

Local Storage Pointers

INTRODUCTION

Local storage describes data that is stored local to an object. In relation to an RTOS, it often refers to Thread/Task Local Storage (TLS) and describes data that is stored in the task control block (TCB).

LOCAL STORAGE POINTERS

Global variables are often used to store data across function calls. If several tasks presently used code that writes to the same global variables, previously stored data could be overwritten before it was read. To address this problem TLS acts like global variables that are private to the Task. That means every Task can have its private set of “global” variables that only this Task may use. In this sense, the use of TLS promotes code reusability, as the Task code can refer to the TLS to reference the Tasks local data instead of system global data. Also the association of the data to the Task is much clearer.

EXAMPLE

As an example, consider a system with several PI controllers to control several processes. Each PI controller contains an integrator value which must be stored for the next controller step to be calculated. Figure 1 shows the global variable approach. It is shown that every task needs to access the correct integrator value in the global variable space. Therefore each controller task needs a specific implementation that uses the correct references.

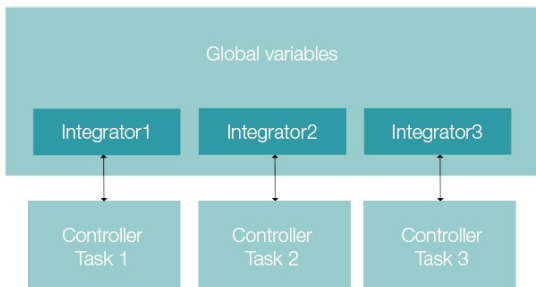


Figure 1. Example with global variables



Figure 2. Example with TLS

Figure 2 shows the same system but with the usage of TLS. Each controller Task could have the same implementation of the control algorithm that is using the Task local storage to reference its integrator value.

LOCAL STORAGE POINTERS AND SAFERTOS®

To allow the application designer the most flexible local storage interface, SAFERTOS® implements it as a void pointer. It is therefore called a local storage pointer. This allows for each task to have its own unique data object. In SAFERTOS® not only do tasks have local storage pointers but also timers, where a void pointer to the timer local storage is stored in the timer control block. Timer local storage pointers behave the same as task local storage pointers.

How the data is stored depends on the size of the data and the application design. The size of a void pointer is architecture dependent, but usually on a 32-bit architecture a void pointer has a size of 32 bits or 4 bytes. That means if the size of the data to be stored in the TLS is 4 bytes or less, it can be directly saved in the storage pointer. If 4 bytes is not enough, the data object must be pre-allocated in memory (SAFERTOS® does not allow dynamic memory allocation) and the pointer to the data object can be saved in the TLS pointer.

By using a void pointer, the data stored at the end of the pointer is not predefined but must be defined by the application designer. It can be something trivial as an array of integers or some more complex data types, like structures.

The local storage pointer is set up at the time the task or timer is created via the task or timer parameter structure. To access the data, the local storage pointer can be retrieved from the TCB via an API function(`pvTaskTLSObjectGet()`, `pvTimerTLSObjectGet()`) which returns the void pointer.

WITTENSTEIN high integrity systems

Worldwide Sales and Support

Americas: +1 408 625 4712

ROTW: +44 1275 395 600

Email: sales@highintegritysystems.com

Web: www.highintegritysystems.com



WITTENSTEIN