Ievads OLIMEX LPC2478-STK

Kjells Enbloms

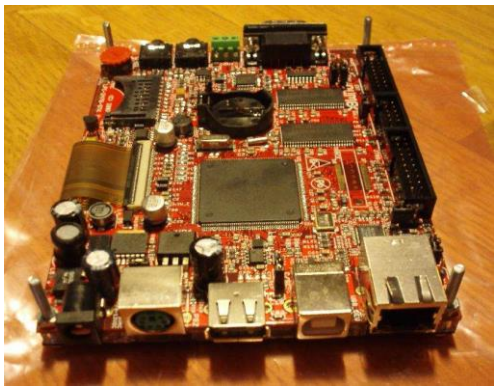
Lysator

2009. gada septembris

Šo prezentāciju, dokumentāciju un parauga kodus var apskatīt
http://www.lysator.liu.se/~kjell-e/embedded/olimex/

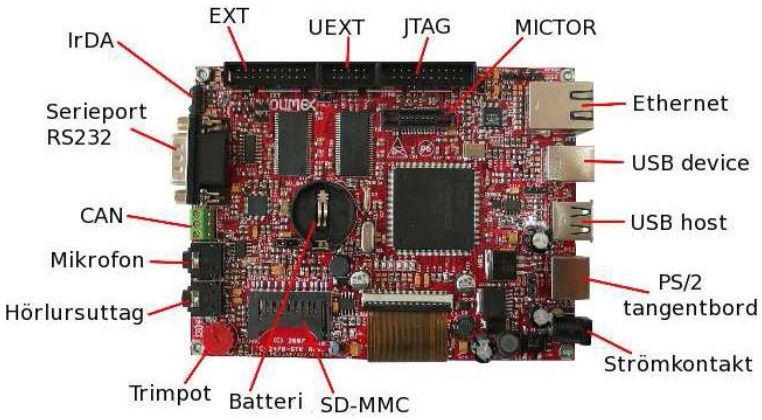
Prezentācija ir pieejama arī
http://www.lysator.liu.se/~kjell-e/tekla/linux/dokument.html

LPC-2478-STK ir izstrādes plate iegultajām sistēmām.
Iegultās sistēmas ir datori, kas iestrādā televizoros, mikroviļņu krāsnīs, automašīnās, digitālos televizoros, mobilos telefonos, plaukstdatoros un citās ierīcēs.
Iegultās sistēmas bieži vien ir ierobežoti resursi, piemēram, salīdzinoši neliels atmiņas apjoms, nav cietā diska, maz zibatmiņas, lēnāks CPU nekā mūsdienu darbstacijām un serveriem.
Iegultās sistēmas var būt pieejami dažāda veida I / O porti un seriālie porti, I2C, I2S, CAN un citi. Iegultās sistēmas var nebūt atmiņas aizsardzības. Vairākās mazās iegultās sistēmās nav operētājsistēmas.



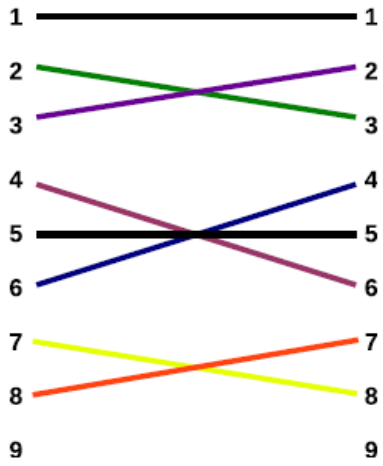
Šīs prezentācijas mērķis iepazīstināt ar šo karti, lai varētu veikt laboratorijas darbus.
Saturs:

Parametri:
ARM7-CPU (ARM7TDMI-S, little endian).
3.5 collu TFT ekrāns ar apgaismojumu un 320x240 izšķirtspēju
MP3 dekodētais
64 MB SDRAM
512 KB
Slots SD/MMC kartei
CAN
JTAG
Ethernet
Seriālie porti
USB vadītājs / USB ierīce
U-boot

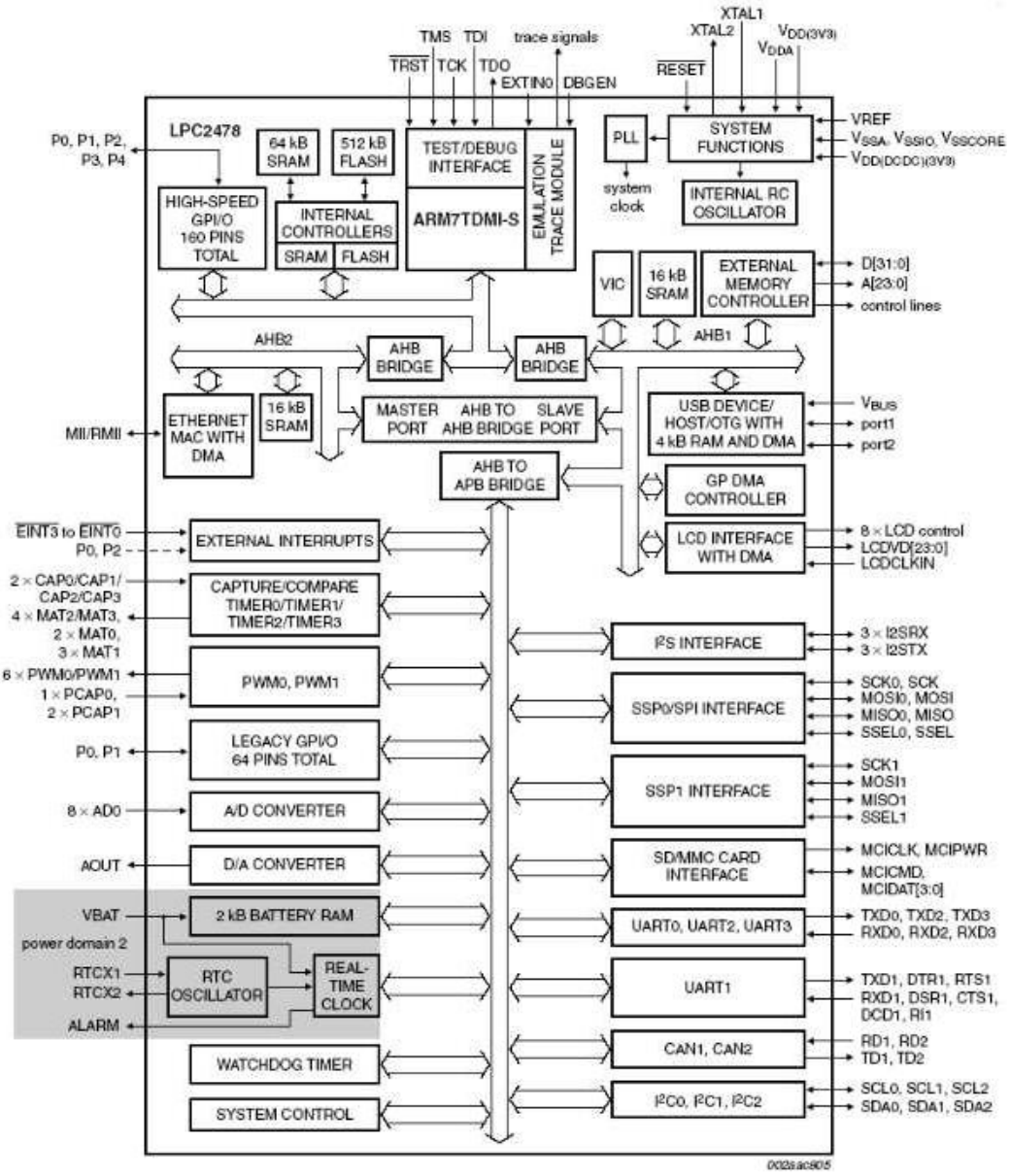


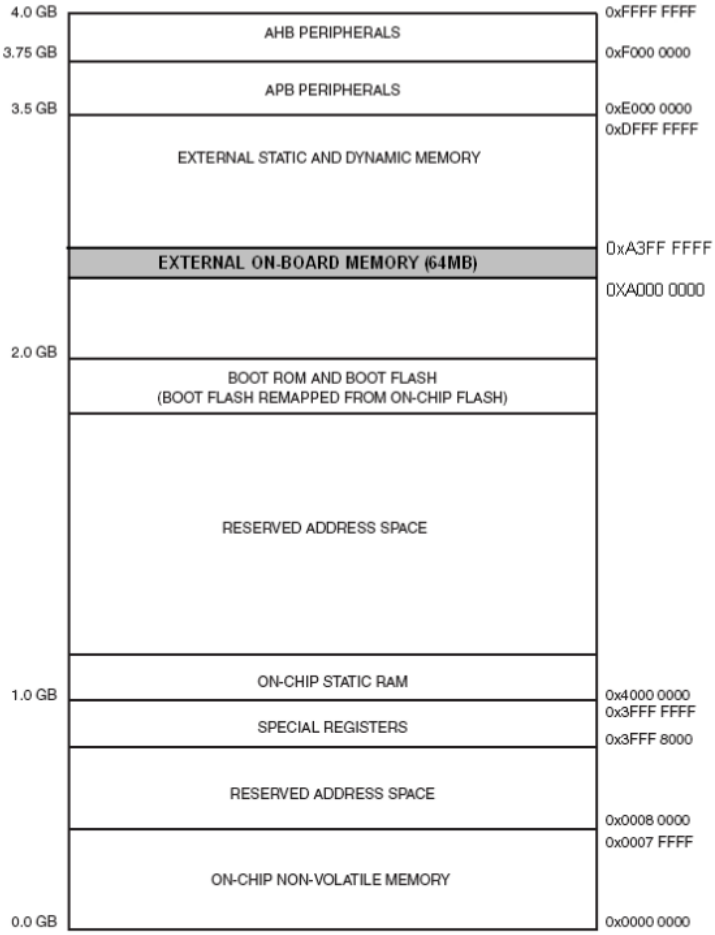
OLIMEX LPC-2478-STK var darbināt vai ar maiņstrāvas spriegumu (6-9 V) vai līdzstrāvas spriegumu (9-12 V).
Savienotāju polaritāte nav svarīgi, ja ir tilta taisngriezis pēc kontakta. Tas tad baro sprieguma stabilizatoru, kas nodrošina 5V un 3.3V.
Klases Ohlsson 32-2314 strāvas adapteris darbojas lieliski.
Seriālais ports ir nokonfigurēts uz 115200 bodiem, 8 datu biti, nav paritātes.
Lai pievienotu seriālo portu pie datora ir nepieciešams null modema kabelis, kur kontakti 2 un 3 sakrustojas.

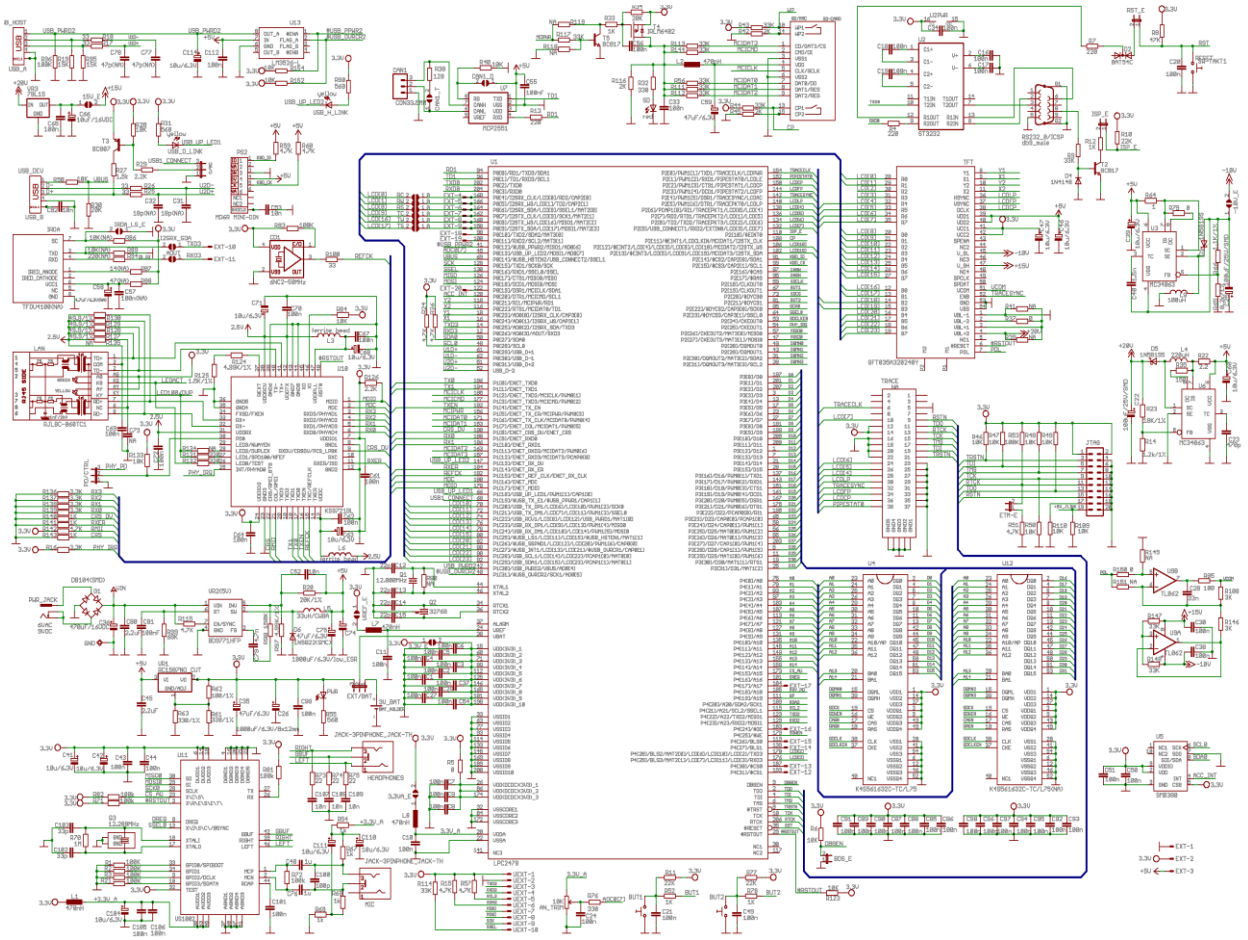
9-pin D-sub sērijas savienotājs izskatās šādi:
1 - Frame ground (N/U)
2 - Rx Dati
3 - Tx Dati
4 - DTR
5 - Signal Ground
6 - DSR
7 - RTS
8 - CTS
9 - N/U



Null modema kabeli (9 kontaktu), ir pieejami labi uzkrāti elektronikas veikalos, piemēram, Klasi Ohlsson. USB arī ir pieejami elektronikas veikalos.




4.0 GB
AHB PERIPHERALS
0xFFFF FFFF
3.75 GB
0xF000 0000
APB PERIPHERALS
3.5 GB
0xE000 0000
0xDFFF FFFF
EXTERNAL STATIC AND DYNAMIC MEMORY
0xA3FF FFFF
EXTERNAL ON-BOARD MEMORY (64MB)
0XA000 0000
2.0 GB
BOOT ROM AND BOOT FLASH
(BOOT FLASH REMAPPED FROM ON-CHIP FLASH)
RESERVED ADDRESS SPACE
ON-CHIP STATIC RAM
1.0 GB
0x4000 0000
0x3FFF FFFF
SPECIAL REGISTERS
0x3FFF 8000
RESERVED ADDRESS SPACE
0x0008 0000
0x0007 FFFF
ON-CHIP NON-VOLATILE MEMORY
0.0 GB
0x0000 0000



Uz plates pēc noklusējuma ir uzstādīts U-boot ielādētājs.
Plates seriālais ports komunicē ar parametriem 115200 bps, 8 datu biti, nav paritātes.
Komunicēt ar plati var ar kermit vai picocom konsoļu palīdzību.
Pieslēgšanas piemērs: picocom -b 115200 /dev/ttyUSB0
Pēc pieslēgšanas portam var pieslēgt izstrādes plati strāvai.
Pirms U-Boot ir iztecējis laiks, lai palaistu ielādētāja konsoli nospiediet jebkuru taustiņu. U-Boot ir nokonfigurēts 3 sekunžu taimauts pēc noklusējuma.
Visi iestatījumi, ieskaitot taimautu, glabājas U-Boot kā mainīgie.

Ierakstiet help, lai izvadītu uz ekrāna sarakstu ar iespējamām komandām.
Ierakstot help „komandas nosaukums", var iegūt vairāk informācijas par konkrēto komandu.
printenv parāda visus mainīgos un to vērtības:.
b o o t a r g s=r o o t=/dev /ram i n i t r d=0xa0800000 ,4000 k c o n s o l e=t tyS0 ,115200N8
bootcmd=run usb bo o t
b o o t d e l a y=3
baudr a t e=115200
t f t p b o o t=t f t p b o o t a0008000 l i n u x . b i n ; t f t p b o o t a1800000 romf s . b i n ; go a0008000
nand boot=nand r e ad 0 xa0008000 0 x0 0 x220000 ; nand r e ad 0 xa0800000 0 x400000 0 x2200
usb bo o t=usb s t a r t ; f a t l o a d usb 0 0 xa0800000 r omf s 5 . img ; f a t l o a d usb 0 0 xa0008000
mmc boot=mmc; f a t l o a d mmc 0 a0008000 l i n u x . b i n ; f a t l o a d mmc 0 a1800000 romf s . b i n ; g
upda t e uboot=t f t p b o o t a1000000 u−boot . b i n ; p r o t e c t o f f 0 2 f f f f ; e r a s e 0 2 f f f f ; cp . b
updat e nand=nand e r a s e ; t f t p b o o t a1000000 l i n u x . b i n ; nand wr i t e a1000000 0 200000;
i p a d d r =192.168.0.158
netmask =255.255.255.0
s e r v e r i p =192.168.0.240
e thaddr =00: de : ad : b0 : 0 5 : 0 3
c o n s o l e=s e r i a l
s t d i n=s e r i a l
s t d o u t=s e r i a l
s t d e r r=s e r i a l

Failus platē var lejupielādēt no USB atmiņas, SD / MMC kartes vai no TFTP servera.
USB un SD / MMC ir šādi:
usb 0 norāda pirmo USB ierīci ar fat failu sistēmu (VFAT).
mmc 0 norāda pirmo MMC ierīci ar fat failu sistēmu (VFAT).
fatls <interface> <dev[:part]> [directory] – izvada failu sarakstu.
fatload <interface> <dev[:part]> <addr> <filename> [bytes] – ielāde failu addr atmiņas adresē.
Piemērs: fatls usb 0
Piemērs: fatload usb 0 0xa0800000 romfs 5.img

Lai varētu piekļūt USB atmiņas kartes saturam, vispirms ir jāieslēdz USB atbalsts.
USB palaiž ar usb start
USB atbalstu var atslēgt ar usb stop.

USB partīcijas var apskatīt ar usb part.
usb help komanda sniedz informāciju par USB izmantošanas iespējam U-boot ietvaros.

tftpboot [loadAddress] [[hostIPaddr:]bootfilename] – ielādēt failu bootfilename atmiņas adresē loadAddress no tftp servera ar ip-adresi hostIPaddr.
Piemērs:
tftpboot 0xa0008000
192.168.0.1:/linux-install/olimex/vmlinux.bin
go addr [arg ...] – palaist programmu no atmiņas adreses addr.
Piemērs:
go 0xa0008000

run var [...] - palaist komandu virkni ar mainīgo var.
Piemērs: run tftp boot
Mainīgais tftp_boot palaiž šādas komandas:
tftpboot 0xa0800000 192.168.0.1:/linux-install/olimex/romfs 5.img; tftpboot 0xa0008000 192.168.0.1:/linux-install/olimex/vmlinux.bin;go a0008000
Darbam var uzskatīt mainīgo par shell skriptu.

Mainīt mainīgā saturu var sekojoši:
setenv mainīgā vērtība
Piemērs:
setenv bootargs ’root=/dev/ram initrd=0xa0800000,4000k console=ttyS0,115200N8’
Mainīgo vērtības saglabā zibatmiņā ar komandu saveenv.
bdinfo parāda vairāk informācijas par karti (print Board Info structure).
ping host sūta paketes uz norādīto ip-adresi. Piemērs: ping 192.168.0.1

Ielādes komandas pēc noklusējuma U-Boot glabājās mainīgajā bootcmd.
Pēc noklusējuma ir nokonfigurēts, ka faili romfs_5.img un vmlinux.bin tiks ielādēti platē no VFAT formāta USB kartes.
romfs 5.img satur romfs ar saknes failu sistēmu, bet vmlinux.bin satur linux kodolu.
Šie divi faili atrodas uz CD diska, kas tiek piedāvāts komplektā ar Olimex plati.
Tajā pašā direktorijā atrodas arī U-boot-bin.hex, kas ir U-Boot attēls.
U-boot tiek ielādēts platē iebūvētajā zibatmiņa.

Lai ielādētu sistēmu no USB, vispirms ir nepieciešams sākt USB atbalstu, ielādēt saknes failu sistēmu, ielādēt kodolu un beidzot palaist kodolu no atmiņas:
usb start
fatload usb 0 0xa0800000 romfs 5.img
fatload usb 0 0xa0008000 vmlinux.bin
go a0008000
Šie iestatījumi ir pieejami kā standarta mainīgais usb_boot. Tāpēc var palaist run usb_boot un bez komandu ievadīšanas ielādēt sistēmu no USB.

Lai ielādētu sistēmu no MMC, vispirms ir nepieciešams sākt MMC atbalstu, ielādēt kodolu, ielādēt saknes failu sistēmu un beidzot palaist kodolu no atmiņas:
mmc
fatload mmc 0 a0008000 linux.bin
fatload mmc 0 a1800000 romfs.bin
go a0008000
Šie iestatījumi ir pieejami kā standarta mainīgais mmc_boot. Tāpēc var palaist run mmc_boot un bez komandu ievadīšanas ielādēt sistēmu no MMC.
Lai lejupielādētu failus no TFTP servera ir nepieciešams strādājošs TFTP serveris ar attiecīgajiem failiem un Ethernet pieslēgums.
Zemāk ir parādīta TFTP servera konfigurācija, kas palaižas no xinetd:
service tftp
{
sockettype = dgram
protocol = udp
wait = yes
user = root
server = /usr/sbin/in.tftpd
serverargs = −s/tftpboot
disable = no
per_source = 11
cps = 100 2
flags = IPv4
}
TFTP servera nokonfigurētais katalogs ir /tftpboot.

Turpmāk sniegtajā piemērā faili serveri ir ievietoti direktorijā /tftpboot/linux-install/olimex/
Servera ip-adrese ir 192.168.0.1.
Izstrādes plate ir mērķis.
Izpildāmās komandas:
tftpboot 0xa0800000 192.168.0.1:/linux-install/olimex/romfs_5.img
tftpboot 0xa0008000 192.168.0.1:/linux-install/olimex/vmlinux.bin
go a0008000

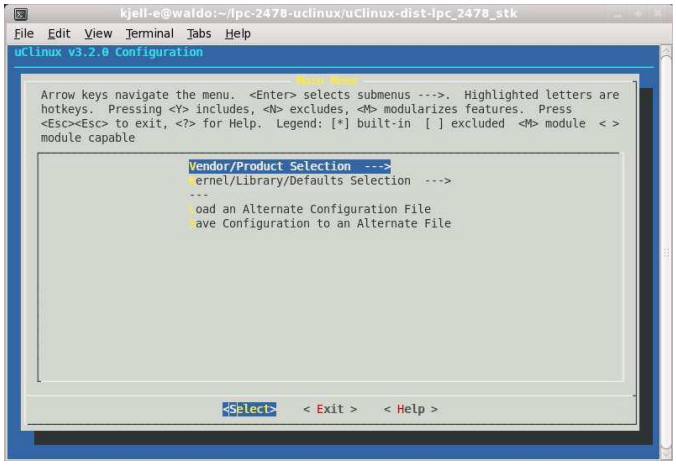
Darbam ar Olimex platēm tiek piedāvāts uClinux.
uClinux tiek izmantots galvenokārt uz sistēmām, kurām trūkst MMU.
MMU nav paredzēts:
Nav virtuālās atmiņas (VM), un nav swap.
Nav atmiņas aizsardzības. Tas nozīmē, ka process var rakstīt citu procesu atmiņā.
Nav tmpfs.
Tā kā sistēma nav MMU, tad pastāv tikai binārā formātā Flat (bFLT) programmās. Visas pārējās binārā formāta programmas funkcionē ar virtuālo mašīnu.
Kompilatoram ir jāatbalsta Flat, kā arī no pozīcijas neatkarīgais kods (PIC) un palaist uz vietas (XIP).
Kaudze tiek piešķirta kompilēšanas laikā un nevar tikt izmainīta izpildes laikā.

Uzdot kaudzes apjomu var ar flthdr.
Piemērs: lpc-2478-uclinux/buildroot/toolchain build arm/elf2flt/flthdr -s 20k program
Kaudze arī darbojas savādāk ucLinux, ja atmiņa tiek piešķirta no globālās atmiņā kopas.

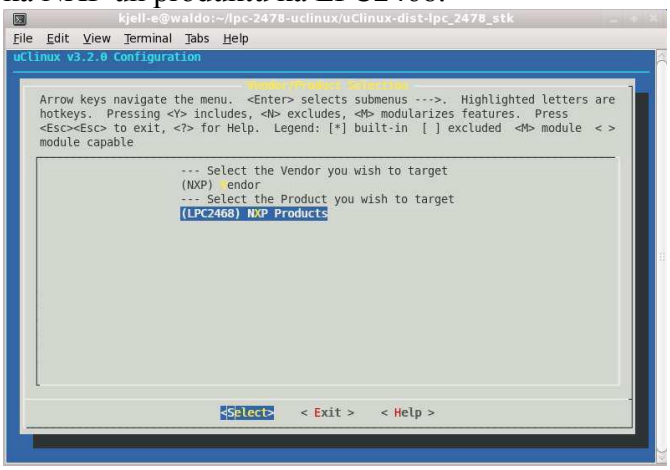
Apraksts par to, kā nokompilēt uClinux (buildroot, kodolu, uc).
Izveidot direktoriju struktūru: mkdir -p ~/lpc-2478-uclinux/snapgear-cross
Ieiet pēdējā direktorijā no struktūras: cd ~/lpc-2478-uclinux/snapgear-cross
tar zxf /mnt/cdrom/Utils/arm-linux-tools-20061213.tar.gz
Uzstādīt PATH, iekļaujot direktoriju ar instrumentiem:
PATH=~/lpc-2478-uclinux/snapgearcross/usr/local/bin:$PATH

Tagad ir pienācis laiks sastādīt uClinux (tostarp avota kodu un kodolu), nokonfigurēt un nokompilēt to.
Sākumā ir jāieiet LPC2478-ucLinux direktorijā. cd ~/lpc-2478-uclinux/
Tad jāatarhivē failu ar pirmkodu
tar zxf /mnt/cdrom/uClinux/uClinux-dist-lpc_2478_stk-20081007.tgz
Tagad ir nepieciešams doties uz leju direktoriju un konfigurēt uClinux
cd uClinux-dist-lpc_2478_stk
make menuconfig
Pārliecinieties, ka uzstādītais ražotājs ir NXP un produkts ir LPC2468.

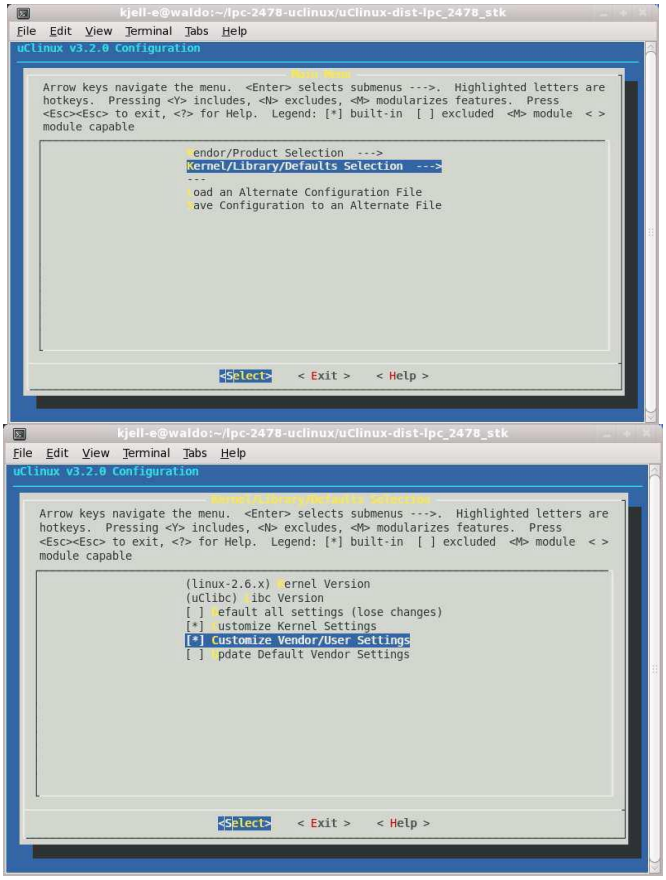
Ieiet ’Vendor/Product Selection’



Nokonfigurēt ražotāju kā NXP un produktu kā LPC2468.



Lai konfigurētu kodolu, Busybox, utt ir jāieiet ’Kernel/Library/Defaults Selection’, tad ’Customize Kernel Settings’, lai nokonfigurēt kodolu, un ’Customize Vendor/User Settings’, lai konfigurēt busybox. Kad menuconfig pabeigts, var konfigurēt kodolu atbilstoši lietojumam.



Tagad ir pienācis laiks nokompilēt uClinux.
Lai kompilētu uClinux un veidotu rootfs nepieciešama programma genromfs.
Kompilē ar komandu make.
Ja kompilācija gāja labi, tad rezultāta jābūt izveidotiem failiem romfs un vmlinux.
Lai veidotu savu kodolu, jums ir to nepieciešams sastādīt ar make uImage.
Ir arī nepieciešams, lai uz datora būtu instalēts genromfs.

Lai varētu ielādēt un uzstādīt saknes failu sistēmu no NFS servera nepieciešams iekļaut papildinātu (patched) kodolu.
Papildinājums atrodas zem ~/lpc-2478-uclinux/uClinux-dist-lpc_2478_stk/linux-2.6.x/
Saglabāt šo failu nfsroot.patch.

```
−−− fs/nfs/oldfile.c 2009−06−24 18:13:46.000000000 +0200
+++ fs/nfs/file.c 2009−06−24 18:13:49.000000000 +0200
@@ −281,9 +281,13 @@
status = nfs_revalidate_mapping(inode,file−>f_mapping);
if (!status) {
+#ifndef CONFIG_MMU
+ status = generic_file_mmap(file,vma) ;
+#else
vma−>vm_ops = &nfs_file_vm_ops ;
vma−>vm_flags |= VM_CAN_NONLINEAR;
file_accessed(file) ;
+#endif
}
return status ;
}
```

Papildināt, izmantojot patch -p0 < nfsroot.patch.

Tad jums ir nepieciešams konfigurēt kodolu ar make menuconfig direktorijā, kur atrodas kodola avots.
'Networking'−>'Networking options' un atļaut TCP/IP atbalstu, 'kernel level autoconfiguration' un atļaut 'DHCP support' un 'BOOTP support'.
'File systems'−>'Network File Systems' un aktivizēt 'NFS file system support', 'Provide NFSv3 client support', 'Root file system on NFS'.
Visbeidzot jāieiet 'Boot options' un jānokonfigurē 'Default kernel command string' uz
root=/dev/nfs console=ttyS0,115200N8 rw nfsroot=192.168.0.1:/export/olimex/root
ip=192.168.0.158:192.168.0.1:192.168.0.1:255.255.255.0:olimex::off
rootpath=/export/olimex/root/ init=/bin/sh

Kompilēt uClinux un uzstādīt jaunu kodolu.
cd ..
make
cp images/vmlinux.bin /tftpboot/linux-install/olimex/vmlinux-nfs.bin
Palaist un ielādēt kodolu:
tftpboot 0xa0008000 192.168.0.1:/linux-install/olimex/vmlinux-nfs.bin
go 0xa0008000
Ja viss strādā, tad var palaist Olimex plati un uzstādīt saknes failu sistēmu no NFS servera 192.168.0.1.
Saknes failu sistēmu var kopēt no direktorija rootfs.
cp –a ~/lpc-2478-uclinux/uClinux-dist-lpc_2478_stk/romfs /export/olimex/root
Zemāk ir screenshots ar NFS saknes failu sistēmas ielādi.
lpc−2478−s t k # t f t p b o o t 0 xa0008000 1 9 2 . 1 6 8 . 0 . 1 : / l i n u x−i n s t a l l / o l ime x / vml inux . b i n
emac : che ck phy − (22 , 1619)
emac : l i n k s t a t u s = 100Mbps , f u l l dupl e x
emac : MAC a d d r e s s = 0 : de : ad : b0 : 5 : 3
TFTP f rom s e r v e r 1 9 2 . 1 6 8 . 0 . 1 ; our IP a d d r e s s i s 1 9 2 . 1 6 8 . 0 . 1 5 8
Fi l ename ' / l i n u x−i n s t a l l / o l ime x / vml inux−n f s . b i n ' .
Load a d d r e s s : 0 xa0008000
Loading : ###
done
Byt e s t r a n s f e r r e d = 2236072 (221 ea8 hex)
lpc−2478−s t k # go 0 xa0008000
S t a r t i n g a p p l i c a t i o n at 0 xA0008000 . . .
´uLinux v e r s i o n 2.6.24.2−uc0 (k j e l l−e@waldo . dyndns . org) (gcc v e r s i o n 3 . 4 . 4) #14 S
CPU: NXP−LPC2468 [2 4 6 8 0 0 0 0] r e v i s i o n 0 (ARMvundef ined /unknown) , c r=a0229ec0
Machine : Ol imex LPC−2478−STK
Warning : bad c o n f i g u r a t i o n page , t r y i n g to cont inue
Bu i l t 1 z o n e l i s t s in Zone orde r , mo b i l i t y g r o u p i n g on . Tot a l page s : 16256
Ke r n e l command l i n e : r o o t=/dev / n f s c o n s o l e=t tyS0 ,115200N8 rw n f s r o o t =1 9 2 . 1 6 8 . 0 . 1
PID hash t a b l e e n t r i e s : 256 (o r d e r : 8 , 1024 b y t e s)
LPC22XX Cl o c k i n g Fin=12000000Hz Fcco=288000000Hz M=11 N=0
Fc c l k=57600000 PCLKSEL=55515555 11555455
Consol e : c o l o u r dummy d e v i c e 80 x30
Dent ry cache hash t a b l e e n t r i e s : 8192 (o r d e r : 3 , 32768 b y t e s)
Inode−cache hash t a b l e e n t r i e s : 4096 (o r d e r : 2 , 16384 b y t e s)

```
Memory : 64MB = 64MB t o t a l
Memory : 62628KB a v a i l a b l e (2000K code , 181K data , 92K i n i t )
Mount−cache hash t a b l e e n t r i e s : 512
ne t name spac e : 64 b y t e s
NET: Re g i s t e r e d p r o t o c o l f ami l y 16
e th0 : Link down .
e th0 : LPC22XX e t h e r n e t at 0 x f f e 0 0 0 0 0 i n t=21 10−Fu l lDu p l e x ( 0 0 : 1 a : f 1 : 0
0 : 0 0 : f 6 )
e th0 : Mi c r e l PHY at 1
block2mtd : v e r s i o n $Re v i s i o n : 1 .3 0 $
TCP c u b i c r e g i s t e r e d
NET: Re g i s t e r e d p r o t o c o l f ami l y 1
RPC: Re g i s t e r e d udp t r a n s p o r t module .
RPC: Re g i s t e r e d t cp t r a n s p o r t module .
e th0 : Link down .
IP−Co nf i g : Complete :
d e v i c e=eth0 , addr =1 9 2 .1 6 8 .0 .1 5 8 , mask=2 5 5 .2 5 5 .2 5 5 .0 , gw=1 9 2 . 1 6 8 . 0 . 1
,
ho s t=ol ime x , domain=, n i s−domain=(none ) ,
b o o t s e r v e r =1 9 2 . 1 6 8 . 0 . 1 , r o o t s e r v e r =1 9 2 . 1 6 8 . 0 . 1 , r o o t p a t h=
Looking up p o r t o f RPC 100003/2 on 1 9 2 . 1 6 8 . 0 . 1
Looking up p o r t o f RPC 100005/1 on 1 9 2 . 1 6 8 . 0 . 1
VFS : Mounted r o o t ( n f s f i l e s y s t em ) .
BusyBox v1 . 0 0 (2008.10.07−02:27+0000) Bu i l t−i n s h e l l (msh )
Ent e r ' h e l p ' f o r a l i s t o f b u i l t−i n commands .
#
#
# l s
b i n dev e t c home l i b mnt proc s b i n tmp u s r v a r
```

Pievienojot jaunu pieteikumu ucLinux nepieciešams, lai izveidotu mapi un pievienot dažas rindiņas dažos failos.
Ja projekts nosaukts foo, tad jāveido šādus failus un direktorijas:
Direktorija ~/lpc-2478-uclinux/uClinux-dist-lpc 2478 stk/user/foo/ atrodas projekta faili.
Izpildīt makefile direktorijā foo.
~/lpc-2478-uclinux/uClinux-dist-lpc_2478_stk/user/Makefile
Neliels palīdzības materiāls ~/lpc-2478-uclinux/uClinux-distlpc_2478_stk/config/Configure.help.
Šis palīdzības teksts ir par ucLinux konfigurāciju.
Vienu vai vairākas līnijas failā ~/lpc-2478-uclinux/uClinuxdist-lpc_2478_stk/config/config.in var izvēlēties konfigurāciju, lai izveidotu uz ucLinux.
Izveidot direktoriju ~/lpc-2478-uclinux/uClinux-dist-lpc 2478 stk/user/foo/ un kopēt projekta failus tur.
Izveidot Makefile projektā direktorijā. Zemāk ir dots piemērs:

```
EXEC = foo
OBJS = foo . o
a l l : $ (EXEC)
$ (EXEC) : $ (OBJS)
$ (CC) $ (LDFLAGS) −o $@ $ (OBJS) $ ( LDLIBS )
romf s :
$ (ROMFSINST) / b i n /$ (EXEC)
c l e a n :
−rm −f $ (EXEC) ∗. e l f ∗. gdb ∗. O
```

Ja projekts sastāv no vairākiem failiem, tad Makefile vajadzētu izskatīties šādi:
EXECS = foo bar
OBJS = foo . o bar . o
a l l : $ (EXECS)
$ (EXECS) : $ (OBJS)
$ (CC) $ (LDFLAGS) −o $@ $@ . o $ (LDLIBS)
romf s :
$ (ROMFSINST) −e CONFIG_USER_FOO_FOO / b i n / foo
$ (ROMFSINST) −e CONFIG_USER_FOO_BAR / b i n / bar
Protams, ka Makefile var būtu daudz sarežģītākas nekā divi iepriekšējie piemēri.

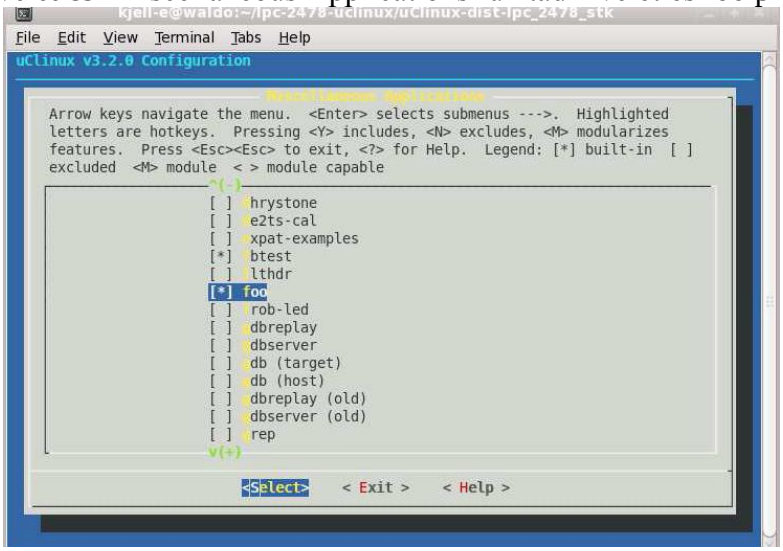
Pievienot līniju dir_$ (CONFIG_USER_FOO_FOO) += foo
˜/lpc-2478-uclinux/uClinux-distlp_2478_stk/user/Makefile
Šī līnija piebilst direktoriju foo to būvēt.
Projekta daļām, kuras tiks būvētas, nevajag būt alfabētiskā secībā, bet tas nodrošina labāku kārtībā.
dir$ (CONFIG_USER_FNORD_HTTPD) += fnord
dir$ (CONFIG_USER_FLASHW_FLASHW) += flashw
dir$ (CONFIG_USER_FLATFSD_FLATFSD) += flatfsd
dir$ (CONFIG_USER_FLTHDR_FLTHDR) += flthdr
dir$ (CONFIG_USER_FOO_FOO) += foo
dir$ (CONFIG_USER_FREESWAN) += freeswan
dir$ (CONFIG_USER_FROB_LED_FROB_LED) += frob−led
dir$ (CONFIG_USER_FROX_FROX) += frox
dir$ (CONFIG_USER_FSWCERT_FSWCERT) += fswcert
dir$ (CONFIG_USER_FTP_FTP_FTP) += ftp

Pievienot palīdzības tekstu failā ˜/lpc-2478-uclinux/uClinuxdist-lpc_2478_stk/ config/ Configure.help
CONFIG_USER_FCONFIG_FCONFIG
A program that lets you manipulate your RedBoot configuration from Linux .
CONFIG_USER_FOO_FOO
This program does fooey things to your bars.
CONFIG_USER_GETTYD_GETTYD
Another getty program.
Approx. binary size: 16k
Ņemiet vērā, ka palīdzības teksta rindām jāsākas ar tieši divām telpām.
Kopumā nedrīkst būt mazāks par 70 rakstzīmēm.
Tukšas rindas nav atļautas.

Visbeidzot, attiecībā uz vienu vai vairākām pozīcijām ~ / lpc-2478-uclinux/uClinux-dist-lpc_2478_stk/config/config.in izvēlēties uClinux konfigurāciju, lai izveidotu projektu, veicot make menuconfig.
Piemērs: bool ’foo’ CONFIG_USER_FOO_FOO

Pievienot šīs rindas atbilstošai izvēlnes iespējai, piemēram, ’Miscellaneous Applications’
mainmenu_option next_comment
comment ’Miscellaneous Applications’
bool ’7za’ CONFIG_USER_P7ZIP_7ZA
bool ’a60’ CONFIG_USER_LANG_A60
if [”$CONFIG_USER_LANG_A60” = ”y”]; then
bool ’examples’ CONFIG_USER_LANG_A60_EGS
bool ’tests’ CONFIG_USER_LANG_A60_TEST
fi
bool ’flthdr’ CONFIG_USER_FLTHDR_FLTHDR
bool ’foo’ CONFIG_USER_FOO_FOO
bool ’frob−led’ CONFIG_USER_FROB_LED_FROB_LED
bool ’gdbreplay’ CONFIG_USER_GDB_GDBREPLAY

FOO un CONFIG_USER_FOO_FOO projekts sastāv no vairākām programmām, kas tiks apkopotas.
Ja projekta foo ir foo bārs programmas, kas var tikt būvētas atsevišķi, tad pievienot atsevišķas rindas konfigurācijai.
bool ’foo’ CONFIG_USER_FOO_FOO
bool ’bar’ CONFIG_USER_FOO_BAR
Ieiet direktorijā ˜/lpc-2478-uclinux/uClinux-dist-lpc_2478_stk/ un palaist make menuconfig.
Izvēlēties ’Kernel/Library/Defaults Selection’ un aktivizēt ’Customize Vendor/User Settings’.
Kad darbs ar menuconfig ir pabeigts, var veikt lietojumprogrammas konfigurācijas.
Jaunajā izvēlnē izvelēties ’Miscellaneous Applications’ un tad izvēlēties foo projektu.



Pēc tam, kad uzstādīšana ir pabeigta, var izpildīt make menuconfig.
Gatavā programma/programmas tagad ir pieejams fails romfs_5.img un direktorija romfs/bin.

Lai varētu kompilēt programmas izstrādes platei, izmantojot uClinux veidotu sistēmu, vajag mainīt savu ceļu uz iekļaut šādu informāciju:
PATH=˜/ lpc−2478−uclinux/uClinux−dist−lpc_2478_stk/tools:\
˜/ lpc−2478−uclinux/snapgear−cross/usr/local/bin/:$PATH

Izveidot Makefile, kur norādīt CC, CXX, CFLAGS,CXXFLAGS, LDFLAGS...

```
CC=u c f r o n t−gcc arm−l i n u x−gcc
CXX=u c f r o n t−g++ arm−l i n u x−g++
CFLAGS=−Os −g −fomi t−f rame−p o i n t e r −p i p e \
−msof t−f l o a t −fno−common −fno−b u i l t i n −Wal l \
−DEMBED −D PIC −f p i c −ms ingl e−pi c−bas e \
−Dl inux −D l i n u x −Dunix −D uCl i n u x
LDFLAGS=−Wl,−−f a t a l−wa rning s −Wl,− e l f 2 f l t \
−msof t−f l o a t −D PIC −f p i c −ms ingl e−pi c−bas e
CXXFLAGS=−Os −g −fomi t−f rame−p o i n t e r −p i p e \
−msof t−f l o a t −fno−common −fno−b u i l t i n −Wal l \
−DEMBED −D PIC −f p i c −ms ingl e−pi c−bas e \
−Dl inux −D l i n u x −Dunix −D uCl i n u x \
−n o s t d i n c++ −fno−e x c e p t i o n s
```

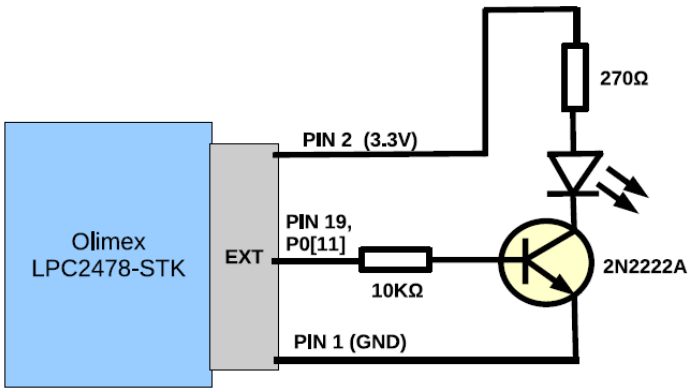
Piemērs Makefile bez G+ +.

```
CC=u c f r o n t−gcc arm−l i n u x−gcc
CFLAGS=−Os −g −fomi t−f rame−p o i n t e r −p i p e \
−msof t−f l o a t −fno−common −fno−b u i l t i n −Wal l \
−DEMBED −D PIC −f p i c −ms ingl e−pi c−bas e \
−Dl inux −D l i n u x −Dunix −D uCl i n u x
LDFLAGS=−Wl,−−f a t a l−wa rning s −Wl,− e l f 2 f l t \
−msof t−f l o a t −D PIC −f p i c −ms ingl e−pi c−bas e
EXEC = foo
OBJS = foo . o
a l l : $ (EXEC)
$ (EXEC) : $ (OBJS)
$ (CC) $ (LDFLAGS) −o $@ $ (OBJS) $ ( LDLIBS )
romf s :
$ (ROMFSINST) / b i n /$ (EXEC)
c l e a n :
−rm −f $ (EXEC) ∗. e l f ∗. gdb ∗. o ∗˜
```

I/O atmiņa OLIMEX LPC2478-STK.
Reģistri kontrolē pin funkciju un konkrētai funkcijai vajadzētu būt PINSEL.
Reģistri kontrolē arī datu virzienu (IODIR,FIODIR).
Citi reģistri ir izmantoti, lai iestatītu izejas uz 0 un 1 (rakstīt datus). (IOSET, IOCLR, FIOSET, FIOCLR).
Lasīšanai izmantojamie datus reģistri (IOPIN, FIOPIN).
Visi reģistri satur 32 bitus.

Reset pin 100 uz LPC2478 (par EXT ligzdu 19 pin) jādara šādi:
Iestatīšana bitu 22 un 23 PA adresi 0xE002C000 (PINSEL0). Tas jādara lai iestatītu tā, lai būtu P0 [11].Pats pin var būt RXD2/SCL2/MAT3 [1].
Nosakot 11 bitu PA Adrese 0xE0028008 (IO0DIR) nokonfigurē P0 [11], ka izeju.
Izeja ir iestatīta uz 1, rakstot 1.-11 bitu pēc adreses 0xE0028004 (IO0SET).
Ieeja ir iestatīta uz 0, rakstot 1.-11 bitu pēc adreses 0xE0028004 ((IO0CLR).
Sīkāku informāciju var atrast dokumentācijā user.manual.lpc24xx.pdf.

Gaismas diožu kontroles piemēri.




```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <inttypes.h>

#define PINSEL0 (*((uint32_t volatile *)0xE002C000))
#define IO0DIR  (*((uint32_t volatile *)0xE0028008))
#define IO0SET  (*((uint32_t volatile *)0xE0028004))
#define IO0CLR  (*((uint32_t volatile *)0xE002800C))

void ledInit(void);
void blink();
void toggle_led(void);

int main(int argc,char **argv)
{
  printf("\n");
  ledInit();
  blink();
  exit(EXIT_SUCCESS);
}

void ledInit(void)
{
  PINSEL0 &= ~((1 << 22) | (1 << 23));
  IO0DIR |= (1 << 11);
}

void blink()
{
  int i;
  for(i=0;i<121;i++)
  {
    toggle_led();
    usleep(100000);
  }
}

void toggle_led(void)
{
```

```
 static int LED_ON = 1;
 if(LED_ON)
 {
  LED_ON = 0;
  IO0CLR |= (1 << 11);
 }
 else
 {
  LED_ON = 1;
  IO0SET |= (1 << 11);
 }
}
```

FIO0-FIO4 kontrolē:
PINSEL piemērs: PINSEL9 - P4.
FIODIR piemērs: FIO4DIR - P4.
FIOSET piemērs l: FIO4SET - P4.
FIOCLR piemērs: FIO4CLR - P4.

Ņemiet vērā, ka user.manual.lpc24xx.pdf ir kļūdas 159. tabulā uz 198 lpp.
FIO0DIR - 0x3FFF C000
FIO1DIR - 0x3FFF C020
FIO2DIR - 0x3FFF C040
FIO2DIR - 0x3FFF C060
FIO2DIR - 0x3FFF C080
Pēdējiem diviem ir jābūt FIO3DIR un FIO4DIR.

Lai mirgotu LED P4 [31], 12 pinu ligzda jānokonfigurē sekojoši:

```
#define PINSEL9 (∗((uint32_t volatile∗)0xE002C024))
#define FIO4DIR (∗((uint32_t volatile∗)0x3FFFC080))
#define FIO4SET (∗((uint32_t volatile∗)0x3FFFC098))
#define FIO4CLR (∗((uint32_t volatile∗)0x3FFFC09C))
```

30. un 31. bitu PINSEL9 nepieciešams iestatīt uz 0:

```
PINSEL9 &= ˜((1<<30)|(1<<31));
```

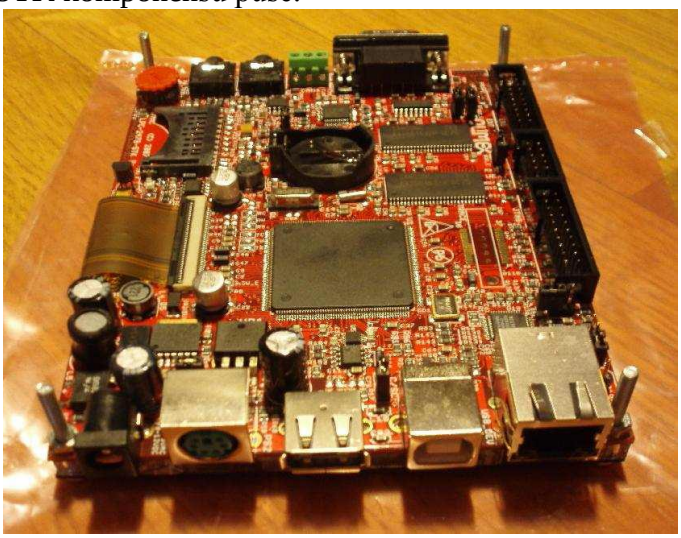
31. bitu FIO4DIR iestatīts uz 1, lai deklarētu to, kā izeju.

```
FIO4DIR |= (1<<31);
```

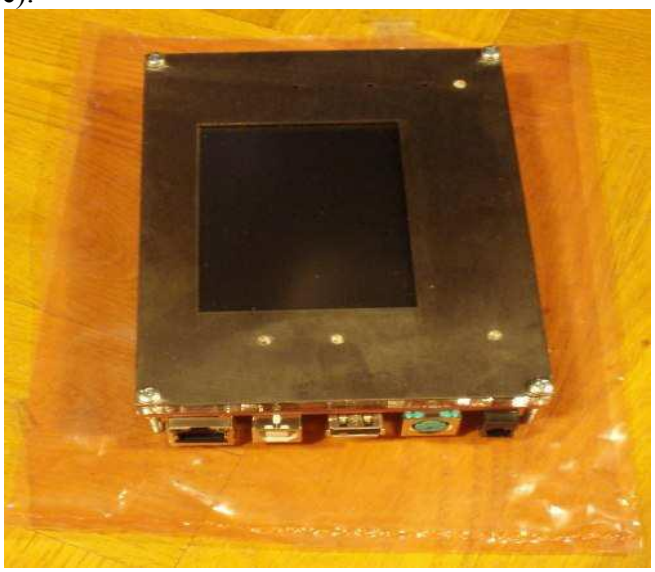
```
#define PINSEL9 (∗((uint32_t volatile∗)0xE002C024))
#define FIO4DIR (∗((uint32_t volatile∗)0x3FFFC080))
#define FIO4SET (∗((uint32_t volatile∗)0x3FFFC098))
#define FIO4CLR (∗((uint32_t volatile∗)0x3FFFC09C))

void ledInit(void)
{
/∗ initialize LED, P4 [ 3 1 ] on Ol imex LPC2478−STK c a rd ∗/
PINSEL9 &= ˜((1<<30)|(1<<31));/∗ Set bit 30:31 to 0 to define pin as GPIO p4[31] ∗/
FIO4DIR |= (1<<31) ; /∗Declare P4.31 as output ∗/
}
```

```
void toggle_led(void)
{
/∗ toggle LED, P4[31] on Olimex LPC2478−STK card ∗/
/∗ Set pin P4[31] to 0 when IO0CLR is 1 ∗/
/∗ Set pin P4[31] to 1 when IO0SET is 1 ∗/
static int LED_ON = 1;
if(LED_ON){
LED_ON = 0 ;
FIO4CLR |= (1<< 31);
}
else{
LED_ON = 1;
FIO4SET |= (1<<31);
}
}
```

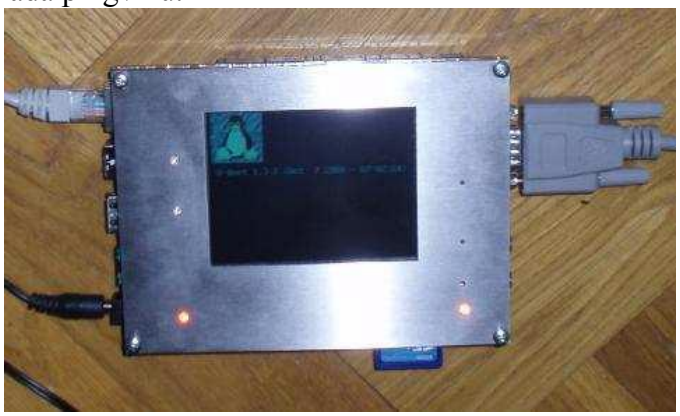
OLIMEX LPC-2478-STK komponenšu pusē.



Augšpusē (displeja pusē).



U-Boot ielādēts un parāda pingvīnu.



Šeit programma fbtest attēlo grafiku, izmantojot ekrāna buferi.