Signalling with SIGF_SINGLE

by John Orr

The ROM Kernel Reference Manuals state that sixteen of a task's 32 signal bits are reserved for the operating system's private use, but, like any good rule, there is an exception. One of these sixteen bits, the `SIGF_SINGLE` bit, can be useful to some applications, if used correctly.

Many system functions need to put their task to sleep while waiting for a single event, which requires using one of the task's signals. Rather than forcing each of these system functions to allocate a signal, then Wait(), then deallocate the signal, the operating system has permanently allocated one signal, the `SIGF_SINGLE`, for this type of signalling. When a system function needs stop a task to Wait() for a single signal, it can use `SIGF_SINGLE`.

The only purpose a program can use `SIGF_SINGLE` for is Wait()ing because the task cannot call any system functions while it is using `SIGF_SINGLE`. A program that calls system functions while using `SIGF_SINGLE` can cause itself and the operating system serious problems because the system functions can use `SIGF_SINGLE` as well. If a program calls a system function while using `SIGF_SINGLE`, two bad things can happen:

> 1) The errant task's event takes place before the system function waits on `SIGF_SINGLE` (or while the system function is waiting on `SIGF_SINGLE`). In this case, the system function will think its event has taken place because its signal became set. The errant task will never find out that its event has taken place, as the system function will clear the `SIGF_SINGLE` bit after Wait()ing on it.

> 2) The errant task's event and the system function's event take place while the system function is waiting on `SIGF_SINGLE`. In this case, the system function will function normally, clear the `SIGF_SINGLE` bit, and exit. The errant task will never know that its event has taken place.

Before Wait()ing on `SIGF_SINGLE`, clear it using SetSignal():

```
SetSignal(0L, SIGF_SINGLE);
```

This step is necessary because it is possible that the last system function that used the `SIGF_SINGLE` signal did not clear the `SIGF_SINGLE` bit.

Also, an application should not wait on other signals while it is waiting on `SIGF_SINGLE`. Waiting on other signals at the same time makes it possible for a program to wake up while the `SIGF_SINGLE` is still outstanding. If this happens, the program will still have to go back to sleep, which requires calling a system function.

SIGF_Single.c is a simple example of using the `SIGF_SINGLE` signal. It starts a child process and waits for that child process to signal the main process using the `SIGF_SINGLE` signal.