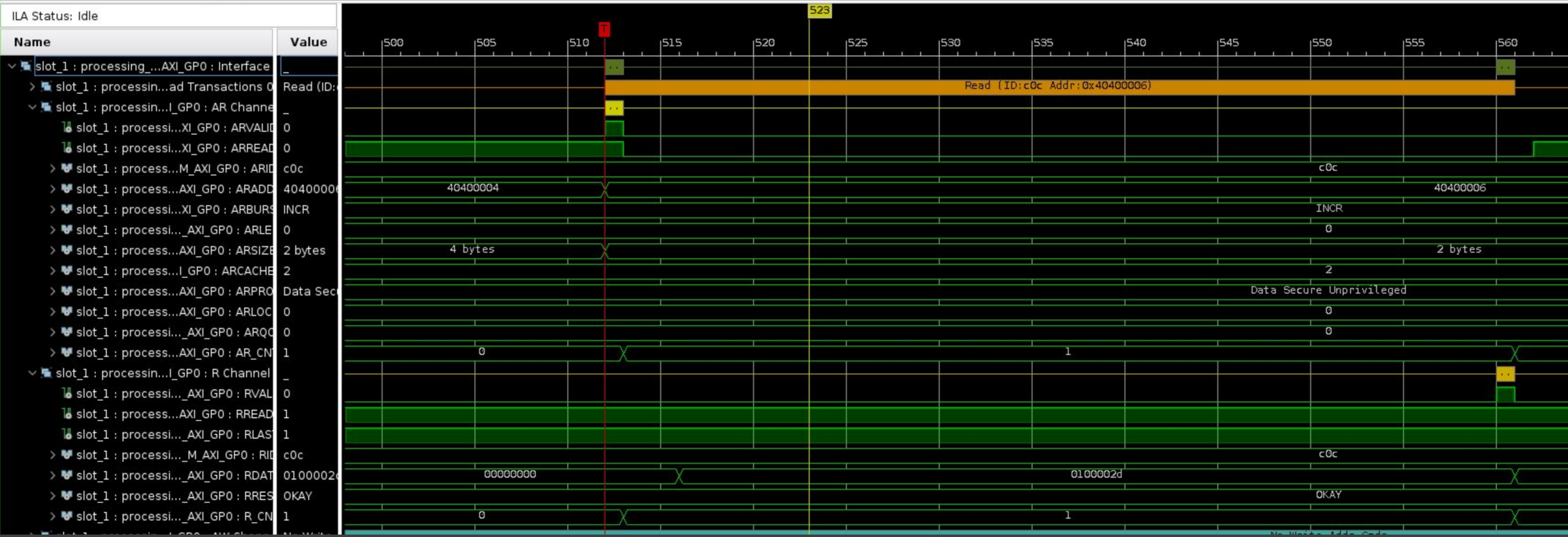
[✓] Enable Data Width Matching   [ ] Enable SYNC Mode

Support for data width match when the peripheral device data width is less than the AXI data width

**Timing Parameters**(In Pico Seconds)

| | |
|---|---|
| Address Setup Period(Tsu) | 1000 |
| Address Hold Period (Th) | 1000 |
| Min. Width of Address Strobe | 1000 |
| Chip Select Setup Period | 1000 |
| Chip Select Hold Period | 1000 |
| Min. Write Pulse Width | 5000 |
| Min. Write Cycle Period | 5000 |
| Data Bus Setup Period | 10000 |
| Data Bus Hold Period | 10000 |
| Min. Read Pulse Width | 15000 |
| Min. Read Cycle Period | 15000 |
| Min. Read Data Valid Period | 80000 |
| Read Data High Impledence Period | 1000 |
| max. Ready Assertion Period | 1000 |
| Ready Valid Period | 1000 |

# Data Width Matched Trace



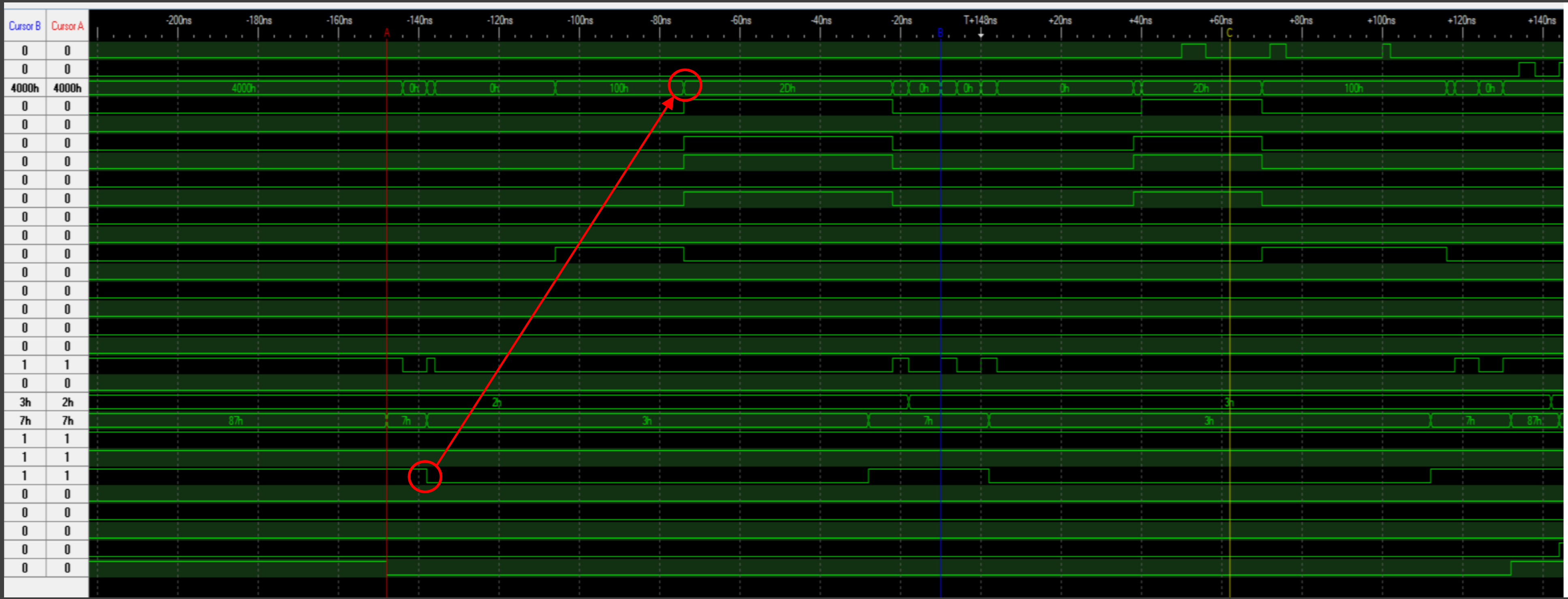
ILA Status: Idle

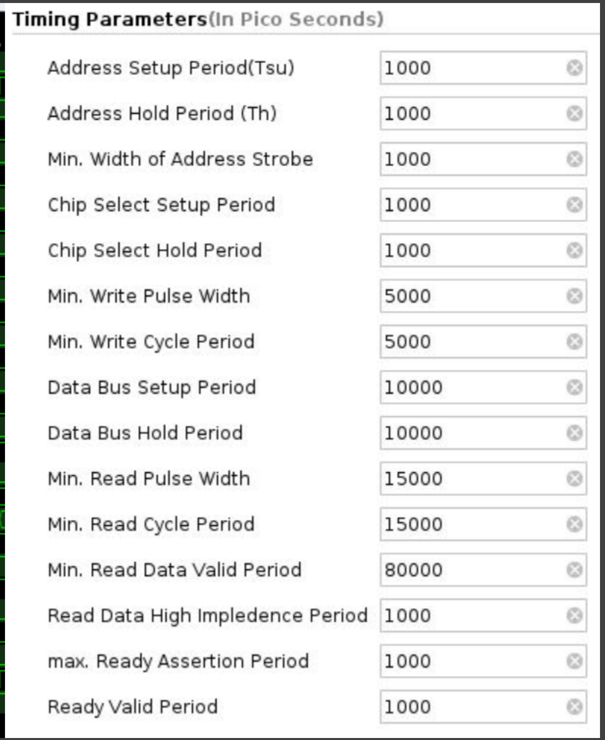
| Name | Value |
|---|---|
| slot_1 : processing_...AXI_GP0 : Interface | _ |
| slot_1 : processin...ad Transactions 0 | Read (ID: |
| slot_1 : processin...I_GP0 : AR Channe | _ |
| slot_1 : processi...XI_GP0 : ARVALID | 0 |
| slot_1 : processi...XI_GP0 : ARREAD | 0 |
| slot_1 : process...M_AXI_GP0 : ARID | c0c |
| slot_1 : process...AXI_GP0 : ARADD | 4040000 |
| slot_1 : processi...XI_GP0 : ARBURS | INCR |
| slot_1 : processi..._AXI_GP0 : ARLE | 0 |
| slot_1 : process...AXI_GP0 : ARSIZE | 2 bytes |
| slot_1 : process...I_GP0 : ARCACHE | 2 |
| slot_1 : process...AXI_GP0 : ARPRO | Data Sec |
| slot_1 : process...AXI_GP0 : ARLOC | 0 |
| slot_1 : processi..._AXI_GP0 : ARQO | 0 |
| slot_1 : process...AXI_GP0 : AR_CN | 1 |
| slot_1 : processin...I_GP0 : R Channel | _ |
| slot_1 : processi..._AXI_GP0 : RVAL | 0 |
| slot_1 : process...AXI_GP0 : RREAD | 1 |
| slot_1 : processi..._AXI_GP0 : RLAST | 1 |
| slot_1 : processi..._M_AXI_GP0 : RID | c0c |
| slot_1 : processi..._AXI_GP0 : RDAT | 0100002 |
| slot_1 : processi..._AXI_GP0 : RRES | OKAY |
| slot_1 : processi..._AXI_GP0 : R_CN | 1 |

## Register Access Summary

- ARM expects 32 bit data to manipulate into 16 bit
- AXI doesn’t care one way or another
- EPC, if configured so, will interpret an AXI transaction to accommodate ARM
- BPM only provides 16 bits per transaction
- So, we do 2 transactions to BPM for any single AXI transaction

# While we’re here…





**Timing Parameters**(In Pico Seconds)

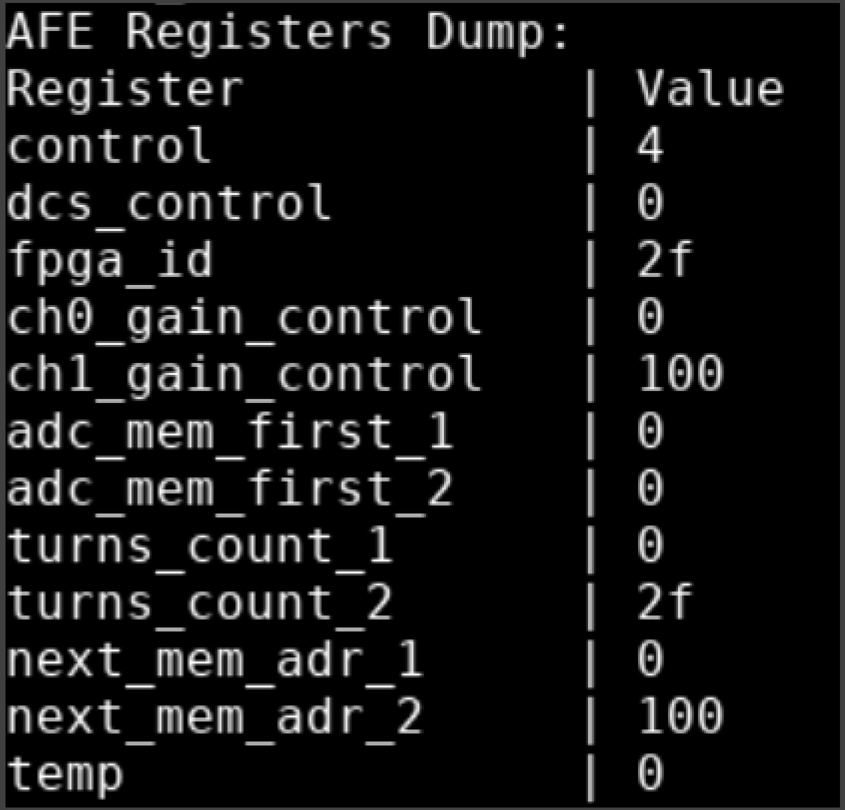
| | |
|---|---|
| Address Setup Period(Tsu) | 1000 |
| Address Hold Period (Th) | 1000 |
| Min. Width of Address Strobe | 1000 |
| Chip Select Setup Period | 1000 |
| Chip Select Hold Period | 1000 |
| Min. Write Pulse Width | 5000 |
| Min. Write Cycle Period | 5000 |
| Data Bus Setup Period | 10000 |
| Data Bus Hold Period | 10000 |
| Min. Read Pulse Width | 15000 |
| Min. Read Cycle Period | 15000 |
| Min. Read Data Valid Period | 80000 |
| Read Data High Impledence Period | 1000 |
| max. Ready Assertion Period | 1000 |
| Ready Valid Period | 1000 |

Critical Parameters are Read Data Valid Period and Data Bus Hold Period

Before: ~1.6 us     Now: ~250 ns

## Other Issues

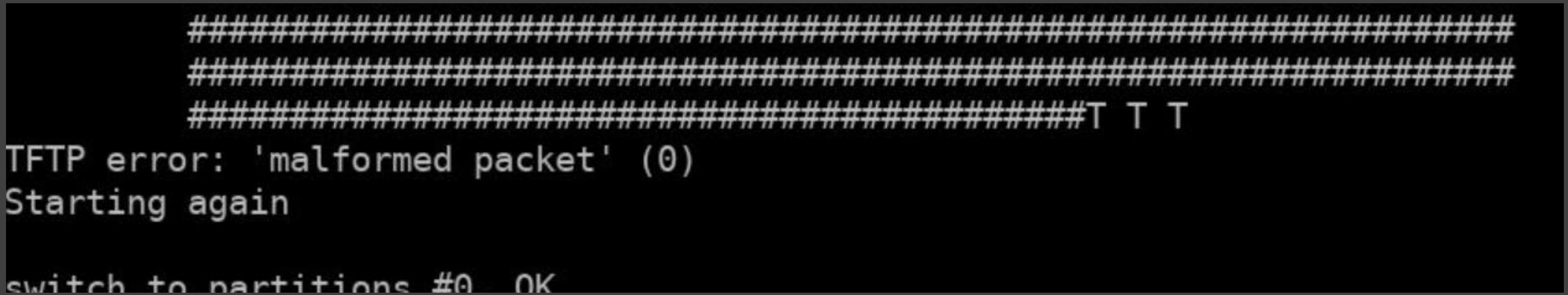
- Coincident with the memory alignment problem
- Seems like the upper address bits are stuck
- Swapped the backplane board, works now
- Investigation ongoing



```
AFE Registers Dump:
Register            | Value
control             | 4
dcs_control         | 0
fpga_id             | 2f
ch0_gain_control    | 0
ch1_gain_control    | 100
adc_mem_first_1     | 0
adc_mem_first_2     | 0
turns_count_1       | 0
turns_count_2       | 2f
next_mem_adr_1      | 0
next_mem_adr_2      | 100
temp                | 0
```

# Other Issues (2)

- Finally caught on my local prototype via serial output
- Malformed packet interrupts tftp load, doesn’t recover gracefully
- bootloader needs polish anyway, can be worked around for now



```
         #################################################################
         #################################################################
         ############################################T T T
TFTP error: 'malformed packet' (0)
Starting again

switch to partitions #0, OK
```

## Next Targets

- Time sweep code
- AFE buffer test code
- Heartbeat code to know when a module is up