C) PF 2 is incorrect because this line of code indicates a causal dependency. Originally only PF1 would be toggled, but now the entire PortF is being toggled, which may or may not overwrite the interrupts toggle of PF2. Therefore, PF2 is incorrect.

In the original code, the program will toggle r0 by continuously loading the previous value from memory into r0, and will EOR that value with 0x02. Then it will use R1 as a placeholder to store this value back into the memory location that R0 accesses in the next cycle of the program

0x0000134C 480E      LDR           r0,[pc,#56]  ; @0x00001388

0x0000134E 6800      LDR           r0,[r0,#0x00]

0x00001350 F0800002  EOR           r0,r0,#0x02

0x00001354 490B      LDR           r1,[pc,#44]  ; @0x00001384

0x00001356 F8C103FC  STR           r0,[r1,#0x3FC]

In the new code, the value of R0 is constantly incrementing, and this is due to the retrieval process not retrieving the correct new R0, which should always be 0x00 or 0x02. In this case, the value of R0 increments till it is 0x08. This is not the proper way to toggle an led because the other bits, may have some significance elsewhere.

Other than bit specific addressing, another method to prevent this causal dependency would be to initiate an atomic portion of this chunk of code, so that only one thread may access this portion on the read-write phase.