

We find unexpected behavior in below code pieces.

```

2400     if(sInvDuty.f32Phase[1] > 0)
2401     {
2402         if(RTDB.sInst.f32InvCurr[1] > 0)
2403         {
2404             sInvDuty.f32Phase[1] += RTDB.sTestRef.f32DBAdjust;
2405         }
2406         else
2407         {
2408             sInvDuty.f32Phase[1] -= RTDB.sTestRef.f32DBAdjust;
2409         }
2410         if(sInvDuty.f32Phase[1] < 0)
2411         {
2412             sInvDuty.f32Phase[1] = 0;
2413         }
2414         EPwm10Regs.CMPA.bit.CMPA = 0;
2415         EPwm9Regs.CMPA.bit.CMPA = (sInvDuty.f32Phase[1]) * EPwm9Regs.TBPRD;
2416     }
2417     else
2418     {
2419         if(RTDB.sInst.f32InvCurr[1] > 0)
2420         {
2421             sInvDuty.f32Phase[1] += RTDB.sTestRef.f32DBAdjust;
2422         }
2423         else
2424         {
2425             sInvDuty.f32Phase[1] -= RTDB.sTestRef.f32DBAdjust;
2426         }
2427         if(sInvDuty.f32Phase[1] > 0)
2428         {
2429             sInvDuty.f32Phase[1] = 0;
2430         }
2431         EPwm9Regs.CMPA.bit.CMPA = 0;
2432         EPwm10Regs.CMPA.bit.CMPA = (-sInvDuty.f32Phase[1]) * EPwm10Regs.TBPRD;
2433     }

```

程序运行此段代码后出错。

Error happens in this line of code

The last "normal" line of code is 00e04b: E203002A MOV32 @0x2a, R0H

The screenshot shows the debugger interface with the C code on the left and the assembly on the right. The C code is the same as the one provided in the previous block. The assembly window shows the instructions for the code. A red box highlights the line 'sInvDuty.f32Phase[1] -= RTDB.sTestRef.f32DBAdjust;' in the C code, which corresponds to the assembly instruction 'MOV32 @0x2a, R0H' at address 00e04b. A red box also highlights the assembly instruction 'MOV32 @0x2a, R0H' at address 00e04b. A red box highlights the assembly instruction 'MOV32 @0x2a, R0H' at address 00e04b. A red box highlights the assembly instruction 'MOV32 @0x2a, R0H' at address 00e04b.

After that variable f32Phase[1] gets abnormal result as shown in the red part.

The screenshot shows a debugger interface with three main panes: Variables, Disassembly, and Memory Map. In the Variables pane, the variable `f32Phase[1]` is highlighted in red, showing a value of `-224.337006`. A red arrow points from this value to a red box in the Disassembly pane. This box contains the assembly instruction `MOV32 @0x2a, R0H` under the label `CSL286`. To the right of the assembly pane, a red text box contains the Chinese text: "此段代码执行后，数据就出现异常" (After this code is executed, the data becomes abnormal).

We also find sentence "`sInvDuty.f32Phase[1] += RTDB.sTestRef.f32DBAdjust;`" in two different places have different disassembly code. As shown below, one and two with circle.

The screenshot shows a C code snippet with two instances of the statement `sInvDuty.f32Phase[1] -= RTDB.sTestRef.f32DBAdjust;` circled in red and labeled ① and ②. To the right of the code, a red text box contains the Chinese text: "对于①、②两个相同的C语言，对应的汇编不一样。不知道是不是此处的异常。具体请看汇编指令。" (For ① and ②, which are two identical C statements, the corresponding assembly is different. I don't know if this is the abnormal result. Please see the assembly instructions for details.)

Corresponding disassembly code are shown below in details. They differences are marked in the red circle.

```

000TeT: 053B SB C$L285, LEQ
2402 if(RTDB.sInst.f32InvCurr[1] > 0)
00dff0: 761F0441 MOVW DP, #0x441
00dff2: E2AF0002 MOV32 R0H, @0x2, UNCF
00dff4: E5A0 CMPF32 R0H, #0.0
00dff5: AD14 MOVST0 NF,ZF
00dff6: 6510 SB C$L282, LEQ
2404 sInvDuty.f32Phase[1] += RTDB.sTestRef.f32DBAdjust;
00dff7: 761F0468 MOVW DP, #0x468
00dff9: E2AF002A MOV32 R0H, @0x2a, UNCF
00dffb: 761F0455 MOVW DP, #0x455
00dffd: E2AF0128 MOV32 R1H, @0x28, UNCF
00dfff: E7100040 ADDF32 R0H, R0H, R1H
00e001: 761F0468 MOVW DP, #0x468
00e003: E203002A MOV32 @0x2a, R0H
00e005: 6F0E SB C$L283, UNC
2408 sInvDuty.f32Phase[1] -= RTDB.sTestRef.f32DBAdjust;
C$L282:
00e006: 761F0455 MOVW DP, #0x455
00e008: E2AF0028 MOV32 R0H, @0x28, UNCF ①
00e00a: 761F0468 MOVW DP, #0x468
00e00c: E2AF012A MOV32 R1H, @0x2a, UNCF
00e00e: E7200008 SUBF32 R0H, R1H, R0H
00e010: 7700 NOP
00e011: E203002A MOV32 @0x2a, R0H
2410 if(sInvDuty.f32Phase[1] < 0)
C$L283:
00e013: E5A0 CMPF32 R0H, #0.0
00e014: AD14 MOVST0 NF,ZF
00e015: 6304 SB C$L284, GEQ
2412 sInvDuty.f32Phase[1] = 0;
00e016: E590 ZERO R0
00e017: E203002A MOV32 @0x2a, R0H
2414 EPwm10Regs.CMPA.bit.CMPA = 0;
C$L284:
00e019: 761F0125 MOVW DP, #0x125
00e01b: 2B2B MOV @0x2b, #0
2415 EPwm9Regs.CMPA.bit.CMPA = (sInvDuty.f32Phase[1]) * EPwm9Regs.TBPRD;

```

```

2419      if(RTDB.sInst.f32InvCurr[1] > 0)
C$L285:
00e02a: 761F0441  MOVW      DP, #0x441
00e02c: E2AF0002  MOV32     R0H, @0x2, UNCF
00e02e: E5A0      CMPF32     R0H, #0.0
00e02f: AD14      MOVST0    NF,ZF
00e030: 6510      SB         C$L286, LEQ
2421      sInvDuty.f32Phase[1] += RTDB.sTestRef.f32DBAdjust;
00e031: 761F0468  MOVW      DP, #0x468
00e033: E2AF002A  MOV32     R0H, @0x2a, UNCF
00e035: 761F0455  MOVW      DP, #0x455
00e037: E2AF0128  MOV32     R1H, @0x28, UNCF
00e039: E7100040  ADDF32    R0H, R0H, R1H
00e03b: 761F0468  MOVW      DP, #0x468
00e03d: E203002A  MOV32     @0x2a, R0H
00e03f: 6F0E      SB         C$L287, UNC
2425      sInvDuty.f32Phase[1] -= RTDB.sTestRef.f32DBAdjust;
C$L286:
00e040: 0001      ABORTI
00e041: 0455      SUB        ACC, *-SP[21] << 16
00e042: E2AF0028  MOV32     R0H, @0x28, UNCF
00e044: 761F0468  MOVW      DP, #0x468
00e046: E2AF012A  MOV32     R1H, @0x2a, UNCF
00e048: E7200008  SUBF32    R0H, R1H, R0H
00e04a: 7700      NOP
00e04b: E203002A  MOV32     @0x2a, R0H
2427      if(sInvDuty.f32Phase[1] > 0)
C$L287:
00e04d: E5A0      CMPF32     R0H, #0.0
00e04e: AD14      MOVST0    NF,ZF
00e04f: 6504      SB         C$L288, LEQ
2429      sInvDuty.f32Phase[1] = 0;
00e050: E590      ZERO      R0
00e051: E203002A  MOV32     @0x2a, R0H
2431      EPwm9Regs.CMPA.bit.CMPA = 0;
C$L288:
00e053: 761F0121  MOVW      DP, #0x121
00e055: 2B2B      MOV        @0x2b, #0
2432      EPwm10Regs.CMPA.bit.CMPA = (-sInvDuty.f32Phase[1]) * EPwm10Regs.TBPRD;

```

此处多了这两行汇编