

Re: How WinCE bootloader is Jumping to SDRAM after copying itself to i

Source:

<http://www.tech-archive.net/Archive/WindowsCE/microsoft.public.windowsce.platbuilder/2006-01/msg00064.html>

- *From:* "Yannick Chamming's [eMVP]" <yhammings_nospam@xxxxxxxxxxx>
 - *Date:* Wed, 4 Jan 2006 07:43:42 +0100
-

The bootloader uses boot.bib file to define its internal address mapping.

If you look at a function named "EverythingRelocate" or something like that in the bootloader source code, you'll see it is using the datas from the bib file through a specific structure to relocate itself in SDRAM.

--

Yannick Chamming's (eMVP)
ADENEO (ADESET)
Windows Embedded Manager
yhammings AT adeset DOT com
<http://www.adeset.com>
Tél : +33 (0)4.72.18.57.77
Fax : +33 (0)4.72.18.57.78

"Peter" <peter> a écrit dans le message de news:

18CC7CF5-B455-47D1-9088-93DBD619C410@xxxxxxxxxxxxxxxxxxxx

> Hi,

>

> I have a thearotical questions. As such, my bootloader is working fine.

>

> Here my query is:

> The Eboot (Lubbock BSP for PXA250), copies itself from Flash to SDRAM.

> Then,

> it calculates the relative PC address of a lable it has to jump. Then,

> loads

> this address to the PC.

> Now, How the compiler knows that this part of the code goes to SDRAM while

> i

> am building a eboot for Flash Address ? Becoz, the compiler is generating

> the

> code for the Flash and the eboot.map file is as follows.

>

> Address Publics by Value Rva+Base Lib:Object

>

Re: How WinCE bootloader is Jumping to SDRAM after copying itself to i

> 0001:00000000 StartUp a0001000 fwp2.obj
> 0001:00000020 Undefined_Handler a0001020 fwp2.obj
> 0001:00000024 SWI_Handler a0001024 fwp2.obj
>
> Please help me to understand.
>
> Thanks & Regards,
> Arockiarajesh Peter

-
- Prev by Date: ***Re: Yet another random FAT corruption posts – WinCE 4.1***
 - Next by Date: ***Re: How to unload a driver ?***
 - Previous by thread: ***How to unload a driver ?***
 - Next by thread: ***Multiple USB Device (IsochTransfer) Problem***
 - Index(es):
 - ◆ ***Date***
 - ◆ ***Thread***