

## Re: Thread in driver

---

*Source:*

<http://www.tech-archive.net/Archive/WindowsCE/microsoft.public.windowsce.platbuilder/2008-04/msg00360.html>

---

- *From:* Winston <[winstonhyypia@xxxxxxxxxx](mailto:winstonhyypia@xxxxxxxxxx)>
  - *Date:* Sun, 13 Apr 2008 20:11:25 -0700 (PDT)
- 

On Apr 11, 4:01 pm, Valter Minute

<[v\\_a\\_l\\_t\\_e\\_r\\_m\\_i\\_n\\_u\\_t\\_e@xxxxxxxxxxxxxx](mailto:v_a_l_t_e_r_m_i_n_u_t_e@xxxxxxxxxxxxxx)> wrote:

winstonhy...@xxxxxxxxxx wrote

[innews:12efb42e-b16b-4920-a82b-a87763e0298a@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx](mailto:innews:12efb42e-b16b-4920-a82b-a87763e0298a@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)

:

Actually I would like to create a driver which I can pass a function pointer in. And the driver is able to do the callback whenever there is an interrupt. Here is my program...

This kind of solution will work only if your callback is inside another driver (and in CE 6.0 both driver must be kernel mode drivers or user mode driver of the same "group" and so running inside the same process).

If you do that calling your driver from an application the callback code will not be mapped inside the context of the caller (unless you are in CE 5.0 and the code is inside a DLL that is shared by the driver and the process) and, anyway, it will run in the context of the caller process (device.exe in CE 5, the kernel or user-mode device process in CE 6) and so it won't be able to access your process memory.

If you need to implement such a mechanism, use a named event and set it inside the IST. The app will spin a thread that will wait on the same event and perform the operations that the callback was supposed to perform when the event is set (you can call your current callback inside the thread to reduce code changes).

If you need to way for the callback completion before you can accept a new interrupt you can use another named event that the app will set and the driver will wait on.

If you need to exchange data between the IST and your app you have a wide choice of solutions: shared memory block, a IOCTL implemented by the driver to return information stored inside the IST, a point-to-point message queue (in this case you won't need the named event because the queue can be use to synchronize the IST thread and the

Re: Thread in driver

application thread.

--

Valter Minutewww.fortechembeddedlabs.it  
Training, support and development for Windows CE  
(the reply address of this message is invalid)

Let me show you a different version

File Driver\_Main.c

```
void SomeISTThread()
{
while (TRUE)
{
/* Wait for the IST event to be triggered */
WaitForSingleObject(Someevent, INFINITE);
Callback();
InterruptDone(SomeIntr);
}
}
```

```
BOOL WINAPI DllEntry(HANDLE hInstDll, DWORD dwReason, LPVOID
lpvReserved)
{
SomeThread = CreateThread(NULL, 0,
(LPTHREAD_START_ROUTINE)SomeISTThread, NULL, 0, NULL);
return TRUE;
}
```

File Callback.c

```
UInt callback_number = 1;
```

```
void Callback()
{
DEBUGMSG (TRUE,(_T("Callback callback_number 0x%X.\r\n"),
callback_number));
}
```

```
void Set_Callback(UInt input_number)
{
callback_number = input_number;
DEBUGMSG (TRUE,(_T("Set_Callback callback_number 0x%X.\r\n"),
callback_number));
}
```

I export the function Set\_Callback to the external program. What I observed is I got a break from Set\_Callback which the number equals to the input\_number. When I trigger the interrupt I got a break from

Re: Thread in driver

Re: Thread in driver

Callback as well. However the callback\_number shown in debugger is the initialized number (1 in this case), no matter how I set the number on the application.

.