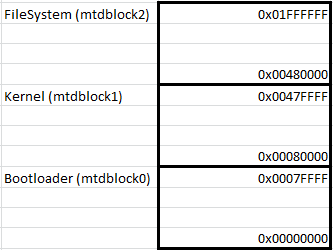
**Übung 2**

**1. Aufgabe**

Partitionierung des Flashs:



**2. Aufgabe**

2.1 Erstellen des Filesystems:

target # mkfs.ext2 /dev/mtdblock2

2.2 Entpacken und kopieren des Inhalts:

target # mkdir /flashfs

target # mount /dev/mtdblock2 /flashfs

host # tar -xjf rootfs\_min2.tbz2

host # cp -r rootfs /root/pxadev/rootfs/

target # cp -a /. /flashfs

In diesem Fall wird cp verwendet, da hier der Inhalt der Dateien in rootfs dupliziert wird und die Besitzer-, Gruppen- und Zugriffsrechte und Erstellungs-, Modifikations- und Zugriffsdaten beibehalten bleiben. Dies geschieht mithilfe des Parameters –a. Der Parameter –r ermöglicht, dass alle Verzeichnisse rekursiv kopiert werden (Unterverzeichnisse eingeschlossen).

Starten von Flash:

Beim Starten muss der Autobootvorgang gestoppt werden und die Environment-Variable bootargs geändert werden:

printenv → bootargs root=/dev/nfs ip=:::::eth0: console=ttyS0,9600n8

setenv bootargs root=/dev/mtdblock2 console=ttyS0,9600n8

saveenv

boot

**3. Aufgabe**

Sourcecode für Programm, dass aktuellen Zustand der Schalter auf LEDs ausgibt:

#include <stdio.h>

#include <fcntl.h>

#include <sys/types.h>

#include <sys/stat.h>

int main()

{

char ledByte;

int gpio = open("/dev/gpio0", O\_RDWR);

if(gpio == 0)

{

printf("Opening of gpio has failed.\n");

return -1;

}

while(1)

{

//read button status

if(read(gpio, &ledByte, 1) != sizeof(ledByte))

{

printf("Reading of gpio has failed.\n");

return -2;

}

//write new led status

if(write(gpio, &ledByte, 1) != sizeof(ledByte))

{

printf("Writing of gpio has failed.\n");

return -3;

}

}

return 0;

}

**4.Aufgabe**

Einbinden des Programms in Root File System:

cp gpio\_test /home/root/gpio

**5.Aufgabe**

Das System wurde so konfiguriert, dass das Kernel-Modul automatisch geladen und im Programm gestartet wird.