

What is an Invalid Memory Reference?

Author: Ben Kropf

Associated Project: No

Associated Part Family: CY8C21x23, CY8C21x34, CY8C24x23A, CY8C27x43, CY8C29x66

Software Version: PSoC Designer 4.2

Associated Application Notes: None

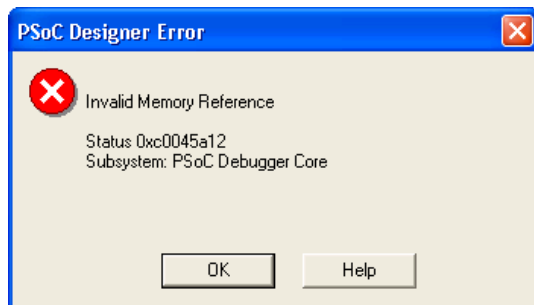
Abstract

Users of the PSoC Designer™ debugger may occasionally encounter an Invalid Memory Reference error. There are multiple causes for this error message. This Application Note covers possible causes for the error, ways to diagnose the specific cause, and methods for fixing the root cause of the error.

Introduction

The Invalid Memory Reference (IMR) error occurs when the internal states of the In-Circuit Emulator (ICE) and Pod diverge. There are multiple possible causes for this divergence. An IMR is reported when the PSoC Designer debugger detects a mismatch between the emulated M8C registers and SRAM in the ICE-base station, and the registers and SRAM in the PSoC® device on the emulation Pod. You can think of an IMR as the result of the ICE losing track of what the Pod is doing.

Figure 1. Invalid Memory Reference Dialog Box



The IMR errors must be fixed before you can continue debugging. Many reasons why IMR errors occur are fixed in the most recent versions of PSoC Designer and current generations of the PSoC microprocessor. You can work around IMRs or prevent the remaining IMR issues using proper methodology in project code. An IMR does not necessarily imply that your application will contain faults when programmed into a part outside of the debugging environment.

General Considerations

Use version 4.2 or later of PSoC Designer. Earlier versions contained bugs that led to Invalid Memory Reference errors. Also make sure that you are not using obsolete PSoC devices.

PLL Issues

Using the Phase Locked Loop (PLL) while debugging may prevent the debugger from operating correctly. To avoid this error, debug with the PLL off. To turn the PLL off, choose the Interconnect View in the Device Editor of PSoC Designer and in the Global Resources section set PLL_Mode to Disable. Make sure to disconnect any external crystal hardware you may have attached to the PSoC device.

Sleep Issues

Use care when debugging code that sets the sleep bit.

- Do not attempt to use the debugger to step over an IO instruction that sets the sleep bit. This will cause an invalid memory reference.
- Do not attempt to set a breakpoint immediately after an IO instruction that sets the sleep bit. This will cause an invalid memory reference.

System Supervisor Call Issues

Follow these guidelines when executing the system supervisor call (SSC) instruction.

- Do not attempt to execute the system supervisor call (SSC) instruction without using the `SSC_Action(OpCode)` macro. The macro is provided in the `m8ssc.inc` file that can be included in your PSoC projects. If you do not use this macro, it is very likely that you will generate an IMR error.
- If you set a breakpoint immediately after either the `SSC_Action(OpCode)` macro or an SSC instruction, the breakpoint will be skipped and the program will not halt at that instruction.
- It is not necessary to use SSC instructions or the `SSC_Action(OpCode)` macro to access the SROM. Many functions are provided in the API libraries that implement SROM access functionality.

See the Assembler Includes section for instructions on how to include `m8ssc.inc`.

Flash Read Issues

If you are using a CY8C27x43 device and your code erases part of the Flash and does not rewrite it with definite data, then you will get an IMR error when you try to read that part of the Flash with the `ROMX` instruction. PSoC Designer initializes all of the Flash memory with data, even if the project code does not require the entire program memory space. However, in all CY8C27x43 devices, Flash memory does not get reinitialized to a definite value after it is erased.

CY8C21xxx/CY8C24x23A/CY8C29x66 device families do not have this problem as they use a Flash memory that is defined as 0 after it is erased.

If the `ReadBlock` SROM function is used to read Flash data into RAM, then no error will occur, even if the Flash is erased and still contains indefinite data. It only occurs with a read from Flash using the `ROMX` instruction.

To prevent these errors when debugging a CY8C27x43 device, either use the functions described in the `flashblock.inc` file or the E2PROM User Module. Using either of these mechanisms makes it much easier to erase, read, and write data to the Flash without having to worry about IMR errors.

Power Issues

Some IMRs are caused by power fluctuations in the Pod or target system. If a project has two conflicting drivers for the same global line (e.g., digital block A drives a global line low and digital block B drives it high), the resulting power drain can collapse the Pod power rails. This causes the Pod to crash and an IMR to result. The same thing can happen with a Pod in a system with noisy power. For example, a system with a marginal power supply and an electric motor can have IMRs when the motor turns on.

Assembler Includes

Assembly language include files, such as `m8ssc.inc` and `flashblock.inc`, are located in the directory where you installed PSoC Designer. For example, if you are using a CY8C27x43 PSoC device, then the include files for that particular family are located in the following directory path:

```
...\PSoC Designer\tools\include\CY8C27000\
```

If you are using a different device family, then you just change the last folder of the directory path to be a different family, such as CY8C29000. To include these files using assembly language, type `INCLUDE "flashblock.inc"` at the top of your `main.asm` file.

Conclusion

Many of the causes for getting Invalid Memory Reference errors are fixed in the most recent version of PSoC Designer. However, there are still some ways to cause the ICE to lose track of what has happened in the Pod. There is always a reason why these errors occur and there is always a way to prevent these errors. The best approach is to prevent them before they occur.

About the Author

Name: Ben Kropf

Title: Co-op Bachelor
Cypress Semiconductor

Background: Currently an electrical engineering senior at Seattle Pacific University. Kropf is using the summer break to work as an intern for Cypress Semiconductor at their Washington Design Center. Specializing in rewriting technical documents, doing tech. support, and fetching coffee at this point in time. Check back in a few years and this section will be longer.

Contact: btk@cypress.com

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709
Phone: 408-943-2600
Fax: 408-943-4730
<http://www.cypress.com>

© Cypress Semiconductor Corporation, 2006. The information contained herein is subject to change without notice. Cypress Semiconductor Corporation assumes no responsibility for the use of any circuitry other than circuitry embodied in a Cypress product. Nor does it convey or imply any license under patent or other rights. Cypress products are not warranted nor intended to be used for medical, life support, life saving, critical control or safety applications, unless pursuant to an express written agreement with Cypress. Furthermore, Cypress does not authorize its products for use as critical components in life-support systems where a malfunction or failure may reasonably be expected to result in significant injury to the user. The inclusion of Cypress products in life-support systems application implies that the manufacturer assumes all risk of such use and in doing so indemnifies Cypress against all charges.