

X Linux Development Guide

Reserved to: FW

Revision History

Authors: G. Filosofi
Revision: 21 Last change: 05/09/2017 16:17:00

<i>revision</i>	<i>Date</i>	<i>Author – Description</i>
0	23/06/2010	GF – First emission of the document
1	13/07/2010	GF – TFTP
2	31/08/2010	GF – Serial Flash Programming Utility
4	08/09/2010	GF – NFS, Qt
5	16/11/2010	GF – Installazione Ubuntu
6	12/05/2011	GF – SDK General Updating
7	06/06/2011	GF – DBus, QtEmbedded-4.7.3
9	07/12/2011	GF – SDK General Updating
10	13/01/2012	GF – CCSv5 General Updating
11	7/02/2012	GF – MTD-Utilities, JFFS2, UBIFS
12	23/02/2012	GF – RAM disk
13	13/07/2012	GF – driver MAG3110
14	09/08/2012	GF – driver MMA955x
15	25/06/2013	GF – CVS, SVN, EGit, SD card
16	30/10/2013	GF – GIT, KEYs & LEDs support
17	17/02/2014	GF – MIC3291 support, System calls
18	15/04/2014	GF – Creare una SD Card di aggiornamento FW
19	28/04/2014	GF – nand dump and other commands
20	11/11/2015	GF – upgrade to Ubuntu 14.04 32-bit
21	16/03/2016	GF – upgrade to Ubuntu 14.04 64-bit

Foreword

Questo documento non è un trattato su Linux, ma una guida operativa allo sviluppo in ambiente Linux Embedded, ad uso interno della X, divisione FW.

Vi sono raccolte le indicazioni pratiche per il setup dell'host e del target, l'installazione dei tools di cross compilazione e di debug, la preparazione del bootloader, del kernel e del file system, le utilities di programmazione della flash, i comandi più usati del bash, ecc.

L'obiettivo è quello di avere a disposizione un prontuario per la risoluzione pratica di problemi già incontrati.

Summary

1 Sommario

X Linux Development Guide	1
Revision History	1
Foreword.....	1
Summary.....	2
Termini e Abbreviazioni.....	12
Utilizzo dei fonts	13
Avvertenze	14
1 Versioni correnti dei principali SW.....	14
1 Definizione degli ambienti di sviluppo	14
1.1 Configurazione Server	15
1.2 Configurazione PC	15
1.3 Configurazione host	15
1.4 Configurazione target	16
2 Ubuntu su host	16
2.1 Releases page	16
2.2 Download page	16
2.3 Installazione Ubuntu su partizione HD.....	17
2.4 Installazione Ubuntu su VM	17
2.5 Upgrade a Ubuntu 14.04.....	17
2.5.1 VMware.....	18
2.5.2 VirtualBox	18
2.6 Configurazione Firefox.....	18
2.7 Cartelle principali	18
2.8 Configurazione Software Sources.....	19
2.9 Connessioni di rete	20
2.10 Configurazione VNC.....	21
2.11 Configurazione Client di Posta Elettronica	21
3 TI SDK e PSP su host	23
3.1 Installazione e configurazione SDK.....	23
3.2 Ridefinire le cartelle dei sorgenti per board kx.....	25
3.3 Installazione PSP completo	26
3.4 Repository e nome delle immagini u-boot, kernel e rootfs	26
4 Yocto	27
5 ARM Toolchain.....	28
5.1 CodeSourcery	28
5.2 Arago ARM Toolchain gcc4.5.3	29

Engineering Specification - confidential

6	Eclipse su host.....	29
6.1	Installazione JRE	30
6.2	Eclipse download page	31
6.3	Installazione Eclipse.....	31
6.4	Installazione Eclipse-CDT Galileo da Synaptic Manager	32
6.5	Installazione GNU ARM Eclipse Plug-In	32
6.6	Configurazione Eclipse e creazione progetto.....	34
7	Tools di utilità generale	39
7.1	Installazione minicom	39
7.2	Installazione gnome-commander.....	39
7.3	Installazione tmux	39
7.4	Installazione vim	40
8	TI CCSv5	40
8.1	Installazione e configurazione CCS v5.1 su host	40
8.2	Creare un nuovo progetto	44
8.3	Importare e compilare progetto U-Boot	44
8.4	Importare e compilare progetto Linux	48
8.5	Importare e compilare progetto Qt matrix_gui (only SDK 5.02)	52
8.6	Tips	54
8.7	Configurare e utilizzare RSE	54
8.8	Configurare e utilizzare GDB server.....	57
8.9	Collegamento al Target	59
9	Team Foundation Server (TFS).....	61
10	TFTP.....	61
10.1	Installazione e configurazione.....	61
11	NFS.....	62
11.1	Installazione e configurazione.....	62
12	Gestire le patches	63
12.1	Metodo classico.....	63
12.2	quilt.....	64
13	UBL.....	65
14	TI DVSDK su host	65
14.1	Installazione e configurazione DVSDK 4.02	65
15	Kernel	67
15.1	Re-build del kernel.....	68
15.2	Configurare il kernel	68
15.3	Compilazione e installazione dei moduli.....	69
15.4	Registrazione del numero di macchina board kx	69

Engineering Specification - confidential

15.5	Creazione supporto kernel per board kx	70
15.6	Patch 0009-kx-kernel-basic	71
15.7	Patch 0010-kx-kernel-config	72
15.8	Patch 0011-kx-add-dpot-lcd-support	73
15.9	Patch 0012-kx-kernel-add-mag3110-support	75
15.10	Patch 0013-kx-kernel-add-mma955x-support.....	77
15.11	Patch 0014-kx-kernel-add-bootup-logo.....	79
15.12	Patch 0015-kx-kernel-add-ad7689-support	80
15.13	Patch 0016-kx-kernel-add-musb-peripheral-support.....	81
15.14	Creazione patch 0017-kx-kernel-port2kxdb.....	83
15.15	Patch 0018-kx-kernel-add-mic3291-support	84
15.15.1	Display Backlight per KX	84
15.15.2	Display Backlight per Spartacus	85
15.16	Patch 0019-kx-kernel-add-mlp3115a2-support.....	85
15.17	Patch 0019-kx-kernel-add-kxpowersupport.....	86
15.18	Patch 0020-kx-kernel-add-fiq-support.....	88
15.19	Patch 0019-kx-kernel-add-ant-support.....	89
15.20	Patch 0019-kx-kernel-add-hrecg-support.....	89
16	Bootloader	90
16.1	Download page	90
16.2	Riferimenti.....	90
16.3	Re-build u-boot.....	90
16.3.1	KX	Errore. Il segnalibro non è definito.
16.3.2	XNRG	Errore. Il segnalibro non è definito.
16.4	Creazione supporto u-boot per board KX.....	91
16.5	Creazione patch 0001-kx-uboot-basic	91
16.6	Creazione patch 0002-kx-uboot-cmd-cfg	95
16.7	Creazione patch 0003-kx-uboot-port2kxdb	95
17	File System	95
17.1	Download page	96
17.2	Montare il root FS da NFS	96
17.3	Montare il root FS da SD card.....	96
17.4	Creare uno script per u-boot.....	98
17.5	Montare il root FS da dispositivo USB Storage.....	99
17.6	ramdisk	99
17.7	Aggiornare il contenuto di una immagine ramdisk preesistente	99
17.8	Creare un nuovo ramdisk appoggiandosi a NFS	99
17.9	Copiare nella flash SPI l'immagine ramdisk da u-boot.....	100
17.10	Boot da flash SPI con RAM Disk.....	100
17.11	Compilare e installare le utility MTD	101

Engineering Specification - confidential

Qui sopra, nel caso cross-compilazione, ovviamente potremmo anche usare <rootfs> invece che <rootpath>	102
17.12 Preparare la directory del rootfs	103
17.13 JFFS2	104
17.14 Creare l'immagine JFFS2	104
17.15 Copiare nella flash l'immagine JFFS2 appoggiandosi a NFS	104
17.16 Copiare nella flash l'immagine JFFS2 da u-boot.....	105
17.17 Boot da NAND con JFFS2.....	106
17.18 UBIFS/UBI.....	106
17.19 Creare un volume UBI da NFS.....	106
17.20 Creare l'immagine UBIFS sul host.....	108
17.21 Copiare nella flash l'immagine UBI da NFS	109
17.22 Copiare nella flash l'immagine UBI da u-boot.....	110
17.23 Copiare nella flash l'immagine UBI da SD card	110
17.24 Boot da NAND con UBIFS	111
17.25 Creare un volume UBI per archivio test/pazienti	111
17.26 Creare SD card per aggiornamento FW.....	113
17.27 Patch rootfs-kx-0001-add-kmain	115
17.28 Patch rootfs-kx-0002-add-kboot	116
17.29 Patch rootfs-kx-0003-add-dropbear	116
17.30 Patch rootfs-kx-0004-add-antdriver (Provvisorio)	117
17.31 Patch rootfs-kx-0004-add-antdriver (Provvisorio)	118
17.32 Patch rootfs-kx-000x-add-iwrapdriver (Provvisorio)	118
17.33 Patch SpO2	122
17.34 Patch rootfs-kx-000x-add-poweroff-at-halt (Provvisorio).....	123
17.35 Patch rootfs-kx-000x-add-reset-at-reboot (Provvisorio)	123
17.36 Patch rootfs-kx-000x-add-network-cfg (Provvisorio)	124
17.37 Patch rootfs-kx-000x-add-kxsensor-driver (Provvisorio).....	124
17.37.1 kxsensors per KX.....	126
17.37.2 ksampler per Spartacus	126
17.38 Patch rootfs-kx-000x-add-psplash (Provvisorio).....	127
17.39 Patch rootfs-kx-000x-add-console-logo (Provvisorio).....	128
17.40 Patch rootfs-kx-000x-add-mpl3115a2	128
18 Fundamentals on flash devices	128
18.1 NAND	128
19 Flash programming utilities	129
19.1 Programmazione via seriale	129
19.1.1 Programmazione via u-boot.....	131
19.2 Programmazione via CCS	132

Engineering Specification - confidential

20	<i>Init e inittab</i>	134
21	<i>Qt Framework</i>	134
21.1	DBus	134
21.2	Tslib	135
21.2.1	tslib per KX	136
21.2.2	tslib per Spartacus	136
21.3	Qt Libraries for Embedded Linux	136
21.4	Ricompilare demo matrix_gui	140
21.5	Set the toolchain paths	140
21.6	Ricompilare XNRG qt-helloX	140
21.7	Qt Creator (standalone)	140
21.8	Qt SDK	141
21.9	Qt Host Libaries	143
21.10	Configurare versioni in Qt Creator	143
21.11	Progetto Qt MIDAS demo	148
21.12	Progetto Qt Arago demo	148
21.13	Uso di Qt con VNC	148
21.14	Uso di qmake da shell	148
21.15	Qt - Tips	149
22	<i>GDB</i>	149
23	<i>SSH</i>	150
23.1	Principi della comunicazione sicura su una rete pubblica	150
23.2	SSH login su Server	150
23.3	SSH login su Target	151
23.4	Usare Teraterm SSH da Windows	151
24	<i>SCP</i>	152
25	<i>http server</i>	152
26	<i>CVS</i>	152
26.1	Creare una nuova repository	153
26.2	CVS in Eclipse	153
27	<i>SVN</i>	156
27.1	SVN in Eclipse	157
28	<i>Prontuario dei comandi vim</i>	162
28.1	Creare e lavorare con i tags in vi	164
28.2	Editare files binari su Host con xxd, dd e cat	164
29	<i>GIT</i>	165
29.1	Configurare GIT	165
29.2	Creare repo remoto (su server)	165
29.3	Creare repo locale	166

Engineering Specification - confidential

29.4	Creare un branch	166
29.5	Utilità e comandi vari	166
29.6	Utilizzare repo locale kx-uboot	170
29.7	Creare git tags	170
30	KX How To	171
30.1	Firmware versioning.....	171
30.2	KX Firmware Update	172
30.3	Settings	173
30.4	QML	173
30.5	KX – How to get a screenshot	173
30.6	How to get access to linux shell with password	174
30.7	How to get event logs	174
30.8	KX – How to activate hw modules via SD card.....	175
30.9	Bitbucket.....	175
30.10	Egit	175
31	Librerie esterne	178
31.1	qwt	179
31.1.1	Qwt per KX	179
31.1.2	Spartacus	180
31.2	libusb	181
31.3	QExtSerialPort	181
31.4	Dropbear	182
31.5	Util-linux	183
31.5.1	Util-linux per KX.....	183
31.5.2	Util-linux per Spartacus	183
31.6	fbset	183
31.6.1	fbset per Spartacus	183
32	Database	184
32.1	SQL Lite.....	184
32.1.1	Sqlite per KX.....	184
32.1.2	Sqlite per Spartacus	184
32.2	KX database.....	184
33	Arago matrix GUI	185
33.1	Aggiungere il submenu kx a matrix_gui	185
34	Licenze.....	185
34.1	GPL	186
34.2	LGPL.....	186
34.3	Tivoization	186
35	Device Drivers	186
35.1	Creare e caricare un driver module	186
35.2	Integrare un driver module nel kernel tree	189

Engineering Specification - confidential

35.3	Programming tecniques - mutex.....	190
35.4	Programming tecniques - spinlock	190
35.5	Programming tecniques - tasklet	191
35.6	Programming tecniques - atomic operations	191
35.7	Programming tecniques - lists.....	191
35.8	Programming tecniques - delay	191
35.9	Programming tecniques - interrupt.....	191
36	<i>Creare una distribuzione from scratch</i>	192
36.1	Buildroot	192
36.2	Yocto.....	195
37	<i>KX remote debug.....</i>	196
37.1	Dynamic DNS	196
37.2	SSH tunneling.....	196
37.3	UMTS	197
38	<i>Creare ed eseguire un'applicazione</i>	197
38.1	Accedere da userspace a un char device driver.....	197
38.2	Accedere da userspace al MAG3110 device driver (eCompass)	199
38.3	Accedere da userspace al MMA9553 device driver (Pedometer)	200
39	<i>Creazione ed esecuzione di scripts.....</i>	201
39.1	Creazione ed esecuzione script da Init runlevel	201
39.2	Creazione ed esecuzione di script da udev	201
39.3	Creazione ed esecuzione di script makefile su host.....	202
40	<i>Debuggare in kernel space</i>	204
40.1	Dindbg.....	204
41	<i>Debuggare un'applicazione</i>	204
42	<i>Modificare la mappa di memoria del target.....</i>	204
43	<i>Aspetti da affrontare</i>	204
43.1	Porting al Kernel 2.6.37-rc6	204
43.2	zlib	205
43.3	Developing with Graphics	205
43.4	How to Connect to an EVM via Telnet	205
43.5	How to Setup a Samba Server.....	205
43.6	Understanding the Boot Sequence.....	205
43.7	How to install a debian package	205
43.8	Pin Setup Utility	206
44	<i>PRU.....</i>	206
45	<i>KX application.....</i>	207
46	<i>Comandi bash e shell</i>	210

Engineering Specification - confidential

47 System Calls.....	220
47.1 open	220
47.2 read.....	220
47.3 write.....	220
47.4 close	220
47.5 wait	220
47.6 fork	220
47.7 exit	220
47.8 brk	221
47.9 kill.....	221
47.10 mmap.....	221
47.11 stat	221
47.12 select	222
47.13 pipe	222
48 Library Calls (API)	222
48.1 malloc e free.....	222
48.2 htop.....	222
48.3 inotify	223
49 Comandi di System administration.....	224
49.1 vmstat	224
50 Lavorare con Taskjuggler.....	224
50.1 Creazione ed esecuzione script da Init runlevel	225
51 Kernel RT Preemp.....	225
52 Opzioni di compilazione gcc	226
53 Other.....	226
53.1 make clean, mrproper, distclean.....	226
53.2 Redirezione	226
53.3 Link simbolici e hard	227
53.1 Integrare un RTC.....	227
53.2 Thread e processi	228
53.3 Garbage Collection e Reference Counting.....	229
53.4 Sicurezza informatica (Cybersecurity)	230
53.5 Memory Swap.....	230
53.6 Isolamento tra HW e SO	230
53.7 Prontuario dei comandi di U-Boot (KX).....	230
53.8 Prontuario dei comandi di U-Boot (BBB).....	231
53.9 Opzioni di Boot.....	232
<i>Nota: la rete viene disabilitata. Sarà il kernel a configurarla.....</i>	232

Engineering Specification - confidential

53.10	Ripristinare le variabili di ambiente di default in u-boot.....	234
53.11	Leggere e scrivere la Flash SPI da u-boot.....	234
53.12	Leggere e scrivere una periferica su I2C da u-boot	234
53.13	Leggere e scrivere la Flash SPI da kernel.....	234
53.14	Leggere e scrivere una periferica su I2C da kernel	235
53.15	Inviare comandi iWRAP al modulo BT da kernel.....	236
53.16	Usare iWRAP con i programmi di test	237
53.17	Ripristinare il MAC Address nella Flash SPI da u-boot.....	237
53.18	Ripristinare il settore kx config. nella Flash SPI da u-boot	237
53.19	Inizializzare o modificare il serial number da u-boot.....	238
53.20	Leggere e scrivere la Flash NAND da u-boot	238
53.21	Leggere e scrivere la Flash NAND da linux.....	238
53.22	Debuggare comunicazione cuml con OMNIA	239
53.23	Comunicazione cuml tramite SSH.....	240
53.24	Testare USB Host interface da u-boot (XNRG)	240
53.25	Testare I2C da u-boot (XNRG).....	240
53.26	Debuggare comunicazione cuml con host bluetooth.....	240
53.27	Inviare o ricevere file al/dal KX con zmodem	242
53.28	Debuggare GPS	243
53.29	Definire un nuovo comando in u-boot.....	243
54	<i>Other</i>	244
54.1	Tecniche di ottimizzazione (todo)	244
54.2	Tecniche di programmazione parallela.....	244
54.3	Codyng Style.....	245
54.4	Ridirigere la shell linux su PC tramite /dev/ttyGS0.....	247
54.5	Cambiare il MAC address e IP address da shell linux	248
54.6	Utilizzare il supporto Frequency Scaling nel kernel.....	248
54.7	Utilizzare il supporto GPIO nel kernel	248
54.8	Utilizzare il supporto LEDS nel kernel	250
54.8.1	LEDS per KX.....	250
54.8.2	LEDS per Spartacus.....	250
54.9	Utilizzare il supporto KEYS nel kernel.....	251
54.10	Utilizzare il supporto ECAP-PWM davinci nel kernel.....	252
54.10.1	eCap per KX	253
54.10.2	eCap per Spartacus	253
54.11	Utilizzare il supporto ECAP-PWM Turbina nel kernel	253
54.11.1	PWM per KX	254
54.11.2	PWM per Spartacus	254
54.12	Utilizzare la connessione USB verso il PC	254
54.13	Patch 0016-kx-kernel-add-usb-support	257
54.14	Utilizzare l'utilità dtest sul target.....	258

Engineering Specification - confidential

54.15	Utilizzare l'utilità <i>dcals</i> sul target.....	259
54.16	Utilizzare l'utilità <i>dmap</i> sul target	260
54.17	Aggiungere un parametro nello step	261
54.18	Generare facilmente una password sicura su host.....	261
54.19	Installazione driver Intel su host	261
54.20	Scrivere un log message da shell o da applicazione.....	261
54.21	Utilizzare la shared memory	262
54.22	Compilazione host distribuita	263
54.23	Utilizzare gnuplot.....	263
54.24	Aggiungere supporto runtime a 32-bit su host Ubuntu a 64-bit.....	264
54.25	Compilazione CPP di una libreria C	265
54.26	Attivare iAP protocol su Bluetooth	265
55	<i>XNRG</i>	<i>Errore. Il segnalibro non è definito.</i>
55.1	Creare git repo locale e remota Spartacus.....	266
55.2	DTS.....	267
55.3	eMMC (BBB).....	267
55.3.1	Formato ext4 del rootf.....	269
55.4	Inizializzazione eMMC di una BBB nuova connessa in rete	269
55.5	Aggiornamento FW da rete.....	270
55.6	Aggiornamento FW da USB.....	270
55.6.1	boot_mmc	282
55.7	Update MLO.....	282
55.8	Update U-Boot.....	282
55.8.1	boot_net	284
55.9	Micro SD recovery (BBB).....	285
55.10	Install Qt Creator (standalone).....	285
55.11	Compile qmlrun for host using QtCreator	287
56	<i>Applicazioni Desktop Varie</i>	289
56.1	Utilizzare GIMP	289
56.2	Utilizzare XFIG	291
56.3	Utilizzare Dropbox	291
56.4	Utilizzare Python3.....	292
56.5	Utilizzare Tensor Flow.....	292
56.6	Utilizzare Ubertooth	292
56.6.1	Installare wireshark	292
56.6.2	Flashare il firmware in Ubertooth One	292
56.6.3	Ricompilare libbrbb	293
57	<i>Debugging tips</i>	293
58	<i>Links utili</i>	298

Termini e Abbreviazioni

host la workstation dove gira la distribuzione GNU/Linux
PC la workstation dove gira Windows
Server la workstation dove gira la distribuzione GNU/Linux (condiviso in rete da tutti gli sviluppatori)
VM Virtual Machine
EVM la scheda di valutazione per processore AM1808 (ZOOM Evaluation Module)
Target EVM o HW proprietario basato su AM1808/OMAPL138/DA850
SDK Software Development Kit
DVSDK Digital Video SDK
PSP Linux Platform Support Package
CCS Code Composer Studio
IDE Integrated Developement Environment
UBL User (or Secondary) Bootloader
NFS Network File System
SFH Serial Flasher Host
VNC Virtual Network Computinf
DVCS Distributed Version Control System
SVN Subversion
CVS Concurrent Versions System
GIT ?
TFS Team Foundation Server
FHS File-System Hierarchy Standard
DTS Device Tree Specification
BBB BeagleBone Black Rev.C
<VID> vendor ID (0x1211)
<PID> product ID
 Per KX: (0x15)
 Per XNRG: (0x17)
dirname Nome generico di una directory (comprensivo della path)
filename Nome generico di un file (comprensivo della path)
<hostname> fub
<fwserverip> Indirizzo IP del Serverfirmware (p.es. 192.168.0.232) OBSOLETO !!
<fwserverip> Indirizzo IP del serverfv (p.es. 192.168.0.203)
<fwrepo> path del SW repository (p.es. /home/X/common/kx)
<svnrepo> path della repository SVN su Serverfirmware (p.es. /srv/svn/ o /home/<user>/svnrepo)
<cvsrepo> path della repository CVS su Serverfirmware (p.es. /srv/cvs/ o /home/<user>/cvsrepo)
<user> Nome dell'utente sull'host (p.es. gabriele, gianni,...)
<ipaddr> Indirizzo IP del target (p.es. 192.168.0.149, default for KX is 192.168.0.139)
<serverip> Indirizzo IP del server tftp (p.es. 192.168.0.142)
<nfsserverip> Indirizzo IP del server nfs (p.es. 192.168.0.232)
<ethaddr> Indirizzo Ethernet del target (00:08:ee:04:34:d2)
<netmask> Maskera IP del target (p.es. 255.255.255.0)
<gatewayip> Indirizzo IP del Gateway (p.es. 192.168.0.196)
<proxyip> Indirizzo IP del server proxy (p.es. 192.168.0.221:8080)
<tftpbootdir> path della directory tftp (default: /var/lib/tftpboot)
<target> nome del target hardware (p.es. kx, kxevb, evm)
<tools> path della directory per i vari programmi, tools, utilities (p.es. /home/<user>/programs)
<sdk> path del SDK (default: /home/<user>/sdk_x_xx_xx_xx)
<sdkpsp> path del PSP "ridotto" (tipicamente una sottocartella di <sdk>, board-support)
<psp> path del PSP integrale (default: /home/<user>/DaVinci-PSP-SDK-xx.xx.xx.xx)
<linux> path dei sorgenti uboot (<sdk>/board-support/src/kx/kernel)
<ext-modules> path dei sorgenti moduli kernel esterni (<sdk>/board-support/src/kx/kernel/kx-linux-kernel/modules)

Engineering Specification - confidential

<user-modules> path dei sorgenti drivers userspace (<sdk>/board-support/src/kx/kernel/kx-linux-kernel/usermode-drivers)
<mtd> path dei sorgenti per il build delle mtd utilities (<tools>/mtd oppure <tools>/mtd_arm)
<dev> path della repo git di progetto (GITROOT)
 Per KX: /home/<user>/kx-dev
 Per XNRG: /home/<user>/xnrg-dev
<apps> path della repo git di generazione KX application (def: < dev>/apps)
<usermodedrv> path della repo git di generazione KX application (def: <dev>/usermode-drivers)
<fs> path della repo git di generazione filesystem e ubi images (def: <dev>/fs)
<uboot> path della repo git di generazione u-boot
 Per KX: <dev>/uboot/src/u-boot
 Per XNRG: <dev>/u-boot/src/u-boot
<rootfs> path della directory del target root file system sull'host (default: <dev>/fs/rootfs/fs)
<rootpath> path del NFS-exported target root file system sull'host
 For KX: /var/lib/nfs-kx-rootfs -> <dev>/fs/rootfs/fs
 For XNRG: /var/lib/nfs-Rootfs -> <dev>/fs/rootfs/fs
<archive> path della directory dell'archivio test/pazienti (default sull'host:
/home/<user>/workdir/kx/archive; default sul target; /mnt/archive)
<qtsdk> path della directory di installazione Qt SDK (default: /opt/qtsdk-2010.05)
<qtdirhost> path librerie Qt per host (default: /opt/qtsdk-2010.05/qt)
<qtdir> path librerie Qt per ARM (default: /usr/local/Trolltech/Qt-4.7.x)
<qlibarm> path librerie Qt Embedded ARM (default: /usr/local/Trolltech/QtEmbedded-4.7.x-arm)
<password> password utente <user> su host (p.es. gbr.rt4)
<armtoolch> nome della Toolchain
 Per KX: arm-none-linux-gnueabi, arm-arago-linux-gnueabi
 Per XNRG: arm-epy-linux-gnueabihf
<CCS_INSTALL_DIR> directory di installazione CCS (default: /home/<user>/ti/ccsv5)

Utilizzo dei fonts

Con il font

ippo

denotiamo l'IO da una shell

Prompt	Shell	Esempio
>	MX-DOS	> dvflasher.exe -p "COM3" -enor
host \$	Linux host	host \$ mkdir -p rootfs
target #	U-Boot	target # setenv rootpath /home/gabriele/workdir/kx/filesys
target \$	Linux target	target \$ cat /proc/mtd

Con lo stesso font denotiamo le path e i nomi dei files.

Con lo stesso font in corsivo riportiamo codice sorgente.

Esempio:

```
void oss_close(int fd)
{
    close(fd);
    printf("close sucessfully/n");
}
```

Con lo stesso font in grassetto riportiamo la sequenza di esplorazione menu grafici

Esempio:

Application->Internet->Remote Desktop Viever

Con il font **colorato in celeste** indichiamo parti obsolete

Con il font **colorato in rosso** indichiamo parti non ancora definitive

Avvertenze

- Se non diversamente specificato, i comandi descritti in questo documento devono essere dati essendo registrati come <user>, non come root; devono essere dati in /home/<user>; vengono eseguiti sull'host, non sul Server
- Sul Server anche andranno applicati, ma tenendo presente le problematiche relative alla gestione multiutente

1 Versioni correnti dei principali SW

Di seguito è indicata la versione corrente stabile dei principali pacchetti SW

KX SW Package	Version	Download Page	Date
SDK_AM18x	5.07.00	http://downloads.ti.com/sitara_linux/esd/AM180xSDK/latest/index_FDS.html	5 APR 2013
SDK_AM18x	6.00.00	http://downloads.ti.com/sitara_linux/esd/AM180xSDK/latest/index_FDS.html	4 LUG 2013
PSP	03.21.00.04	http://software-dl.ti.com/dsps/dsps_public_sw/psp/LinuxPSP/ Kernel 2.6.37 U-Boot v2010.12	4 MAY 2011
PSP	03.22.00.06	http://software-dl.ti.com/dsps/dsps_public_sw/psp/LinuxPSP/ Kernel 3.3.0 U-Boot v2012.04.01	? 2013
Kernel	3.3.0		
U-Boot	2012.01.01		
Cross Tool		arm-linux-gnueabi-gcc	
GCC	5.3		11 MAR 2011
Eclipse CDT	1.5.2	Juno	JUNE 2012
CCS	5.3.0.00090	within SDK	5 APR 2013
Qt	4.8.3		
Ubuntu LTS	12.04	http://www.ubuntu.com/download/ubuntu/download	APR 2012
FlashAndBootUtils	2.40	http://sourceforge.net/projects/dflashutils/files/OMAP-L138/	12 APR 2012
PRU	1.03	http://www.ti.com/tool/sprc940	9 JUN 2010
XNRG SW Package	Version	Download Page	Date
SDK_AM437x	2.0.0		7 OCT 2015
Kernel	4.4.39		DIC 2016
U-Boot	2016.11		NOV 2016
Cross Tool		epy-arm-linux-gnueabihf-	
GCC	4.9.3		26 FEB 2013
PinMux		https://dev.ti.com/pinmux/app.html#/config/AM437x/ID_500/ID_0/null/null	
Qt	5.7		

1 Definizione degli ambienti di sviluppo

Per lavorare, ciascuno sviluppatore necessita di una macchina Linux (host) e un target. Opzionalmente si usa anche una macchina Windows (PC).

L'host è principalmente usato per:

- far girare l'applicazione del terminale seriale (p.es. Tera Term)

- usare la utility di scrittura in flash da seriale
- far girare CCS (ricompilare UBL e utility di scrittura in flash, per il debug HW, per lo sviluppo di applicazioni DSP)
- installare l'SDK e il DVSDK
- ricompilare le versioni release di U-Boot e del kernel Linux
- sviluppare applicazioni ARM con CCS, Qt Creator, Eclipse, ecc.
- far girare il TFTP server (per trasferire al target immagini beta kernel e filesystem da U-Boot tramite rete)
- far girare il NFS server (per eseguire il boot del target con NFS come root filesystem)
- far girare il l'applicazione del terminale seriale (opzionale)

Il Server è principalmente usato per:

- far girare CVS e/o SVN e/o GIT server
- far girare SSH e VNC server
- conservare il sorgente nelle repository
- conservare i vari pacchetti SW e librerie di terze parti
- far girare il TFTP server (per trasferire al target immagini release kernel e filesystem da U-Boot tramite rete)
- far girare il NFS server (per eseguire il boot del target con NFS come root filesystem)

1.1 Configurazione Server

- PC workstation, VM con distribuzione Linux
- Distribuzione Linux: Ubuntu 10.04 LTS

Hostname: <serverfirmware>
serverip: <fwserverip>
password: firmware

1.2 Configurazione PC

- PC workstation con Windows OS, terminale seriale e connessione di rete
- OS suggerito: XP

1.3 Configurazione host

- PC workstation standalone, oppure partizione sull'HD del PC Windows, oppure VM¹ con distribuzione Linux
- Distribuzione suggerita: Ubuntu 14.04 LTS 64-bit

Hostname: <hostname>
serverip: <serverip>
password: <password>

Partizionare il disco fisso come segue

Per /: 50GB primary ext4

¹ La Virtual Machine (VM) è un software che gira su un macchina fisica (host) e che crea una macchina virtuale (guest). Nel nostro caso sull'host gira Windows e sul guest gira una distribuzione Linux. Il vantaggio della VM rispetto alla partizione HD è che non occorre riavviare il PC.

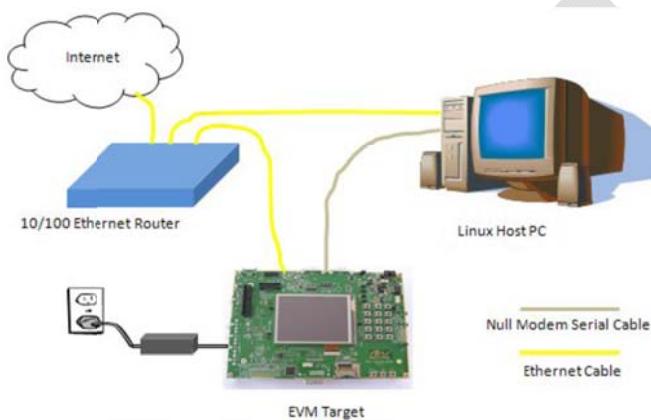
Per home: 190GB primary ext4

Per swap: 10GB primary swap

1.4 Configurazione target

La configurazione minimale dell'HW prevede un cavo seriale NULL MODEM, un PC host, un cavo Ethernet CROSS e un target basato su AM1808/OMAPL138/DA850, proprietario o di acquisto

- EVM P/N: TI TMDXEVM1808L
<http://www.logicpd.com/products/system-on-modules/zoom-am1808-evm/>
- KXEVB: X KXEVB
- OMAPL138-based SOM P/N: LogicPD SOMOMAPL138-10-1602QHIR
- AM1808-based SOM P/N: LogicPD SOMAM1808-10-1602QHIR-B



2 Ubuntu su host

2.1 Releases page

La versione Ubuntu GNU/Linux desktop da utilizzare è la versione LTS suggerita dal SDK. Le release disponibili sono riportate in

<https://wiki.ubuntu.com/Releases>

Attualmente usiamo Ubuntu 14.04 LTS.

2.2 Download page

<http://www.ubuntu.com/download/ubuntu/download>

2.3 Installazione Ubuntu su partizione HD

Inserire disco di installazione Ubuntu 10.04 LTS e riavviare

Selezionare data e ora locali, tipo tastiera.

Quando richiesto selezionare partizione manuale e definire

Partizione primaria (> 30GB, ext3 o ext4, per montare il root filesystem /, formattazione abilitata) (p.es. /dev/sda2)

Area di swap (> 500MB) (p.es. /dev/sda5)

Esempio



Quando richiesto inserire nome user (<user>), password (<password>) e nome computer (es. "fub").

Sotto "Advanced" specificare dove effettuare l'installazione del bootloader (grub2)

Nota. Se è presente su HD anche una partizione con windows 7, allora installare il bootloader nella stessa partizione di Linux (p.es. /dev/sda2), in quanto in questo caso converrà utilizzare un loader di windows per gestire il multiboot all'avvio (p.es. EasyBCD), altrimenti installare bootloader nel MBR (/dev/sda)

Specificare Server Proxy di default (p.es. 192.168.0.221:8080)

Completare l'installazione guidata.

2.4 Installazione Ubuntu su VM

In alternativa, nel caso si voglia installare Ubuntu su una VM, seguire

http://processors.wiki.ti.com/index.php/How_to_Build_a_Ubuntu_Linux_host_under_VirtualBox

Spesso a corredo di un HW linux viene fornita una VM contenente una distribuzione linux host (p.es. Ubuntu) con all'interno tutti i tools di sviluppo del target. Ad esempio

LogicPD fornisce la VM VirtualBox *1024065_AM1808_OMAP-L138_Linux_PSP_SDK_03-21-00-04_VM.ova* a corredo della EVM OMAPL138

ENG/CAM fornisce la VM VMware *bsp-imx6.vmx* a corredo della EVM M6

2.5 Upgrade a Ubuntu 14.04

Una volta disponibile la versione 14.04 si può procedere all'aggiornamento.

Una volta effettuato l'upgrade provvedere a lanciare il comando

```
host $ sudo visudo
```

e andare a commentare le opzioni di default *env_reset* e *secure_path*.

Questo è necessario per consentire il caricamento di path senza restrizioni all'interno degli script di compilazione e installazione utilizzati.

Se si è installata la versione a 64-bit, aggiungere il supporto runtime a 32-bit a un host PC con installato 64-bit Ubuntu, seguendo le istruzioni riportate al paragrafo relativo.

2.5.1 VMware

<http://www.vmware.com/products/player/overview.html>

2.5.2 VirtualBox

<https://www.virtualbox.org/wiki/Downloads>

Occorre installare VirtualBox e VirtualBox Extension Pack

VirtualBox-4.2.14-86644-Win.exe

Oracle_VM_VirtualBox_Extension_Pack-4.2.14-86644.vbox-extpack

Poi si lancia VirtualBox e si importa la VM

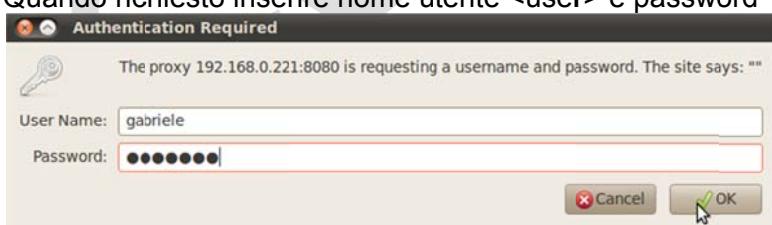
1024065_AM1808_OMAP-L138_Linux_PSP_SDK_03-21-00-04_VM.ova

2.6 Configurazione Firefox

- 1) Lanciare Firefox. Da **Edit->Preferences->Advanced->Network->Settings** selezionare il Server Proxy in rete per l'accesso a internet



Quando richiesto inserire nome utente <user> e password <password>



2.7 Cartelle principali

Creare <tools>, la directory per i vari programmi, tools, utilities
host \$ mkdir -p /home/<user>/programs

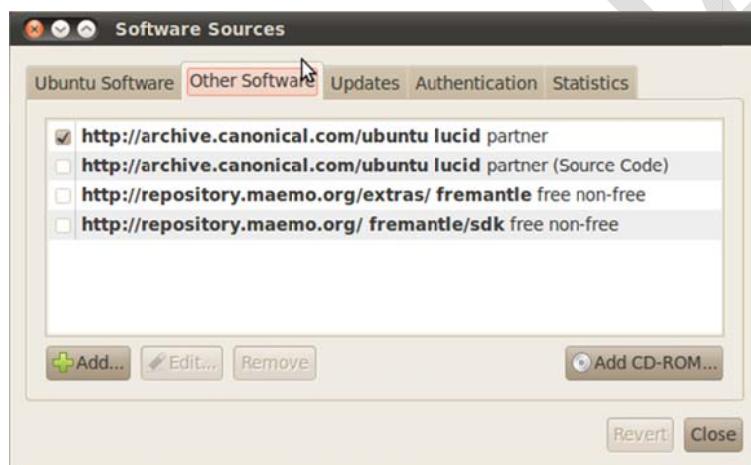
2.8 Configurazione Software Sources

2) In System->Administration->Software Sources

Su tab “Ubuntu Software” impostare Download from *Server for United States*



Su tab “Other Software” selezionare repository di default e aggiungere altre se desiderato



Su tab “Updates” lasciare il default



Eseguire **Close** e **Reload**

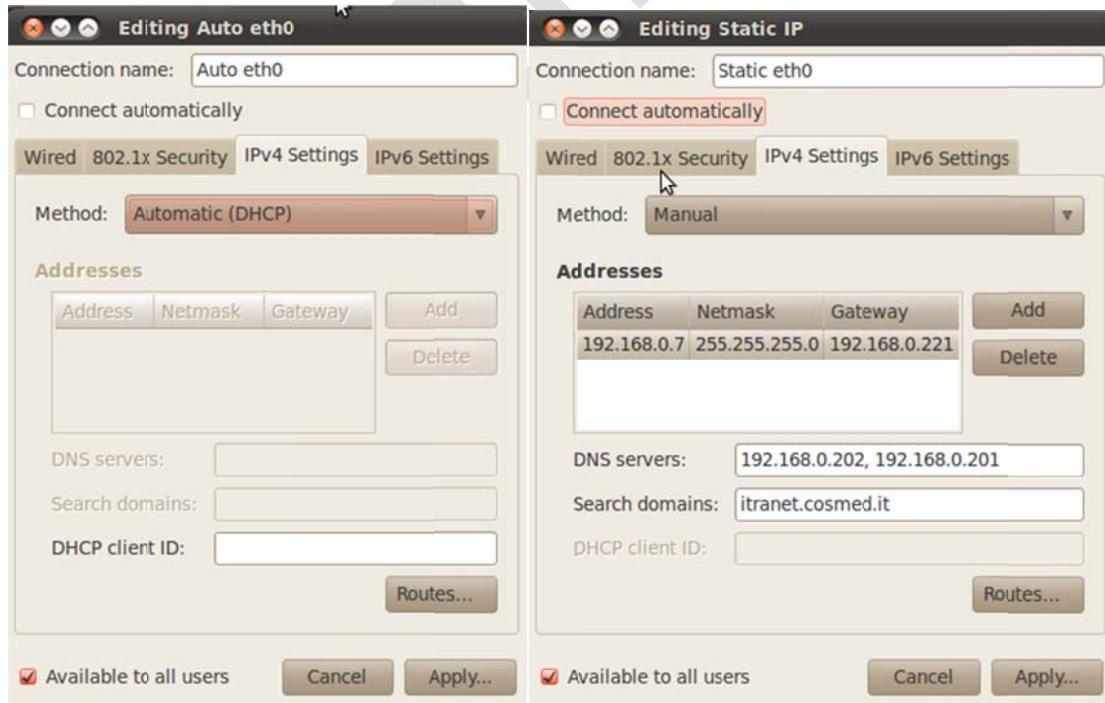
Aggiornare il sistema (da fare ogni tanto)

```
host $ sudo apt-get update
```

Da Update Manager verranno eventualmente notificati degli upgrade. **Check** e **Install**.

2.9 Connessioni di rete

- 3) L'indirizzo IP della connessione di rete è per default assegnato mediante DHCP ("Auto eth0"). Talvolta può essere utile abilitare un indirizzo statico. Da **System->Preferences-> Network Connections** aggiungere una nuova connessione "Static eth0"



2.10 Configurazione VNC

Il PC deve potersi connettere al Server con un terminale remoto VNC.

- 1) Da **System->Administration->Synaptic Package Manager** selezionare e installare il pacchetto VNC client

`xtightvncviewer`

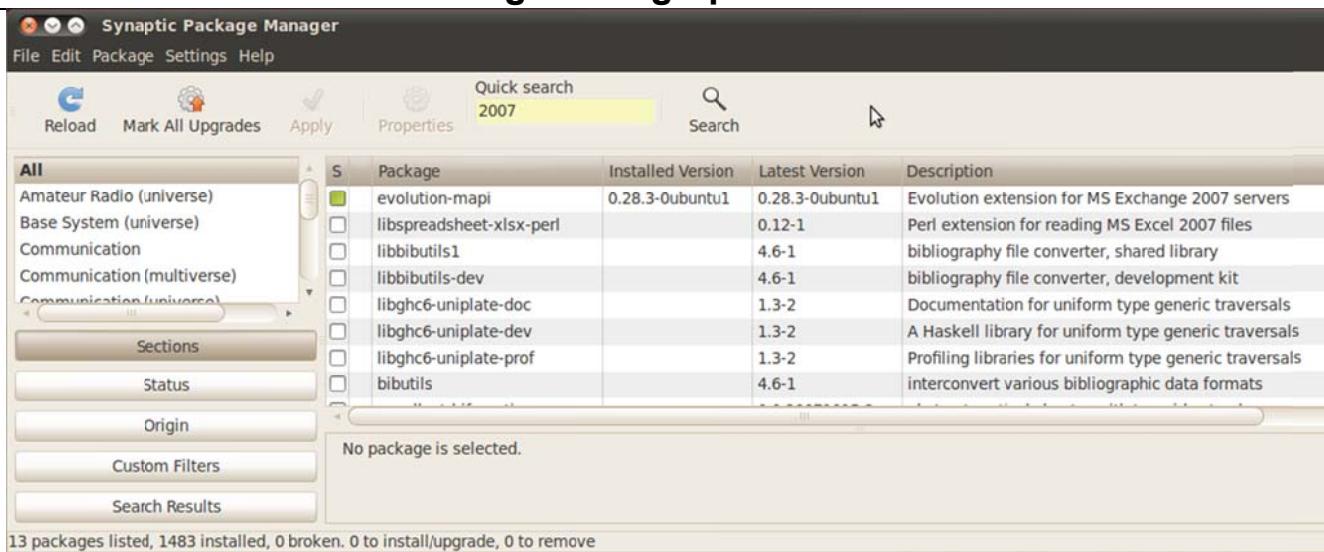
- 2) Da **Applications->Internet->TerminalServerClient**, inserire l'IP del Server, selezionare il protocollo VNC e **Connect**



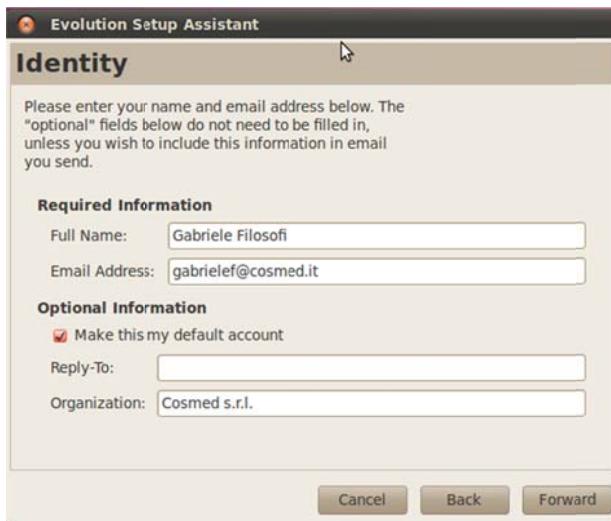
2.11 Configurazione Client di Posta Elettronica

- 4) Da **System->Administration->Synaptic Packet Manager**
Installare *evolution-mapi*

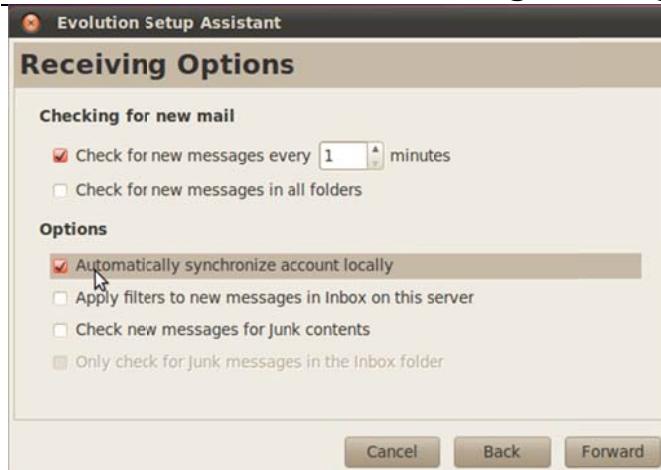
Engineering Specification - confidential



Lanciare Applications>Office->EvolutionMail



Quando richiesto inserire <password>



3 TI SDK e PSP su host

L'SDK è un insieme di pacchetti SW già testati a corredo di un target di valutazione (EVM).

Nel nostro caso l'SDK è relativo al processore AM1808.

Usare sempre la versione più recente del SDK, ma solo se apporta variazioni al supporto AM18xx.
Si assume che la toolchain per ARM sia stata già attiva

Download site:

<http://focus.ti.com/docs/tools/folders/print/linuxsdk-am1x.html>

3.1 Installazione e configurazione SDK

- Scaricare il pacchetto di installazione specifico dell'AM180x e anche il pacchetto di installazione CCSv5 in /home/<user>

```
host $ cd /home/<user>
host $ wget ti-sdk-am180x-evm-05.03.02.00-Linux-x86-Install AM180x EVM SDK
http://software-dl.ti.com/dspsw/dspsw\_public\_sw/sdo\_sb/targetcontent/sdk/AM1x/latest/exports/ti-sdk-am180x-evm-xx.xx.xx.xx-Linux-x86-Install
```

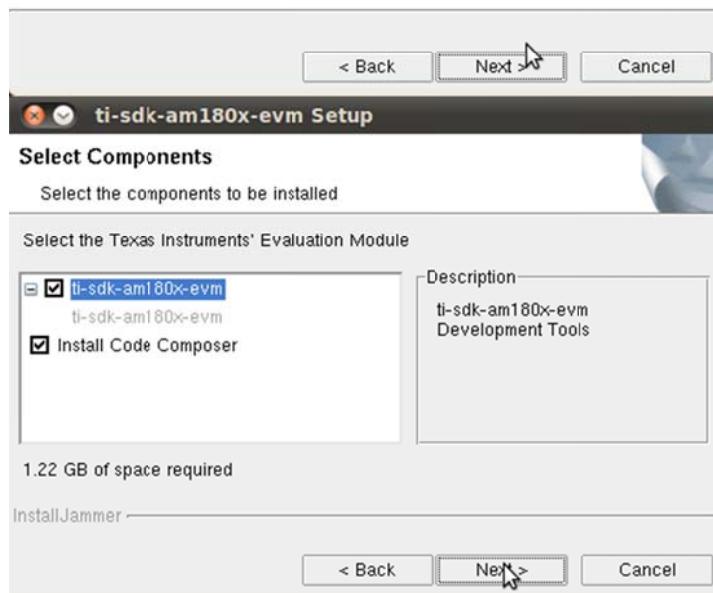
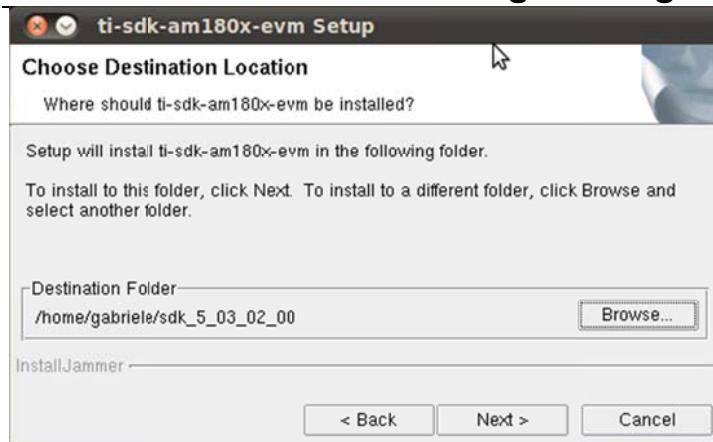
- Ora abbiamo i due pacchetti in /home/<user>

```
ti-sdk-am180x-evm-xx.xx.xx.xx-Linux-x86-Install
CCS-x.x.x.x-Sitara-ARM.tar.gz
host $ tar xzvf CCS_x.x.x.x_Sitara_ARM.tar.gz
host $ chmod 755 ti-sdk-am180x-evm-xx.xx.xx.xx-Linux-x86-Install
host $ ./ti-sdk-am180x-evm-xx.xx.xx.xx-Linux-x86-Install
```

Quando richiesto, specificare la path della directory di installazione.

Noi useremo

/home/<user>/sdk_x_xx_xx_xx (Esempio: sdk_5_03_02_00)



- 1) Se si seleziona anche "install Code Composer" proseguire poi con l'installazione del CCSv5 (vedi paragrafo relativo)
- 3) Eseguire

```
host $ cd sdk_x_xx_xx_xx
host $ ./setup.sh
```

Attenzione. Questo script installerà se necessario altri pacchetti (tftp,nfs,minicom,..)

Engineering Specification - confidential

Quando richiesto, specificare la path della directory dove scompattare il target file system per l'NFS
<rootpath> = /home/<user>/workdir/kx/filesys, ma non prima di avere rinominato una eventuale directory preesistente, che altrimenti verrebbe sovrascritta.

Per quanto riguarda la directory tftp specificare la cartella <tftpbootdir>.

Queste path possono essere rieditate successivamente nel file Rules.make

Per le altre opzioni richieste, accettare sempre il default.

3.2 Ridefinire le cartelle dei sorgenti per board kx

- 4) Riorganizzare le directory interne in board_support dove sistemare i sorgenti kernel e u-boot

```
host $ cd board_support
host $ mkdir -p src/kx/u-boot
host $ mkdir -p src/kx/kernel
host $ mv linux-2.6.37-psp03.21.00.04.sdk src/kx/kernel
host $ mv u-boot-2010.12-psp03.21.00.04.sdk src/kx/u-boot
host $ cd src/kx/kernel
host $ cp -Rf linux-2.6.37-psp03.21.00.04.sdk linux-01.00          (cartella <linux>)
host $ cd ../u-boot
host $ cp -Rf u-boot-2010.12-psp03.21.00.04.sdk u-boot-01.00        (cartella <uboot>)
```

- 5) Editare Rules.make aggiornando le seguenti informazioni

```
PLATFORM=da850_kx
UBOOT_MACHINE=da850kx_config
DESTDIR=/home/gabriele/workdir/kx/filesys
LINUXKERNEL_INSTALL_DIR=$(TI_SDK_PATH)/board-support/src/kx/kernel/linux-01.00
```

Aggiungere anche

```
TFTPBOOTDIR=/var/lib/tftpboot/kx
```

- 6) Editare Makefile

Correggere la riga dove è definito il nome del file di configurazione kernel nella sezione make linux con

```
$(MAKE) -C $(LINUXKERNEL_INSTALL_DIR) ARCH=arm
CROSS_COMPILE=$(CROSS_COMPILE) $(PLATFORM)_defconfig
```

Aggiornare la directory dei sorgenti u-boot con

```
$(MAKE) -C $(TI_SDK_PATH)/board-support/src/u-boot/u-boot-01.00
CROSS_COMPILE=$(CROSS_COMPILE) $(UBOOT_MACHINE)
```

Aggiungere anche la seguente riga nella sezione make linux_install

```
install $(LINUXKERNEL_INSTALL_DIR)/arch/arm/boot/ulimage $(TFTPBOOTDIR)
```

Aggiungere anche la seguente riga nella sezione make u-boot_install

```
install $(TI_SDK_PATH)/board-support/src/u-boot/u-boot-01.00/u-boot.bin $(TFTPBOOTDIR)
```

- 7) Applicare le patches a <linux> e <uboot>

- 8) Ora si può lavorare nelle cartelle <linux> e <boot>, procedendo ad esempio alla ricompilazione, ecc.

```
host $ cd sdk_x_xx_xx_xx
host $ make clean
host $ make linux
host $ make linux_install
```

- 9) Per ricompilare e reinstallare anche l'applicazione demo matrix_gui

```
host $ make all  
host $ make install
```

(ricompila anche uboot, il kernel e i moduli)

3.3 Installazione PSP completo

L'SDK contiene del PSP solo alcune parti. Mancano i progetti CCS delle utility per la flash.

- 10) Scaricare il pacchetto di installazione specifico dell'AM180x in /home/<user> e scompattare

```
host $ cd /home/<user>  
host $ wget http://software-  
dl.ti.com/dsps/dsps_public_sw/psp/LinuxPSP/DaVinci_03_21/03_21_00_04//exports/DaVinci-PSP-  
SDK-03.21.00.04.tgz  
host $ tar zxf DaVinci-PSP-SDK-03.21.00.04.tgz
```

Attualmente è disponibile il PSP-SDK-03.22.00.02, con i seguenti miglioramenti

- Linux Kernel Version 3.3
- U-Boot Version 2012.04.01
- U-Boot SPL support for SPI/NAND/NOR flash and MMC/SD
- Serial flashwriter
- USB dual role OTG support

Ma al momento restiamo su PSP-SDK-03.21.00.04

3.4 Repository e nome delle immagini u-boot, kernel e rootfs

Una volta creata una immagine del kernel, di u-boot o del rootfs (versione xx.xx, patch level pppp, versione sorgente originale #####) può servire conservare tale immagine (eventualmente insieme a files di configurazione utilizzati) in una cartella chiamata images. Questa cartella si deve trovare insieme alla cartella dei sorgenti.

Creare <uboot-images>

```
host $ cd <uboot>  
host $ mkdir -p ..images
```

Creare <linux-images>

```
host $ cd <linux>  
host $ mkdir -p ..images
```

Creare <rootfs-images>

```
host $ cd <rootfs>  
host $ mkdir -p ..images
```

All'immagine si assegna nome

imagename#####-xx.xx-pppp.imageextension

Esempio-1:

```
host $ cd <uboot>  
host $ cp u-boot.bin ..images/u-boot-2011.12-rc1-01.00-0002.bin  
host $ cp u-boot-ubl.bin ..images/u-boot-ubl-2011.12-rc1-01.00-0002.bin
```

Esempio-2:

```
host $ cd <linux>
```

Engineering Specification - confidential

```
host $ cp arch/arm/boot/uImage .../images/uImage-2.6.37-0009-01.00  
host $ cp .config .../images/2.6.37-01.00-0009.config
```

Esempio-3:

```
host $ cd <rootfs>  
host $ cp ..//rootfs-sum.jffs2 ..//images/rootfs-sum-sdk-0012-01.00.jffs2  
host $ cp ..//rootfs.ubi ..//images/ rootfs-sdk-01.00-0012.ubi
```

4 Yocto

Arago sono le distribuzioni linux create da TI utilizzando i tools OE (Open Embedded) e Yocto. Le distribuzioni Arago sono basate su OE-Classic mentre le più nuove Meta-Arago sono basate su Oe-core. Il build di una distribuzione Arago parte da Recipes (oggetti meta-data) e Layers (gruppi di Recipes e altri files). La struttura a layers è del tipo

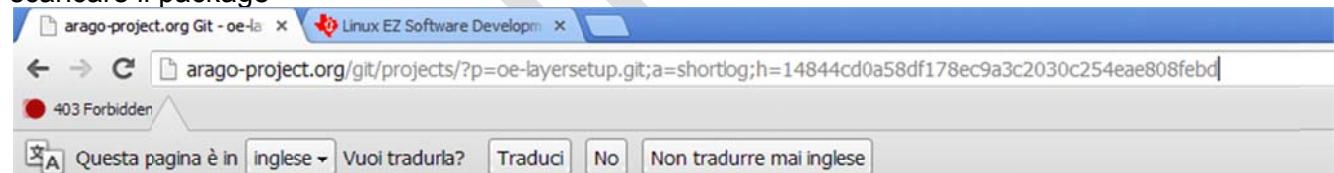
```
meta-arago  
meta-ti  
meta-oe  
oe-core
```

Le distribuzioni Sitara TI Arago per ARM sono Yocto compatibili.

<http://e2e.ti.com/blogs/b/toolsinsider/archive/2013/04/16/ti-sitara-linux-is-yocto-project-compatible.aspx>

<http://arago-project.org/git/projects/?p=oe-layersetup.git;a=shortlog;h=14844cd0a58df178ec9a3c2030c254eae808febd>

scaricare il package



<http://arago-project.org/git/projects/oe-layersetup.git/shortlog>

[summary](#) | [shortlog](#) | [log](#) | [commit](#) | [commitdiff](#) | [tree](#)
[HEAD](#) · prev · next

oe-layersetup.git

2013-05-02 Franklin S... Lock commits for amsdk 05.07.00.00 official release
2013-05-01 Denys Dmytryienko coresdk-2013.04.00-config: move meta-arago to include...
2013-04-26 Denys Dmytryienko coresdk-2013.04.00-config: pin down all the layers...

[commit](#) | [commitdiff](#) | [tree](#) | [snapshot](#)
[commit](#) | [commitdiff](#) | [tree](#) | [snapshot](#)
[commit](#) | [commitdiff](#) | [tree](#) | [snapshot](#)

```
host $ tar zxf oe-layersetup-14844cd0a58df178ec9a3c2030c254eae808febd.tar.gz  
host $ cd oe-layersetup  
host $ ./oe-layer-tool-setup.sh -f configs/amsdk/amsdk-05.07.00.00-config.txt  
host $
```

Uno dei tool principali di Yocto è BitBake, per creare una distribuzione Linux a partire da Recipes, packages e metadata. Altri tools di Yocto permettono di creare SDK specifici per ciascuna distribuzione.

Yocto

X Srl,

File: Linux_Development_Guide_rev21.docx

page 27/299

1. After access git: git clone git://arago-project.org/git/projects/oe-layersetup.git, the TI Yocto project files can be get to local.
2. "./oe-layer-tool-setup.sh -f configs/amsdk/amsdk-<version>-config.txt" is used to choose a SDK version in this file. Then which version can be chosen to build Ti335X SDK?
3. The document shows build can be started in "oe-layersetup/build", but it does not show how to build the module, package or image? Need more detail about this.
4. If some other specific module is to be added to existing project, how to add?
5. If some module is to be deleted from existing project, how to delete?
6. Is there some more detailed documents about TI Yocoto?

5 ARM Toolchain

La toolchain è un insieme di SW tools necessari a generare un eseguibile. Comprende linker, assemblatore, archiver, compilatore e libreria C.

L'architettura dell'host è x86, mentre l'architettura del target è ARM, quindi occorre una cross-platform toolchain per ARM

5.1 CodeSourcery

Una possibile cross-platform toolchain GNU per ARM, da installare sull'host, è CodeSourcery Sourcery G++ Lite 2009q1-203. Esiste anche una Commercial Edition a pagamento.

<http://www.codesourcery.com/sjpp/lite/arm/portal/release858>

Scaricare il pacchetto “IA32 GNU/Linux TAR”, in quanto quello di installazione grafico non funziona bene su Ubuntu

Packages

Download	MD5 Checksum
Recommended Packages	
IA32 GNU/Linux Installer	c8833da1cde15cb77e76635139256fb2
IA32 Windows Installer	902bd1f8a4e6517c325c6bf439797a9a
Advanced Packages	
IA32 GNU/Linux TAR	1a6e88782f08b09a0e6cef545a1712ec
IA32 Windows TAR	e605f1c851fd9b0f7b95d5c70081d04c
Source TAR	db7bfaa8dfba06503de28459f906b1b5

- 2) Dopo aver scaricato il pacchetto, in /var/local, estrarre con

```
host $ cd /var/local  
host $ bunzip2 package.tar.bz2
```

dove <package> è il nome del pacchetto, p.es.

arm-2009q1-203-arm-none-linux-gnueabi-i686-pc-linux-gnu

- 3) Creare la directory di installazione ed entrarci

```
host $ mkdir -p CodeSourcery  
host $ cd CodeSourcery
```

- 4) Estrarre

```
host $ tar xf ..package.tar
```

- 5) Editare .bashrc e aggiungere le righe per esportare sia la path dei tools che le man pages

```
export PATH=/var/local/CodeSourcery/arm-2009q1/bin:$PATH  
export MANPATH=/var/local/CodeSourcery/arm-2009q1/share/doc/arm-arm-none-linux-gnueabi/man:$MANPATH
```

- 6) Aggiornare le variabili di ambiente e verificare l'installazione

```
host $ source ~/.bashrc  
host $ arm-none-linux-gnueabi-gcc
```

reply: "no input files"

- 7) Creare link simbolici per Eclipse

```
host $ cp <tools>/mkl /var/local/CodeSourcery/arm-2009q1/bin  
host $ cd /var/local/CodeSourcery/arm-2009q1/bin  
host $ ./mkl
```

5.2 Arago ARM Toolchain gcc4.5.3

- 8) I nuovi SDK (ver 5) non usano più Code Sourcery Lite ma una toolchain basata su GCC Open Embedded. Quindi non c'è più bisogno di scaricare separatamente una toolchain. Infatti i tools stanno nella cartella linux-devkit dell'SDK.
Con questa toolchain, quando si ricompila u-boot o il kernel, usare

arm-none-linux-gnueabi-

invece di

arm-arago-linux-gnueabi

- 9) Editare .bashrc e aggiungere le righe per esportare sia la path dei tools che le man pages

```
export PATH=<sdk>/linux-devkit/bin:$PATH
```

6 Eclipse su host

L'IDE è l'ambiente grafico di sviluppo che integra i comandi della toolchain, del debugger simbolico, ecc. Eclipse è l'IDE di sviluppo delle applicazioni ARM che utilizzeremo sull'host

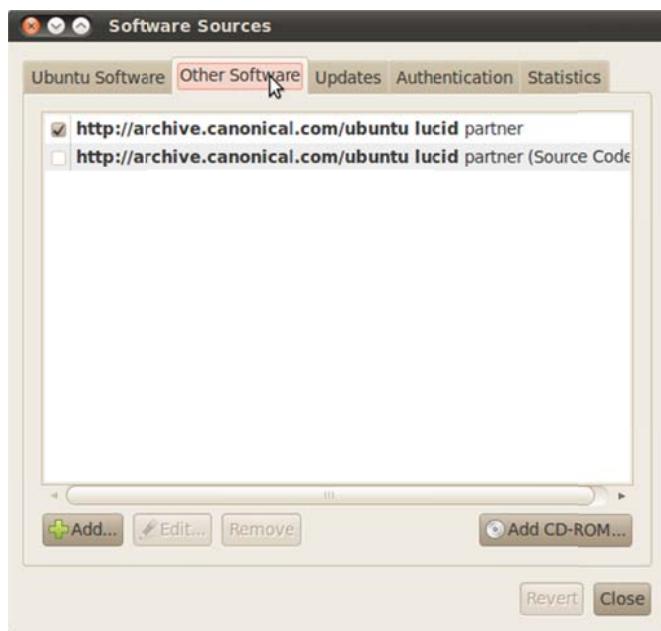
Il pacchetto suggerito è Eclipse IDE for C/C++ Developers (Juno CDT).

Esiste anche la versione Helios

6.1 Installazione JRE

Dapprima è necessario installare il JRE (Java Runtime Environment) della Sun.

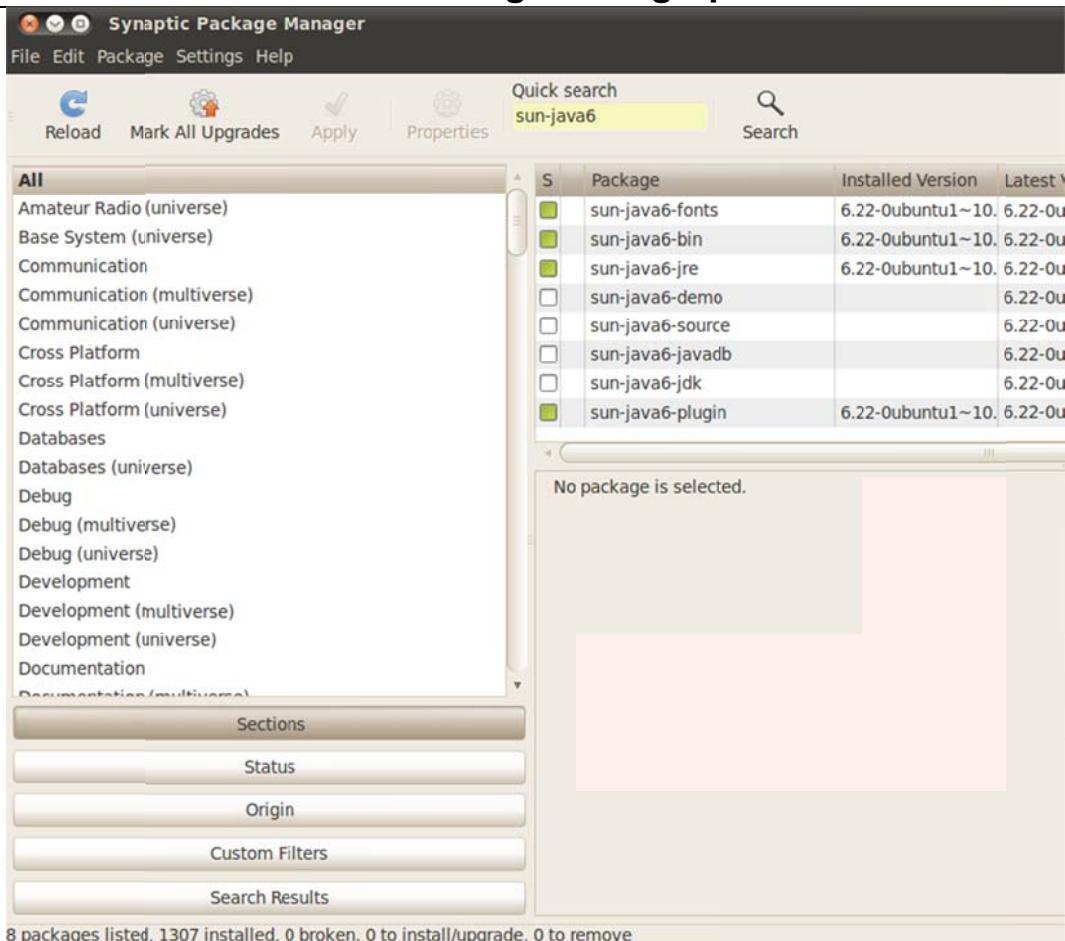
- 3) Assicurarsi di avere una connessione di rete a internet.
In **System->Administration->SoftwareSources** assicurarsi di avere la seguente repository



- 4) Da **System->Administration->SynapticPackageManager** selezionare e installare i pacchetti

sun-java6-bin
sun-java6-fonts
sun-java6-jre
sun-java6-plugin

Apply



6.2 Eclipse download page

<http://www.eclipse.org/downloads/download.php?file=/technology/epp/downloads/release/galileo/SR2/eclipse-cpp-galileo-SR2-linux-gtk.tar.gz>

6.3 Installazione Eclipse

<https://help.ubuntu.com/community/EclipseWebTools>

- 5) Dopo aver scaricato il pacchetto, ad esempio in /opt, estrarre con

```
host $ cd /opt  
host $ tar xzf eclipse-cpp-galileo-SR2-linux-gtk.tar.gz  
host $ rm eclipse-cpp-galileo-SR2-linux-gtk.tar.gz
```

- 6) Per creare il collegamento sul desktop all'applicazione, right click sul desktop ->**Create Launcher**



Cliccando sull'icona predefinita si può sostituirla con l''icona di eclipse, che si trova in /opt/eclipse/icon.xpm)

6.4 Installazione Eclipse-CDT Galileo da Synaptic Manager

Assicurarsi di avere una connessione di rete a internet.

In **System->Administration->SynapticPackageManager** installare eclipse-cdt utilizzando la repository

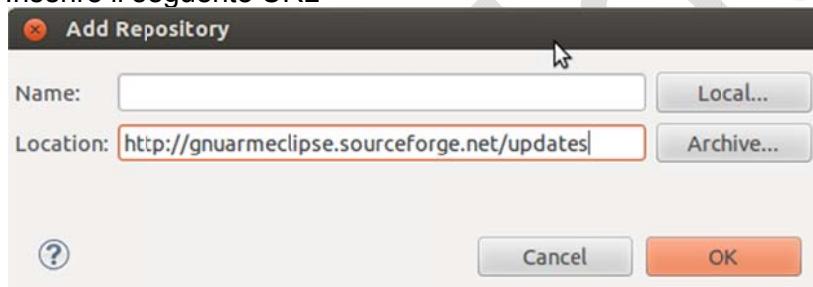
<http://download.eclipse.org/tools/cdt/releases/galileo>

6.5 Installazione GNU ARM Eclipse Plug-In

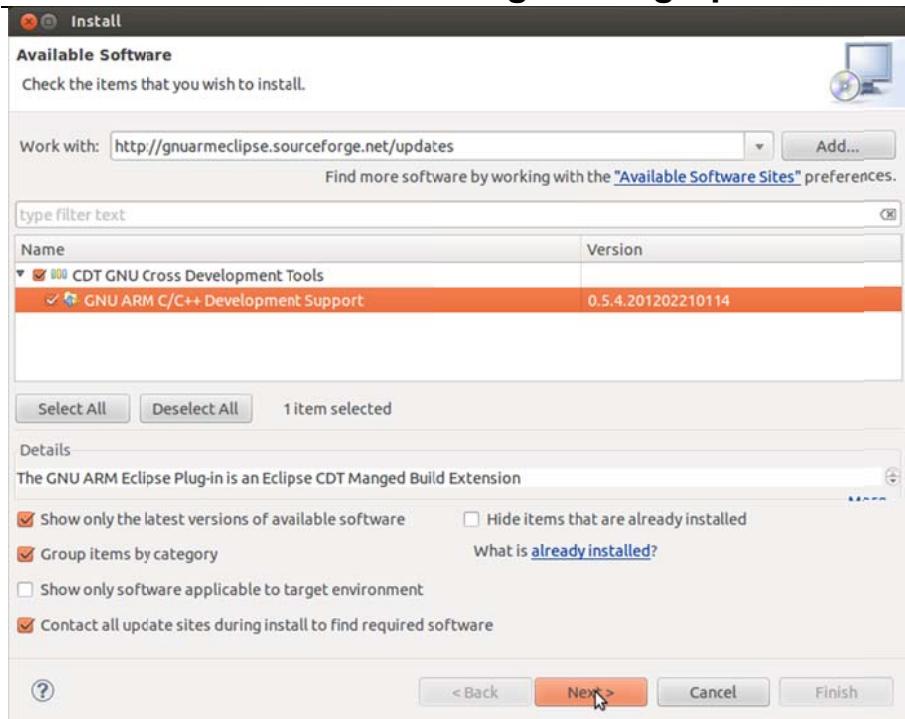
E' necessario installare la Plug-In per il cross developement ARM

- 7) Da **Help->Install New Software->Add**

Inserire il seguente URL



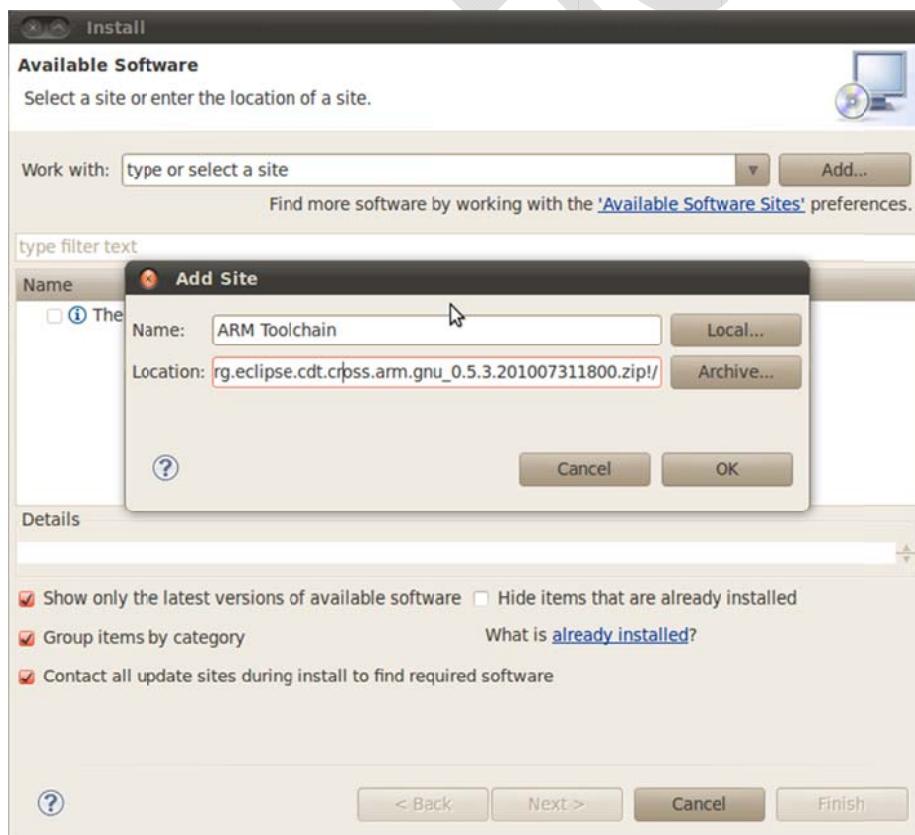
Selezionare e completare l'installazione



- 8) Un modo alternativo è scaricare il pacchetto, ad esempio in <tools>, dal URL

<http://sourceforge.net/projects/gnuarmeclipse/>
 package: org.eclipse.cdt.cross.arm.gnu_0.5.3.201007311800.zip
 package: org.eclipse.cdt.cross.arm.gnu_0.5.4.201202210114.zip

e da **Help->Install New Software->Add->Archive**
 si va a puntare il file .zip del Plug-In



e proseguire fino a installazione completata.

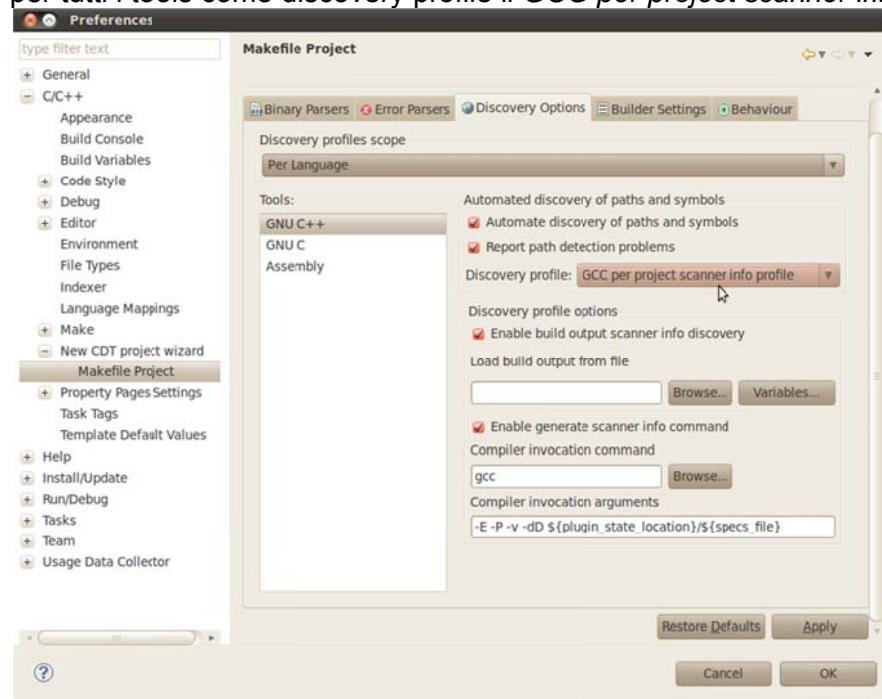
Questo modo è sconsigliato in quanto non si beneficia della funzione **Help->Check For Updates**

- 9) Riavviare Eclipse

6.6 Configurazione Eclipse e creazione progetto

- 10) Da **Window->Preferences->C/C++**

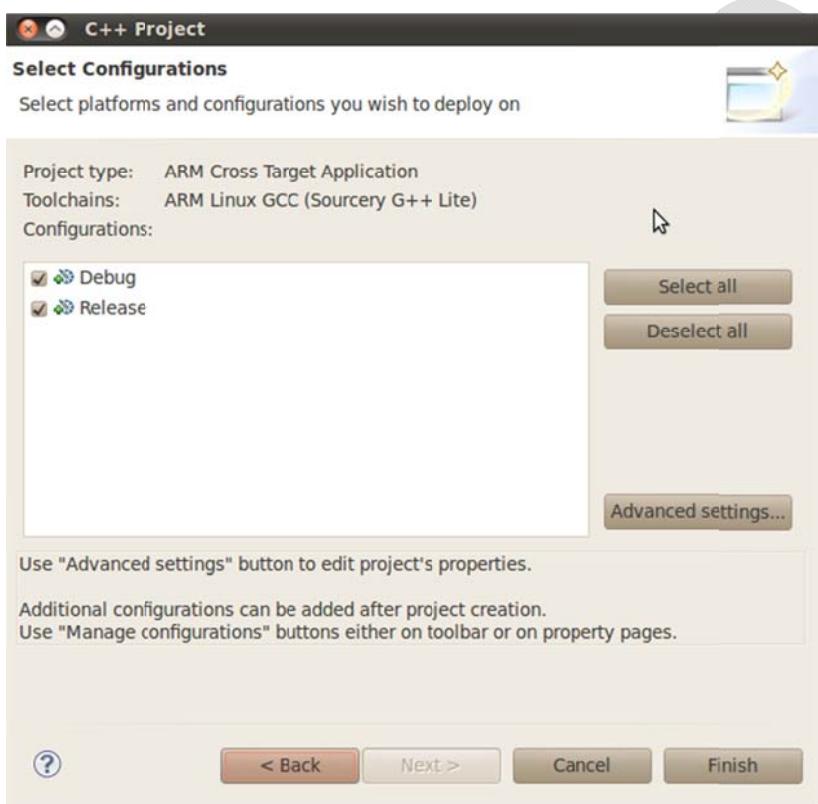
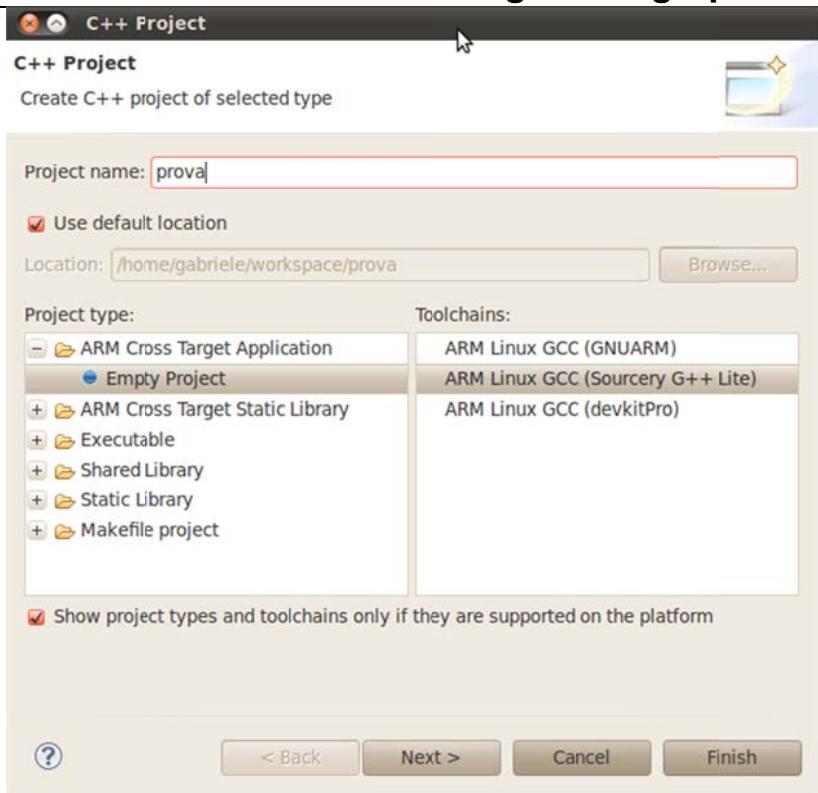
sotto **NewCDTProject Wizard->MakefileProject DiscoveryOptions** specificare “Per Language” e per tutti i tools come discovery profile il **GCC per project scanner info profile Sourcery G++ Lite**



- 11) Creare un nuovo progetto da **File->New->C++Project**, e specificare la Toolchain **Sourcery G++ Lite**.

Next

Engineering Specification - confidential



Sotto Advance Settings
C/C++General->PathsAndSymbols aggiungere la path della libc
/var/local/CodeSourcery/arm-2009q1/arm-none-linux-gnueabi/libc/usr/include



Selezionare anche le opzioni

Add to all configurations

Add to all languages

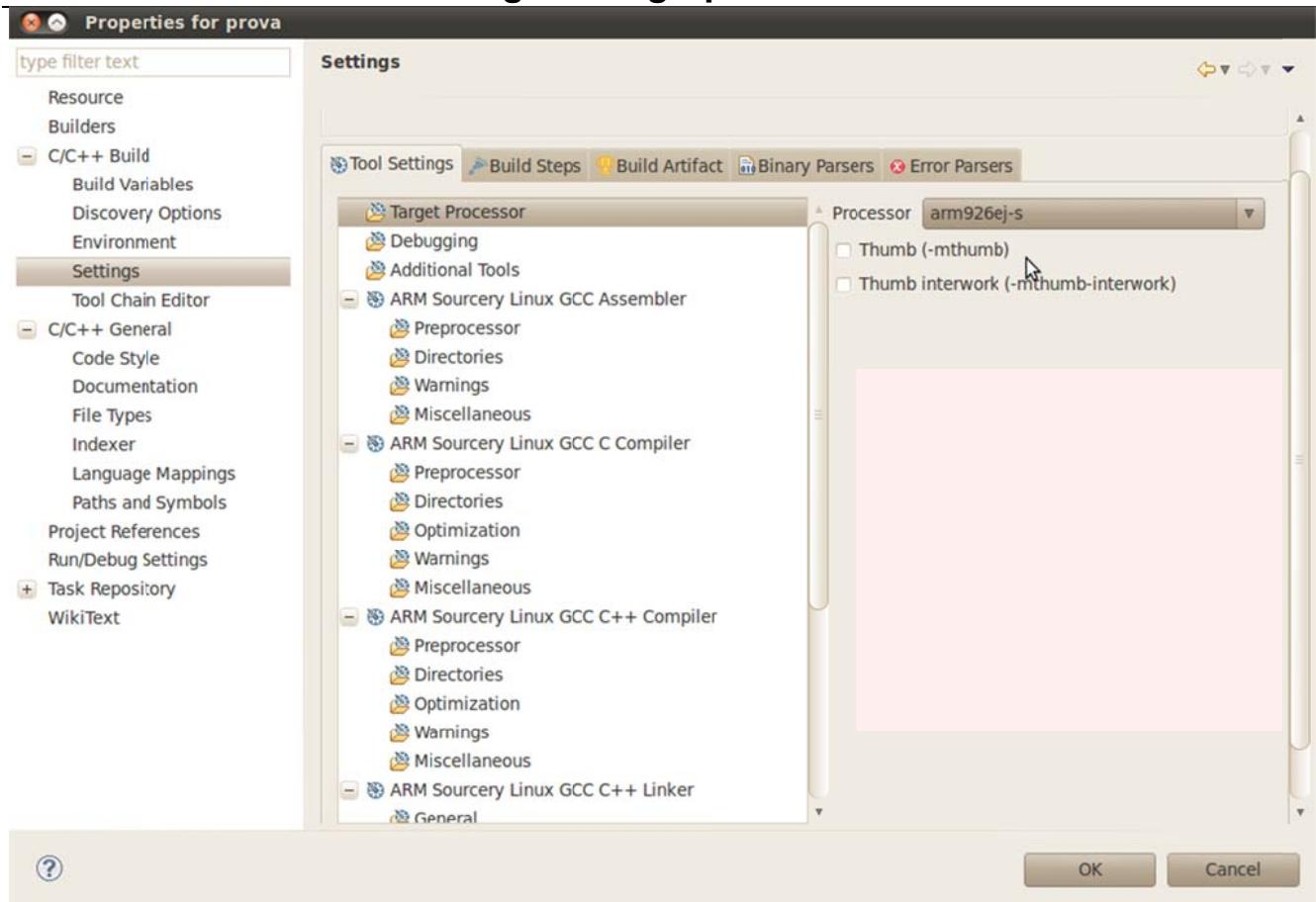
Is a workspace path

Apply

Finish

- 12) Selezionare il progetto nel Project Explorer. Right Click e in Properties sotto **C/C++Build->SettingsTargetProcessor** selezionare **arm926ej-s**. Deselezionare opzione *Thumb*

Engineering Specification - confidential

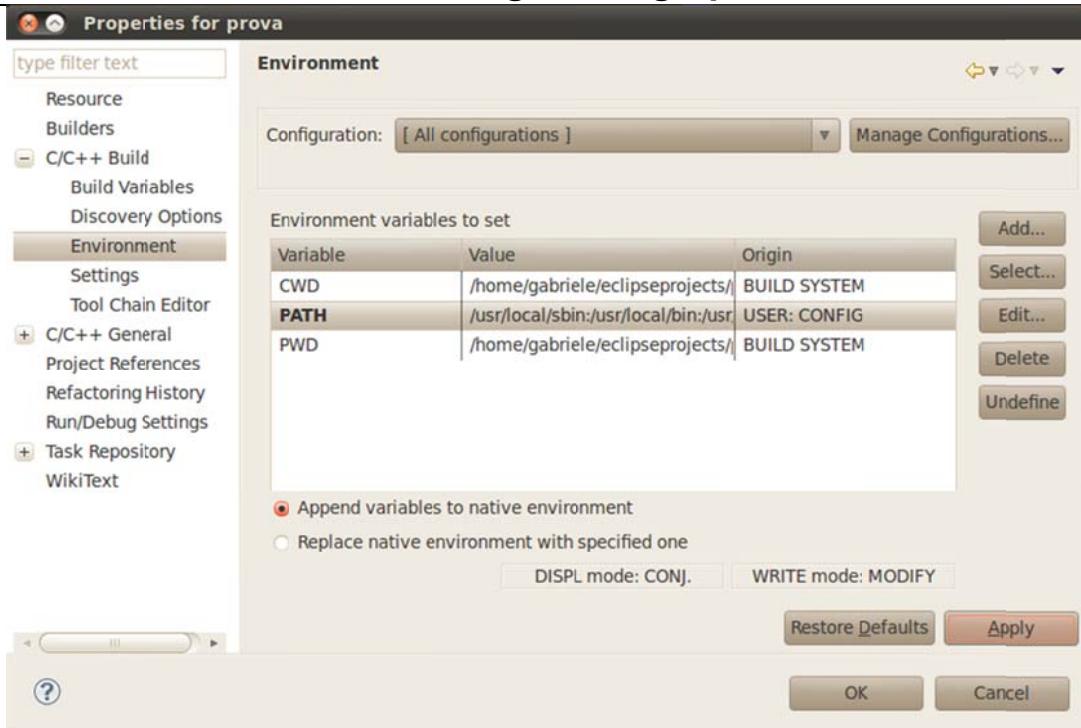


- 13) Selezionare il progetto nel Project Explorer. Right Click e in Properties sotto **C/C++Build->Environment** aggiungere la variabile di ambiente PATH appendendo a mano la path della toolchain ARM. La variabile completa sarà qualcosa del tipo

/usr/local/sbin:/usr/local/bin:/usr/sbin:/sbin:/bin:/usr/games:/var/local/CodeSourcery/arm-2009q1/bin:\${PATH}

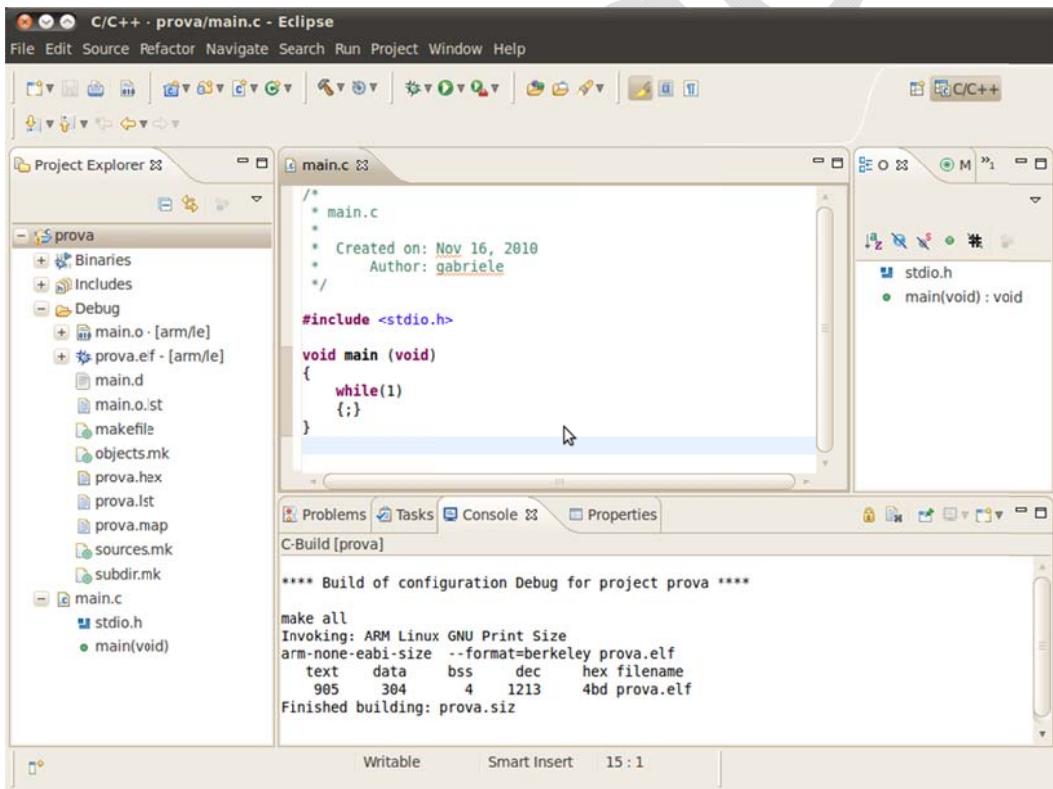
Se presente, selezionare anche
Add to all configurations

Engineering Specification - confidential



14) Per creare un file di main nel progetto, selezionare il progetto nel Project Explorer. Right Click, **New->FileFromTemplate**. Chamarlo "main.c", **Finish**.

15) Editare qualcosa di semplice (vedi figura)



16) Selezionare il progetto nel Project Explorer. **Project->Clean**, **Project->BuildProject**

7 Tools di utilità generale

7.1 Installazione minicom

```
host $ sudo apt-get install minicom
```

- 1) Lanciare utilità di configurazione

```
host $ sudo minicom -s
```

- 2) Sotto **SerialPortSetup** impostare

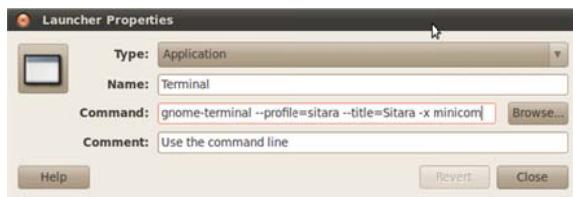
Serial device: /dev/ttyUSB0
 Bps/Par/Bits: 115200 8N1
 Flow Control: No

e salvare configurazioni
 Save setup as dfl

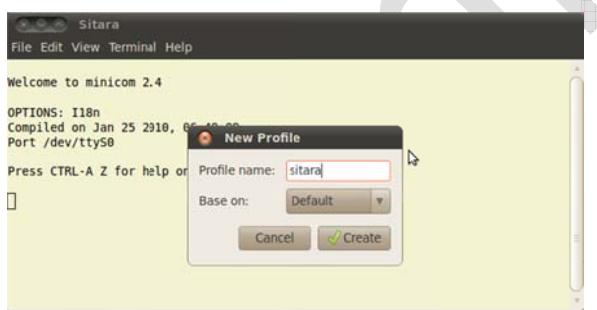
Note: bisogna impostare l'opzione “Serial device” in base al nodo di IO che si intende utilizzare.
 Per lanciare minicom da terminale

```
host $ minicom
```

- 3) Per creare il collegamento sul desktop, right click sul desktop ->**Create Launcher**



Associare un'icona al collegamento e da **File->NewProfile**, creare il profilo “sitara” a piacimento



7.2 Installazione gnome-commander

```
host $ sudo apt-get install gnome-commander
```

7.3 Installazione tmux

Tmux è un terminal multiplexer. Permette di aprire più terminali e passare facilmente dall'uno all'altro

Engineering Specification - confidential

It lets you switch easily between several programs in one terminal, detach them (they keep running in the background) and reattach them to a different terminal

```
host $ sudo apt-get install tmux
```

To split tmux vertically, just press (Ctrl-b) + %

To split tmux horizontally, press (Ctrl-b) + "

Move Left : (Ctrl-b) + Left arrow

Move Right : (Ctrl-b) + Right arrow

Move Up : (Ctrl-b) + Up arrow

Move Down : (Ctrl-b) + Down arrow

Move to the next pane : (Ctrl-b) + o

More command on

<http://linoxide.com/how-tos/install-tmux-manage-multiple-linux-terminals/>

7.4 Installazione vim

Vim è un editor

```
host $ sudo apt-get install vim
```

8 TI CCSv5

L'SDK contiene il CCSv5 che gira sull'host. Comunica col target tramite TCP.

La licenza serve solo se si vuole debuggare con un JTAG debugger diverso da emulatore Blackhawk USB100v2.

8.1 Installazione e configurazione CCS v5.1 su host

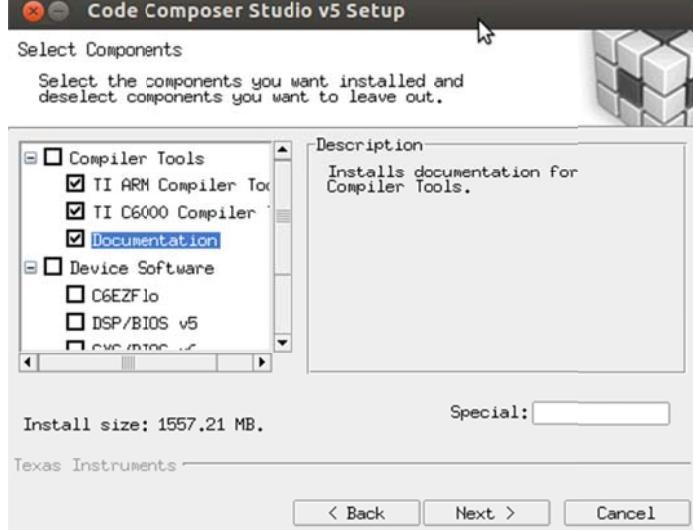
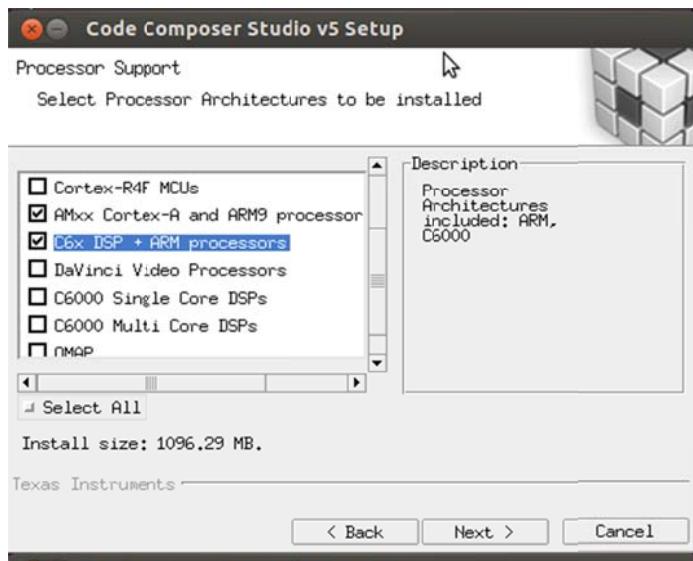
CCS v5.3 è basato su [Eclipse 3.6](#) e [CDT 7.0 \(Helios\)](#).

http://processors.wiki.ti.com/index.php/CCSv5_Getting_Started_Guide

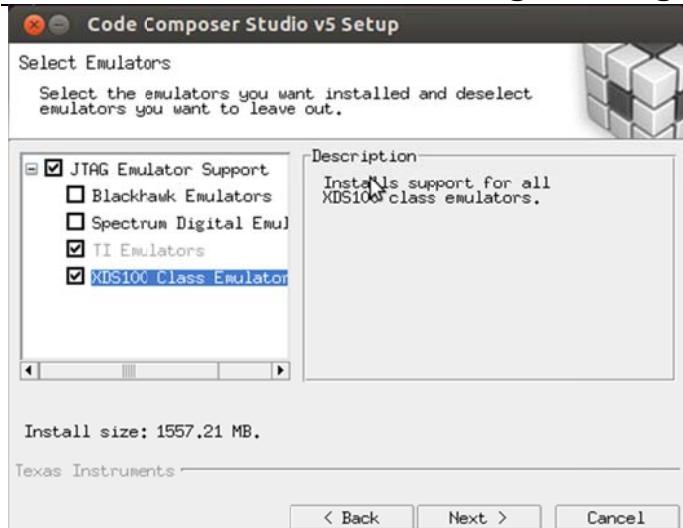
L'installazione del CCSv5 viene avviata contestualmente a quella del SDKv5 (vedi paragrafo relativo), oppure può essere lanciata individualmente dalla cartella in cui è stato estratto il pacchetto con

```
host$ tar xvzf CCS_5.x.x.xxxxx-Sitara-ARM.tar.gz
host$ cd CCS
host$ chmod 777 ccs_setup_5.x.x.xxxxx.bin
host$ sudo ./ccs_setup_5.x.x.xxxxx.bin --setupfile ccs_installini.xml
```

- 11) Quando richiesto, specificare la path <CCS_INSTALL_DIR> della directory di installazione. Il default è /home/<user>/ti/ccsv5



Non selezionare "Add TI plug-ins into an existing Eclipse install".
Selezionare i pacchetti richiesti. E' sufficiente selezionare ARM9 processors e C6x DSP + ARM processors. Per i tools è sufficiente selezionare ARM e C6000.
DSP/BIOS e SYS/BIOS sono solo per dispositivi con DSP.
Successivamente è sempre possibile aggiungere altri pacchetti lanciando il programma di installazione.



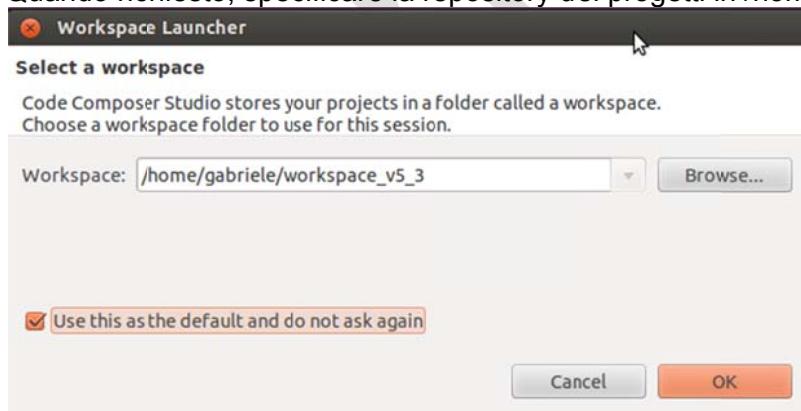
12) Al termine dell'installazione eseguire

```
host $ sudo ./ti/ccsv5/install_scripts/ti_xds100_linux_install.sh  
host $ sudo ./ti/ccsv5/install_scripts/install_drivers.sh
```

13) Lanciare CCSv5 dal desktop

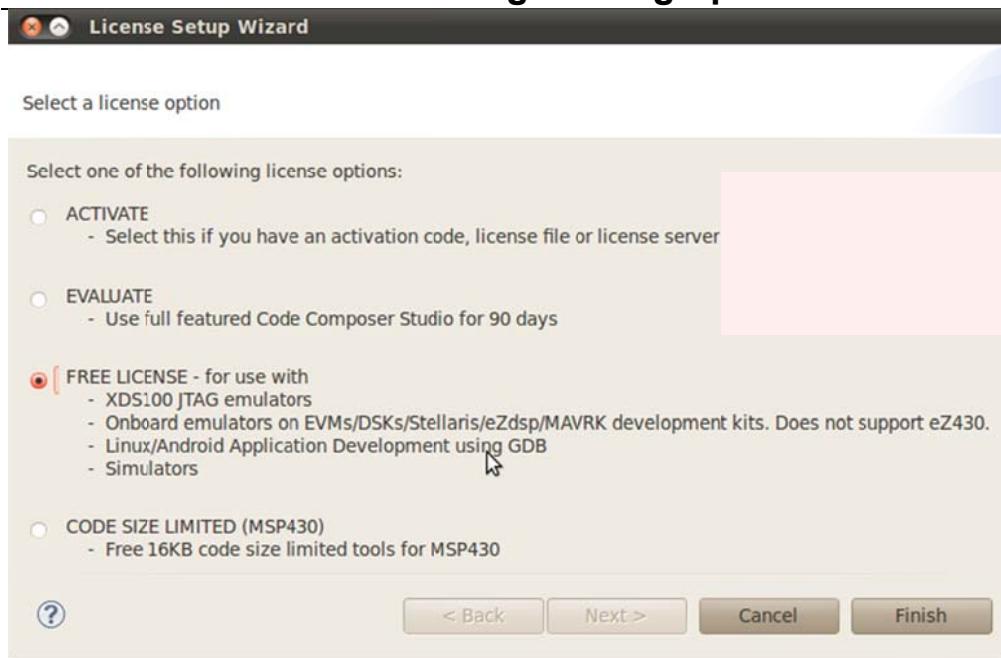


Quando richiesto, specificare la repository dei progetti in /home/<user>/workspace_v5_3

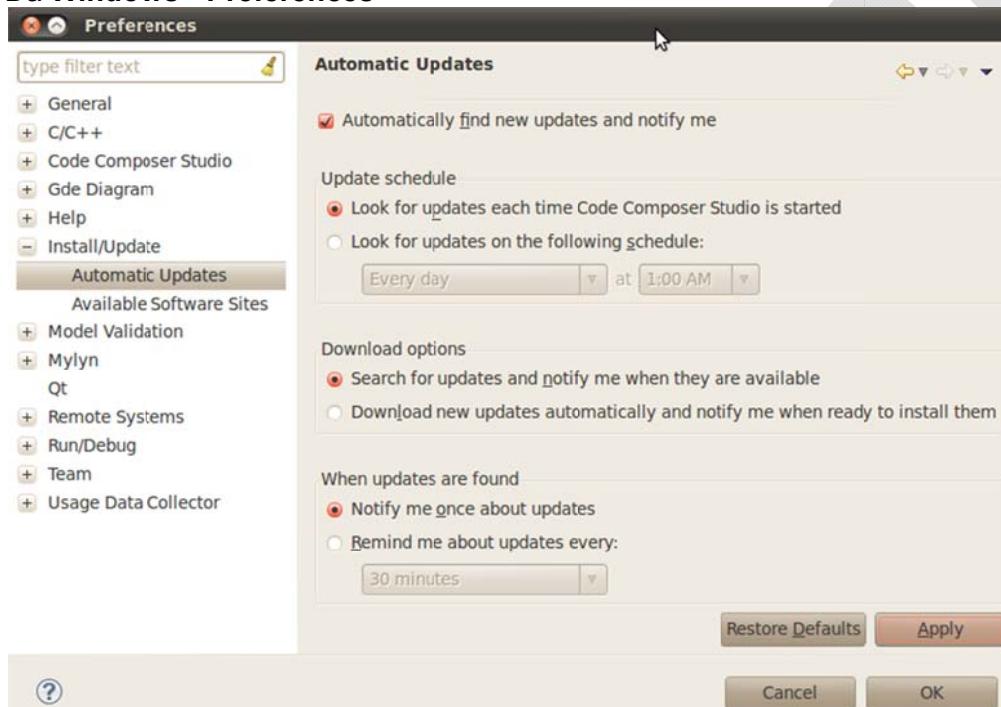


Quando richiesto, specificare l'opzione di licenza

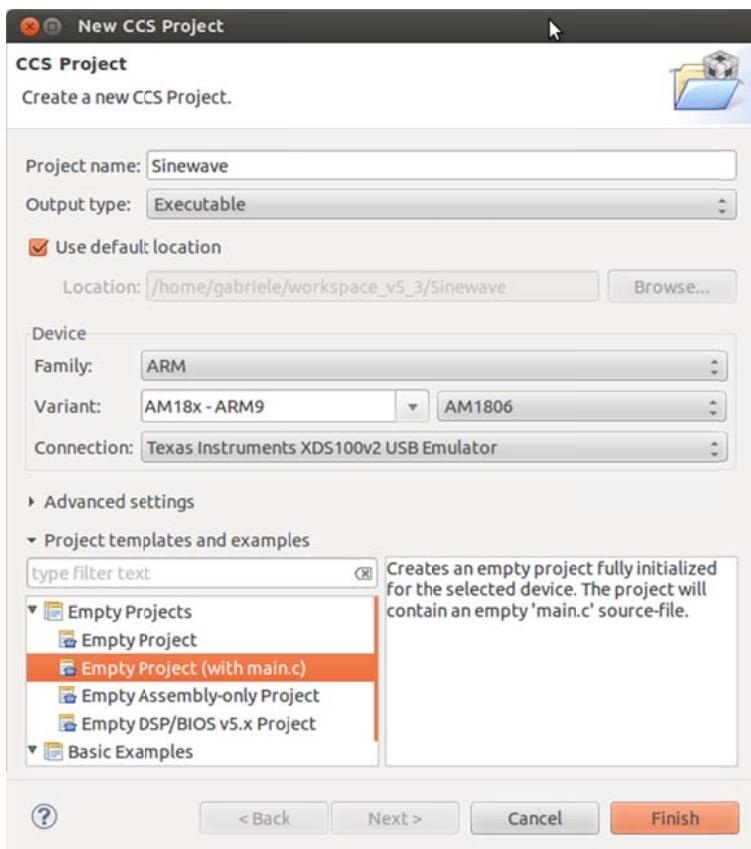
Engineering Specification - confidential



Da Windows->Preferences



8.2 Creare un nuovo progetto

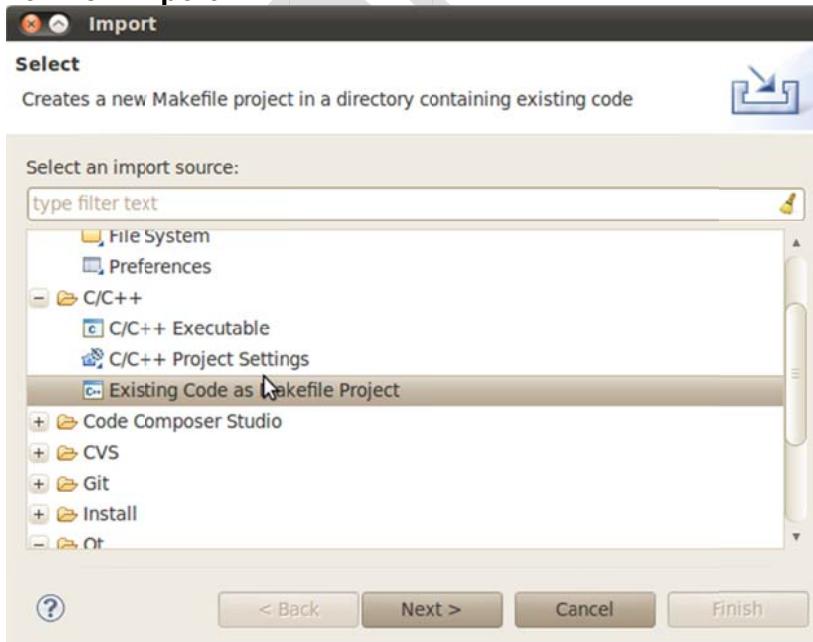


To do ..

8.3 Importare e compilare progetto U-Boot

Si assume che SDKv5 sia installato

Da **File->Import...**

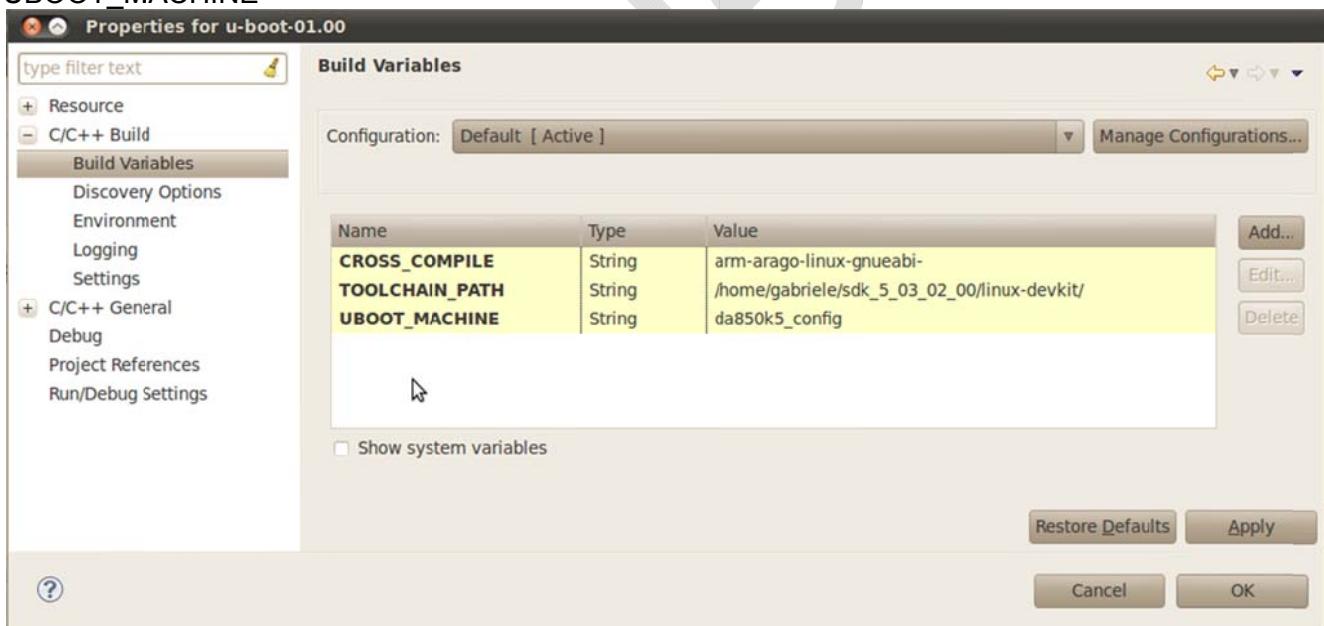




Premere **Finish**.

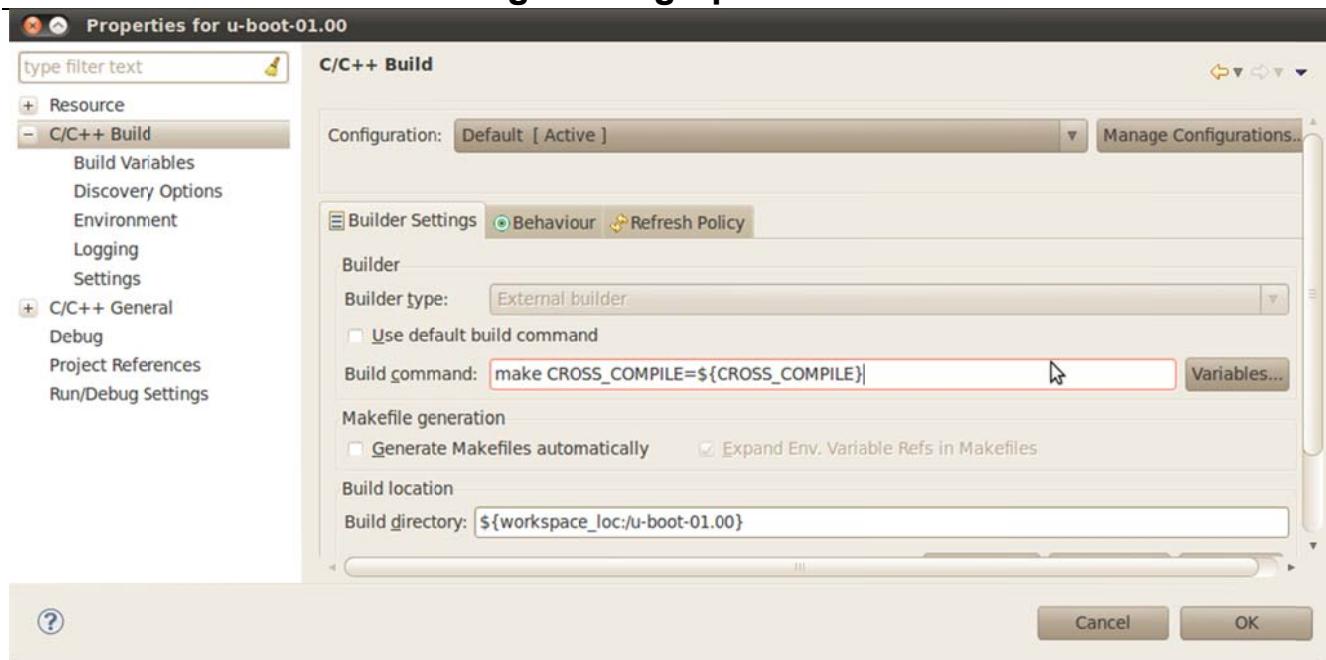
Puntando il mouse sul progetto, **BuildOptions->C/C++Build->BuildVariables**

Premendo il pulsante **Add** definire le variabili CROSS_COMPILE, TOOLCHAIN_PATH, UBOOT_MACHINE

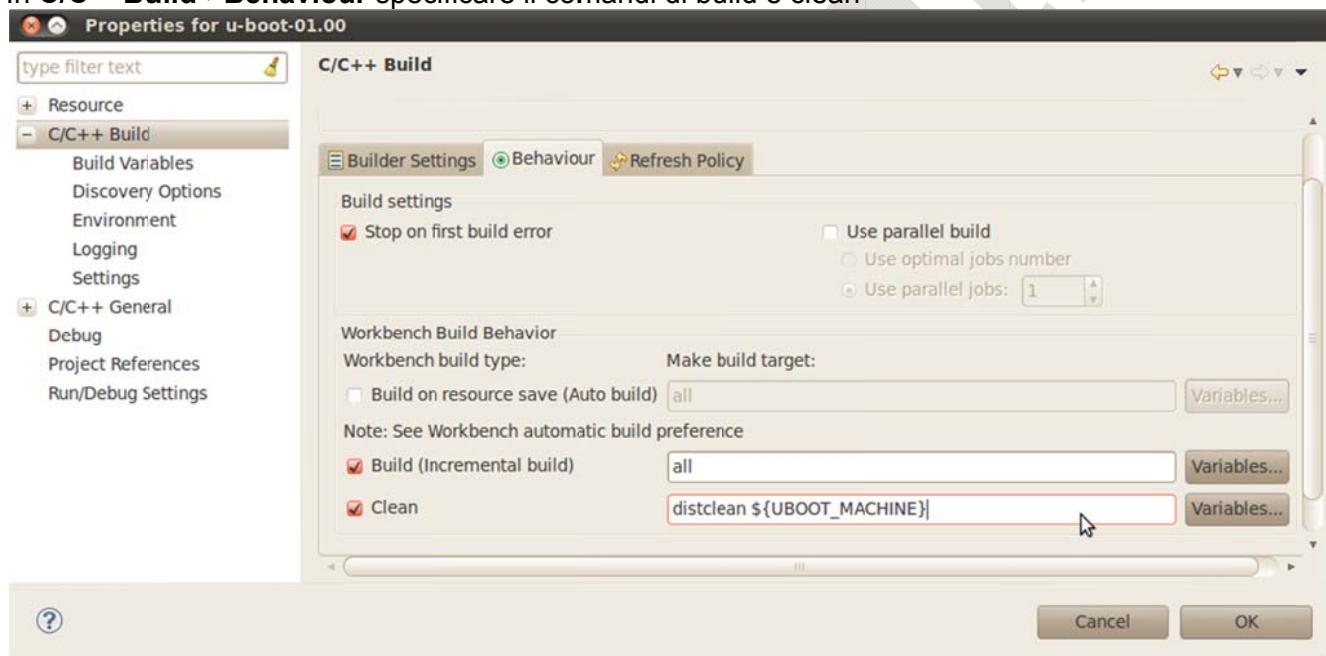


In **C/C++Build->Configuration**, specificare il Build Command

Engineering Specification - confidential

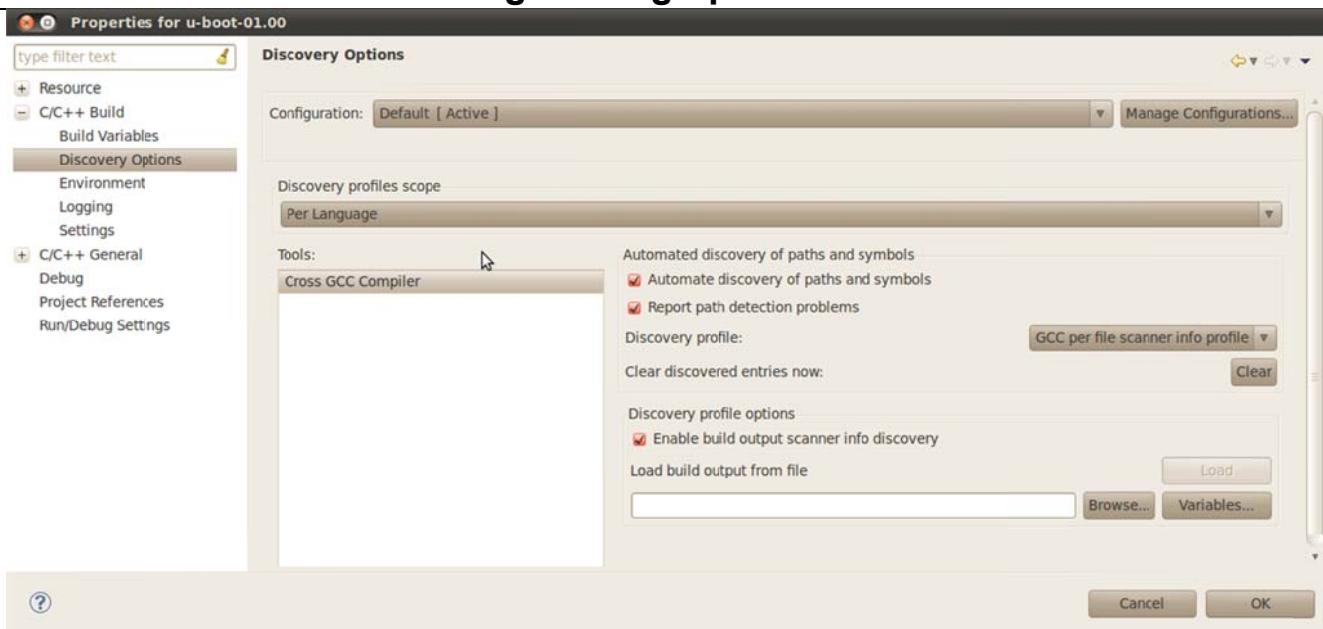


In C/C++Build->Behaviour specificare il comandi di build e clean

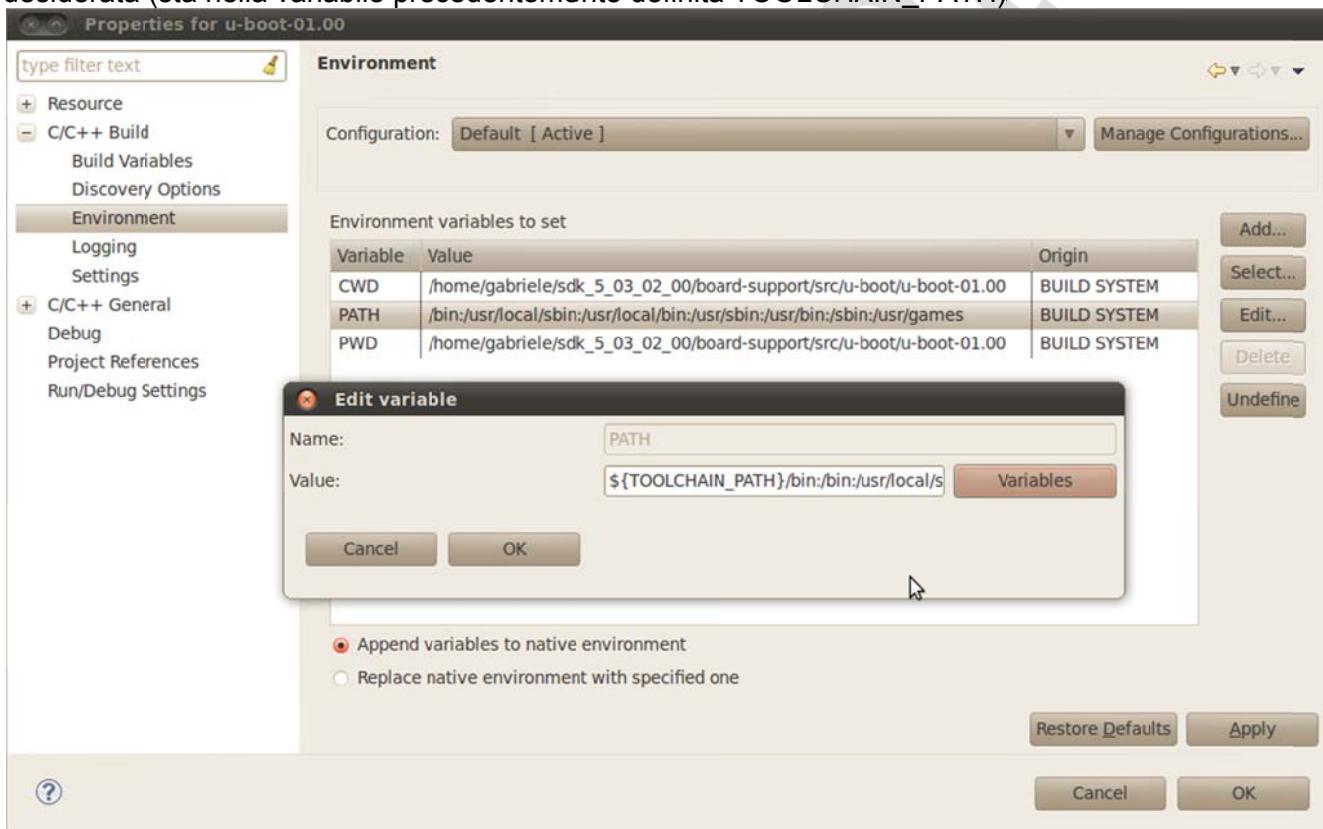


Verificare le seguenti opzioni di Discovery

Engineering Specification - confidential

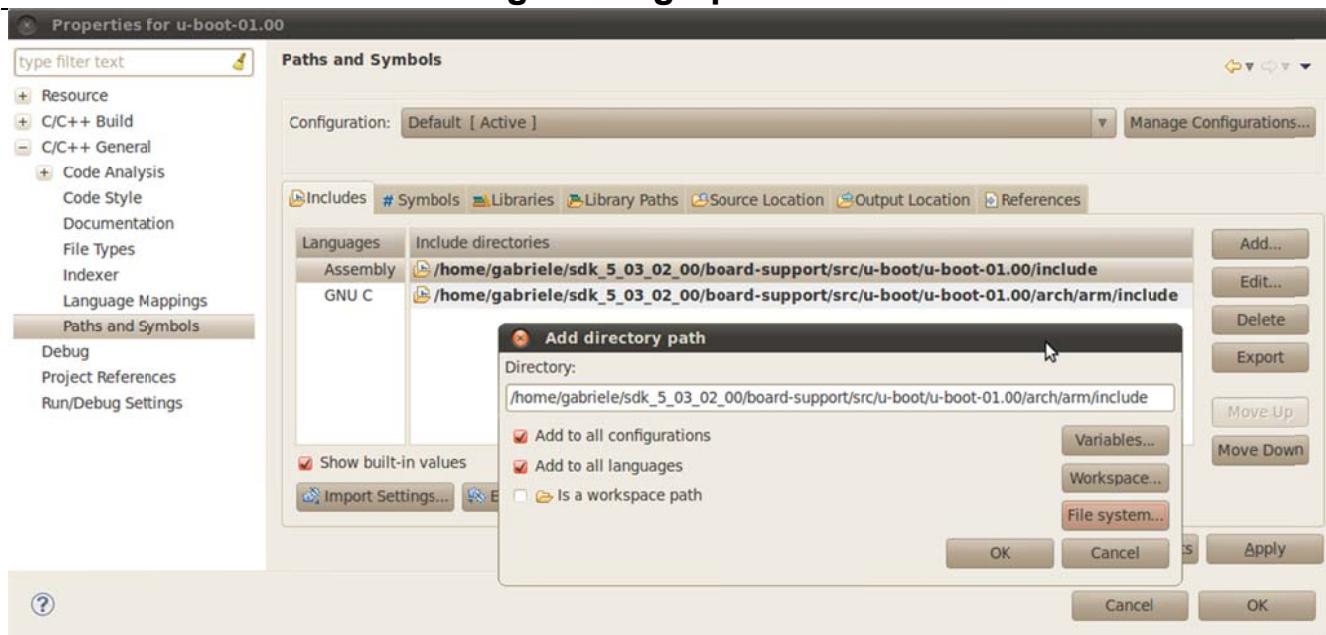


In **C/C++Build->Environment** editare la variabile PATH e appendere il percorso della toolchain desiderata (sta nella variabile precedentemente definita TOOLCHAIN_PATH)



In **C/C++General->PathsAndSymbols** sotto **Includes** aggiungere i percorsi seguenti

Engineering Specification - confidential

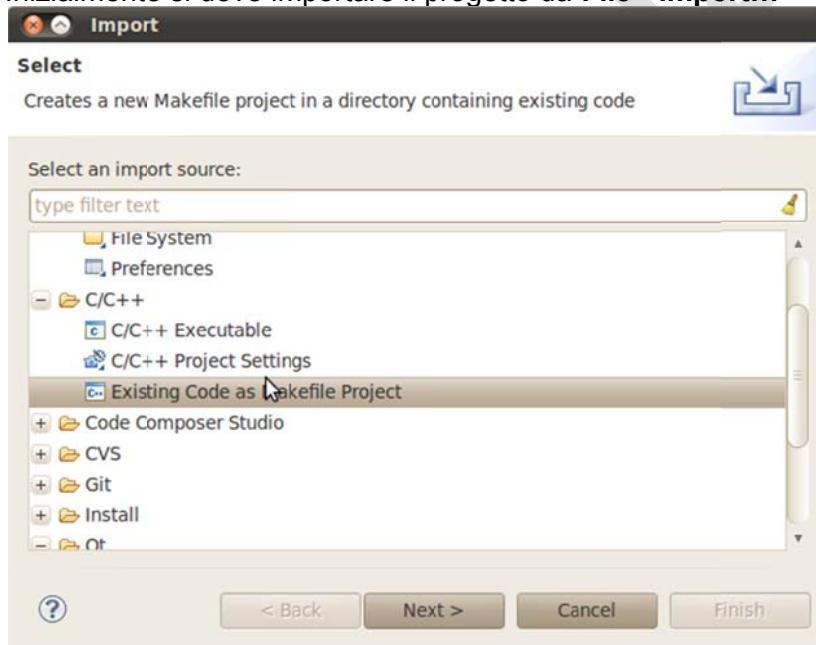


Ricompile con Project->Clean, Build

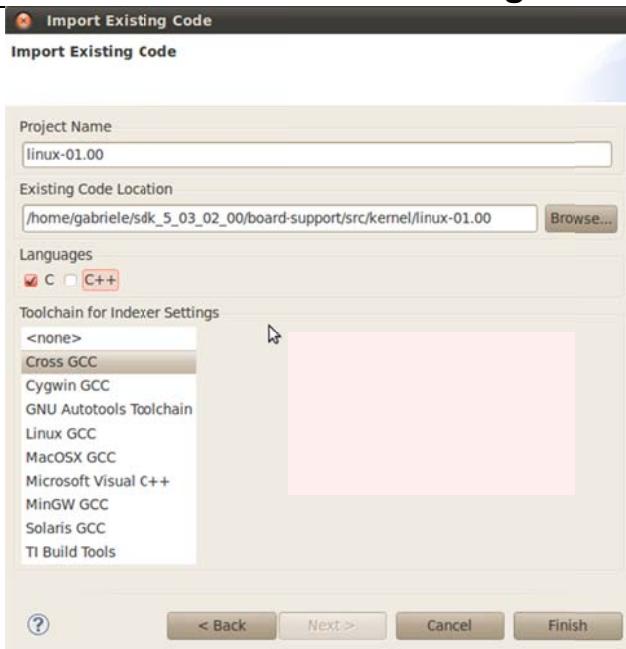
8.4 Importare e compilare progetto Linux

Si assume che SDKv5 sia installato.

Inizialmente si deve importare il progetto da **File->Import...**



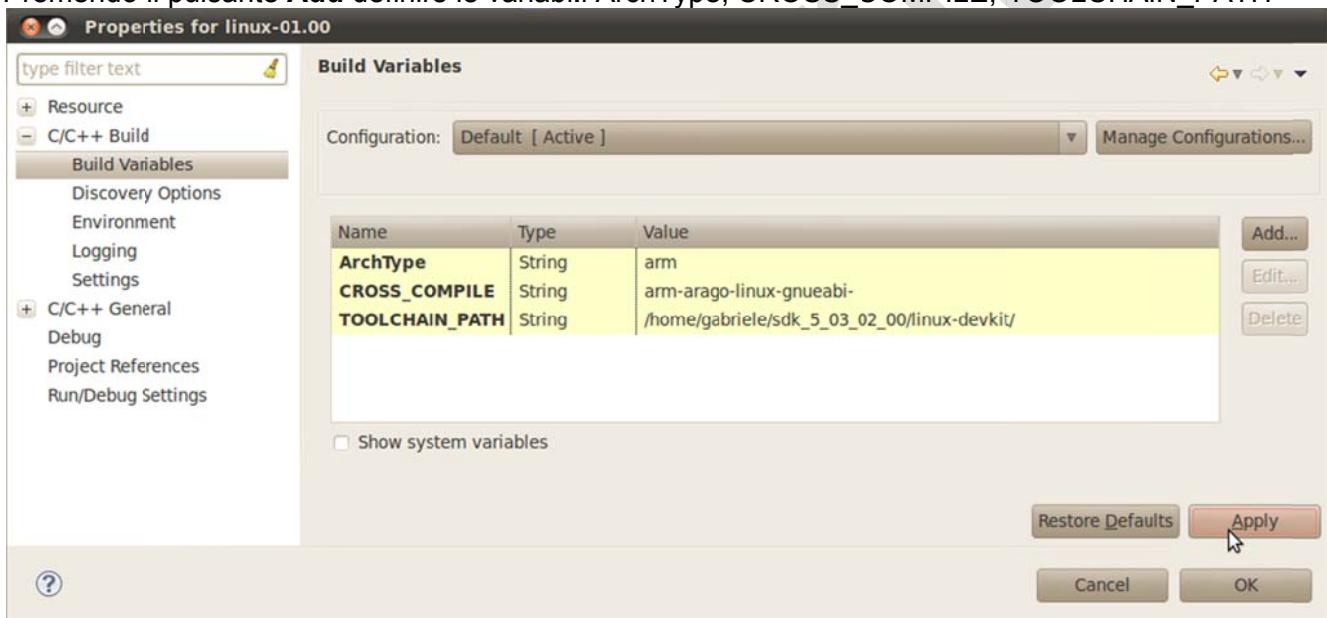
Engineering Specification - confidential



Premere **Finish**.

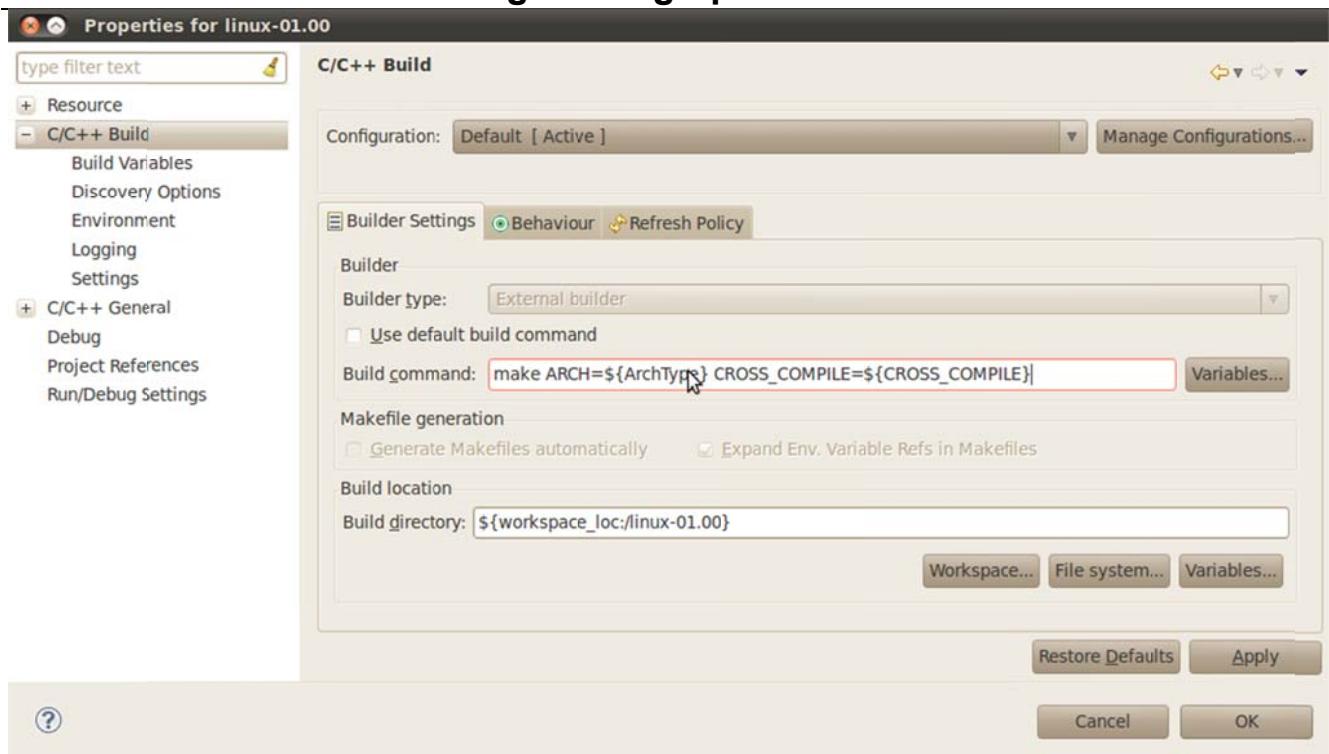
Puntando il mouse sul progetto, **BuildOptions->C/C++Build->BuildVariables**

Premendo il pulsante **Add** definire le variabili ArchType, CROSS_COMPILE, TOOLCHAIN_PATH

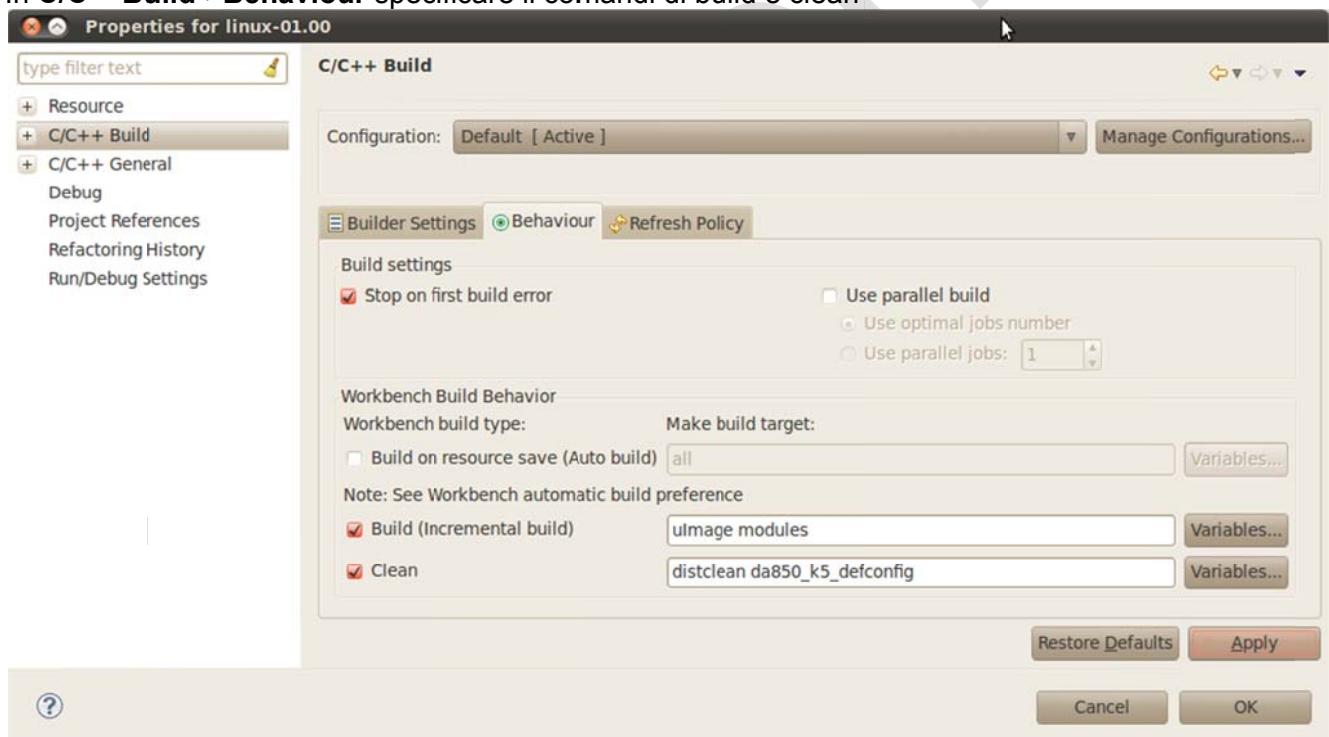


In **C/C++Build**, Tab **Configuration**, specificare il Build Command

Engineering Specification - confidential

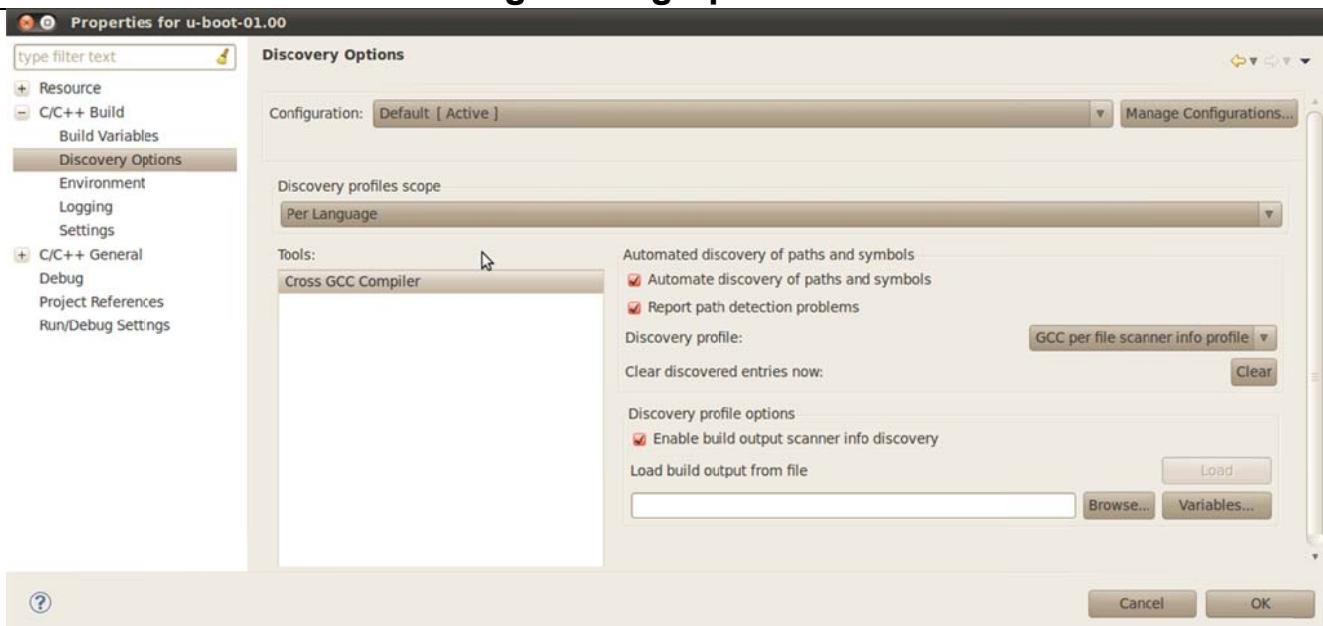


In C/C++Build->Behaviour specificare il comandi di build e clean

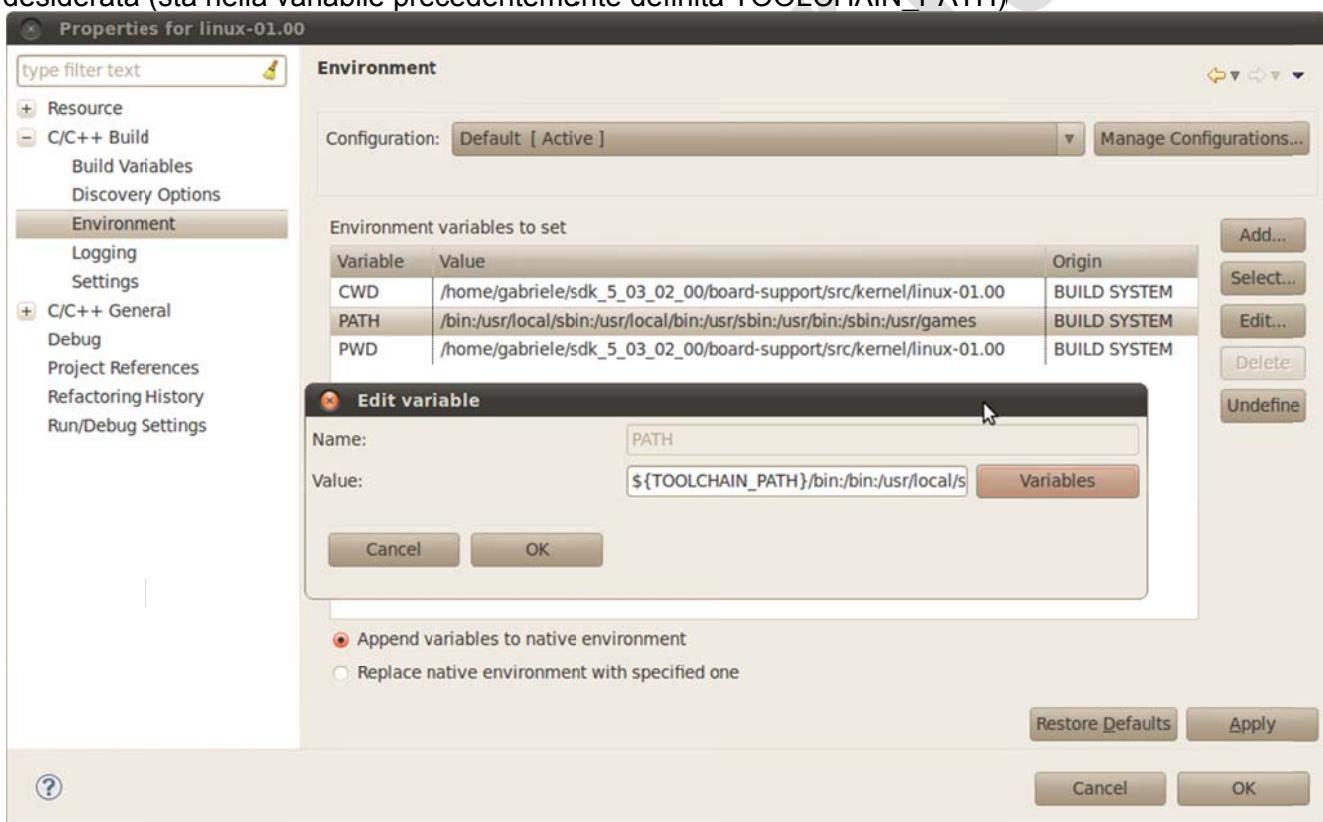


Verificare le seguenti opzioni di Discovery

Engineering Specification - confidential

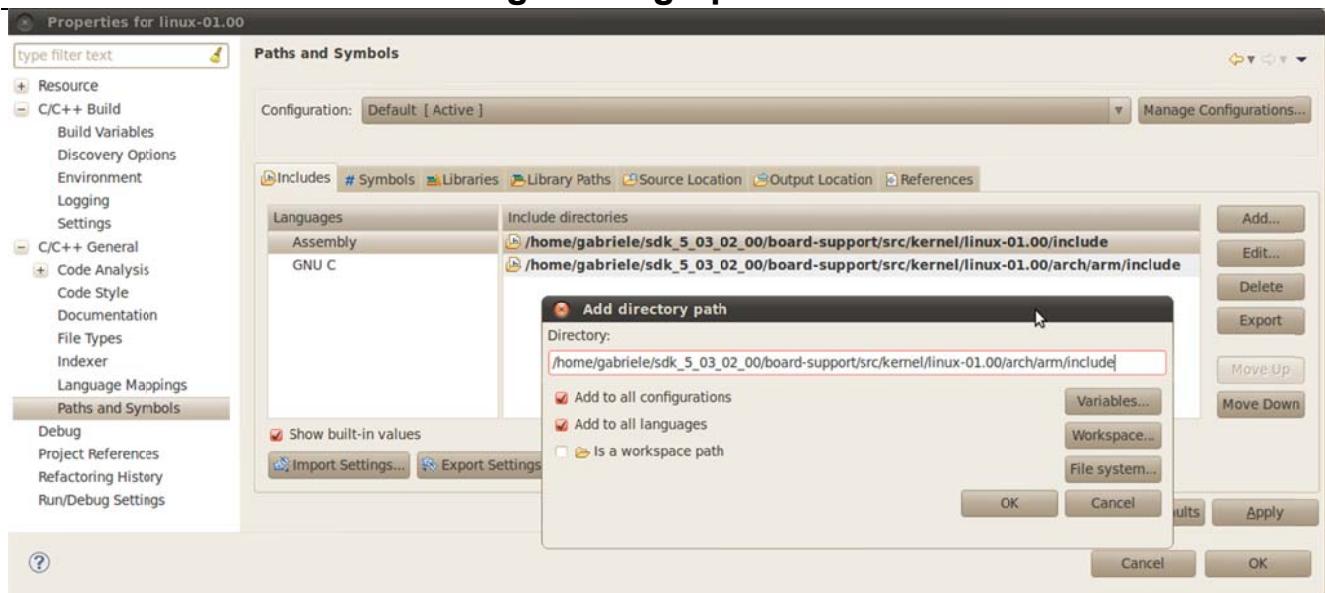


In **C/C++Build->Environment** editare la variabile PATH e appendere il percorso della toolchain desiderata (sta nella variabile precedentemente definita TOOLCHAIN_PATH)



In **C/C++General->PathsAndSymbols** sotto **Includes** aggiungere i percorsi seguenti

Engineering Specification - confidential



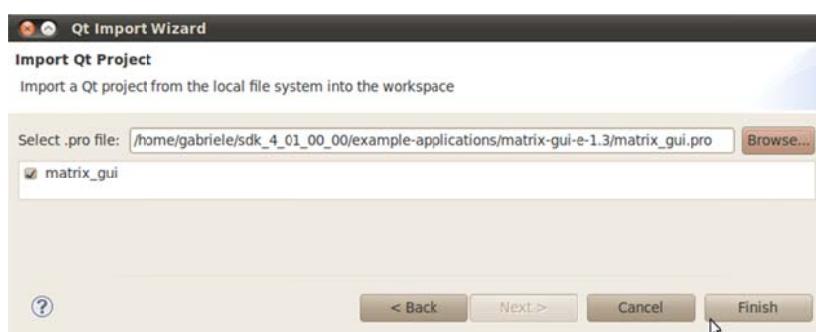
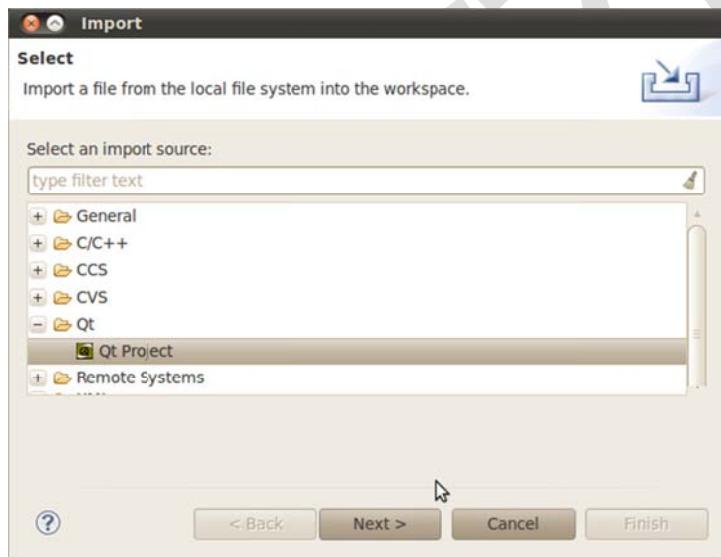
Ricompile con Project->Clean, Build

8.5 Importare e compilare progetto Qt matrix_gui (only SDK 5.02)

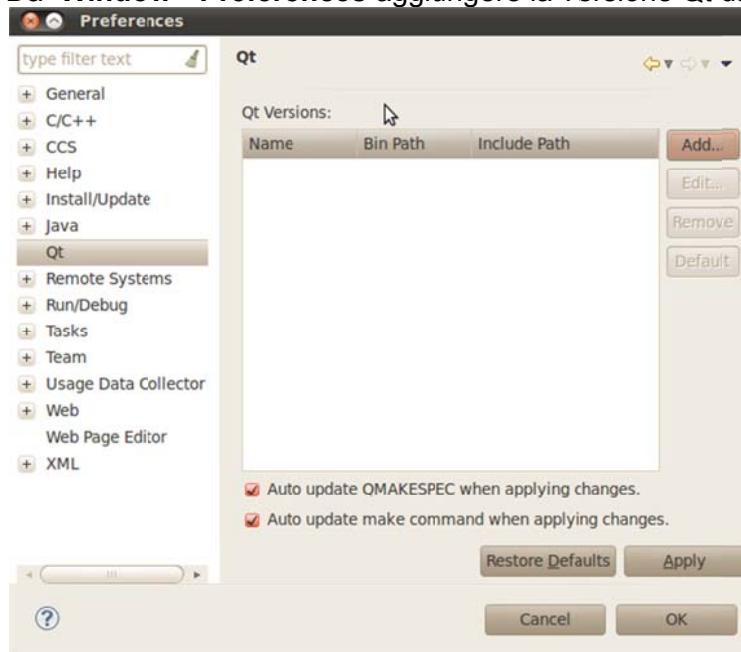
Si assume che SDKv5.02 sia installato

14) Importare il progetto demo matrix_gui

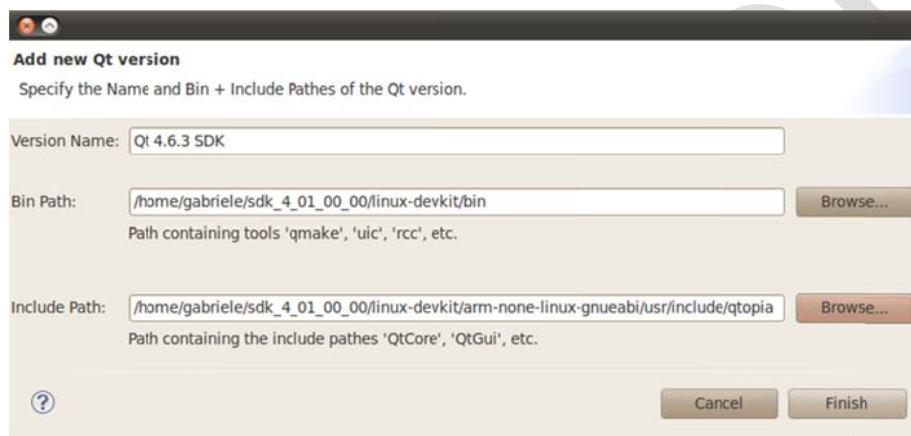
Da File->Import...



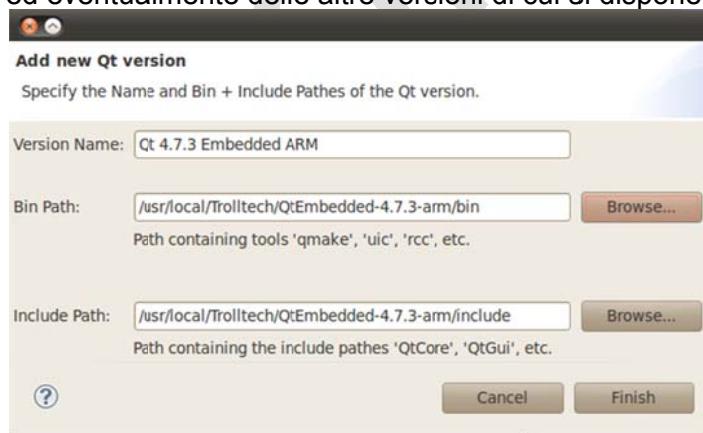
Da **Window->Preferences** aggiungere la versione Qt da associare al progetto (**Add**)



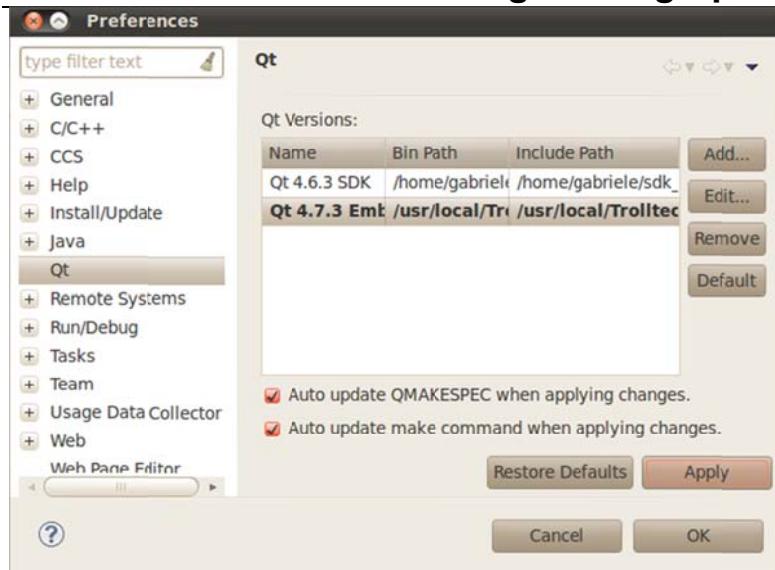
Aggiungere la versione Qt per ARM inclusa nell'SDK



ed eventualmente delle altre versioni di cui si dispone



Con il tasto **Default** si seleziona la versione attiva desiderata



15) Ricompilare

Si può fare o banalmente da CCS (**Project->Clean, Build**) oppure da terminale

host \$ cd <sdk>	
host \$ make clean	
host \$ make matrix_gui	(per la versione release)
host \$ make matrix_gui_debug	(per la versione debug)

16) Installare l'eseguibile sul target file system

Si può fare o da terminale

host \$ sudo make matrix_gui_install	(per la versione release)
host \$ sudo make matrix_gui_install_debug	(per la versione debug)

oppure da CCS utilizzando RSE. In tal caso è necessario rinominare matrix_gui in matrix_guiE e rendere eseguibile il file

8.6 Tips

In caso di problemi alla GUI, **Window->ResetPerspective...**

In caso di problemi vari, lanciare CCS con l'opzione –clean

In caso di problemi vari cancellare la cartella .metadata nel workspace

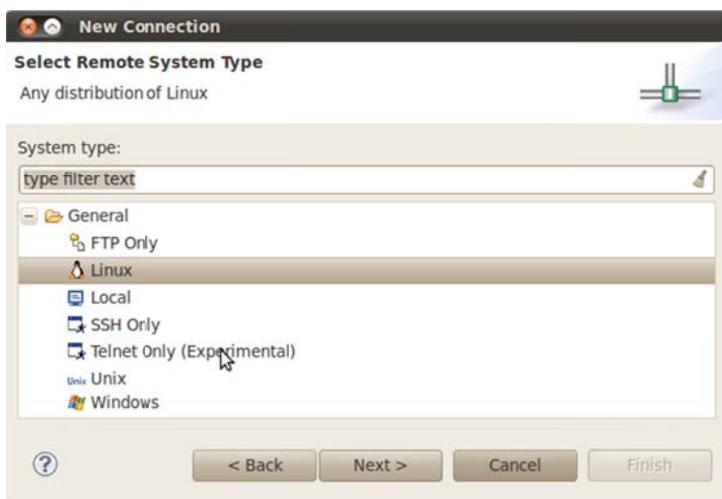
8.7 Configurare e utilizzare RSE

Il Remote System Explorer (RSE) è un plugin di CCS che permette di stabilire una connessione SSH tra host e target, per aprire un terminale sul target, killare processi, creare e cancellare files nel target filesystem, ecc.

http://processors.wiki.ti.com/index.php/How_to_setup_Remote_System_Explorer_plug-in

Da **File->New->Other..**

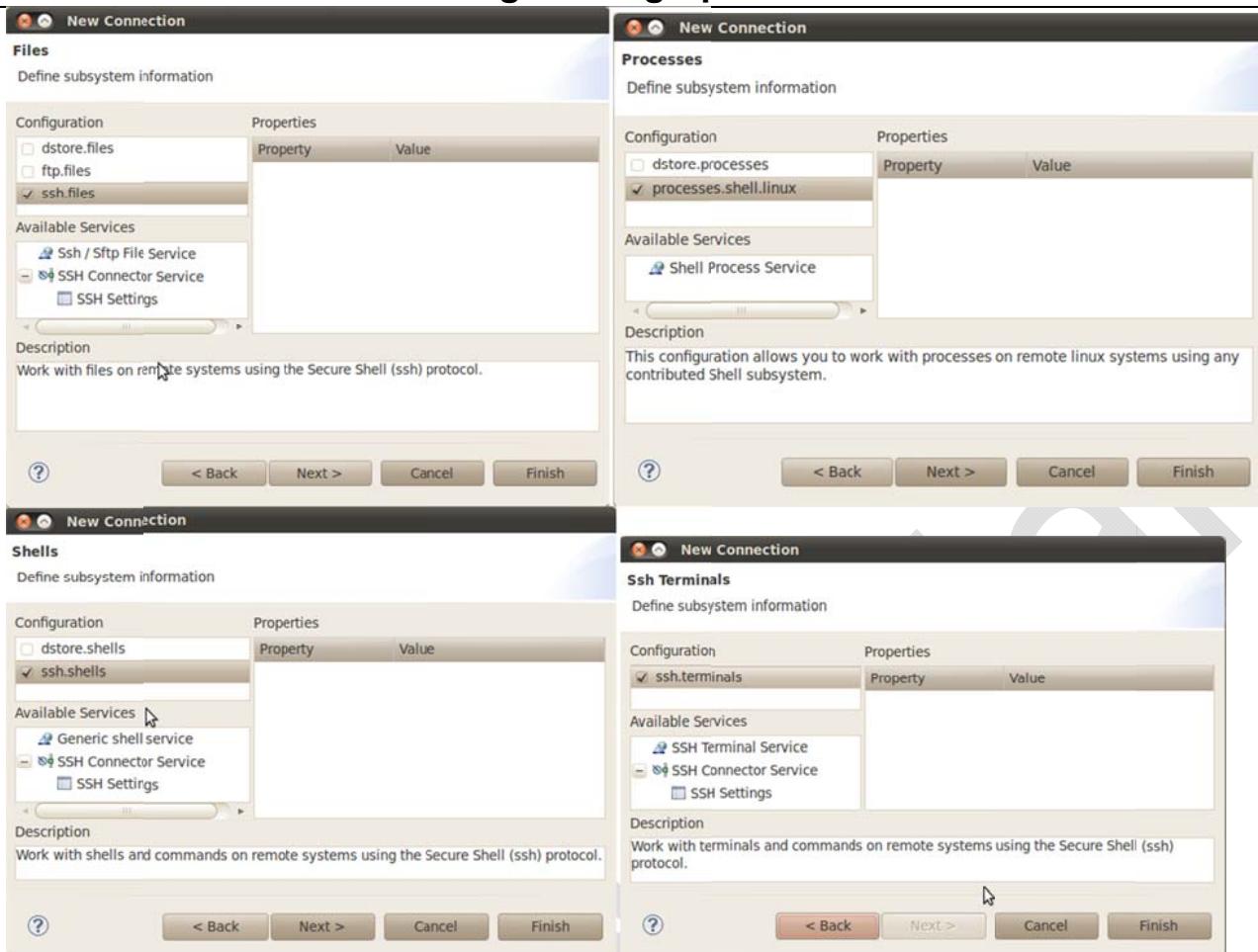
Engineering Specification - confidential



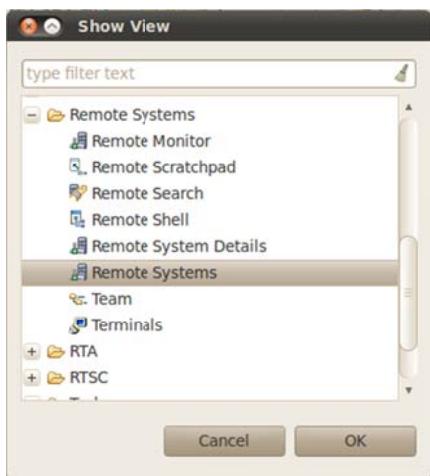
Inserire l'indirizzo IP del target <ipaddr>



Engineering Specification - confidential

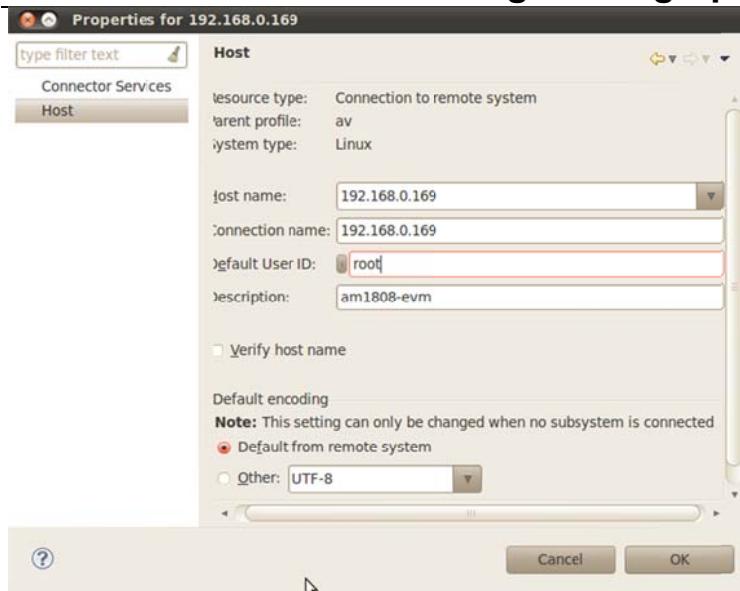


Da Window->Show View->Other..

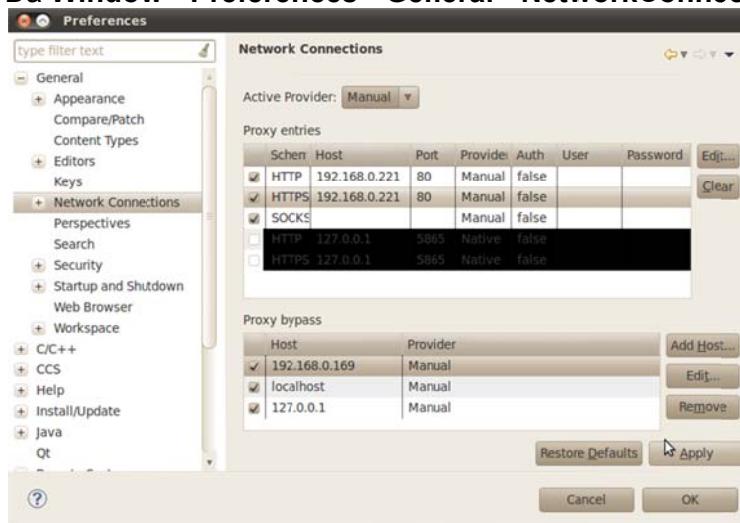


Right click sulla nuova connessione, Properties->Host

Engineering Specification - confidential



Da Window->Preferences->General->NetworkConnections



Dalla finestra Remote Systems espandere All Processes

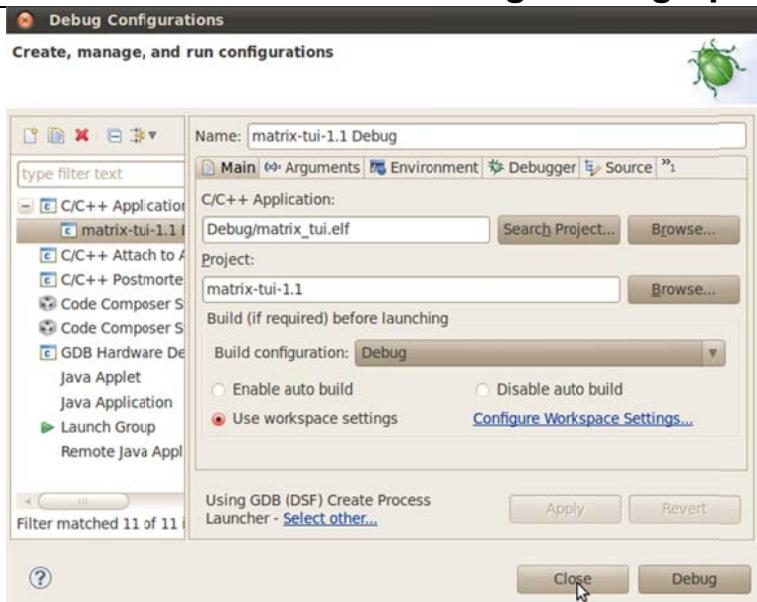


8.8 Configurare e utilizzare GDB server

Importare e rendere attivo il progetto da debuggere, p.es. matrix_tui
Assicurarsi che la **Build Configuration** attiva sia Debug (se necessario ricompilare la configurazione Debug).

Da **Run->Debug Configurations** doppio click su C/C++ Applications

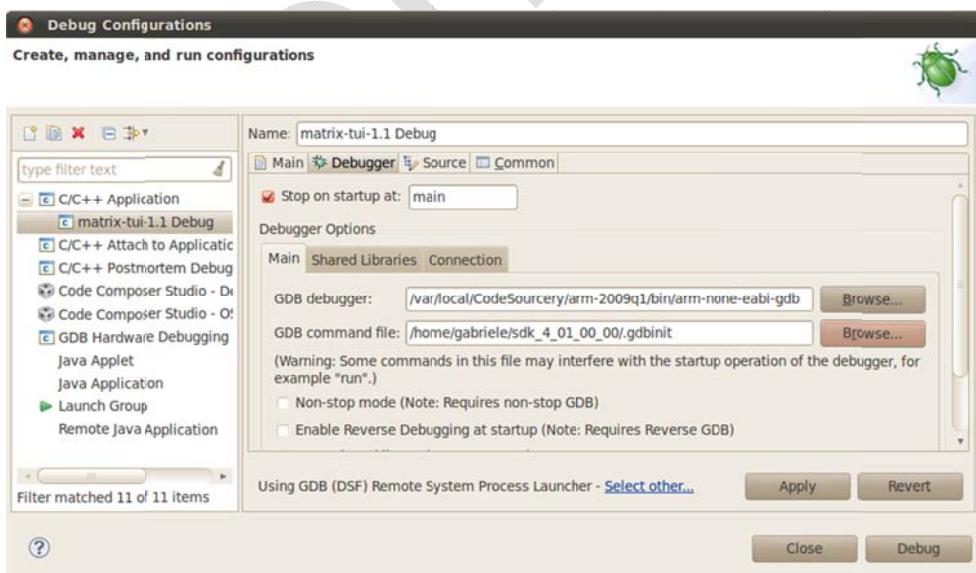
Engineering Specification - confidential



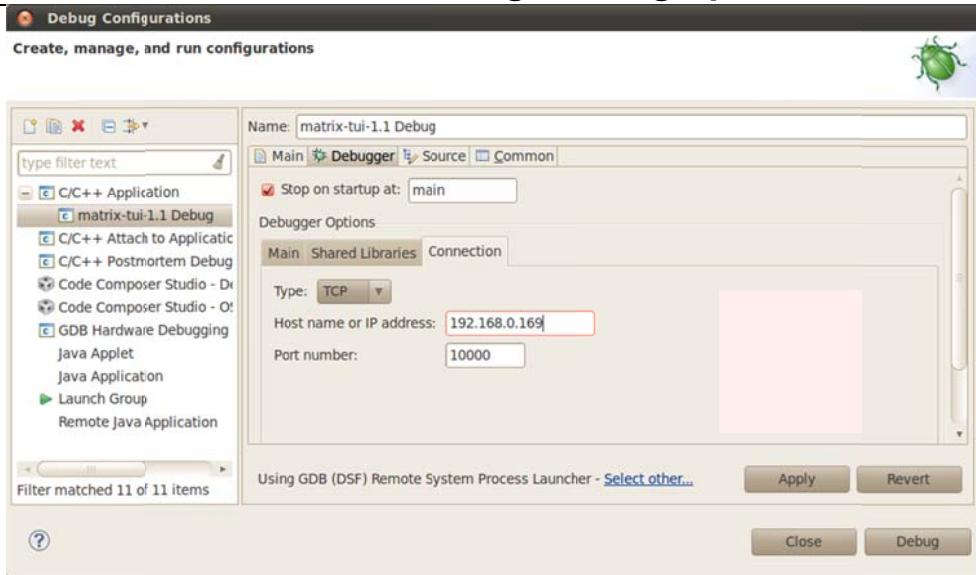
Click su **Select Other..** per selezionare il Launcher



Passare al Debugger tab e impostare l'eseguibile debugger della toolchain e il file .gdbinit del SDK



Nel Connection tab impostare il tipo di connessione (TCP) la porta (10000) e l' <ipaddr> del target



Copiare l'eseguibile gdbserver sul target utilizzando l'RSE oppure, se si è in NFS, con

```
host $ cp <sdk>/linux-devkit/arm-arago-linux-gnueabi/usr/bin/gdbserver <rootpath>/usr/bin/
```

e se necessario cambiare al file i permessi per poterlo eseguire sul target

```
target $ chmod 755 /usr/bin/gdbserver
```

Sul target, lanciare il gdbserver in ascolto sulla porta 10000

```
target $ gdbserver :10000 /usr/bin/matrix_tui
> Process matrix_tui created; pid = 1506
> Listening on port 10000
```

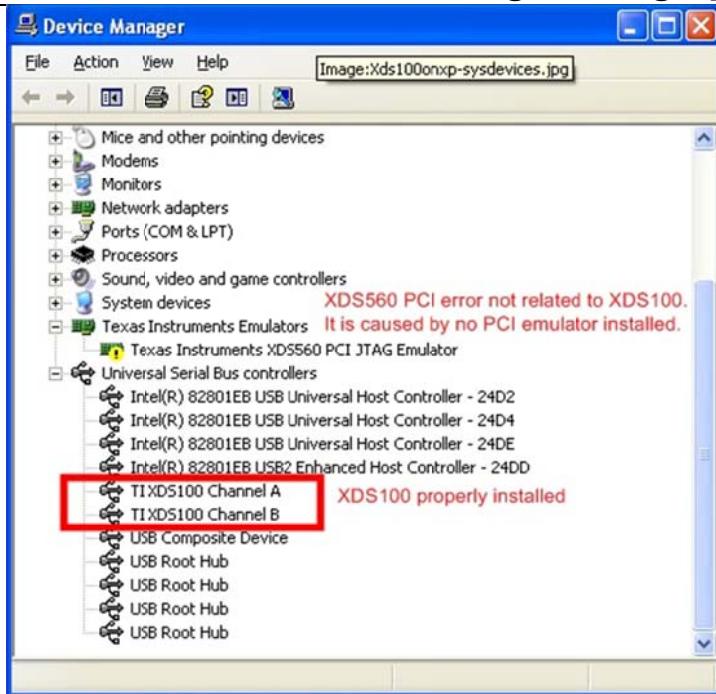
Lanciare il prospetto Debug (pulsante bagarozzo).

Nota: non cliccare **Run->Debug** altrimenti parte una nuova session di debug

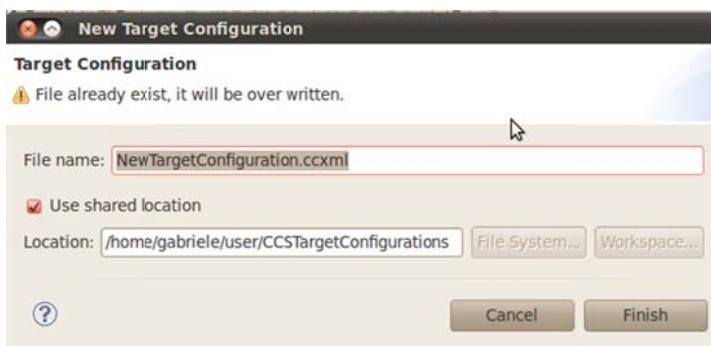
8.9 Collegamento al Target

- 1) Collegare l'emulatore XDS100v2 al PC e al target EVM

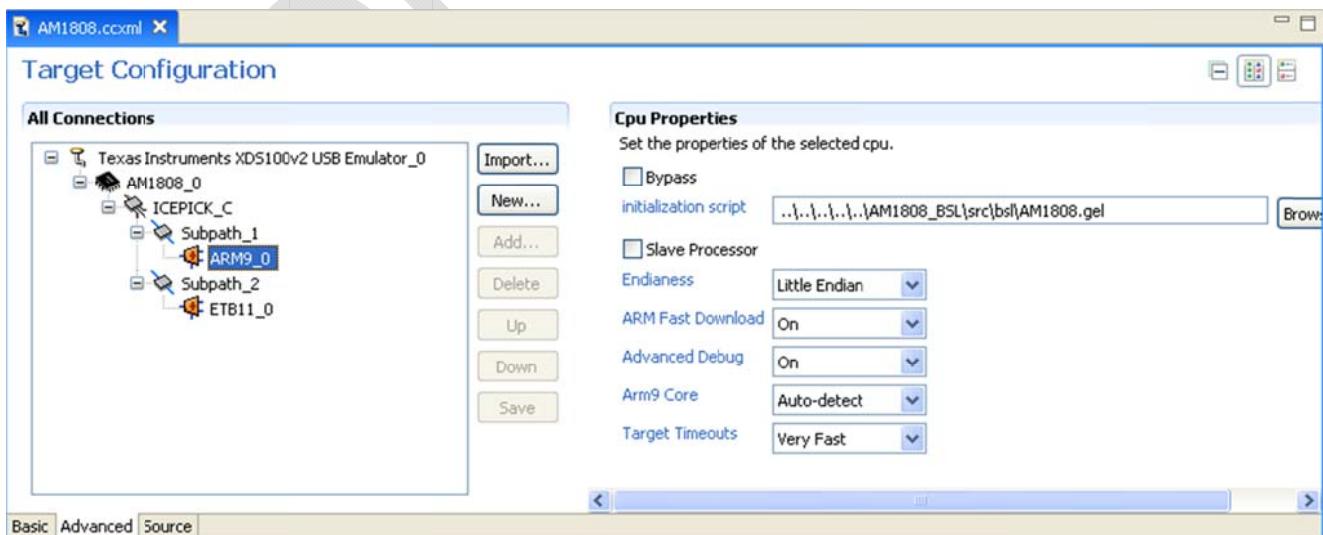
Engineering Specification - confidential

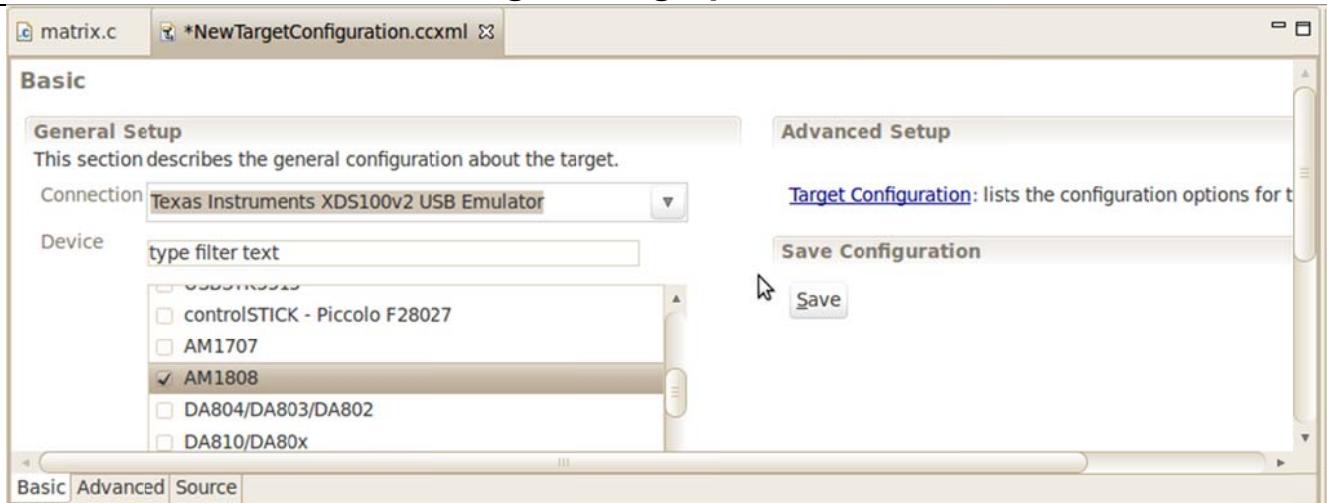


- 2) Lanciare CCS
- 3) Creare un nuovo *Target Configuration* basato su XDS100v2.



- 4) Avendo installato xxx, un Target Configuration già pronto è C:\AM1808_BSL\src\bsl\AM1808.ccxml





- 5) Predisporre il target in Emulation boot Mode (sulla EVM basta mettere a ON gli switch S7.5 e S7.8)

S7:8	S7:7	S7:6	S7:5
ON	OFF	OFF	ON

9 Team Foundation Server (TFS)

<http://blogs.microsoft.co.il/blogs/morgen/archive/2010/06/07/using-tfs-in-linux-ubuntu.aspx>

10 TFTP

Grazie ai TFTP files possono essere trasferiti dall'host al target attraverso la connessione di rete. Questo capitolo spiega come installare il TFTP server sull'host e come utilizzarlo.

10.1 Installazione e configurazione

http://processors.wiki.ti.com/index.php/Setting_Up_a_TFTP_Server#Using_Ubuntu

- 6) Da Synaptic, o da shell, installare tftpd-hpa

```
host $ cd /home/<user>
host $ sudo apt-get install tftpd-hpa
```

- 7) Verificare il file di configurazione, ed eventualmente cambiare qualcosa

```
host $ sudo gedit /etc/default/tftpd-hpa
```

```
TFTP_USERNAME="tftp"
TFTP_DIRECTORY="/var/lib/tftpboot"
```

*TFTP_ADDRESS="0.0.0.0:69"
TFTP_OPTIONS="--secure"*

- 8) Verificare che la directory tftpboot sia al suo posto, altrimenti crearla. Rendere la directory totalmente accessibile

```
host $ chmod -R 777 /var/lib/tftpboot (<tftpbootdir>)
```

- 9) Copiare l'immagine del kernel desiderata in tftpboot

```
host $ cp <sdkpsp>/prebuilt-images/uImage-am180x-evm.bin <tftpbootdir>/kx/uImage
```

- 10) Lanciare server

```
host $ sudo /etc/init.d/tftpd-hpa start
```

Successivamente, questa operazione verrà fatta automaticamente all'avvio dell'host

- 11) Da terminale seriale arrestare il boot del target durante il countdown. Dalla shell U-Boot verificare con printenv la correttezza dell'indirizzo IP dell'host (<serverip>) e bootfile=uImage. Invocare tftp boot

```
target # tftpboot  
Using device  
TFTP from server 192.168.0.70; our IP address is 192.168.0.53  
Filename 'uImage'.  
Load address: 0xc0700000  
Loading: #####  
#####  
#####  
done  
Bytes transferred = 2113752 (2040d8 hex)
```

- 12) Verificare correttezza dell'immagine in RAM

```
target # iminfo  
Checking Image at c0700000 ...  
Legacy image found  
Image Name: Linux-2.6.33-rc4  
Image Type: ARM Linux Kernel Image (uncompressed)  
Data Size: 2113688 Bytes = 2 MB  
Load Address: c0008000  
Entry Point: c0008000  
Verifying Checksum ... OK
```

- 13) Conviene strutturare internamente <tftpbootdir> per prodotto/progetto

```
host $ mkdir -p <tftpbootdir>/<target>
```

- 14) ATTENZIONE: Su Ubuntu 12.04 il file di configurazione TFTP è

/etc/xinetd.d/tftp

11 NFS

NFS (Network File System) è il protocollo nativo Linux per condividere directory tra computer diversi connessi in rete. Grazie al NFS il processore sul target monta il suo root file system su una directory fisicamente presente e modificabile sull'host.

Questo capitolo spiega come installare ed esportare il NFS sull'host e come utilizzarlo.

11.1 Installazione e configurazione

Perché il target possa montare il file system, questo deve essere prima "esportato" sull'host.

- 1) Da Synaptic, o da shell, installare nfs

```
host $ sudo apt-get install nfs-kernel-server
```

- 2) Se non ancora fatto, creare le directory da esportare <rootfs> (p.es. /home/<user>/workdir/kx/rootfs)
- 3) Dentro /var/lib creare un link simbolico *nfs-kx-rootfs* che punta a <rootfs>
- 4) Editare /etc/exports per aggiungere la directory da esportare

```
host $ sudo gedit /etc/exports
```

```
/var/lib/nfs-kx-rootfs *(rw, async, no_root_squash, no_subtree_check, nohide, insecure)  
/home/<user>/workdir/evm/filesys *(rw, async, no_root_squash, no_subtree_check, nohide, insecure)
```

- 5) Settare la variabile di ambiente u-boot rootpath nel seguente modo

```
rootpath="/var/lib/nfs-kx-rootfs"
```

- 6) Lanciare il server

```
host $ sudo /etc/init.d/nfs-kernel-server start
```

Successivamente, questa operazione verrà fatta automaticamente all'avvio dell'host

- 7) Verificare il funzionamento seguendo il paragrafo relativo al boot da NFS

12 Gestire le patches

Per poter tenere traccia delle modifiche da apportare al kernel è consigliabile utilizzare il sistema di gestione delle patch. In alternativa si può usare il metodo classico basato su *diff* e *patch*

12.1 Metodo classico

```
host $ diff -Nru orgdir newdir > mypatch
```

- 1) Assicurarsi di avere installato patch

```
host $ sudo apt-get install patch
```

- 2) Supponiamo di avere nella directory corrente una cartella di file orgdir/ e una cartella newdir/, versione modificata e rinominata di /orgdir. Per creare la patch che contempla le differenze,

```
host $ diff -Nru orgdir newdir > mypatch
```

- 3) Supponiamo adesso di avere la cartella orgdir/ e la patch mypatch che la deve trasformare in newdir/

```
host $ cd orgdir/  
host $ patch -p1 < ../mypatch
```

Adesso il contenuto di orgdir/ è lo stesso di newdir/

Per **rimuovere** la patch applicata si esegue lo stesso comando usato per applicarla aggiungendo l'opzione **-R**:

```
host $ cd orgdir/  
host $ patch -R -p1 < ../mypatch
```

Engineering Specification - confidential

Per quanto riguarda il nome della patch, una buona convenzione è usare un progressivo a 4 cifre Per quanto riguarda il nome della patch, usiamo un progressivo a 4 cifre seguito da un nome esplicativo dello scopo della patch, e l'estensione finale .patch.

La parte esplicativa inizia specificando il prodotto (p.es. kx), poi il campo di applicazione (p.es. uboot), ecc.

12.2 quilt

quilt è un programma che permette di gestire le patches in maniera più semplice e automatizzata rispetto al tradizionale metodo con *diff* e *patch*.

Questo paragrafo spiega come installare e utilizzare *quilt*.

4) Installazione

```
host $ cd /home/<user>/  
host $ sudo apt-get install quilt
```

5) Usando la variabile di ambiente QUILT_PATCHES si può definire la cartella dove andranno memorizzate le patches. Ad esempio

```
host $ export QUILT_PATCHES=/home/<user>/sdk_x_xx_xx_xx/patches
```

Tuttavia, siccome preferiamo avere cartelle separate per bootloader, kernel, filesystem, ecc. non seguiamo questo metodo. Invece creiamo una cartella patches all'interno di ciascuna delle directory principali interessate a modifiche. In tal modo i comandi quilt si riferiranno automaticamente alla prima cartella patches che si incontra risalendo la path della directory corrente. Quindi, creiamo le seguenti repository

per u-boot:

```
host $ cd <uboot>  
host $ mkdir -p .../patches
```

(<uboot-qpatches>)

per il root file system (assumendo che sia in /home/<user>/workdir/kx/rootfs):

```
host $ cd <rootfs>  
host $ mkdir -p .../patches
```

(<rootfs-qpatches>)

per il kernel la cosa è diversa in quanto il sorgente fornito consdk ha già una sua dir patches interna:

```
host $ cd <linux>  
host $ mkdir -p /patches
```

(<linux-qpatches>)

6) Creare una patch. Il nome della patch è un progressivo a 4 cifre seguito da un nome esplicativo dello scopo della patch, e l'estensione finale .patch. Ad esempio

```
host $ cd <uboot>  
host $ quilt new 0001-kx-uboot-shell-prompt.patch
```

Questo creerà la patch nella cartella <uboot-qpatches>

7) Aggiungere a questa patch il primo file che dovrà essere soggetto a modifiche (o creato ex novo)

```
host $ quilt add include/configs/da850evm.h
```

Aggiungere, se necessario altri files.

8) Applicare le modifiche al (ai) file

```
host $ gedit include/configs/da850evm.h
```

modificare la costante CONFIG_SYS_PROMPT come segue

```
#define CONFIG_SYS_PROMPT "KX > /* Command Prompt */
```

9) Registrare le modifiche nella patch

```
host $ quilt refresh
```

10) Verificare

```
host $ cat .../patches/0001-kx-uboot-shell-prompt.patch
```

11) Per tornare al punto di partenza basta togliere tutte le patch (per ora ne abbiamo solo una)

```
host $ quilt pop -a  
senza l'opzione -a viene rimossa solamente l'ultima patch applicata
```

- 12) Per applicare nuovamente tutte le patch

```
host $ quilt push -a  
senza l'opzione -a viene applicata solamente l'ultima patch rimossa  
Talvolta occorre forzare con l'opzione -f.
```

Altre patch possono essere create (new), applicate (push) e rimosse (pop), come su uno stack.

- 13) Per applicare una patch ottenuta da fonti esterne (sia nuova_patch), rinominandola ad esempio mypatch

```
host $ quilt import nuova_patch -P mypatch  
host $ quilt push mypatch
```

- 14) Per togliere un file filename da mypatch

```
host $ quilt delete filename  
host $ quilt pop  
host $ quilt push
```

- 15) Per rinominare una patch

```
host $ quilt rename -P mypatch newpatchname
```

13 UBL

UBL è il bootloader di primo livello.

Al power-up l'ARM del AM1808/OMAPL-138 esegue UBL (in formato AIS file). UBL inizializza il PLL e la RAM esterna, copia U-Boot nella RAM e lo esegue.

Il KX utilizza l'UBL incluso nel PSP. UBL richiede CCS per essere ricompilato.

Questo capitolo spiega dove reperire i sorgenti, come modificare, ricompilare e utilizzare UBL.

14 TI DVSDK su host

Il DVSDK è un insieme di pacchetti SW già testati a corredo di un target di valutazione DaVinci (EVM). A differenza del SDK, il target processor deve avere un core dedicato all'elaborazione del video.

Nel nostro caso il DVSDK è relativo al processore OMAP-L138.

Usare sempre la versione più recente del DVSDK.

Si assume di avere Ubuntu 10.04, e di avere già installato la toolchain per ARM CodeSourcery arm-2009q1.

14.1 Installazione e configurazione DVSDK 4.02

http://software-dl.ti.com/dsps/dsps_public_sw/sdo_sb/targetcontent/dv sdk/DVSDK_4_00/latest/index_FDS.html

Build date: 03252011

Engineering Specification - confidential

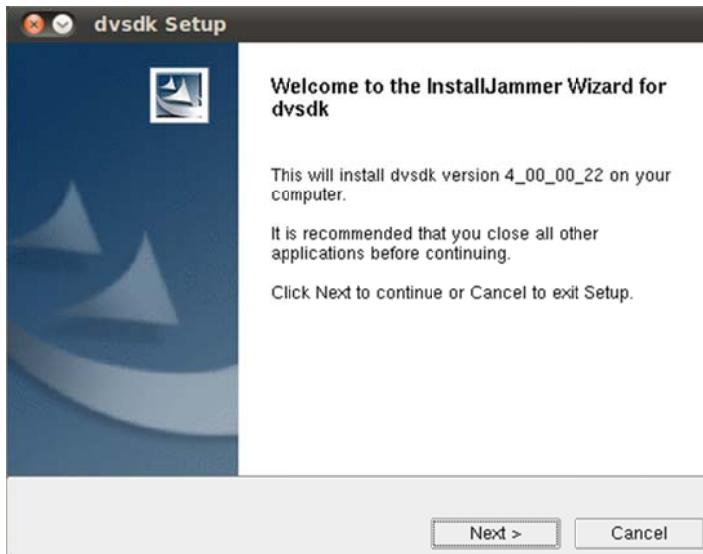
```
host $ sudo dpkg-reconfigure dash
```

alla domanda rispondere No.

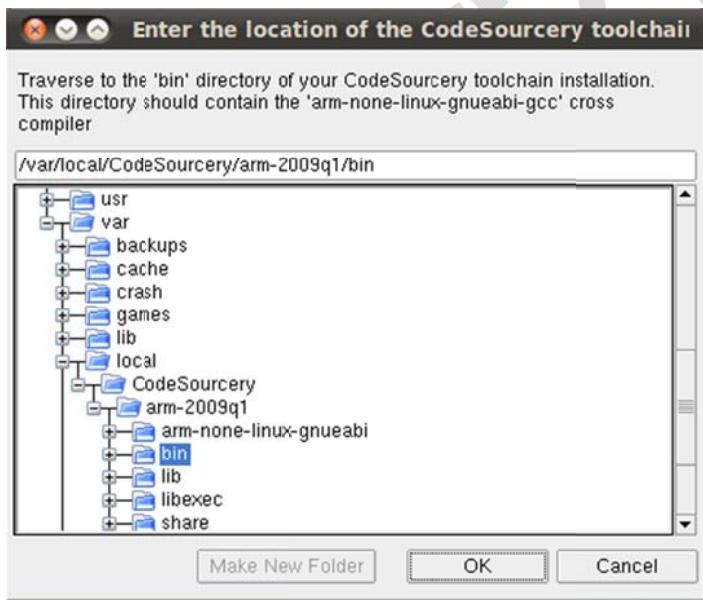
```
host $ sudo apt-get install fakeroot
```

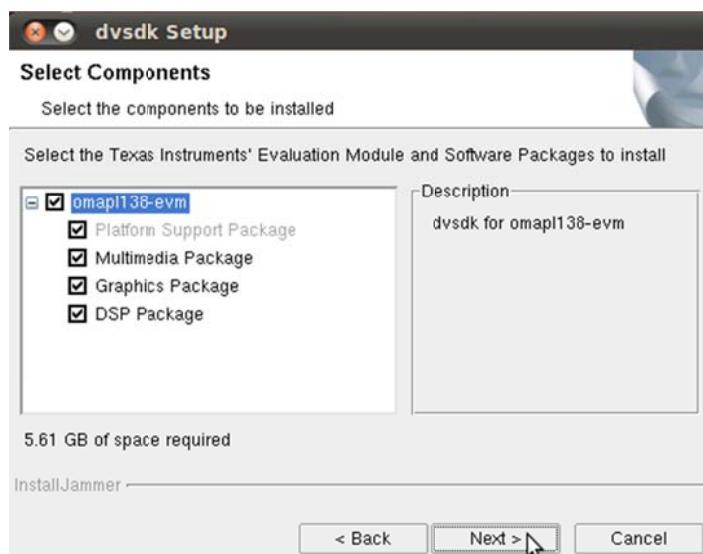
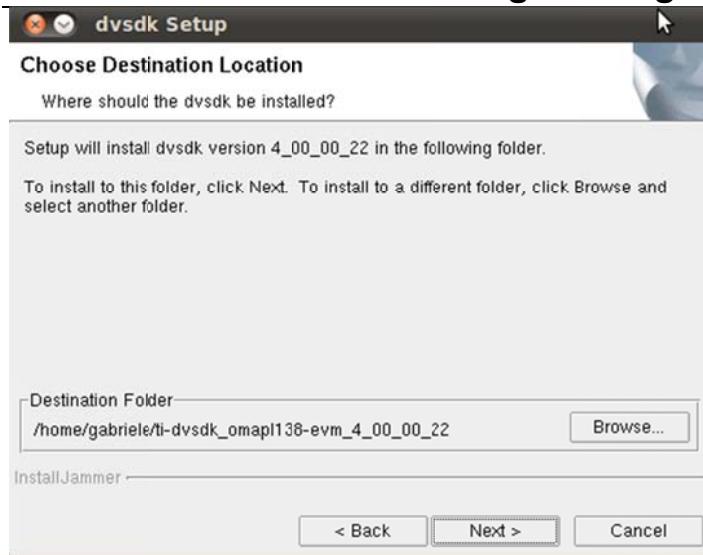
- 17) Dopo aver scaricato il pacchetto di installazione specifico dell'OMAPL138 in /home/<user>, lanciare l'installazione

```
host $ ./dvsdk omapl138-evm_4_00_00_22_SetupLinux
```



Quando richiesto, specificare la path della toolchain CodeSourcery





Proseguire fino al completamento dell'installazione

- 18) Editare .bashrc e aggiungere la riga per esportare la path dove si trova il DVSDK

```
export DVSDK="/home/<user>/ti-dvdk OMAPL138-evm_4_00_00_22"
```

Continuare seguendo il documento
OMAPL138 Software Developers Guide

15 Kernel

Il kernel è lo strato che fornisce per le risorse HW il livello di astrazione richiesto da applicazioni e librerie. Interagisce col root filesystem.

Il KX utilizza il kernel incluso nel PSP TI, a sua volta derivato dal GIT tree.

[L'SDK 5.05 include il PSP 03.21.00.04, che è basato sul kernel v2.6.37.](#)

L'SDK 5.07 include il PSP 03.21.00.04, che è basato sul kernel v2.6.37.

15.1 Re-build del kernel

http://processors.wiki.ti.com/index.php/GSG:_Building_Software_Components_for_OMAP-L1#Rebuilding_the_Linux_kernel

Si assume che la Toolchain sia installata e che u-boot sia stato già ricompilato.

- 19) Editare .bashrc e aggiungere la riga per esportare la path dove si trova l'utility mkimage

```
export PATH= <uboot>/tools:$PATH
```

- 20) Entrare nella cartella dei sogenti

```
host $ cd <linux>
```

- 21) Rebuild della configurazione di default

```
host $ make ARCH=arm CROSS_COMPILE=<armtoolch>- distclean  
host $ make ARCH=arm CROSS_COMPILE=<armtoolch>- da850_omap138_defconfig  
host $ make ARCH=arm CROSS_COMPILE=<armtoolch>- ulimage
```

Il file generato è arch/arm/boot/ulimage

ulimage è la versione compressa di <linux>/vmlinux (formato ELF), con l'header di u-boot

Nota. clean cancella solo i file oggetto, distclean cancella anche .config (la configurazione)

Nota. Non specificando ulimage la sola immagine che viene creata è <linux>/vmlinux.

Nota. Se si vuole vedere TUTTI gli step di build l'output, usare V=1

```
host $ make ARCH=arm CROSS_COMPILE=<armtoolch>- ulimage V=1
```

Nota. Se l'host è un multi-core con N core, è possibile velocizzare la compilazione del kernel usando l'opzione -j#, dove # è 2N. Ad esempio, su un dual-core

```
host $ make ARCH=arm CROSS_COMPILE=<armtoolch>- ulimage -j4
```

- 22) Copiare l'immagine del kernel nella cartella predefinita per il trasferimento TFTP

```
host $ cp arch/arm/boot/ulimage <tftpbootdir>/<target>
```

15.2 Configurare il kernel

- 23) Per modificare le opzioni di configurazione del kernel con l'interfaccia a menu

```
host $ make ARCH=arm CROSS_COMPILE=<armtoolch>- menuconfig
```

Questo comando produce il file di configurazione < linux>/.config

E il corrispondente include file <linux>/include/linux/autoconf.h

Nota. Se si ottiene l'errore "Unable to find the ncurses libraries" occorre installarle.

Da Administration->**Synaptic PackageManager**, Mark for install

libncurses5

libncurses5-dbg

libncurses5-dev

libncursesw5

Nota. Vi sono molti altri comandi per configurare il kernel, tra cui

config Update current config using a line-oriented program

xconfig Update current config using a QT-based front end

gconfig Update current config using a GTK-based front end

oldconfig Update current config using a provided .config as the base

defconfig New config with default answer to all options

allmodconfig New config that selects modules, when possible

15.3 Compilazione e installazione dei moduli

- 24) Per compilare le features configurate come moduli (M)

```
host $ make ARCH=arm CROSS_COMPILE=<armtoolch>- modules
```

- 25) Per installare i moduli nel target root file system
si assume che un'immagine del target file system sia sull'host

```
host $ make ARCH=arm INSTALL_MOD_PATH=<rootpath> modules_install
```

dove <rootpath> è la path del target root file system sull'host

15.4 Registrazione del numero di macchina board kx

<http://www.arm.linux.org.uk/developer/kernelnotes.php>

- 1) Registrarsi su <http://www.arm.linux.org.uk>

Welcome, Gabriele

A web request from [94.94.20.200] was received to register your email address (gabrielef@X.it) on the ARM Linux Web Site. This request has been successfully processed, and your account has been created. Your password for logging into the site is **4edb3c**

- 2) Richiedere un numero che identificherà la board.

La lista aggiornata delle boards registrate è in

<http://www.arm.linux.org.uk/developer/machines/download.php>

<http://www.arm.linux.org.uk/developer/machines/>

che poi viene inserito nel kernel come file

<linux>/arch/arm/tools/mach-types

Per richiedere il numero

<http://www.arm.linux.org.uk/developer/machines/?action=new>

Engineering Specification - confidential

Machine name: This is the long (English) name for this machine type.

Cosmed K5 Mobile CPET

Machine type: Please give the type of the machine. This must:

- be unique in the machine database
- be a single word
- consist of characters acceptable as a C language identifier
- not be the name or part number of a CPU or SoC
- be no longer than 18 characters

It will be converted to lower case, and used to generate the configuration and machine type variables.

DA850_K5

Directory suffix: This suffix for the arch/arm/mach- directory.

- This may be the name or part number of a CPU or SoC
- This should consist of lower case characters
- This must only contain 7 bit ASCII alphanumeric characters

davinci

Web site: If you have a website for this machine, please give its address here. Please give a complete URL including the protocol to find more information on this machine or product.

http://www.cosmed.it/index.php?option=com_content&view=article&id=249&Itemid=174&lang=en

It is a portable, breath by breath metabolic cart for professional Cardio Pulmonary Exercise Testing and Spirometry

Queste impostazioni determinano le seguenti funzioni e macro di sistema

*machine_is_da850_kx()
CONFIG_DA850_KX
MACH_DA850_KX
BOARD DIRECTORY: /arch/arm/mach-davinci*

3) Definire nuovo numero di macchina

Sostituire <http://www.arm.linux.org.uk/developer/machines/download.php>
a <linux>/arch/arm/tools/mach-types

15.5 Creazione supporto kernel per board kx

Creare il BSP per la board kx significa partire da una copia del kernel sorgente fornito nel SDK (pulito con distclean), assegnando la versione xx.xx (inizialmente 01.00)

```
host $ cd <sdkpsp>/src/kx/kernel  
host $ cp -R linux-##.##.##.## linux-xx.xx/
```

(<linux>)

e applicarvi una sequenza di patches preventivamente create. Le patch vengono create e applicate con quilt. Se nella cartella contenente <linux> c'è già una sottocartella patches/ allora questa sarà anche la stessa cartella per le patch nuove, altrimenti la si deve creare allo stesso livello di <linux>. Ogni patch deve includere la modifica della macro EXTRAVERSION presente in <linux>/Makefile. Le patches vengono periodicamente compresse

```
host $ cd <linux>  
host $ tar czf ..../linux-qpatches-xx.xx-ddmmyyyy.tar.gz patches/
```

e archiviate in

/Serverfirmware/X/common/kx/kernel/vxx/

Quando si vuole riapplicare le patches al kernel

```
host $ cd <sdkpsp>/src/kx/kernel  
host $ cp -R linux-##.##.##.## linux-xx.xx/  
host $ cd <linux>  
host $ quilt pop -a  
host $ rm -Rf patches/ .pc  
host $ tar zxf ..../linux-qpatches-xx.xx-ddmmyyyy.tar.gz  
host $ quilt push -a
```

(si rimuovono tutte le patches preesistenti)

(si applicano tutte le patches desiderate)

Opzionale: sottoporre le patch a davinci-linux-open-source@linux.davincidsp.com in modo che vengano inserite nel kernel. Dal momento della richiesta inizia un intenso scambio di e-mail con i responsabili di turno finalizzato a confezionare le patch nel modo giusto. Come riferimento consultare nell'archivio della mailing-list i casi delle board enbw-cmc, mityomapl138, omapl138-hawk

15.6 Patch 0009-kx-kernel-basic

Questa patch crea l'infrastruttura di base, ma non introduce modifiche HW-dipendenti dalla scheda KXDB.

Pur essendo la prima, a questa patch assegnamo numero 9 in quanto già esistono 8 patches nel kernel fornito con l'SDK. Le patches per kx si accodano alle patch preesistenti

Patch Name:

0009-kx-kernel-basic.patch

Patched Files:

Makefile
arch/arm/ftdi/da850_kx_defconfig
arch/arm/configs/da850_omapl138_defconfig
arch/arm/mach-davinci/board-da850-kx.c
arch/arm/mach-davinci/include/mach/uncompress.h
arch/arm/mach-davinci/Kconfig
arch/arm/mach-davinci/Makefile
drivers/media/video/davinci/Kconfig
sound/soc/davinci/davinci-evm.c
sound/soc/davinci/Kconfig
sound/soc/davinci/Makefile

- 26) Usando la macro EXTRAVERSION presente nel Makefile modificare la versione del kernel appendendo –pppp-xx.xx (pppp è il numero di patch, xx.xx la versione del FW)

```
#define EXTRAVERSION "–0009-01.00"
```

- 27) Creare il BSP per la board

```
host $ cd <linux>  
host $ cp arch/arm/configs/da850_omapl138_defconfig arch/arm/configs/da850_kx_defconfig  
host $ cp arch/arm/mach-davinci/board-da850-evm.c arch/arm/mach-davinci/board-da850-kx.c
```

Nei due file appena creati e nei vari altri punti dove sono usati

davinci_da850_evm
CONFIG_DAVINCI_DA850_EVM
MACH_DAVINCI_DA850_EVM

Sostituire o abbinare in OR con

da850_kx
CONFIG_DA850_KX
MACH_DA850_KX

- 28) Rebuild della configurazione appena creata

```
host $ make ARCH=arm CROSS_COMPILE=<armtoollch>- distclean  
host $ make ARCH=arm CROSS_COMPILE=<armtoollch>- da850_kx_defconfig  
host $ make ARCH=arm CROSS_COMPILE=<armtoollch>- ulimage  
host $ make ARCH=arm CROSS_COMPILE=<armtoollch>- modules  
host $ make ARCH=arm INSTALL_MOD_PATH=<rootpath> modules_install
```

- 29) Il numero di macchina usato deve essere inserito anche nel bootloader. Eseguire quindi i passaggi descritti nel paragrafo **Aggiungere la board kx a u-boot**. Una volta ricompilato e trasferito in flash il bootloader, provare il reboot del sistema operativo.

Le prime righe del boot saranno:

```
Linux version 2.6.37 (gabriele@fub1004) (gcc version 4.3.3 (GCC) ) #1
PREEMPT Tue Dec 6 14:56:20 CET 2011
CPU: ARM926EJ-S [41069265] revision 5 (ARMv5TEJ), cr=00053177
CPU: VIVT data cache, VIVT instruction cache
Machine: X KX Mobile CPET
```

15.7 Patch 0010-kx-kernel-config

Questa patch introduce modifiche HW-dipendenti dalla scheda KXDB.

Tra le altre cose introduce il driver per SHT21 ed espande il numero dei banchi di GPIO supportati

Patch Name:

0010-kx-kernel-config.patch

Patched Files:

Makefile
arch/arm/mach-davinci/board-da850-kx.c
arch/arm/configs/da850_kx_defconfig
drivers/hwmon/sht21.c
drivers/hwmon/Kconfig
drivers/hwmon/Makefile
arch/arm/configs/da850.c
arch/arm/mach-davinci/include/mach/mux.h

Configurazione kernel:

Da menuconfig:

- 4) Disabilitare FileSystems->kernel automounter version 4 support
- 5) Disabilitare FileSystems->Ext3 support
- 6) Disabilitare FileSystems->MiscellaneousFileSystems->Minix File System support
- 7) Disabilitare NetworkingSupport->NetworkingOptions->IPV6 Protocol
- 8) Disabilitare SystemType->DaVinci Implementations->DaVinci McBSP
- 9) Disabilitare SystemType->DaVinci Implementations->Enable Mistral daughter board support
- 10) Disabilitare NetworkingSupport->Bluetooth subsystem support
- 11) Disabilitare DeviceDrivers->Serial ATA and Parallel ATA drivers
- 12) Disabilitare DeviceDrivers->InputDeviceSupport->Keyboards
- 13) Disabilitare DeviceDrivers->MMC/SD/SDIO (in quanto almeno inizialmente impedisce l'utilizzo di NAND e NOR)
- 14) Abilitare DeviceDrivers->MTD Support->NAND Device Support->DaVinci + generic
- 15) Abilitare DeviceDrivers->MTD Support->JFFS2 Support
- 16) Abilitare DeviceDrivers->MTD Support->UBIFS Support
- 17) Abilitare File System->JFFS2 Support
- 18) Abilitare File System->UBIFS Support
- 19) Disabilitare definizione partizioni MTD per flash NAND e NOR in quanto vengono passate da u-boot
- 20) Modificare definizione partizioni MTD per flash SPI (allineandole a quelle definite in u-boot)
- 21) Aggiornare EXTRAVERSION nel Makefile a “-0010-01.00”
- 22) Abilitare DeviceDrivers->USB Support->USB SerialConverterSupport->USB FTDI SinglePortSerialDriver

Consente di vedere le porte USB host che fanno capo al USB-To-QuadUART bridge (FT4232HQ)

- 23) Abilitare DeviceDrivers->USB Support->USB SerialConverterSupport->USB CP210x family of UART Bridge Controllers

Consente di vedere le una ANT key sulla porta USB host che fa capo al USB-Quad Hub (USB2514)
Non è strettamente necessario all'applicazione KX, ma potrebbe essere di ausilio.

- 24) Abilitare DeviceDrivers->USB Support->USB Modem (CDC ACM) support

Consente di vedere la porta USB host che fa capo al BLE112 (Bluetooth 4.0 module)

- 25) Abilitare DeviceDrivers->GPIO Support->/sys/class/gpio/...

- 26) Abilitare DeviceDrivers->LED Support->LED Class Support

- 27) Abilitare DeviceDrivers->LED Support->LED Trigger Support (all stuff)

- 28) Abilitare DeviceDrivers->Hardware Monitoring Support ->Sensirion humidity and temperature sensors. SHT21 and compat

- 29) In board-da850-kx.c adattare all'HW specifico della scheda KXDB

Verifiche:

- 30) Verifica funzionamento driver per SHT21

```
target $ cat /sys/class/hwmon/hwmon0/device/modalias  
i2c:sht21
```

15.8 Patch 0011-kx-add-dpot-lcd-support

Questa patch introduce modifiche HW-dipendenti dalla scheda KXDB.

In particolare introduce il supporto driver al Trimmer digitale e al display LCD TIANMA TM035HBHT1

Patch Name:

0011-kx-kernel-add-dpot-lcd-support.patch

Patched Files:

Makefile

arch/arm/configs/da850_kx_defconfig

arch/arm/mach-davinci/Makefile

arch/arm/mach-davinci/board-da850-kx.c

arch/arm/mach-davinci/da850.c

arch/arm/mach-davinci/devices-da8xx.c

arch/arm/mach-davinci/hx8238.c

arch/arm/mach-davinci/ili934x.c

arch/arm/mach-davinci/include/mach/da8xx.h

arch/arm/mach-davinci/include/mach/hx8238.h

arch/arm/mach-davinci/include/mach/ili934x.h

arch/arm/mach-davinci/include/mach/mux.h

drivers/spi/davinci_spi.c

drivers/video/da8xx-fb.c

Configurazione kernel:

Da menuconfig:

- 30) Abilitare DeviceDrivers->SPI Support->UserModeSPIdeviceDriverSupport

- 31) Abilitare DeviceDrivers->Misc Devices->AnalogDevicesDigitalPotentiometers->SupportSPIbusConnection

- 32) Abilitare DeviceDrivers->Graphics support->ConsoleDisplayDriverSupport->FramebufferConsoleSupport->FramebufferConsoleRotation

Engineering Specification - confidential

- 33) Definire la struttura del display TIANMA TM035HBHT1 nel driver da8xx-fb.c
34) Definire la struttura del display URT UMSH8065MD19T nel driver da8xx-fb.c
35) In board-da850-kx.c adattare all'HW specifico della scheda KXDB aggiungendo in da850_kx_spi_info[]

```
[3] = { /* SPI1.3 for Digital Potentiometer AD5231 */
    .modalias = "ad_dpot",
    .platform_data = "ad5231",
    .max_speed_hz = 3000000,
    .bus_num = 1,
    .chip_select = 3,
},
```

Verifiche:

- 31) Verifica funzionamento driver per potenziometro digitale ad_dpot

```
target $ cd /sys/bus/spi/devices/spi1.3
target $ cat eeprom0
0
target $ echo 10 > eeprom0
target $ cat eeprom0
10
target $ cat rdac0
5
target $ echo 3 > rdac0
target $ cat rdac0
3
```

- 32) Verifica funzionamento driver spidev

Dopo avere cross-compilato e copiato nel file system il programma /drivers/spi/spidev_test
target \$./spidev_test -D /dev/spidev1.3

Per progetto Spartacus (XNRG):

Nel progetto XNRG il supporto driver al Trimmer digitale e al display LCD DIGIWISE_13-101XMLAIC1-S viene inserito nel DTS

</linux>/arch/arm/boot/dts/am335x-xnrg.dts

In particolare, per il trimmer digitale, la sezione del DTS rilevante è la seguente

```
&spi1 {
    status = "okay";
    pinctrl-names = "default";
    pinctrl-0 = <&spi1_pins>;
    /* Inverting D0 and D1 meaning */
    ti,pindir-d0-out-d1-in;

    ad5235@1 { /* digital trimmer */
        reg = <1>; /* chip select */
        /* Acutal max frequency is 50 MHz - 40 MHz is just enough */
        spi-max-frequency = <40000000>; /* 40 MHz */
        compatible = "ad5235";
    };
};
```

- 33) Verifica funzionamento driver per potenziometro digitale ad_dpot

```
target $ cd /sys/bus/spi/devices/spi2.1
target $ cat eeprom0
0
target $ echo 10 > eeprom0
target $ cat eeprom0
10
target $ cat rdac0
5
target $ echo 3 > rdac0
target $ cat rdac0
3
```

NOTA: c'è un problema di disallineamento del numero del bus

15.9 Patch 0012-kx-kernel-add-mag3110-support

MAG3110 è un magnetometro. Indichiamo il vettore di campo magnetico con $B_p = (B_{px}, B_{py}, B_{pz})$.

Questa patch introduce il supporto al dispositivo MAG3110 per la scheda KXDB.

Esistono due versioni del driver, una versione semplice che qui non utilizzeremo e una versione full-featured per linux 2.6.35 che si può scaricare da

http://www.freescale.com/webapp/sps/site/prod_summary.jsp?code=SNSDRV SANDROID&fsrch=1

Seguire inizialmente le istruzioni su Readme.txt per integrare il driver. In particolare occorrerà abilitare la compilazione statica nel tree da

DeviceDrivers->Hardware Monitoring Support->Freescale MAG3110 e-compass sensor

La compilazione sotto linux 2.6.37 (versione SDK corrente) si otterrà solo apportando alcune modifiche a mag_core.c e mag_3110.c.

Patch Name:

0012-kx-kernel-add-mag3110-support.patch

Patched Files:

Makefile
arch/arm/configs/da850_kx_defconfig
arch/arm/mach-davinci/board-da850-kx.c
drivers/hwmon/Kconfig
drivers/hwmon/Makefile
drivers/hwmon/mag3110/mag3110.c (versione semplificata del driver)
drivers/hwmon/mag3110/Kconfig
drivers/hwmon/mag3110/Makefile
drivers/hwmon/mag3110/mag_3110.c
drivers/hwmon/mag3110/mag_3110.h
drivers/hwmon/mag3110/mag_char.c
drivers/hwmon/mag3110/mag_core.c
drivers/hwmon/mag3110/mag_input.c
drivers/hwmon/mag3110/mag_regs.h
drivers/hwmon/mag3110/mag_sysfs.c
drivers/hwmon/mag3110/mag_sysfs.h
include/linux/mag3110.h

Configurazione kernel:

Da menuconfig:

36) Abilitare DeviceDrivers->Hardware Monitoring Support ->Freescale MAG3110 e-compass Sensor

Il driver esporta dei file in sysfs, sotto /sys/class/mag/mag con i seguenti attributi che servono a settare alcune modalità operative

ATTR	Default	Description
calibration_offset	0,0,0	
data_output_mode	scaled	
device_id	0xc4	
die_temperature	7 (valore misurato)	
max_range	1.0	
name	MAG3110	
odr	1.250	
operation_mode	wake, standby	
oversampling	normal	

Engineering Specification - confidential

oversampling_values	normal;low_noise;High_resolution;low_power	
poll_time	50	
precision	30.517 nTesla	
range	-1.0:1.0;-1.0:1.0	
range_high	1.0	
range_low	-1.0	
resolution	16	
resolutions	pChip->pSupportedResolution[i] = 16 pChip->pSupportedResolution[i] = 8	
sample_rate	10.0	
supported_odr	80.0;40.0;20.0;10.0;5.0;2.500;1.250;0.630	
type	1	
value	current value of geomagnetic field	Read only
vend	Freescale Semiconductors	
version	1	

Il driver è accessibile da user space tramite il char device node

/dev/input/event2

Verifica di funzionamento:

```
target $ ls /sys/class/mag/mag/
calibration_offset oversampling_values subsystem
data_output_mode poll_time supported_odr
dev power trigger
device_id precision type
die_temperature range uevent
max_range range_high value
name range_low vendor
odr resolution version
operation_mode resolutions
oversampling sample_rate
```

Ricevere caratteri dal device node (FIFO)

target \$ od -x /dev/mag

oppure dal device event node

target \$ od -x /dev/input/event2

```
root@am180x-evm:/opt# ls /sys/class/mag/mag/
calibration_offset oversampling_values subsystem
data_output_mode poll_time supported_odr
dev power trigger
device_id precision type
die_temperature range uevent
max_range range_high value
name range_low vendor
odr resolution version
operation_mode resolutions
oversampling sample_rate
root@am180x-evm:/opt# cat /sys/class/mag/mag/value
3697,-3377,-1602
```

attribute file nodes exported under sysfs. They can be read/written in order to set operative modes.

```
root@am180x-evm:/opt# od -x /dev/mag
00000000 ffff 000e 0e73 f2d5 f9b7 0e75 f2d0 f9c2
00000020 0e72 f2d1 f9be 0e70 f2d1 f9cc 0e77 f2cd
00000040 f9b1 0e69 f2cc f9c5 0e73 f2d0 f9bb 0e75
00000060 f2ca f9bd 0e76 f2d0 f9bb 0e70 f2cc f9bc
00000100 0e76 f2cb f9b1 0e71 f2cf f9c4 0e70 f2ce
00000120 f9be 0e70 f2cf f9af ffff 0004 0e6f f2d1
00000140 f9bb 0e71 f2ce f9bd 0e75 f2cc f9c3 0e71
00000160 f2cd f9b4 ffff 0004 0e75 f2ce f9b8 0e70
00000200 f2d0 f9bd 0e6f f2ce f9bb 0e6e f2cf f9c2
```

magnetometer outputs retrieved from FIFO. They are read in blocks starting with ffff followed by number of valid B 3-D samples. The last value is the most recent.

```
root@am180x-evm:/opt# od -x /dev/input/event1
00000000 1298 4eed f51b 0008 0003 0001 f2d3 ffff
00000020 1298 4eed f539 0008 0000 0000 0000 0000
00000040 1298 4eed 2656 0009 0003 0000 0e70 0000
00000060 1298 4eed 2674 0009 0003 0001 f2d2 ffff
00000100 1298 4eed 2680 0009 0000 0000 0000 0000
00000120 1298 4eed 5604 0009 0003 0002 f9be ffff
00000140 1298 4eed 561f 0009 0000 0000 0000 0000
00000160 1298 4eed 866d 0009 0003 0001 0000 0000
root@am180x-evm:/opt#
```

magnetometer outputs (only variations are reported as async events). Each line is an event with its type and value.

Engineering Specification - confidential

Nota: il file può essere anche /dev/input/event1 nel caso in cui non è stato registrato il device mma955x (accelerometro)

```
root@am180x-evm:/opt# od -x /dev/mag
00000000 ffff 000d 0de4 f287 fc18 0de4 f290 fc1d
00000020 0de4 f28f fc17 0de4 f289 fc1a 0de8 f28a
00000040 fc22 0de0 f28b fc24 0de4 f28a fc20 0de4
00000060 f287 fc1c 0de0 f286 fc16 0de2 f290 fc26
00000100 0de0 f28d fc12 0de1 f289 fc25 0de0 f28f
00000120 fc0f ffff 0004 Odda f291 fc0d 0de1 f288
00000140 fc18 0de3 f288 fc1a 0ddf f28d fc0f ffff
00000160 0004 0de2 f28c fc19 0de2 f290 fc05 0dde
00000200 f28b fc0a 0de3 f28f fc14 ffff 0004 0de3
00000220 f285 fc0c 0de4 f28b fc1b 0de1 f28f fc20
00000240 0dde f287 fc14 ffff 0004 0de2 f288 fc1e
00000260 0de1 f282 fc25 0de1 f287 fc16 0de3 f288
00000300 fc18 ffff 0004 0ddf f289 fc20 0de0 f291
00000320 fc22 0de3 f28a fc14 0dde f28b fc27 ffff
00000340 0004 0de0 f28e fc26 0de4 f28b fc1d 0de1
00000360 f288 fc15 0de3 f292 fc15 ffff 0004 0ddd
```

Verifica di funzionamento:

The magnitude of the geomagnetic field varies from 25 μT in South America to about 60 μT over Northern China. Let's find a rough measure of the actual value

1. First, use a compass to determine the direction of magnetic north.
2. Point the x axis of the MAG3110 northwards and downwards by the inclination angle (typically 50 degrees downwards) and get the following sensor output (units .1uT)

```
target $ cat /sys/class/mag/mag/value
3098,-2757,-411
```

3. Point the x axis of the MAG3110 in the opposite direction (ie aligned against the geomagnetic field) and get the following sensor output

```
target $ cat /sys/class/mag/mag/value
2148,-2666,319
```

4. So you can find the rough magnitude of the geomagnetic field as the half difference of the reported x axis fields
 $(3098 - 2148)/2 = 475 .1\mu\text{T} = 47.5 \mu\text{T}$

Verifica di funzionamento:

Cfr. paragrafo "Accedere da userspace al MAG3110 device driver (eCompass)"

References:

<http://blogs.freescale.com/2012/03/09/source-form-for-ecompass-software-and-4-and-7-element-magnetic-routines-yes/>

15.10 Patch 0013-kx-kernel-add-mma955x-support

MMA955x è un accelerometro. Indichiamo il vettore di accelerazione con $\mathbf{Gp}=(Gpx, Gpy, Gpz)$. Questa patch introduce il supporto al dispositivo MMA955x per la scheda KXDB.

In particolare occorrerà abilitare la compilazione statica nel tree da DeviceDrivers->Hardware Monitoring Support->FreescaleMMA955xIntelligentMotion-SensingPlatform Il parametro ->FreescaleMMA955xIntelligentMotion-SensingPlatformSetting serve ad applicare una rotazione al vettore di accelerazione in funzione dell'orientazione del device rispetto alla grafica del display.

Engineering Specification - confidential

Sul KX, montando il dispositivo in maniera tale che i suoi assi coincidano col riferimento XYZ (o NED) del sistema, non ci sarà bisogno di applicare rotazioni, sicchè mettiamo a 0 questo parametro.

Patch Name:

0013-kx-kernel-add-mma955x-support.patch

Patched Files:

Makefile
arch/arm/configs/da850_kx_defconfig
arch/arm/mach-davinci/board-da850-kx.c
drivers/hwmon/Kconfig
drivers/hwmon/Makefile
drivers/hwmon/mma955x/Kconfig
drivers/hwmon/mma955x/Makefile
drivers/hwmon/mma955x/mma_955x.c
drivers/hwmon/mma955x/mma_955x.h
drivers/hwmon/mma955x/mma_sysfs.c
drivers/hwmon/mma955x/mma_sysfs.h
drivers/hwmon/mma955x/mma_input.c
include/linux/mma955x.h

Configurazione kernel:

Da menuconfig:

37) Abilitare DeviceDrivers->Hardware Monitoring Support ->MMA955xIntelligentMotion-SensingPlatform

Il driver esporta dei file in sysfs, i seguenti attributi che servono a settare alcune modalità operative

ATTR	Default	Values	Description
acc_invert	0	0,1	Accelerometer data inversion for all axes
acc_output_stage	0	0..5	Selects the AFE stage when reading event node
acc_range	4	2,4,8	Accelerometer Full-Span selection (units: g)
calibration_offset	0,0,0	16b,16b,16b	units: 0.244 mg/LSB
position	0	0..7	Select a vector rotation for both acc and gyro
gyro_invert	0	0,1	Gyroscope data inversion for all axes
gyro_range	500	250,500,2000	Gyroscope Full-Span selection (units: dps)
poll	50	1..500	polling time (ms)
min	1	-	min polling time (ms)
max	500	-	max polling time (ms)
sleep	0	0,1	enable/disable sleep mode for mma955x

Il driver è accessibile da user space tramite il char device node

/dev/input/event1

Verifica di funzionamento:

34) Verifica funzionamento driver mma955x

```
target $ ls /sys/devices/virtual/input/input1/
acc_invert          id          position
acc_output_stage   max          power
acc_range           min          sleep
calibration_offset modalias    subsystem
capabilities       name        uevent
event1              phys        uniq
gyro_invert         poll        
```

Ricevere caratteri dal device event node

```
target $ od -x /dev/input/event1
```

Mettere il dispositivo in SLEEP mode

```
target $ echo 1 > /sys/devices/virtual/input/input1/sleep
```

Verificare che il dispositivo è entrato in modalità sleep provando a ricevere dati dal device event node

Mettere il dispositivo in RUN mode

```
target $ echo 0 > /sys/devices/virtual/input/input1/sleep
```

Verificare che il dispositivo è in RUN mode provando a ricevere dati dal device event node.

Nota: La risoluzione dei ciascuna componente del vettore accelerazione è 16-bit, cioè -32767..32768. Selezionando p.es. range=4 si avrà che 32768 corrisponde a 4g, quindi la sensitività è 0.122 mg/LSB.

Il parametro data_output_stage seleziona il particolare stadio del AFE dove il dato viene prelevato nel momento in cui si legge da /dev/input/event1. Di default è impostato 0 e, come in tutti i casi differenti da 3, il valore letto è riscalato in maniera che 4096 corrisponda a 1g.

Soltanto impostando data_output_mode=3 il valore letto mostrerà la sua dipendenza da range.

La variabile position determina una rotazione del vettore G come lo si legge da /dev/input/event1. Il valore position=3 (default) non applica rotazioni rispetto alla normale convenzione degli assi data nel datasheet.

La variabile data_invert permette di applicare una inversione delle componenti di G come lo si legge da /dev/input/event1. Il valore data_invert=0 (default) non applica inversione, sicché quando ad esempio il dispositivo è orientato con asse z verso il basso esso misura G_p = (0,0,-1g) a riposo. L'inversione non viene applicata se data_output_mode=2 (corrispondente al valore assoluto dell'accelerazione).

Il parametro sleep serve a attivare/disattivare la modalità di sleep.

15.11 Patch 0014-kx-kernel-add-bootup-logo

Di default la distribuzione mostra un pinguino TUX in alto a sinistra come logo al boot.

Questa patch introduce un nuovo logo al boot del KX.

Si parte da un'immagine *kxsplash.png* con formato 320x240, ad esempio questa



Patch Name:

0014-kx-kernel-add-bootup-logo.patch

Patched Files:

Makefile

arch/arm/configs/da850_kx_defconfig

drivers/video/logo/Kconfig

drivers/video/logo/Makefile

drivers/video/logo/logo.c

Engineering Specification - confidential

drivers/video/logo/logo_kx_clut224.ppm
include/linux/linux_logo.h

- 1) Installare i tools necessari

```
host $ sudo apt-get install netpbm
```

- 2) Convertire il file

```
host $ pngtopnm kxsplash.png | ppmquant -fs 223 | pnmtoplainpnm > logo_kx_clut224.ppm
```

- 3) Copiare

```
host $ cp logo_kx_clut224.ppm <linux>/drivers/video/logo
```

Editare e modificare i vari file in modo tale che anche questo logo possa essere incluso

Configurazione kernel:

Da menuconfig:

Disabilitare DeviceDrivers->GraphicsSupport->Bootup logo ->Standard black and white Linux logo
Disabilitare DeviceDrivers->GraphicsSupport->Bootup logo ->Standard 16-color Linux logo
Disabilitare DeviceDrivers->GraphicsSupport->Bootup logo ->Standard 224-color Linux logo
Abilitare DeviceDrivers->GraphicsSupport->Bootup logo ->Standard 224-color X KX logo

References:

http://en.gentoo-wiki.com/wiki/Linux_Logo_Hack

15.12 Patch 0015-kx-kernel-add-ad7689-support

Questa patch introduce il supporto per ADC AD7689

Patch Name:

0015-kx-kernel-add-ad7689-support.patch

Patched Files:

Makefile
arch/arm/configs/da850_kx_defconfig
drivers/staging/iio/adc/Kconfig
drivers/staging/iio/adc/Makefile
drivers/staging/iio/adc/ad7689_core.c
drivers/staging/iio/adc/ad7689_ring.c
drivers/staging/iio/adc/ad7689_trigger.c
drivers/staging/iio/adc/ad7689.h

Source:

Progetto in collaborazione con Aptasys (ing. Luciano Neri)

Configurazione kernel:

Da menuconfig:

- 38) Abilitare DeviceDrivers->StagingDrivers->Industrial IO support->Enable ring buffer support within IIO
- 39) Abilitare DeviceDrivers->StagingDrivers->Industrial IO support->Enable triggered sampling support
- 40) Abilitare DeviceDrivers->StagingDrivers->AnalogDevices AD7682/7689 ADC Drivers

Il driver esporta dei file in sysfs, i seguenti attributi che servono a settare alcune modalità operative

ATTR	Default	Values	Description
channels	4	4,8	Number of channels

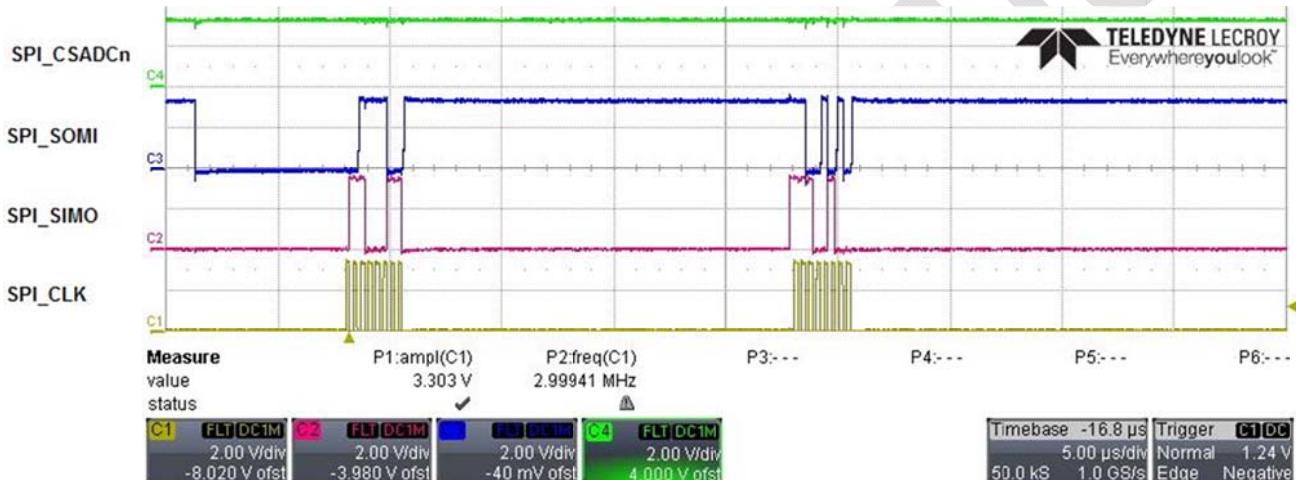
Engineering Specification - confidential

full_bw_filter	0	0,1	
frequency			Sampling frequency
in_scale	0.000		
in0_raw			Read channel 0
in1_raw			Read channel 1
in2_raw			Read channel 2
in3_raw			Read channel 3
in4_raw			Read channel 4
in5_raw			Read channel 5
in6_raw			Read channel 6
in7_raw			Read channel 7

Verifica di funzionamento:

35) Verifica funzionamento driver

```
target $ cd /sys/devices/platform/spi_davinci.1/spi1.1/device0
target $ cat in0_raw
VAL:3CCĀ CFG:63e4
15562
```



36) Verifica funzionamento driver come modulo. Dopo averlo compilato e installato,

```
target $ modinfo ad7689
filename:  /lib/modules/2.6.37-01.00-0015/kernel/drivers/staging/iio/adc/ad7689.ko
alias:    spi:ad7689
license:  GPL v2
description: Analog Devices AD7689 ADC
author:   Luciano Neri - l.neri@aptasys.eu
depends:  industrialio,ring_sw
staging:  Y
vermagic: 2.6.37-01.00-0015 preempt mod_unload modversions ARMv5
target $ insmod ad7689
...
```

15.13 Patch 0016-kx-kernel-add-musb-peripheral-support

Questa patch introduce il supporto peripheral per USB-0

Patch Name:

0016-kx-kernel-add-musb-peripheral-support.patch

Patched Files:

Makefile

arch/arm/configs/da850_kx_defconfig

Configurazione kernel:

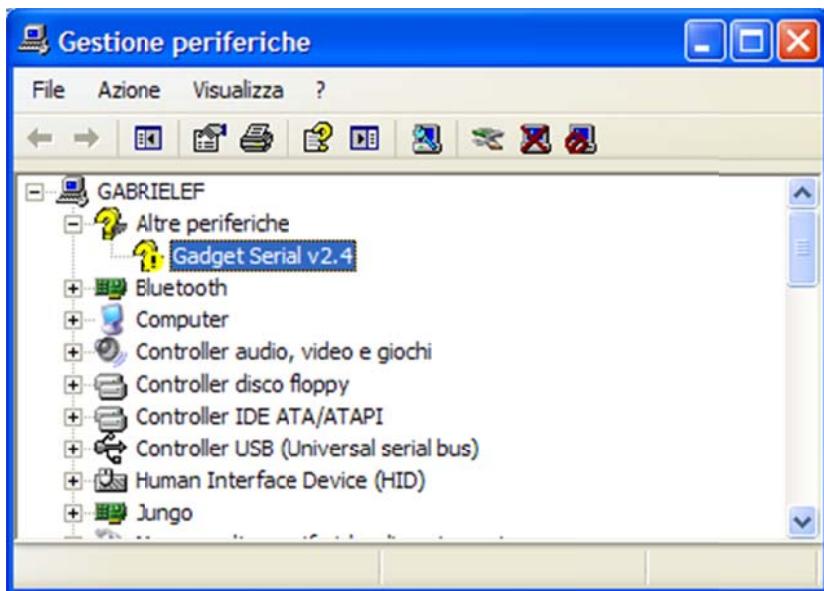
Da menuconfig:

- 41) Abilitare DeviceDrivers->USB Support->Inventra Highspeed Dual Role Controller (TI, ADI, ...)
- 42) Abilitare DeviceDrivers->USB Support->DA8xx/OMAP-L1xx->USB Peripheral (gadget stack)
- 43) Abilitare DeviceDrivers->USB Support-->USB Gadget Support-->USB Peripheral Controller->Inventra ..
- 44) Abilitare DeviceDrivers->USB Support->USB Gadget Support-->USB Gadget Drivers->Gadget Serial

Verifica:

```
target $ cat /sys/devices/platform/musb-da8xx/musb-hdrc/mode  
b_peripheral
```

Quando si collega il KX a un PC windows, sul Pannello di Controllo la periferica viene riconosciuta

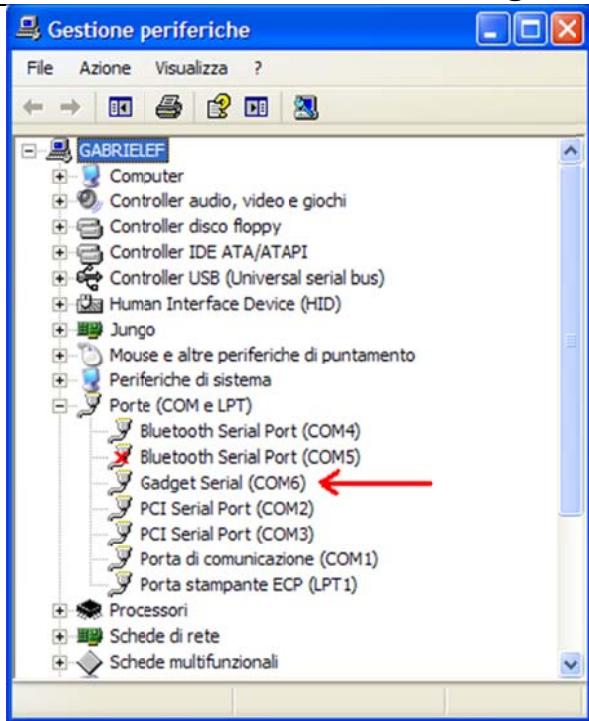


Ottenerne il driver *linux-cdc-acm.inf* e installarlo

Per disabilitare echo sul target

```
target $ stty -F /dev/ttyGS0 -echo
```

Collegare il cavo USB tra host PC e target



Aprire un terminale di emulazione seriale sulla COM port del PC host e comunicare col target

15.14 Creazione patch 0017-kx-kernel-port2kxdb

Con questa patch il kernel diventa compatibile con KXDB Rev.A (<target>=kx) e perde la compatibilità con KXEVB (<target>=kxevb).

Questa patch introduce il supporto per LCD NLT, BLE, ecc.

Patch Name:

0017-kx-kernel-port2kxdb.patch

Patched Files:

```

Makefile
arch/arm/configs/da850_kx_defconfig
arch/arm/mach-davinci/Makefile
arch/arm/mach-davinci/board-da850-kx.c
arch/arm/mach-davinci/devices-da8xx.c
arch/arm/mach-davinci/include/mach/da8xx.h
arch/arm/mach-davinci/include/mach/lcd-kx.h
arch/arm/mach-davinci/include/mach/mux.h
arch/arm/mach-davinci/lcd-kx.c
drivers/misc/ad525x_dpot-spi.c
drivers/misc/ad525x_dpot.c
drivers/misc/ad525x_dpot.h
drivers/video/da8xx-fb.c

```

Configurazione kernel:

Da menuconfig:

45) Abilitare DeviceDrivers->USB Support->USB Gadget Support-->USB Gadget Drivers->Gadget Serial

15.15 Patch 0018-kx-kernel-add-mic3291-support

Questa patch introduce il supporto per LED Driver MIC3291. Il MIC3291 consente di decidere il livello di luminosità del LED di back light display da 0 a 15. Il driver rimappa questi livelli nel range 1..16 e aggiunge il livello 0 che corrisponde allo spegnimento del LED.

Patch Name:

0018-kx-kernel-add-mic3291-support.patch

Patched Files:

```
<ext-modules>/Makefile
<ext-modules>/load.sh
<ext-modules>/unload.sh
<ext-modules>/mic3291_core.c
<ext-modules>/mic3291_core.h
<ext-modules>/mic3291_sysfs.c
<ext-modules>/mic3291_sysfs.h
<ext-modules>/mic3291.h
```

Source:

TBD. SVN ..

Configurazione kernel:

Gestito come modulo esterno

Il driver esporta in sysfs i seguenti files di attributi che servono a interagire col driver da userspace

ATTR	Default	Values	Description
brightness	16	0..16	Level of brightness
delay	0	0..100	Transition step delay (ms)

La scrittura sul file brightness del valore <new_brightness> modifica la luminosità del LED display in un tempo proporzionale al parametro delay.

Verifica di funzionamento:

37) Verifica funzionamento driver

```
target $ modprobe mic3291
target $ lsmod
Module      Size  Used by
mic3291      3087  0
target $ cat /sys/class/mic3291/brightness
16
(target $ echo 6 > /sys/class/mic3291/brightness
mic3291 0.1: brightness set to 6
(target $ echo 0 > /sys/class/mic3291/brightness
target $ echo 6 > /sys/class/mic3291/brightness
target $ echo 16 > /sys/class/mic3291/brightness
mic3291 0.1: brightness set to 16
target $ rmmod mic3291

(<-- il modulo mic3291.ko viene caricato)
(<-- verifica caricamento avvenuto)

(<-- lettura luminosità corrente)
(<-- 16 corrisponde a massima luminosità)
(<-- transizione graduale a luminosità 6)
(<-- il LED si spegne)
(<-- transizione repentina a luminosità 6)
(<-- transizione graduale a luminosità 16)
```

15.15.1 Display Backlight per KX

Quanto detto sopra vale per KX

15.15.2 Display Backlight per Spartacus

Nel progetto XNRG non vi è il mic3291. La luminosità del display è gestita dal DTS

<linux>/arch/arm/boot/dts/am335x-xnrg.dts

```
target $ cat /sys/class/backlight/backlight/brightness          (<- lettura luminosità corrente)
8                                         (<- 8 corrisponde a massima luminosità)
target $ echo 6 > /sys/class/backlight/backlight/brightness    (<- transizione graduale a luminosità 6)
```

15.16 Patch 0019-kx-kernel-add-mpl3115a2-support

Questa patch introduce il supporto per sensore di pressione/altimetro e tempreatura Freescale MPL3115A2.

Patched Files:

```
<ext-modules>/Makefile
<ext-modules>/mpl3115a2_core.c
<ext-modules>/ mpl3115a2_core.h
<ext-modules>/mpl3115a2_io.c
<ext-modules>/ mpl3115a2_io.h
<ext-modules>/mpl3115a2_sysfs.c
<ext-modules>/ mpl3115a2_sysfs.h
```

Source:

Disponibile su GIT branch master

Configurazione kernel:

Gestito come modulo esterno

Il driver esporta in sysfs i seguenti files di attributi che servono a interagire col driver da userspace

ATTR	Default	Values	Description
alt_offset	0	-128..128	Altimeter offset (m)
samples	-		Format: T <ttt> P <ppppp> A <aaaa>
temp_offset	0.00		Temperature offset (°C/100)
pb_offset	0		Barometric Pressure offset (Pa)
sea_level_pascal	101326	P_0, pressure at sea level (Pa)	

Verifica di funzionamento:

38) Al boot del kernel

```
[...]
[ 17.499329] mag3110 1-000e: mag3110 device is probed successfully.
[ 17.829421] mpl3115a2 1-0060: mpl3115a2 irq line 1 is 244
[ 17.834845] mpl3115a2 1-0060: I2C adapter functionalities 0x0eff0009
[ 18.337532] mpl3115a2 1-0060: IRQ config (ctrl_reg4) is 0xff
[ 18.469689] mpl3115a2 1-0060: BAR IN 101326 pa           (<- questo è il valore di P_0, pressione at sea level)
[ 18.702008] mpl3115a2 1-0060: OFF_P 0 Pa, OFF_T 0.00 C, OFF_H 0 m
[ ...]
```

39) Verifica funzionamento driver nella parte accessibile da console

```
target $ ls /sys/bus/i2c/drivers/mpl3115a2/1-0060/
alt_offset      name      samples      temp_offset
driver         pb_offset   sea_level_pascal uevent
modalias       power      subsystem
```

40) Verifica funzionamento

```
target $ cd /sys/bus/i2c/drivers/mlp3115a2/1-0060/
```

```
... TODO
```

15.17 Patch 0019-kx-kernel-add-kxpower-support

Questa patch introduce il supporto per Smart Battery Charger LTC4100 e batteria. Inoltre monitora la linea BLACKOUT per determinare se la batteria è stata disconnessa.

Configurazione kernel:

Da menuconfig:

46) Abilitare DeviceDrivers->PowerSupplyClassSupport

Patched Files:

```
<ext-modules>/Makefile
<ext-modules>/kxpower_core.c
<ext-modules>/kxpower_ltc4100.c
<ext-modules>/kxpower_battery.c
<ext-modules>/kxpower_core.h
<ext-modules>/kxpower_ltc4100.h
<ext-modules>/kxpower_battery.h
<ext-modules>/kxpower_fileops.c
<ext-modules>/kxpower.h
<ext-modules>/test/Makefile
<ext-modules>/test/test_kxpower.c
```

Source:

Disponibile su GIT branch master a partire da [6308f654001b9865f17b8cb5391598c3d2ff4d57](https://github.com/linaro-kernel/kx-kernel-add-kxpower-support/commit/6308f654001b9865f17b8cb5391598c3d2ff4d57)

Configurazione kernel:

Gestito come modulo esterno

Il driver esporta in sysfs i seguenti files di attributi che servono a interagire col driver da userspace

ATTR	Default	Values	Description
capacity	-	0..100	Relative State Of Charge (%)
charge_empty_design	0		(mAh)
charge_full_design	3100		(mAh)
current_avg	-		Average Battery Current (mA). Battery is discharging when negative
current_now	-		Battery Current (mA). Battery is discharging when negative
cycle_count			
health			“Good”/“Dead”
serial_number			
status			“Not charging”/“Charging”
technology			
temp			Battery Temperature (0.1°K)
time_to_empty_avg			(min)
time_to_full_avg			(min)
type			
voltage_now			Battery Voltage (mV)

Integrazione nell'applicativo:

Cercare `kxpower_evt_flag()`

Engineering Specification - confidential

Cfr: <ext-modules>/test/test_kxpower.c

Verifica di funzionamento:

41) Al boot del kernel

```
[...]
[ 15.693848] kxpower 1.0: blackout irq line is 188          (<- registrazione dell'interrupt dalla VBAT, linea BLACKOUT_INT)
[ 15.938789] kxpower 1.0: detected battery nb2037hd, SN 349, Design cap. 3100, Full cap. 2899, cycles 6
[ 16.497166] kxpower 1-0009: ltc4100 irq line is 137          (<- registrazione dell'interrupt dal LTC4100, linea SMBALERT)
[ 16.598829] kxpower 1-0009: I2C adapter functionalities 0x0eff0009
[...]
```

42) Verifica funzionamento driver nella parte accessibile da console

```
target $ ls /sys/class/power_supply/kxbattery/
capacity      manufacturer      technology
charge_empty_design model_name      temp
charge_full     online        time_to_empty_avg
charge_full_design power       time_to_full_avg
current_avg     present       type
current_now    serial_number uevent
cycle_count    status        voltage_now
health         subsystem

target $ cat /sys/class/power_supply/kxpower/capacity
68
target $ cat /sys/class/power_supply/kxpower/charge_empty_design
0
target $ cat /sys/class/power_supply/kxpower/charge_full
2890
target $ cat /sys/class/power_supply/kxpower/charge_full_design
3100
target $ cat /sys/class/power_supply/kxpower/current_avg
-551
target $ cat /sys/class/power_supply/kxpower/current_now
-497
target $ cat /sys/class/power_supply/kxpower/cycle_count
14
target $ cat /sys/class/power_supply/kxpower/health
Good
target $ cat /sys/class/power_supply/kxpower/manufacturer
INSPIRED
target $ cat /sys/class/power_supply/kxpower/model_name
nb2037hd
target $ cat /sys/class/power_supply/kxpower/online
0
target $ cat /sys/class/power_supply/kxpower/present
1
target $ cat /sys/class/power_supply/kxpower/serial_number
253
target $ cat /sys/class/power_supply/kxpower/status
Not charging
target $ cat /sys/class/power_supply/kxpower/technology
Li-ion
target $ cat /sys/class/power_supply/kxpower/temp
3013
target $ cat /sys/class/power_supply/kxpower/time_to_empty_avg
235
target $ cat /sys/class/power_supply/kxpower/time_to_full_avg
0
target $ cat /sys/class/power_supply/kxpower/type
Battery
target $ cat /sys/class/power_supply/kxpower/voltage_now
7555
```

43) Verifica funzionamento driver attraverso interfaccia a eventi

```
target $ test_kxpower
Reading from /dev/kxpower
AC | CHG | BAT | FULL | DEAD | LOW | COND | REPL | BAT% | BATmin
0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 25% | 70
0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 25% | 69
0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 24% | 69
0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 24% | 68
<- Collego AC main adapter
1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 24% | 68
1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 24% | 335
```

Engineering Specification - confidential

```
1| 1| 1| 0| 0| 0| 0| 0| 24%| 201  
1| 1| 1| 0| 0| 0| 0| 0| 24%| 174  
1| 1| 1| 0| 0| 0| 0| 0| 25%| 164  
1| 1| 1| 0| 0| 0| 0| 0| 25%| 159  
1| 1| 1| 0| 0| 0| 0| 0| 25%| 157  
1| 1| 1| 0| 0| 0| 0| 0| 25%| 156  
1| 1| 1| 0| 0| 0| 0| 0| 25%| 155  
<- Collego la batteria  
1| 0| 0| 0| 0| 0| 0| 0| 0%| 0  
<- Collego la batteria  
[ 45.175896] kxpower 1.0: detected battery nb2037hd, SN 349, Design cap. 3100, Full cap. 2899, cycles 6  
1| 1| 1| 0| 0| 0| 0| 0| 25%| 274  
1| 1| 1| 0| 0| 0| 0| 0| 26%| 194  
...
```

Come l'applicazione gestisce la scarica della batteria:

Quanto segue vale con il KX non è connesso al AC/DC adapter.

Quando la capacità della batteria raggiunge il 10%, il driver kxpower lo segnala con una flag specifica all'applicazione

la quale mostra il messaggio a display "Battery LOW!".

Da questo momento, una richiesta di avvio test da Omnia viene abortita inviando a Omnia un messaggio EM_BATTERYLOW, che Omnia gestisce con un messaggio a video "Battery Low".

Quando la capacità della batteria scende sotto il 5% l'applicazione, che comunque monitora sempre tale valore, mostra il messaggio a display "Il test si spegne..", salva e chiude il test eventualmente in corso.

Quando la tensione batteria scende sotto 6150 mV, il driver kxpower lo segnala con una flag specifica all'applicazione

ed esegue lo shutdown forzato dopo 5 sec. Questi 5 sec servono all'applicazione per salvare e chiudere eventuali test in corso.

15.18 Patch 0020-kx-kernel-add-fiq-support

Nel KX, durante i picchi di ventilazione (240 L/min) gli impulsi generati dalla turbina hanno periodicità dell'ordine di 200 us. Ogni impulso genera un'interrupt sulla periferica eCAP che legge il flusso della turbina. I canali standard dell'interrupt controller (irq) del AM1808 hanno una latenza tipicamente maggiore di 200 us, pertanto vi sono dei problemi. L'AM1808 consentirebbe di configurare due canali come fast interrupt (fiq) aventi latenza inferiore a 8 us. Questa patch introduce il supporto arm per i fast interrupt.

Al momento il driver supporta un solo canale fiq, anche se in teoria se ne potrebbe aggiungere un secondo.

Configurazione kernel:

Da menuconfig:

47) Abilitare DeviceDrivers->PowerSupplyClassSupport

Patched Files:

```
<kernel>/arch/arm/Kconfig  
<kernel>/arch/arm/configs/da850_kx_debugconfig  
<kernel>/arch/arm/configs/da850_kx_defconfig  
<kernel>/arch/arm/configs/da850_kx_ubidebugconfig  
<kernel>/arch/arm/kernel/Makefile  
<kernel>/arch/arm/kernel/entry-armv.S  
<kernel>/arch/arm/kernel/vector-fiq-entry.S  
<kernel>/arch/arm/kernel/vector-fiq.c  
...
```

Source:

Disponibile su GIT branch master a partire da 8a60582aebb8805fd734752ffe505faddf0ef3

15.19 Patch 0019-kx-kernel-add-ant-support

Questa patch introduce il supporto per sensore ANT C7.

Su un socket si effettua la comunicazione di controllo tra l'applicazione utente e il demone.

Ad esempio per lanciare una ricerca, ecc.

AntControl_search_sensor(timeout,..) torna una lista di oggetti con tipo, ID e RSSI.

AntControl_open_sensor(ID) per aprire il canale desiderato

Quando la connessione con un sensore è attiva i dati vengono passati su una shared memory.

La funzione di lettura..

Patched Files:

<ext-modules>/Makefile

..

Source:

Disponibile su GIT branch master

Configurazione kernel:

Gestito come modulo esterno

15.20 Patch 0019-kx-kernel-add-hrecg-support

Questa patch introduce il supporto per l'interfaccia HR/TTL.

Patched Files:

```
<ext-modules>/Makefile
<ext-modules>/hrecg/hrecg_core.c
<ext-modules>/hrecg/hrecg_core.h
<ext-modules>/hrecg/hrecg_fileops.c
<ext-modules>/hrecg/hrecg.h
<ext-modules>/hrecg/test/Makefile
<ext-modules>/hrecg/test/test_hr_read.c
<ext-modules>/hrecg/README
```

Source:

Disponibile su GIT branch master a partire da *hash 076918eed..*

Configurazione kernel:

Gestito come modulo esterno

- 44) Verifica funzionamento driver attraverso interfaccia a eventi

```
target $ test_hr_read
Reading from /dev/hrecg
INV | HRV | HR | RR
 0 | 0 | 57 | 1050          (<- Il simulatore HR Polar è acceso, HR_avg = 57 bpm, RR_avg = 1050 ms)
<- Incremento frequenza
 0 | 0 | 78 | 760
<- Riduco frequenza
 0 | 0 | 57 | 1050
 0 | 0 | 56 | 1055
 0 | 0 | 57 | 1050
<- Incremento frequenza
 0 | 0 | 69 | 866
```

```
0 | 0 | 100 | 596
0 | 0 | 102 | 587
<- Incremento frequenza
0 | 0 | 117 | 510
0 | 0 | 121 | 495
0 | 0 | 151 | 396
0 | 1 | 151 | 320
(<- La flag HRV segnala RR<350ms, RR istantaneo = 320 ms)
<- Incremento frequenza
0 | 1 | 158 | 378
0 | 1 | 195 | 307
<- Riduco frequenza
0 | 1 | 174 | 343
0 | 0 | 116 | 513
0 | 0 | 115 | 517
0 | 0 | 116 | 515
0 | 0 | 110 | 543
<- Spengo simulatore
0 | 0 | 54 | 1095
0 | 0 | 49 | 1210
0 | 0 | 39 | 1525
0 | 0 | 30 | 1990
0 | 0 | 69 | 865
1 | 0 | 69 | 865
(<- La flag INV segnala assenza di battiti)
...
```

16 Bootloader

Il bootloader di secondo livello inizializza l'HW minimale del target e dirige il caricamento del sistema operativo in RAM.

Questo capitolo spiega dove reperire i sorgenti, come modificare e ricompilare U-Boot.

16.1 Download page

<http://git.denx.de/cgi-bin/gitweb.cgi?p=u-boot.git;a=summary>

16.2 Riferimenti

<http://www.denx.de/wiki/U-Boot/WebHome>
<http://lists.denx.de/pipermail/u-boot/>

16.3 Re-build u-boot

http://processors.wiki.ti.com/index.php/GSG:_Building_Software_Components_for_OMAP-L1#Rebuilding_U-Boot

16.3.1 KX

Si assume che la Toolchain e l'SDK siano installati.

48) Entrare nella cartella dei sorgenti

```
host $ cd <uboot>
```

49) Rebuild

Engineering Specification - confidential

```
host $ make CROSS_COMPILE=<armtoolch> distclean  
host $ make CROSS_COMPILE=<armtoolch> da850evm_config  
host $ make CROSS_COMPILE=<armtoolch> all
```

dove

<armtoolch> = arm-arago-linux-gnueabi

Il file generato è u-boot.bin.

Le opzioni di build sono specificate in
include/configs/da850evm.h

16.3.2 XNRG

Si assume che la Toolchain e l'SDK siano installati.

50) Entrare nella cartella dei sorgenti

```
host $ cd <uboot>
```

51) Rebuild

```
host $ make CROSS_COMPILE=<armtoolch> distclean  
host $ make CROSS_COMPILE=<armtoolch> am335x_xnrg_defconfig  
host $ make CROSS_COMPILE=<armtoolch> all
```

dove

<armtoolch> = arm-epy-linux-gnueabihf

Il file generato è u-boot.bin.

Le opzioni di build sono specificate in
include/configs/da850evm.h

16.4 Creazione supporto u-boot per board KX

Creare il BSP per la board kx significa partire da una copia del u-boot sorgente fornito nel SDK (pulito con distclean), assegnando la versione xx.xx (inizialmente 01.00)

L'SDK 5.03 include il PSP 03.21.00.04, che è basato su U-Boot 2010.12

Tuttavia il KX non utilizza questa versione, ma la versione più recente U-Boot 2011.12-rc1 ottenuta dal GIT tree.

```
host $ cd <sdkpsp>/src/u-boot
```

```
host $ cp -R u-boot-2011.12-rc1 u-boot-xx.xx/
```

(<uboot>)

e applicarvi una sequenza di patches preventivamente create. Le patch vengono create e applicate con quilt. Nella stessa cartella contenente <uboot> deve essere presente la cartella patches per quilt. Ogni patch deve includere la modifica della macro EXTRAVERSION presente in <uboot>/Makefile.

Le patches vengono periodicamente compresse

```
host $ cd <uboot>
```

```
host $ tar ..czf u-boot-qpatches-xx.xx.tar.gz ..patches
```

e archiviate in

/Serverfirmware/X/common/u-boot/vxx/

Opzionale: sottoporre le patch a u-boot@lists.denx.de

16.5 Creazione patch 0001-kx-uboot-basic

Patch Name:

0001-kx-uboot-basic.patch

Patched Files:

<uboot>/Makefile

Engineering Specification - confidential

```
<uboot>/arch/arm/cpu/arm926ejs/X/Makefile
<uboot>/arch/arm/cpu/arm926ejs/X/cpu.c
<uboot>/arch/arm/cpu/arm926ejs/X/da850_lowlevel.c
<uboot>/arch/arm/cpu/arm926ejs/X/da850_pinmux.c
<uboot>/arch/arm/cpu/arm926ejs/X/dm355.c
<uboot>/arch/arm/cpu/arm926ejs/X/dm365.c
<uboot>/arch/arm/cpu/arm926ejs/X/dm365_lowlevel.c
<uboot>/arch/arm/cpu/arm926ejs/X/dm644x.c
<uboot>/arch/arm/cpu/arm926ejs/X/dm646x.c
<uboot>/arch/arm/cpu/arm926ejs/X/dp83848.c
<uboot>/arch/arm/cpu/arm926ejs/X/et1011c.c
<uboot>/arch/arm/cpu/arm926ejs/X/ksz8873.c
<uboot>/arch/arm/cpu/arm926ejs/X/lowlevel_init.S
<uboot>/arch/arm/cpu/arm926ejs/X/lxt972.c
<uboot>/arch/arm/cpu/arm926ejs/X/misc.c
<uboot>/arch/arm/include/asm/arch-X/aintc_defs.h
<uboot>/arch/arm/include/asm/arch-X/da850_lowlevel.h
<uboot>/arch/arm/include/asm/arch-X/da8xx-fb.h
<uboot>/arch/arm/include/asm/arch-X/davinci_misc.h
<uboot>/arch/arm/include/asm/arch-X/ddr2_defs.h
<uboot>/arch/arm/include/asm/arch-X/dm365_lowlevel.h
<uboot>/arch/arm/include/asm/arch-X/emac_defs.h
<uboot>/arch/arm/include/asm/arch-X/emif_defs.h
<uboot>/arch/arm/include/asm/arch-X/gpio.h
<uboot>/arch/arm/include/asm/arch-X/hardware.h
<uboot>/arch/arm/include/asm/arch-X/i2c_defs.h
<uboot>/arch/arm/include/asm/arch-X/nand_defs.h
<uboot>/arch/arm/include/asm/arch-X/pinmux_defs.h
<uboot>/arch/arm/include/asm/arch-X/pll_defs.h
<uboot>/arch/arm/include/asm/arch-X/psc_defs.h
<uboot>/arch/arm/include/asm/arch-X/sdmmc_defs.h
<uboot>/arch/arm/include/asm/arch-X/syscfg_defs.h
<uboot>/arch/arm/include/asm/arch-X/timer_defs.h
<uboot>/board/X/kx/Makefile
<uboot>/board/X/kx/da830kx.c
<uboot>/board/X/kx/da850kx.c
<uboot>/board/X/kx/misc.c
<uboot>/board/X/kx/misc.h
<uboot>/boards.cfg
<uboot>/include/configs/da850kx.h
<uboot>/tools/Makefile
<uboot>/tools/genublimg.c
```

- 45) Usando la macro EXTRAVERSION presente nel Makefile modificare la versione del kernel appendendo –pppp-xx.xx (pppp è il numero di patch, xx.xx la versione del FW)

```
#define EXTRAVERSION      "-rc1-01.00-0001"
```

- 52) Estrarre la cartella dei sorgenti ed entrarci

```
host $ cd <uboot>
host $ mkdir board/X
host $ cp -R board/davinci/da8xxevm board/X
host $ mv board/X/da8xxevm board/X/kx
```

Engineering Specification - confidential

```
host $ mv board/X/kx/da830evm.c board/X/kx/da830kx.c
host $ mv board/X/kx/da850evm.c board/X/kx/da850kx.c
host $ rm board/X/kx/hawkboard*
host $ cp include/configs/da850evm.h include/configs/da850kx.h
host $ cp -R arch/arm/include/asm/arch-davinci arch/arm/include/asm/arch-X
```

Editare <uboot>/board/X/kx/Makefile e sostituire

CONFIG_MACH_DAVINCI_DA8x0_EVM con CONFIG_MACH_DA8x0_KX
da8x0evm.o con da8x0kx.o

Eliminare riga relativa al build hawkboard

Editare <uboot>/board/X/kx/da850kx.c e <uboot>/include/configs/kx.h e sostituire
CONFIG_MACH_DAVINCI_DA850_EVM con CONFIG_MACH_DA850_KX

Editare <uboot>/boards.cfg e aggiungere riga con
Da850kx arm arm926ejs kx X X

Editare

```
<uboot>/board/X/kx/common.c
<uboot>/board/X/kx/common.h
<uboot>/drivers/net/davinci_emac.c
```

e nei vari punti dove sono usati

```
davinci_da850_evm
CONFIG_DAVINCI_DA850_EVM
MACH_DAVINCI_DA850_EVM
```

Sostituire o abbinare in OR con

```
da850_kx
CONFIG_DA850_KX
MACH_DA850_KX
```

Editare <uboot>/Makefile e aggiungere

```
da850kx_config: unconfig
@$(MKCONFIG) -a kx arm arm926ejs kx X X
```

Editare <uboot>/board/X/kx/Makefile

e sostituire CONFIG_MACH_DAVINCI_DA8x0_EVM con CONFIG_MACH_DA8x0_KX
da8x0evm.o con da8x0kx.o

53) Modificare il numero di macchina.

Il numero di macchina usato nel kernel deve essere uguale a quello usato nel bootloader.
Dopo avere compilato il kernel con il corretto numero di macchina, copiamo il file mach-types
generato nella cartella <uboot>

```
host $ cp <linux>/include/generated/mach-types.h <uboot>/include/asm/mach-types.h
```

Editare <uboot>/board/X/kx/da850kx.c
e in board_init(), sostituire MACH_TYPE_DAVINCI_DA850_EVM con MACH_TYPE_DA850_KX

54) Editare <uboot>/include/configs/ da850kx.h

sostituire
CONFIG_MACH_DAVINCI_DA850_EVM con CONFIG_MACH_DA850_KX

55) Editare <uboot>/include/configs/da850kx.h

Correggere l'indirizzo I2C dell'IO Expander

```
#define CONFIG_SYS_I2C_EXPANDER_ADDR 21
```

(sulla evm è definito 20, che è l'addr del expander sulla UI board)

56) Editare <uboot>/board/X/kx/da850kx.c

Aggiungere

```
#define CONFIG_INIT_I2C_EXPANDER /* Init IO Expander Outputs */
```

Aggiungere in misc_init_r() la call a i2c_hw_init(). Definire i2c_hw_init()

57) Editare <uboot>/include/configs/da850kx.h

Inserire nell'intestazione le informazioni aziendali

Settare PHYS_SDRAM_1_SIZE a (128 << 20) /* SDRAM SIZE 128MB */
(I moduli SOM hanno questo taglio di memoria DDR)

Aggiungere tra gli U-Boot commands anche il support per I2C

```
#define CONFIG_CMD_I2C
```

(da la possibilità di scansionare il bus da u-boot)

Aggiungere nella sezione SoC Configurations anche il supporto per GPIO

```
#define CONFIG_DA8XX_GPIO
```

58) Editare <uboot>/board/X/kx/da850kx.c

Inserire nell'intestazione le informazioni aziendali

Aggiungere

```
#define CONFIG_INIT_GPIO /* Init GPIO */
```

Completere get_board_rev() in maniera da includere la revisione HW del PCB tramite lettura dei pin GP6[7:6]

Aggiungere in misc_init_r() la call a gpio_hw_init(). Definire gpio_hw_init()

59) Rebuild

```
host $ make CROSS_COMPILE=<armtoolch>- distclean  
host $ make CROSS_COMPILE=<armtoolch>- da850kx_config  
host $ make CROSS_COMPILE=<armtoolch>- all
```

Il file generato è u-boot.bin.

Si procede poi a tutta una serie di modifiche

60) Aggiungere tool di post-processing per inserimento dell'header UBL all'immagine u-boot.bin. Il file generato è u-boot-ubl.bin

61) Aggiungere funzione di inizializzazione GPIOs

62) Aggiungere comando di controllo dei GPIOs (kxgpio) e degli ExpanderIOs (kxoexp) nel help menu. Si ottiene implementando i moduli cmd_gpio.c e cmd_ioexp.c

- 63) Aggiungere passaggio del SerialNumber da u-boot a linux. Si ottiene attivando il Tag CONFIG_SERIAL_TAG e definendo la funzione get_board_serial in da850kx.c
http://www.simtec.co.uk/products/SWLINUX/files/booting_article.html#appendix_tag_reference

16.6 Creazione patch 0002-kx-uboot-cmd-cfg

Patch Name:

0002-kx-uboot-add-cmd-cfg.patch

Patched Files:

<uboot>/Makefile
<uboot>/board/X/kx/Makefile
<uboot>/board/X/kx/cmd_cfg.c
<uboot>/board/X/kx/cmd_cfg.h
<uboot>/board/X/kx/da850kx.c

- 46) Aggiornare EXTRAVERSION nel Makefile a “-rc1-0002-01.00”

...

16.7 Creazione patch 0003-kx-uboot-port2kxdb

Con questa patch u-boot diventa compatibile con KXDB Rev.A (<target>=kx) e perde la compatibilità con KXEVB (<target>=kxevb).

Patch Name:

0003-kx-uboot-port2kxdb.patch

Patched Files:

<uboot>/Makefile
<uboot>/board/X/kx/da850kx.c
<uboot>/board/X/kx/misc.c
<uboot>/board/X/kx/misc.h
<uboot>/board/X/kx/cmd_cfg.c
<uboot>/board/X/kx/cmd_cfg.h
<uboot>/board/X/kx/cmd_gpio.c
<uboot>/board/X/kx/cmd_ioexp.c

- 47) Aggiornare EXTRAVERSION nel Makefile a “-rc1-0003-01.00”

...

17 File System

Il filesystem permette di accedere alle informazioni presenti nella memoria di sistema al livello di files e directories. Il root filesystem è il filesystem da dove il kernel carica la prima applicazione e i moduli. Il progetto KX parte dal file system fornito con la versione più recente del SDK. Lo script di setup del SDK permette di installare il file system in <rootpath> e di configurare il NFS, automaticamente. In questo capitolo i passaggi sono esplicitati

17.1 Download page

<http://arago-project.org/git/?p=arago.git;a=shortlog;h=refs/tags/DEV.DaVinciPSP.03.XX.00.36>

17.2 Montare il root FS da NFS

- 1) La cartella del root file system che viene utilizzata per esportare il NFS deve essere creata ex novo, con permessi di scrittura

```
host $ cd /home/<user>
host $ mkdir -p workdir/kx
host $ cp -a -R sdk_x_xx_xx_xx/filesys ./workdir/kx
host $ sudo chmod 777 -R /home/<user>/workdir/kx/filesys
```

- 2) Installare NFS, configurare /etc/exports e lanciare NFS (vedi capitolo NFS)
- 3) Sul Target, dal prompt di u-boot, impostare le seguenti variabili di ambiente (esempio)

```
X-KX > printenv
bootdelay=3
baudrate=115200
ethaddr=00:08:ee:04:34:d2
rootpath=/home/gabriele/workdir/filesys
nfsserverip=192.168.0.7
bootfile=uImage
filesize=202518
fileaddr=C0700000
ipaddr=192.168.0.233
serverip=192.168.0.7
bootargs=console=ttyS2,115200n8 nointrd rw ip=192.168.0.149:192.168.0.7:255.255.255.0::off root=/dev/nfs
nfsroot=192.168.0.142:/home/gabriele/workdir/kx/filesys,M
stdin=serial
stdout=serial
stderr=serial
ver=U-Boot 2009.11 (Nov 25 2010 - 10:19:19)
bootcmd=tftp;bootm
```

Se si intende lavorare con DHCP,TFTP e NFS usare

```
target# setenv bootargs console=ttyS2,115200n8 nointrd rw ip=dhcp root=dev/nfs nfsroot=${nfsserverip}:${rootpath},nolock mem=128M
rootdelay=3
target# setenv bootcmd 'dhcp;bootm'
```

Se si intende lavorare con indirizzo IP statico, TFTP e NFS usare

```
target# setenv bootargs console=ttyS2,115200n8 nointrd rw ip=${ipaddr}:${serverip}:255.255.255.0::off,root=dev/nfs
nfsroot=${nfsserverip}:${rootpath},nolock mem=128M rootdelay=3
target# setenv bootcmd 'tftp;bootm'
```

- 4) Reboot

```
target # boot
```

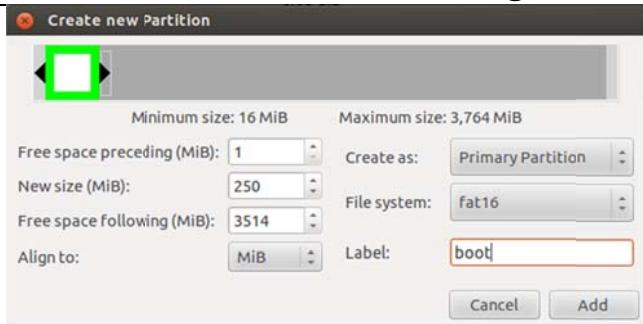
17.3 Montare il root FS da SD card

Questo paragrafo spiega come precaricare il file system sulla card SD. La card deve essere prima formattata e partizionata usando gparted o fdisk

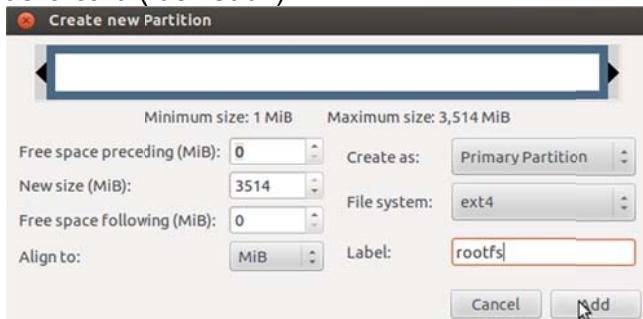
- 1) Esplorando la cartella /dev prima e dopo l'inserimento della card (anche un adattatore SD/USB va bene), vedere qual è il nodo interessato (sdb, sdc, ..). Sia sdb

Con gparted:

- 2) Per cancellare/creare partizioni lanciare gparted (se non disponibile, installarlo con apt-get install)
host \$ sudo gparted
- 3) Selezionare e smontare le partizioni del media (attenzione a non toccare l'hard disc del sistema!!)
- 4) Volendo memorizzare nella card anche u-boot e ulmage, definire una partizione "boot", di tipo FAT32 da 250-500MiB (/dev/sdb1)

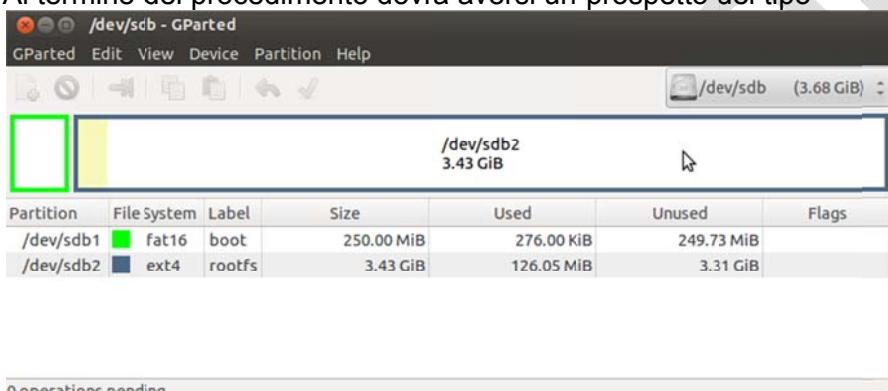


- 5) Per il root filesystem, definire una partizione “rootfs” di tipo ext4 che occupi tutta la parte rimanente della card (/dev/sdb2)



- 6) Applicare le modifiche.

Al termine del procedimento dovrà aversi un prospetto del tipo



- 7) Disinserire e reinserire la SD card, in maniera che le due partizioni vengano ad essere automaticamente montate sulle cartelle /media/boot e /media/rootfs
 8) Copiare l'immagine ubi, kernel, u-boot e rootfs.ubi ed eventuali script per u-boot nella partizione “boot”

```
host $ sudo cp <linux>/arch/arm/boot/uimage /media/boot
host $ sudo cp <uboot>/u-boot-ubl.bin /media/boot
host $ sudo cp <uboot>/myscripts/boot.scr /media/boot
host $ sudo cp <uboot>/myscripts/prog.sh /media/boot
host $ sudo cp <rootfs>/..rootfs.ubi /media/boot
```

- 9) Copiare il target filesystem nella partizione “rootfs”

```
host $ sudo cp -Rf <rootfs>/* /media/rootfs
```

- 10) Smontare e resync

```
host $ sync
host $ sudo umount /media/boot /media/rootfs
```

- 11) Verifica sul Target. Dal prompt di u-boot

```
target # boot_mmc
```

Con fdisk:

Verifica partizioni preesistenti

```
host $ sudo fdisk -l /dev/sdb
```

```
Disk /dev/sdc: 3963 MB, 3963617280 bytes
255 heads, 63 sectors/track, 481 cylinders
Units = cylinders of 16065 * 512 = 8225280 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disk identifier: 0x000c6363
```

Device	Boot	Start	End	Blocks	Id	System
/dev/sdc1	*	1	7	56196	6	FAT16
/dev/sdc2		8	481	3807405	83	Linux

Volendo memorizzare nella card anche u-boot e ulimage, definire una partizione "boot", di tipo FAT16 da 50MiB (/dev/sdb1)

```
host $ sudo fdisk /dev/sdb
Command (m for help): m
Command action
  a  toggle a bootable flag
  b  edit bsd disklabel
  c  toggle the dos compatibility flag
  d  delete a partition
  l  list known partition types
  m  print this menu
  n  add a new partition
  o  create a new empty DOS partition table
  p  print the partition table
  q  quit without saving changes
  s  create a new empty Sun disklabel
  t  change a partition's system id
  u  change display/entry units
  v  verify the partition table
  w  write table to disk and exit
  x  extra functionality (experts only)
```

cancellare le partizioni esistenti:

d 1
d 2

verifica partizioni: p
creare: n p 1 1 7
tipo: t 6

creare: n p 2 8 481
tipo: t 83

bootable: a 1

creazione: w

sudo mkfs.ext3 /dev/sda2

mount
sync

17.4 Creare uno script per u-boot

Per creare uno script eseguibile da u-boot

12) Editare boot.txt

13) Se non si dispone di mkimage installarlo

```
host $ sudo apt-get install mkimage
```

14) Trasformare il file in formato scr

```
host $ mkimage -A arm -O linux -T script -C none -a 0 -e 0 -n 'Execute ulimage' -d boot.txt boot.scr
```

15) Copiare nel supporto desiderato (p.es. nella SD card, partizione boot)

Lo stesso file può essere utilizzato per personalizzare bootargs con altre variabili

Esempio di contenuto del file boot.txt

```
defaultdisplay=tv  
buddy=unknown  
bootargs=console=${console} ${optargs} mpurate=${mpurate} buddy=${buddy}  
camera=${camera} vram=${vram} omapfb.mode=tv:ntsc  
omapdss.def_disp=${defaultdisplay} root=${mmcroot} rootfstype=${mmcrootfstype}  
omapdss.tvcable=composite  
uenvcmd= run loaduiimage; run mmcboot
```

17.5 Montare il root FS da dispositivo USB Storage

To do

17.6 ramdisk

Il KX dispone di un file system di tipo ram disk memorizzato nella flash SPI.

La partizione MTD dedicata si chiama "ramdisk" e ha dimensione 0x3e0000 (4063232 bytes).

Questo significa che l'immagine gzip-compressa non può superare questa dimensione.

Partiamo da un ram disk ext2 preesistente disponibile nel pacchetto 1015906B_OMAP-L138_AM1808_U-Boot_Labs.zip di LogicPD

<http://support.logicpd.com/downloads/1350/>

Questa immagine si chiama ramdisk-base.gz (3377860 bytes compresso, 8MiB uncompressed).

17.7 Aggiornare il contenuto di una immagine ramdisk preesistente

Spostare nella directory di lavoro una copia del ramdisk-base.

```
host $ mv ramdisk-base.gz ramdisk.gz  
host $ mkdir -p tmp  
host $ gunzip ramdisk.gz  
host $ sudo mount -o loop ramdisk tmp
```

Ad esempio vogliamo aggiornare i moduli del kernel

```
host $ make ARCH=arm INSTALL_MOD_PATH=<tmp_path> modules_install
```

Copiare/installare nel rootfs tutte le altre cose che interessa avere. Infine smontare e comprimere

```
host $ sudo umount tmp/  
host $ gzip -9 ramdisk
```

17.8 Creare un nuovo ramdisk appoggiandosi a NFS

Può essere necessario creare un ramdisk ex-novo, per cambiare il tipo o la dimensione massima (in RAM).

La procedura per creare, ad esempio, un RAM disk con N blocchi da 1KB è la seguente

Engineering Specification - confidential

La procedura può essere eseguita sull'host o sul target, ma siccome di default il comando dd non è disponibile sull'host Ubuntu, lavoriamo sul target.

Copiare dall'host in /home la cartella rootfs-xx.xx (<rootfs>), contenente il file system
gome \$ cp -R <rootfs> <rootpath>/home

Fare il boot del target da NFS

```
target $ cd /home
```

Assumiamo esista già un device node /dev/ramX altrimenti lo si deve creare con

```
target $ sudo mknod /dev/ramX b 1 X  
target $ sudo chmod 660 /dev/ramX
```

Supponiamo di volere un ramdisk di tipo ext2 da 16MiB, quindi scegliamo N=16384

```
target $ mkdir -p tmp  
target $ sudo mke2fs -vM0 /dev/ramX 16384  
target $ sudo mount -t ext2 /dev/ramX tmp  
target $ cp -R -a <rootfs> tmp  
target $ umount tmp/  
target $ dd if=/dev/ramX bs=1k count=16384 of=ramdisk  
target $ gzip -9 ramdisk
```

L'ultimo comando avrà creato il file desiderato ramdisk.gz.

Questo comando può anche essere eseguito dall'host, per velocizzare.

Si noti che i comandi suddetti, nel caso li si voglia eseguire sull'host, possono necessitare dei permessi di amministratore, nel qual caso è sufficiente il prefisso sudo.

17.9 Copiare nella flash SPI l'immagine ramdisk da u-boot

Copiare l'immagine JFFS2 nella cartella tftpboot

```
host $ cp ramdisk.gz <tftpbootdir>/<target>
```

Resettere il target e entrare nella shell di u-boot.

Download dell'immagine da TFTP e scrittura in flash

```
target # tftp c1180000 kx/ramdisk.gz  
target # sf probe 0  
target # sf erase 400000 3e0000  
target # sf write c1180000 400000 ${filesize}
```

La stessa cosa si deve ottenere con un solo comando, avendolo correttamente definito in CONFIG_EXTRA_ENV_SETTINGS.

```
target # run upd_sf_rd
```

17.10 Boot da flash SPI con RAM Disk

Esempio di bootargs:

```
root=/dev/ram0 rw initrd=0xc1180000,18M console=ttyS2,115200n8 mtdparts=davinci_nand.1:128k(u-boot_env),128k(ubl),256k(spare),512k(u-boot),4m(kernel),235m(rootfs),16m(archive),235m(rootfs_a),-(rest)
```

Esempio di bootcmd (il kernel viene preso dalla partizione "kernel" della flash SPI):

```
sf probe 0  
sf read c0700000 b0000 350000  
sf read c1180000 400000 3e0000  
bootm c0700000
```

La stessa cosa si deve ottenere con un solo comando, avendolo correttamente definito in CONFIG_EXTRA_ENV_SETTINGS.

```
target # run boot_sf
```

17.11 Compilare e installare le utility MTD

Si parte ovviamente da un target file system sotto forma di directory nell'host.

Per creare una imagine di questo file system adatta a essere scritta in flash occorrono alcune delle MTD-Utilities, un insieme di tools che permettono al kernel di agire sul sottosistema MTD, il quale permette di eseguire operazioni sui dispositivi Flash.

*mkfs.jffs2 – creates JFFS2 images
mkfs.ubifs – creates UBIFS images
ubinize – creates UBI images
ubinfo - provides information about UBI devices and volumes
ubiattach - attaches MTD devices to UBI and creates corresponding UBI devices
ubidetach - the opposite to what ubiattach does
ubimkvol - creates UBI volumes on UBI devices
ubirmvol - removes UBI volumes from UBI devices
ubiupdatevol - updates UBI volumes
ubicrc32 - calculates CRC-32 checksum of a file with the same initial seed as UBI would use;
ubinize - generates UBI images
ubiformat - formats empty flash, erases flash and preserves erase counters, flashes UBI images
mtdinfo - reports information about MTD devices found in the system*

Le MTD-Utilities possono essere compilate per l'host e usate sull'host (preferibile), oppure cross-compilate per il target e usate sul target.

In questo paragrafo è descritta la procedura da seguire in entrambe i casi.

Creare una cartella <mtd> di lavoro dedicata con una sottocartella dove installare i tools.

```
host $ mkdir -p <tools>/mtd/install  
host $ mkdir -p <tools>/mtd_arm/install  
host $ cd <tools>/mtd  
host $ cd <tools>/mtd_arm
```

(<mtd> per compilazione x86)
(<mtd> per cross-compilazione)
(<mtd> per compilazione x86)
(<mtd> per cross-compilazione)

Ottenerne i sorgenti dei seguenti pacchetti:

zlib (da <http://zlib.net/>)

```
host$ wget http://zlib.net/zlib-1.2-6.tar.gz  
host$ wget http://zlib.net/zlib-1.2-8.tar.gz
```

lzo (da <http://www.oberhumer.com/opensource/lzo/download/>)

```
host$ wget http://www.oberhumer.com/opensource/lzo/download/lzo-2.06.tar.gz
```

e2fsprogs (da <http://e2fsprogs.sourceforge.net/>)

mtd-utils (da <http://git.infradead.org/mtd-utils.git>)

Copiarli in <mtd>, e qui estrarli

```
host $ tar zxf zlib-x.x.x.tar.gz  
host $ tar zxf lzo-x.xx.tar.gz  
host $ tar zxf e2fsprogs-x.xx.x.tar.gz  
host $ tar zxf mtd-utils-xxxxxx.tar.gz
```

Configurare zlib

```
host $ cd zlib-x.x.x  
host $ ./configure --prefix=/usr
```

Solo se si vuole cross-compilare, editare il Makefile e inserire

*CROSS = <armtoolch>-
CC = \$(CROSS)gcc
LDSHARED = \$(CROSS)gcc -shared ...*

CPP = \$(CROSS)gcc -E
AR = \$(CROSS)ar rc
RANLIB = \$(CROSS)ranlib

```
host $ make  
host $ make install DESTDIR=<mtd>/install
```

Compilare e installare lzo

```
host $ cd lzo-x.xx  
host $ ./configure --build=i686-pc-linux --prefix=/usr  
host $ ./configure --host=<armtoolch> --prefix=/usr  
host $ make  
host $ make install DESTDIR=<mtd>/install
```

(per compilazione x86)
(per cross-compilazione)

Compilare e installare e2fsprogs

```
host $ cd e2fsprogs-x.xx.x  
host $ ./configure --build=i686-pc-linux --prefix=/usr  
host $ ./configure --host=<armtoolch> --prefix=/usr  
host $ make  
host $ make install DESTDIR=<mtd>/install  
host $ mkdir .../install/usr/include/uuid  
host $ cp lib/uuid/uuid.h .../install/usr/include/uuid  
host $ cp lib/libuuid.a .../installdir/usr/lib
```

(per compilazione x86)
(per cross-compilazione)

Configurare mtd-utils (a luglio 2013 siamo alla versione 1.5.0)

```
host $ cd mtd-utils-xxxxxx
```

Editare il Makefile e inserire

```
ZLIBCPPFLAGS = -I<mtd>/install/usr/include  
LZOCPPFLAGS = -I<mtd>/install/usr/include/lzo  
ZLIBLDFLAGS = -L<mtd>/install/usr/lib  
LZOLDFLAGS = -L<mtd>/install/usr/lib  
WITHOUT_XATTR = 1  
CROSS=<armtoolch>-  
LDFLAGS += -L<mtd>/install/usr/lib  
LDFLAGS += -L<mtd>/install/usr/lib  
CFLAGS += -O2 -g -L$(ZLIBLDFLAGS) -L$(LZOLDFLAGS)  
CPPFLAGS += -I./include -I<mtd>/install/usr/include -I$(BUILDDIR)/include -I./ubi-utils/include  
$(ZLIBCPPFLAGS) $(LZOCPPLFLAGS)
```

(per cross-compilazione)

Compilare e installare

```
host $ make  
host $ make install DESTDIR=<mtd>/install
```

I tools eseguibili generati si trovano nella sottocartella install/usr/sbin.

Copiare i tools che interessano nel file system (di solito sull'host bastano quelli per creare immagini UBI, ecc., mentre sul target si potrebbe anche servire di cancellare la flash, ecc.)

```
host $ cd <mtd>/install/usr/sbin/  
host $ sudo cp mkfs.jffs2 sumtool mkfs.ubifs ubinize /usr/sbin  
host $ cp mkfs.jffs2 sumtool mkfs.ubifs ubinize <rootpath>/usr/sbin  
host $ cp flash* <rootpath>/usr/sbin/  
host $ cp ubi* <rootpath>/usr/sbin/  
host $ cp mtd* <rootpath>/usr/sbin/  
host $ cp jffs2* <rootpath>/usr/sbin/  
host $ cp nand* <rootpath>/usr/sbin/  
host $ cp mkfs.ubifs <rootpath>/usr/sbin/  
host $ cp mkfs.jffs2 <rootpath>/usr/sbin/  
host $ cp sumtool ubinize <rootpath>/usr/sbin/
```

Qui sopra, nel caso cross-compilazione, ovviamente potremmo anche usare <rootfs> invece che <rootpath>

17.12 Preparare la directory del rootfs

Creare il target filesystem per la board kx significa partire da una copia del rootfs di base fornito nel SDK, assegnando la versione xx.xx (inizialmente 01.00)

L'SDK 5.07 include il file archivio *arago-base-tisdk-image-am180x-evm.tar.gz* (43MB)

```
host $ cd /home/<user>/workdir/kx
host $ mkdir -p rootfs-xx.xx
host $ cd rootfs-xx.xx
host $ cp <sdk>/filesystem/arago-base-tisdk-image-am180x-evm.tar.gz .
host $ sudo tar zxf arago-base-tisdk-image-am180x-evm.tar.gz
host $ sudo rm arago-base-tisdk-image-am180x-evm.tar.gz
host $ cd ..
host $ sudo chown -R <user>:<user> rootfs-xx.xx
```

Le patch sono create modificando il <rootfs>, poi compresse

```
host $ cd <rootfs>
host $ tar ..czf rootfs-qpatches-xx.xx.tar.gz ..patches
e infine archiviate sul Server in
```

<fwrepo>/rootfs/

Le patch sono semplicemente i file aggiunti e/o modificati del file system, organizzati nella stessa struttura del <rootfs>, che vanno compresse e archiviate sul Server in

<fwrepo>/rootfs/

Con un nome del tipo *rootfs-kx-<nnnn>-<patchname>.tar*, dove <nnnn> è un numero sequenziale e <patchname> una descrizione sintetica.

Applicare una patch significa semplicemente copiarla sul <rootfs>.

Attenzione: la patch N-esima si deve applicare sull'immagine N-1 esima del <rootfs>

A prescindere dalle patch specifiche del rootfs, i moduli del kernel presenti in <rootfs>/lib/modules e le immagini del kernel e di u-boot presenti in <rootfs>/boot devono essere aggiornati.

Quindi, dopo avere ricompilato u-boot, kernel e moduli kernel,

```
host $ cd <linux>
host $ cp arch/arm/boot/ulimage <rootfs>/boot/ulimage-#.##-xx.xx-pppp
host $ make ARCH=arm INSTALL_MOD_PATH=<rootfs> modules_install
host $ cd <uboot>
host $ cp u-boot.bin <rootfs>/boot/u-boot-#.##-xx.xx-pppp.bin
```

Provvedere a sistemare i link simbolici

```
host $ cd <rootfs>/boot
host $ rm ulimage
host $ ln -s ulimage-#.##-xx.xx-pppp ulimage
host $ rm u-boot.bin
host $ ln -s u-boot-#.##-xx.xx-pppp.bin u-boot.bin
```

in maniera che alla fine si abbia qualcosa tipo

```
host $ ls -la <rootfs>/lib/modules
drwxrwsr-x 3 root root 4096 May 21 18:33 2.6.37-01.00-0014
host $ ls -la <rootfs>/boot
-rw-r-xr-x 1 root root 403920 May 21 18:33 u-boot-2011.12-01.00-0002.bin
lrwxrwxrwx 1 root root    29 May 21 18:33 u-boot.bin -> u-boot-2011.12-01.00-0002.bin
lrwxrwxrwx 1 root root    24 May 21 18:33 ulimage -> ulimage-2.6.37-01.00-0014
-rw-r-xr-x 1 root root 2387156 May 21 18:33 ulimage-2.6.37-01.00-0014
```

Completata la preparazione della directory rootfs, si passa a creare l'immagine UBIFS (o JFFS2).

17.13 JFFS2

http://processors.wiki.ti.com/index.php/Create_a_JFFS2_Target_Image

JFFS2 (Journaled Flash File System v2) è un filesystem per dispositivi Flash. Diversamente dai file system residenti in flash, ma che vengono copiati in ram (p.es. ramdisk, NFS), il JFFS2 resta in flash, in modo tale che le modifiche apportate tra un reboot e l'altro restano permanenti. JFFS2 supporta la compressione, il wear leveling, la protezione contro power-failure, gli hard-link. Per flash di tipo NAND pare che sia preferibile YAFFS, ma siccome il KXDB revA prevede la possibilità di utilizzare sia NOR che FLASH, al momento decidiamo di basare tutto su JFFS2.

Per abilitare il supporto JFFS2 nel kernel:

Per abilitare il supporto JFFS2 nel kernel:

Select

FileSystems->Miscellaneous Filesystems->JFFS2 write-buffering support

17.14 Creare l'immagine JFFS2

Si assume che le MTD-utilities siano installate nell'host e che il file system sia stato preparato nella cartella <rootfs>.

Creare l'immagine JFFS2 di <rootfs> usando l'utilità MTD mkfs.jffs2

```
host $ mkfs.jffs2 -r <rootfs> -o rootfs.jffs2 -e 128 -n -p
```

-r specifica la directory di partenza

-e specifica lo erase block size (in KiB)

-n non aggiunge dei cleanmarker sui settori vuoti

-p esegue il padding a 0xff sul completamento dell'ultimo settore

Aggiungere i nodi dei blocchi cancellati (facoltativo ma consigliato nel caso NAND)

```
host $ sumtool -i rootfs.jffs2 -o rootfs-sum.jffs2 -e 128 -n -p
```

-i specifica il file di input

-o specifica il file di output

-e specifica lo erase block size (in KiB)

-n non aggiunge dei cleanmarker sui settori vuoti

-p esegue il padding a 0xff sul completamento dell'ultimo settore

A questo punto bisogna copiare l'immagine creata nella flash, direttamente da u-boot oppure da linux appoggiandosi a un file system intermedio (p.es. ram disk, NFS).

Copiare l'immagine JFFS2 nel target filesystem NFS (per lavorare successivamente con NFS)

```
host $ cp rootfs-sum.jffs2 <rootpath>/home
```

Copiare l'immagine JFFS2 nella cartella TFTP (per lavorare successivamente con u-boot)

```
host $ cp rootfs-sum.jffs2 <tftpbootdir>/<target>
```

17.15 Copiare nella flash l'immagine JFFS2 appoggiandosi a NFS

Copiare l'immagine JFFS2 nel target filesystem NFS

```
host $ cp rootfs-sum.jffs2 <rootpath>/home
```

Fare il boot del target da NFS.

```
target $ cd /home
```

Engineering Specification - confidential

Se non ci sono, creare i nodi MTD sui quali poter poi montare le partizioni. Ogni partizione richiede due nodi, uno di tipo c (mtd#, con MAJOR NUMBER 90 e MINOR NUMBER 2#) e uno di tipo b (mtblockquote#, con MAJOR NUMBER 31 e MINOR NUMBER #).

```
target $ mknod /dev/mtd0 c 90 0
target $ mknod /dev/mtd1 c 90 2
target $ mknod /dev/mtd2 c 90 4
target $ mknod /dev/mtd3 c 90 6
target $ mknod /dev/mtd4 c 90 8
target $ mknod /dev/mtd5 c 90 10
target $ mknod /dev/mtd6 c 90 12
target $ mknod /dev/mtd7 c 90 14
target $ mknod /dev/mtblockquote0 b 31 0
target $ mknod /dev/mtblockquote1 b 31 1
target $ mknod /dev/mtblockquote2 b 31 2
target $ mknod /dev/mtblockquote3 b 31 3
target $ mknod /dev/mtblockquote4 b 31 4
target $ mknod /dev/mtblockquote5 b 31 5
target $ mknod /dev/mtblockquote6 b 31 6
target $ mknod /dev/mtblockquote7 b 31 7
```

Ottobre

```
target $ for i in $(seq 0 7); do mknod /dev/mtd$i c 90 $((i*2)); done
target $ for i in $(seq 0 7); do mknod /dev/mtblockquote$i b 31 $(i); done
```

Ora è possibile copiare l'immagine del rootfs sulla partizione "rootfs" della flash (corrisponde a /dev/mtd5)

```
target $ flash_erase /dev/mtd5 0
target $ nandwrite -p /dev/mtd5 rootfs-sum.jffs2
target $ flashcp -v rootfs-sum.jffs2 /dev/mtd5
target $ dd if=rootfs-sum.jffs2 of=/dev/mtd5
target $ rm rootfs-sum.jffs2
```

(per la NAND)
(per la NOR)
(alternativo per la NOR)
(opzionale)

Ora possiamo anche verificare il contenuto della flash montando il rootfs su una cartella

```
target $ mkdir -p tmp
target $ mount -t jffs2 /dev/mtblockquotex /tmp
target $ ls -al tmp
....(si deve vedere il contenuto del rootfs)
target $ umount tmp
```

(se già non esiste)

17.16 Copiare nella flash l'immagine JFFS2 da u-boot

Attenzione: questa procedura presenta una limitazione quando l'immagine che si deve scrivere in flash ha dimensioni superiori alla RAM disponibile (128MB nel nostro caso). Per questo motivo si preferisce gestire l'aggiornamento della flash utilizzando una SD card.

Copiare l'immagine JFFS2 nella cartella tftpboot

```
host $ cp rootfs-sum.jffs2 <tftpbootdir>/<target>
```

Resetare il target e entrare nella shell di u-boot.

Verificare che in u-boot le partizioni MTD siano ben definite

```
target # mtddparts
device nand0 <davinci_nand.1>, # parts = 8
#: name      size        offset      mask_flags
0: u-boot_env 0x00020000 0x00000000 0          (<- defined but not used)
1: ubl       0x00020000 0x00020000 0          (<- ubl_AM1808_NAND.bin)
2: ubl-backup 0x00020000 0x00040000 0          (<- other copies of ubl_AM1808_NAND.bin)
3: spare     0x00060000 0x00060000 0
4: u-boot    0x00100000 0x000c0000 0
5: kernel   0x00640000 0x001c0000 0
6: rootfs   0x0f800000 0x00800000 0
7: archive  0x10000000 0x10000000 0
```

Download dell'immagine da TFTP e scrittura in flash

```
target # tftp c20000000 kx/rootfs-sum.jffs2
target # nand erase.part rootfs
target # nand write.e c20000000 rootfs ${filesize}
```

La stessa cosa si deve ottenere con un solo comando, avendolo correttamente definito in CONFIG_EXTRA_ENV_SETTINGS.

```
target # run upd_nand_jffs2
```

17.17 Boot da NAND con JFFS2

Esempio di bootargs:

```
noinitrd root=/dev/mtdblockx rw rootfstype=jffs2 rootwait=1 console=ttyS2,115200n8  
ip=192.168.0.149:192.168.0.142::255.255.255.0::eth0:off panic=1 mtdparts=davinci_nand.1:128k(u-  
boot_env),128k(ubl),256k(spare),512k(u-boot),4m(kernel),235m(rootfs),16m(archive),235m(rootfs_a),-(rest) mem=128M  
mac_addr=80:81:82:83:84:86 sn=2012010001
```

Esempio di bootcmd (il kernel viene preso dalla partizione “kernel” della NAND):

```
nand read c0700000 kernel 400000; bootm c0700000
```

La stessa cosa si deve ottenere con un solo comando, avendolo correttamente definito in CONFIG_EXTRA_ENV_SETTINGS.

```
target # run boot_nand_jffs2
```

17.18 UBIFS/UBI

UBIFS (Unsorted Block Images File System) rappresenta l’evoluzione di JFFS2 (è molto più veloce) Mentre JFFS2 lavora sopra MTD, UBIFS lavora sopra UBI che lavora sopra MTD.

Il sottosistema UBI introduce la distinzione di volumi all’interno di device UBI.

UBI è un “volume manager” e mappa physical erase blocks (PEB) in logical erase blocks (LEB).

L’immagine UBIFS è generata col comando mkfs.ubifs mentre l’immagine UBI si ottiene dall’immagine UBIFS col comando ubinize. Una immagine UBIFS può essere scritta su un volume UBI preesistente col comando ubiupdatevol, ma solo un’immagine UBI può essere scritta direttamente su una partizione MTD, ad esempio da u-boot.

Di default il KX monta un UBIFS root file system.

Per abilitare il supporto UBIFS nel kernel:

Select

FileSystems->Miscellaneous Filesystems->UBIFS file system support
DeviceDrivers->MTD Support->Enable UBI
DeviceDrivers->MTD Support->NAND Device Support->Support NAND on DaVinci SoC

Deselect

DeviceDrivers->MTD Support->NAND Device Support->Verify NAND page writes

E’ possibile anche emulare MTD sopra un volume UBI, magari per farci lavorare sopra JFFS2. Per fare ciò occorre abilitare *gluebi* selezionando nel menuconfig “Emulate MTD devices”

17.19 Creare un volume UBI da NFS

La creazione dell’immagine UBIFS richiede la conoscenza di alcuni parametri legati alla flash, come

- MTD partition size
- Physical Erase Block (PEB) size
- Minimum I/O unit size

- Sub-page size (per le NAND)

Almeno la prima volta conviene fare qualche verifica su alcuni parametri, creando un volume UBI sul target sfruttando il NFS.

Fare il boot del target da NFS.

```
target $ cd /home
```

Se già non esistono, creiamo un device UBI, con una decina di volumi (anche se adesso ne basterebbe uno solo)

```
target $ mknod /dev/ubi_ctrl c 10 62
target $ mknod /dev/ubi0 c 253 0
target $ for i in $(seq 0 9); do mknod /dev/ubi0_$i c 253 $((i + 1)); done
```

Cancellare la partizione MTD che abbiamo deciso di destinare al rootfs. Sia /dev/mtd5 ("rootfs")

```
target $ flash_erase -q /dev/mtd5 0 0
```

Connettere la partizione MTD al device 0 UBI

```
target $ ubiattach /dev/ubi_ctrl -m 5 -O 2048
UBI: attaching mtd5 to ubi0
UBI: physical eraseblock size: 131072 bytes (128 KiB)
UBI: logical eraseblock size: 126976 bytes
UBI: smallest flash I/O unit: 2048
UBI: sub-page size: 512
UBI: VID header offset: 2048 (aligned 2048)
UBI: data offset: 4096
UBI: empty MTD device detected
UBI: max. sequence number: 0
UBI: create volume table (copy #1)
UBI: create volume table (copy #2)
UBI: attached mtd5 to ubi0
UBI: MTD device name: "rootfs"
UBI: MTD device size: 235 MiB
UBI: number of good PEBs: 1879
UBI: number of bad PEBs: 1
UBI: number of corrupted PEBs: 0
UBI: max. allowed volumes: 128
UBI: wear-leveling threshold: 4096
UBI: number of internal volumes: 1
UBI: number of user volumes: 0
UBI: available PEBs: 1857
UBI: total number of reserved PEBs: 22
UBI: number of PEBs reserved for bad PEB handling: 18
UBI: max/mean erase counter: 0/0
UBI: image sequence number: 1692111298
UBI: background thread "ubi_bgt0d" started, PID 1136
UBI device number 0, total 1879 LEBs (238587904 bytes, 227.5 MiB), available 1857 LEBs (235794432 bytes, 224.9 MiB), LEB size 126976 bytes (124.0 KiB)
```

Segnarsi il numero di byte disponibili per il rootfs presente nell'ultima riga (**235794432**) e il LEB count (**1857**) in quanto serviranno successivamente. Queste due quantità esprimono, in unità differenti, la capacità massima del filesystem al netto dell'overhead UBI.

1857 LEB = 1857*(126976 bytes) = 235794432 bytes = 224.9 MiB

Nota: 1 MiB = 1024 * 1024 bytes, mentre 1 MB = 1024 * 1000 bytes

Nota: total number of reserved PEBs: 22, viene da 1857*(CONFIG_MTD_UBI_BEB_RESERVE/100) + 4 = 18 + 4

dove la percentuale riservata per ogni 100 PEBs è data dalla configurazione del kernel
CONFIG_MTD_UBI_BEB_RESERVE = 1

Volendo si possono anche a leggere le caratteristiche del device 0 UBI con ubinfo

```
root@am180x-evm:~# ubinfo /dev/ubi0
ubi0
Volumes count: 0
Logical eraseblock size: 126976 bytes, 124.0 KiB
```

Engineering Specification - confidential

Total amount of logical eraseblocks:	1879 (238587904 bytes, 227.5 MiB)
Amount of available logical eraseblocks:	1857 (235794432 bytes, 224.9 MiB)
Maximum count of volumes	128
Count of bad physical eraseblocks:	1
Count of reserved physical eraseblocks:	18
Current maximum erase counter value:	1
Minimum input/output unit size:	2048 bytes
Character device major/minor:	252:0

Provare a confrontare con le caratteristiche del MTD con mtdinfo

target \$ mtdinfo -u /dev/mtd5	
mtd5	
Name:	rootfs
Type:	nand
Eraseblock size:	131072 bytes, 128.0 KiB
Amount of eraseblocks:	1880 (246415360 bytes, 235.0 MiB)
Minimum input/output unit size:	2048 bytes
Sub-page size:	512 bytes
OOB size:	64 bytes
Character device major/minor:	90:10
Bad blocks are allowed:	true
Device is writable:	true
Default UBI VID header offset:	512
Default UBI data offset:	2048
Default UBI LEB size:	129024 bytes, 126.0 KiB
Maximum UBI volumes count:	128

Disconnettere

```
target $ ubidetach /dev/ubi_ctrl -m 5
```

17.20 Creare l'immagine UBIFS sul host

Si può decidere di eseguire questo processo sul host oppure sul target.

Qui assumeremo che le MTD-utilities siano installate nell'host e che il file system sia stato preparato nella cartella <rootfs>.

Creare l'immagine UBIFS di <rootfs> usando l'utilità MTD mkfs.ubifs

```
host $ mkfs.ubifs -r <rootfs> -m 2048 -e 126976 -c 1856 -F -x lzo -o rootfs.ubifs
```

L'opzione -m specifica il minimum IO unit size

L'opzione -e specifica il logical erase block (LEB) size (126976 equivale a 124KiB)

L'opzione -c specifica il max LEB count

L'opzione -x specifica il tipo di compressione, LZO (default, più veloce), ZLIB (più efficiente). Se non si usa questa opzione l'immagine del file system sarà più grande (in un caso ho misurato 190MB contro 105MB) ma la CPU avrà meno lavoro da fare.

L'opzione -F (funziona solo con ver linux da 3.0)

Si noti che il max LEB count è stato dato proprio uguale al valore sperimentalmente determinato al paragrafo precedente. In realtà è sufficiente mettere un numero maggiore o uguale a questo.

Creare un UBI ini-file ubinize.cfg (descrittore dei volumi) con il seguente contenuto

```
host $ cat ubinize.cfg
[ubifs]
mode=ubi
image=rootfs.ubifs
vol_id=0
vol_size=235794432
vol_type=dynamic
vol_name=rootfs
vol_flags=autoresize
```

In questo descrittore è definito un solo volume, con id 0.

Si noti che a **vol_size** è stato assegnato il numero determinato al paragrafo precedente.

Creare l'immagine UBI

X Srl,

```
host $ ubinize -o rootfs.ubi -m 2048 -p 128KiB -s 2048 -O 2048 ubinize.cfg
```

L'opzione –m specifica il minimum IO unit size

L'opzione –s specifica il sub-page size

L'opzione –p specifica il physical erase block (PEB) size

L'immagine così ottenuta è pronta per essere copiata sulla partizione MTD dedicata al rootfs.
Lo si può fare direttamente da u-boot oppure da linux appoggiandosi a un file system intermedio
(p.es. ram disk, NFS, SD card).

17.21 Copiare nella flash l'immagine UBI da NFS

Queste procedure sono opzionali in quanto la procedura che verrà seguita in fase di produzione è quella di aggiornare la flash da u-boot tramite TFTP o meglio ancora da SD card.

Appoggiandosi a NFS, installare direttamente l'immagine UBI su MTD (se non ci sono volumi UBI preesistenti in /dev):

Copiare l'immagine UBI nel target filesystem NFS

```
host $ cp rootfs.ubi <rootpath>/home
```

Fare il boot del target da NFS

```
target $ cd /home
```

Cancellare la partizione MTD che abbiamo deciso di destinare al rootfs. Sia /dev/mtd5 (“rootfs”)

```
target $ flash_erase -q /dev/mtd5 0 0
```

Copiare l'immagine UBI nella partizione MTD del “rootfs”

```
target $ ubiformat -q -s 2048 -f rootfs.ubi /dev/mtd5 -O 2048
```

Attenzione: questa operazione potrebbe fallire se non c'è abbastanza memoria libera in RAM

(<http://en.it-usenet.org/thread/18514/1074/#post1040>). Per verificare la RAM free usare il comando

```
target $ free -m
```

Nota: sembra che il problema sia stato risolto nella versione v 3.3.0 del kernel.

Verifica funzionamento:

Collegare il device UBI 0, volume 0, alla partizione MTD del “rootfs”

```
target $ ubiattach /dev/ubi_ctrl -m 5 -O 2048
```

Volendo possiamo leggere le caratteristiche del volume con ubinfo

```
root@am180x-evm:~# ubinfo /dev/ubi0_0
Volume ID: 0 (on ubi0)
Type: dynamic
Alignment: 1
Size: 1857 LEBs (235794432 bytes, 224.9 MiB)
State: OK
Name: rootfs
Character device major/minor: 252:1
```

Appoggiandosi a NFS, installare UBIFS su un volume UBI preesistente (/dev/ubi0_0):

Copiare l'immagine UBIFS nel target filesystem NFS

```
host $ cp rootfs.ubifs <rootpath>/home
```

Fare il boot del target da NFS

```
target $ cd /home
```

Aggiornare il volume UBI con l'immagine UBIFS

```
target $ ubiupdatevol /dev/ubi0_0 rootfs.ubifs
```

Possiamo sempre verificare il contenuto della flash montando il volume 0 del device UBI 0 su una cartella

```
target $ mkdir -p tmp  
target $ mount -t ubifs /dev/ubi0_0 tmp  
target $ ls -al tmp  
....(si deve vedere il contenuto del rootfs)  
target $ umount tmp
```

(se già non esiste)

17.22 Copiare nella flash l'immagine UBI da u-boot

Attenzione: questa procedura presenta una limitazione quando l'immagine che si deve scrivere in flash ha dimensioni superiori alla RAM disponibile (128MB nel nostro caso). Per questo motivo si preferisce gestire l'aggiornamento della flash utilizzando una SD card.

Copiare l'immagine UBI nella cartella tftpboot

```
host $ cp rootfs.ubi <tftpbootdir>/<target>
```

Resettere il target e entrare nella shell di u-boot.

Verificare che le partizioni MTD siano ben definite

```
target # mtdparts  
device nand0 <davinci_nand.1>, # parts = 8  
#: name size offset mask_flags  
0: u-boot_env 0x00020000 0x00000000 0 (<- defined but not used)  
1: ubl 0x00020000 0x00020000 0 (<- ubl_AM1808_NAND.bin)  
2: ubl-backup 0x00020000 0x00040000 0 (<- other copies of ubl_AM1808_NAND.bin)  
3: spare 0x00060000 0x00060000 0  
4: u-boot 0x00100000 0x000c0000 0  
5: kernel 0x00640000 0x001c0000 0  
6: rootfs 0x0f800000 0x00800000 0  
7: archive 0x10000000 0x10000000 0
```

Download dell'immagine da TFTP e scrittura in flash

```
target # tftp c20000000 kx/rootfs.ubi  
target # nand erase.part rootfs  
target # nand write.trimms c20000000 rootfs ${filesize}
```

La stessa cosa si deve ottenere con un solo comando, avendolo correttamente definito in CONFIG_EXTRA_ENV_SETTINGS.

```
target # run upd_nand_ubi
```

Nota: si noti che l'operazione tftp di U-Boot fallisce se rootfs.ubi ha dimensioni maggiori della RAM disponibile (128MB). In tal caso bisogna trovare in U-Boot il modo di eseguire il trasferimento e la copia del file "a pezzi"

17.23 Copiare nella flash l'immagine UBI da SD card

16) Copiare l'immagine UBI nella SD card di aggiornamento, nella partizione VFAT16 /boot (p1)

```
host $ sudo cp rootfs.ubi /media/boot
```

17) Editare uno script prog.sh tipo

```
#!/bin/sh  
  
function pause(){  
    read -p "$*"  
}  
  
echo Programming UBI rootfs  
echo A certain amount of free memory is required to complete.
```

```
/etc/init.d/kmain stop
pause 'Press [Enter] key to continue ..'
echo Erasing NAND flash rootfs ..
flash_erase -q /dev/mtd5 0 0
echo UBI format NAND flash rootfs ..
ubiformat -y -q -s 2048 -f rootfs.ubi /dev/mtd5 -O 2048
#echo UBI attach NAND flash rootfs partition ..
#ubiaattach /dev/ubi_ctrl -m 5 -O 2048
/etc/init.d/kmain start
sync
```

- 18) Copiare prog.sh nella SD card di aggiornamento, nel settore VFAT16 /boot (p1)

```
host $ sudo cp prog.sh /media/boot
```

- 19) Inseriamo la SD card nel target e riavviamo da SD card. Ciò può essere fatto da u-boot con un solo comando, avendolo correttamente definito in CONFIG_EXTRA_ENV_SETTINGS.

```
target # run boot_mmc_scr
```

- 20) Montiamo la partizione boot (p1) della SD card su una cartella locale e lanciamo lo script

```
target $ mkdir -p boot
target $ mount -t vfat /dev/mmcblk0p1 boot
target $ cd boot
target $ ./prog.sh
```

17.24 Boot da NAND con UBIFS

Esempio di bootargs:

```
noinitrd rw ubi.mtd=5,2048 root=ubi0:rootfs rootfstype=ubifs chk_data_crc rootwait=1 console=ttyS2,115200n8
ip=192.168.0.149:192.168.0.142::255.255.255.0::eth0:off panic=1 mtdparts=davinci_nand.1:128k(u-
boot_env),128k(ubl),256k(spare),512k(u-boot),4m(kernel),235m(rootfs),16m(archive),235m(rootfs_a),-(rest) mem=128M
```

Esempio-1 di bootcmd (il kernel viene preso dalla partizione “kernel” della NAND):

```
nand read c0700000 kernel 4000000; bootm c0700000
```

La stessa cosa si deve ottenere con un solo comando, avendolo correttamente definito in CONFIG_EXTRA_ENV_SETTINGS.

```
target # run boot_nand_ubifs
```

Esempio-2 di bootcmd (il kernel viene preso, montando il volume UBI, in <rootfs>/boot):

```
ubi part nand0,5 2048; ubifsmount rootfs; ubifsload c0700000 /boot/uImage; ubifsumount; bootm c0700000
```

La stessa cosa si deve ottenere con un solo comando, avendolo correttamente definito in CONFIG_EXTRA_ENV_SETTINGS.

```
target # run boot_nand_ubifs_embk
```

Quando il boot è avvenuto possiamo vedere quanto spazio occupa il file system UBI in NAND

```
target # df -h
Filesystem      Size   Used Available Use% Mounted on
ubi0:rootfs    206.0M  83.0M   123.0M  40% /
none            61.0M   56.0K   61.0M   0% /dev
tmpfs           16.0M  456.0K   15.6M   3% /var/volatile
tmpfs           61.0M     0   61.0M   0% /dev/shm
tmpfs           16.0M     0   16.0M   0% /media/ram
```

17.25 Creare un volume UBI per archivio test/pazienti

Finora abbiamo preparato il volume UBI per il rootfs.

Engineering Specification - confidential

In questo paragrafo vogliamo preparare un volume UBI per la cartella di archivio test/pazienti del KX. Ripercorriamo tutto il procedimento visto prima.

Per prima cosa dobbiamo identificare i parametri da usare in ubinize.

Da NFS cancelliamo la partizione MTD che abbiamo deciso di destinare all'archivio. Sia /dev/mtd6 ("archive")

```
target $ flash_erase -q /dev/mtd6 0 0
```

Connettere la partizione MTD al device 1 UBI

```
target $ ubiattach /dev/ubi_ctrl -m 6 -O 2048
UBI: attaching mtd6 to ubi1
UBI: physical eraseblock size: 131072 bytes (128 KiB)
UBI: logical eraseblock size: 126976 bytes
UBI: smallest flash I/O unit: 2048
UBI: sub-page size: 512
UBI: VID header offset: 2048 (aligned 2048)
UBI: data offset: 4096
UBI: empty MTD device detected
UBI: max. sequence number: 0
UBI: create volume table (copy #1)
UBI: create volume table (copy #2)
UBI: attached mtd6 to ubi1
UBI: MTD device name: "archive"
UBI: MTD device size: 256 MiB
UBI: number of good PEBs: 2048
UBI: number of bad PEBs: 0
UBI: number of corrupted PEBs: 0
UBI: max. allowed volumes: 128
UBI: wear-leveling threshold: 4096
UBI: number of internal volumes: 1
UBI: number of user volumes: 0
UBI: available PEBs: 2024
UBI: total number of reserved PEBs: 24
UBI: number of PEBs reserved for bad PEB handling: 20
UBI: max/mean erase counter: 0/0
UBI: image sequence number: -1552425479
UBI: background thread "ubi_bgt1d" started, PID 1333
UBI device number 1, total 2048 LEBs (260046848 bytes, 248.0 MiB), available 2024 LEBs (256999424 bytes, 245.1 MiB), LEB size 126976 bytes (124.0 KiB)
```

Creare un UBI ini-file ubinize.cfg

```
host $ cat ubinize_archive.cfg
[ubifs]
mode=ubi
image=archive.ubifs
vol_id=0
vol_size=256999424
vol_type=dynamic
vol_name=archive
vol_flags=autoresize
```

Creare l'immagine UBIFS della cartella dell'archivio <archive> usando l'utilità MTD mkfs.ubifs

```
host $ mkfs.ubifs -r archive -m 2048 -e 126976 -c 2024 -F -x lzo -o archive.ubifs; \
host $ ubinize -o archive.ubi -m 2048 -p 128KiB -s 2048 -O 2048 ubinize_archive.cfg; \
```

Copiare l'immagine UBI nella cartella tftpboot

```
host $ cp archive.ubi <tftpbootdir>/<target>
```

Resetta il target e entrare nella shell di u-boot.

Verificare che le partizioni MTD siano ben definite

```
target # mtdparts
device nand0 <davinci_nand.1>, # parts = 8
#: name          size          offset        mask_flags
0: u-boot_env   0x00020000  0x00000000  0          (<- defined but not used)
1: ubl          0x00020000  0x00020000  0          (<- ubl_AM1808_NAND.bin)
2: ubl-backup   0x00020000  0x00040000  0          (<- other copies of ubl_AM1808_NAND.bin)
3: spare         0x00060000  0x00060000  0
4: u-boot        0x00100000  0x000c0000  0
5: kernel        0x00640000  0x001c0000  0
6: rootfs        0x0f800000  0x00800000  0
7: archive       0x10000000  0x10000000  0
```

Download dell'immagine da TFTP e scrittura in flash

```
target # run upd_nand_archive
```

Reboot del target.

Montiamo il volume UBI su una cartella del filesystem

```
target $ ubiattach /dev/ubi_ctrl -m 6 -O 2048  
target $ mount -t ubifs ubi0:archive /mnt/archive
```

Per fare in modo che queste operazioni vengano eseguite automaticamente al boot, aggiungere il seguente comando nel file <rootfs>/etc/init.d/mountall.sh

```
ubiattach -m 6 -d 1 -O 2048 && mount -t ubifs ubi1:archive /mnt/archive
```

Nota: Il motivo per cui si è scelto di utilizzare ubi1 invece che ubi0 è che /dev/ubi0 deve essere associato al rootfs.

17.26 Creare SD card per aggiornamento FW

Bisogna procurarsi una SD Card formattata VFAT. A volte in commercio se ne trovano formattate EXFAT, in tal caso occorre riformattarla VFAT in questo modo

21) Inserire la SD card nel PC host

```
host $ dmesg  
host $ mkfs.vfat -F 32 -v /dev/sdb1      (<- per verificare nodo della card inserita, sia /dev/sdb1)  
host $ mount /dev/sdb1 /media/sd        (<- monto la card su una cartella preesistente /media/sd)  
host $ mkdir -p /media/sd/_X (<- creo la directory che conterrà i files di aggiornamento)  
host $ cp <gen-kx-updfile>/hdr-vrs-x.x/Xkxfw* /media/sd/_X          (<- copio i files)  
host $ umount /media/sd
```

22) Disinserire la SD card dal PC e inserirla nello slot dell'unità KX che si intende aggiornare

23) Accendere l'unità

24) Fermare il boot e da shell di U-Boot digitare il comando *kxfwupd c8000000*

```
target $ kxfwupd c8000000  
kxfwupd: kx firmware update  
* memory buffer at 0xc8000000  
  
Looking for firmware update file:  
  (mmc 0:1)/_X/Xkxfw.bin  
Current KX FW version: 0.1  
Header version 1.0, New FW version 0.1  
Header flags 0x000000ff  
Total update size: 199733248 bytes  
Checking update data for 'pre-script'  
* off: 4096  
* size: 88  
* checksum: 0x9647b25c  
  --- Reading chunk 0, clen 88, file _X/Xkxfw.bin, foff 4096,flen 88  
* checksum: 0x9647b25c == 0x9647b25c? yes!  
Checking update data for 'ubl-nand'  
* off: 8192  
* size: 12528  
* checksum: 0x8c30f785  
  --- Reading chunk 0, clen 12528, file _X/Xkxfw.bin, foff 8192,flen 12528  
* checksum: 0x8c30f785 == 0x8c30f785? yes!  
Checking update data for 'ubl-spi'  
* off: 24576  
* size: 8136  
* checksum: 0x232d9124  
  --- Reading chunk 0, clen 8136, file _X/Xkxfw.bin, foff 24576,flen 8136  
* checksum: 0x232d9124 == 0x232d9124? yes!  
Checking update data for 'uboot'  
* off: 32768  
* size: 412144  
* checksum: 0xe5c37b8d  
  --- Reading chunk 0, clen 412144, file _X/Xkxfw.bin, foff 32768,flen 412144  
* checksum: 0xe5c37b8d == 0xe5c37b8d? yes!  
Checking update data for 'kernel'  
* off: 446464  
* size: 2198696  
* checksum: 0x6c1859d8  
  --- Reading chunk 0, clen 2198696, file _X/Xkxfw.bin, foff 446464,flen 2198696  
* checksum: 0x6c1859d8 == 0x6c1859d8? yes!
```

Engineering Specification - confidential

```
Checking update data for 'rootfs'
* off: 2646016
* size: 191627264
* checksum: 0xda59e646
--- Reading chunk 0, clen 8388608, file _X/Xkxfw.bin, foff 2646016,flen 5742592
----- Reading second part of chunk 0, file _X/Xkxfw.bin.001, flen 2646016
--- Reading chunk 1, clen 8388608, file _X/Xkxfw.bin.001, foff 2646016,flen 5742592
----- Reading second part of chunk 1, file _X/Xkxfw.bin.002, flen 2646016
...
----- Reading second part of chunk 21, file _X/Xkxfw.bin.022, flen 2646016
--- Reading chunk 22, clen 7077888, file _X/Xkxfw.bin.022, foff 2646016,flen 5742592
----- Reading second part of chunk 22, file _X/Xkxfw.bin.023, flen 1335296
* checksum: 0xda59e646 == 0xda59e646? yes!
Checking update data for 'archive'
* off: 194273280
* size: 2097152
* checksum: 0xeda4f4dd
--- Reading chunk 0, clen 2097152, file _X/Xkxfw.bin.023, foff 1335296,flen 2097152
* checksum: 0xeda4f4dd == 0xeda4f4dd? yes!
Checking update data for 'ramdisk'
* off: 196370432
* size: 3361841
* checksum: 0x2fabc852
--- Reading chunk 0, clen 3361841, file _X/Xkxfw.bin.023, foff 3432448,flen 3361841
* checksum: 0x2fabc852 == 0x2fabc852? yes!
Running pre-script...
--- Reading chunk 0, clen 88, file _X/Xkxfw.bin, foff 4096,flen 88
SF: Detected M25P64 with page size 64 KiB, total 8 MiB
Updating UBL on NAND...
Bad block table found at page 262080, version 0x01
Bad block table found at page 262016, version 0x01
Erasing at 0x20000 -- 100% complete.
--- Reading chunk 0, clen 12528, file _X/Xkxfw.bin, foff 8192,flen 12528
Writing to NAND partition ubl (off 131072, part-off 0, len 12528, part-size 131072)...
* 12528 bytes written to NAND partition ubl
Updating UBOOT on NAND...
Erasing at 0xe0000 -- 100% complete.
--- Reading chunk 0, clen 412144, file _X/Xkxfw.bin, foff 32768,flen 412144
Writing to NAND partition u-boot (off 524288, part-off 0, len 412144, part-size 524288)...
* 412144 bytes written to NAND partition u-boot
Updating UBL on SPI MEM...
Updating UBOOT on SPI MEM...
Updating KERNEL on NAND...
Erasing at 0x4e0000 -- 100% complete.
--- Reading chunk 0, clen 2198696, file _X/Xkxfw.bin, foff 446464,flen 2198696
Writing to NAND partition kernel (off 1048576, part-off 0, len 2198696, part-size 4194304)...
* 2198696 bytes written to NAND partition kernel
Updating ROOTFS on NAND...
Erasing at 0xefe0000 -- 100% complete.
--- Reading chunk 0, clen 8388608, file _X/Xkxfw.bin, foff 2646016,flen 5742592
----- Reading second part of chunk 0, file _X/Xkxfw.bin.001, flen 2646016
Writing to NAND partition rootfs (off 5242880, part-off 0, len 8388608, part-size 246415360)...
--- Reading chunk 1, clen 8388608, file _X/Xkxfw.bin.001, foff 2646016,flen 5742592
----- Reading second part of chunk 1, file _X/Xkxfw.bin.002, flen 2646016
...
--- Reading chunk 22, clen 7077888, file _X/Xkxfw.bin.022, foff 2646016,flen 5742592
----- Reading second part of chunk 22, file _X/Xkxfw.bin.023, flen 1335296
Writing to NAND partition rootfs (off 5242880, part-off 184549376, len 7077888, part-size 246415360)...
* 191627264 bytes written to NAND partition rootfs
Updating ARCHIVE on NAND...
Erasing at 0x1efe0000 -- 100% complete.
--- Reading chunk 0, clen 2097152, file _X/Xkxfw.bin.023, foff 1335296,flen 2097152
Writing to NAND partition archive (off 251658240, part-off 0, len 2097152, part-size 268435456)...
* 2097152 bytes written to NAND partition archive
Updating RAMDISK on SPI MEM...
Done!
U-Boot >
```

Nota: nella versione definitiva u-boot lancerà automaticamente il programma di aggiornamento *kxfwupd* se troverà i file di aggiornamento nella cartella *_X* all'interno della SD card, e inoltre non sarà necessario eseguire il reset manualmente al termine dell'aggiornamento stesso

17.27 Patch rootfs-kx-0001-add-kmain

Questa patch:

- aggiunge la cartella dell'applicazione main del KX
- aggiunge uno script di init per il main del KX
- aggiunge le librerie Embedded Qt 4.8.0 per ARM
- sovrascrive la libreria tslib preesistente e relativo file di config
- aggiunge la gestione dello shutdown hw in risposta a un "init 0"
- aggiunge l'applicazione battery charger monitor che viene lanciata da u-boot se AC=1. Per simulare su KXEVB porre SW2.1=ON
- sovrascrive etc/inittab
- sovrascrive console banner "Argao Project" con "X KX"

Questa patch in realtà non è gestita come tale (né con quilt né con diff) ma è gestita a mano a partire dai file forniti da Gianni.

Patched Files:

etc/init.d/halt	gestisce GPIO2 per shutdown
etc/init.d/kmain	script di init per main app
etc/init.d/psplash	script di init per psplash
etc/rc0.d/K15kmain (-> ../init.d/kmain)	
etc/rc1.d/K15kmain (-> ../init.d/kmain)	
etc/rc5.d/K99kmain (-> ../init.d/kmain)	
etc/rc6.d/K97kmain (-> ../init.d/kmain)	
etc/inittab	
etc/ts.conf	
etc/issue	console banner
etc/issue.net	console banner
home/root/kmain/*	
usr/lib/chgmon	applicazione chgmon app
usr/lib/ts/*	
usr/lib/libts-1.0.so.0.0.0	
usr/lib/libts-1.0.so.0 (->libts-1.0.so.0.0.0)	
usr/lib/libts.la	
usr/lib/libts.so	
usr/local/Trolltech/Qt4.8.0arm/*	
usr/bin/psplash (-> psplash-cosmes	banner di start-up
usr/bin/psplash-X	
usr/bin/psplash-write	
usr/bin/chgmon	battery charging monitor

ArchivedAs:

rootfs-kx-0001-add-kmain.tar obsoleto
 rootfs-kx-0005-add-kmain.tar

HowToApply:

```
host $ cd /home/<user>/workdir/kx/temporary
host $ scp <fwserverip>:<fwrepo>/rootfs/rootfs-kx-0005-add-kmain.tar .
host $ sudo tar xf rootfs/rootfs-kx-0005-add-kmain.tar
host $ cp -Rfa rootfs-kx-0005-add-kmain/* ..../rootfs-xx.xx
```

HowToArchiveRootfs:

```
host $ cd /home/<user>/workdir/kx/
host $ tar czf rootfs-xx.xx-0002.tar.gz rootfs-xx.xx/
host $ scp rootfs-xx.xx-0002.tar.gz <fwserverip>:<fwrepo>/rootfs/
```

HowToTest:

Reboot del target

```
target $ cd /home/root/kmain  
target $ ./qmlrun -qws
```

Nota:

Nel funzionamento normale il processo *kmain* viene lanciato dallo script <app>/**kmain_initd_script**

17.28 Patch rootfs-kx-0002-add-kboot

Questa patch crea uno script volatile di init per il KX.

P.es. assegna i criteri di accesso ai nodi /dev/ttyUSB[0..4] creati dal driver ftdisio e imposta i baudrate di default da assegnare alle porte virtuali (VCP).

Questo script è volatile nel senso che una volta eseguito si autodistrugge.

Orgdir:

cartella <rootfs> popolata con il contenuto del file
arago-base-tisdk-image-am180x-evm.tar.gz
presente nel SDK 5.03

Patched Files:

```
etc/rcS.d/S75kboot.sh
```

ArchivedAs:

```
rootfs-01.00-0001.tar.gz
```

HowToApply:

```
host $ cd /home/<user>/workdir/kx  
host $ mkdir -p rootfs-xx.xx  
host $ cd rootfs-xx.xx  
host $ cp <sdk>/filesystem/arago-base-tisdk-image-am180x-evm.tar.gz .  
(<rootfs>  
host $ sudo tar zxf arago-base-tisdk-image-am180x-evm.tar.gz  
host $ sudo rm arago-base-tisdk-image-am180x-evm.tar.gz  
host $ cd ..  
host $ sudo chown -R <user>:<user> rootfs-xx.xx  
host $ cd <rootfs>  
host $ quilt push
```

A prescindere dalle patch specifiche del rootfs, i moduli del kernel presenti in <rootfs>/lib/modules e le immagini del kernel e di u-boot presenti in <rootfs>/boot devono essere aggiornati (vedere paragrafo “Preparare la directori del rootfs”).

HowToArchiveRootfs:

```
host $ cd /home/<user>/workdir/kx/  
host $ tar czf rootfs-xx.xx-0001.tar.gz rootfs-xx.xx/  
host $ scp rootfs-xx.xx-0001.tar.gz <fwserverip>:<fwrepo>/rootfs/
```

HowToArchivePatches:

```
host $ cd /home/<user>/workdir/kx/  
todo
```

17.29 Patch rootfs-kx-0003-add-dropbear

Questa patch:

- aggiunge il supporto SSH dropbear al KX

Patched Files:

etc/init.d/dropbear	script di gestione
etc/rc0.d/K10dropbear (-> ../init.d/dropbear)	
etc/rc1.d/K10dropbear (-> ../init.d/dropbear)	
etc/rc2.d/S10dropbear (-> ../init.d/dropbear)	
etc/rc3.d/S10dropbear (-> ../init.d/dropbear)	
etc/rc4.d/S10dropbear (-> ../init.d/dropbear)	
etc/rc5.d/S10dropbear (-> ../init.d/dropbear)	
etc/rc6.d/K10dropbear (-> ../init.d/dropbear)	
etc/dropbear/	cartella per le chiavi rsa e dss
usr/sbin/dbclient (-> dropbearmulti)	
usr/sbin/dropbear (-> dropbearmulti)	
usr/sbin/dropbearconvert (-> dropbearmulti)	
usr/sbin/dropbearkey (-> dropbearmulti)	
usr/sbin/scp (-> dropbearmulti)	
usr/sbin/ssh (-> dropbearmulti)	

ArchivedAs:

rootfs-kx-0003-add-dropbear.tar

HowToApply:

O si cross-compila per ARM oppure si copia da file system TI (strada seguita, cfr. paragrafo relativo a dropbear)

17.30 Patch rootfs-kx-0004-add-antdriver (Provvisorio)

Questa patch:

- aggiunge il supporto driver ANTC7 Receiver al KX

Patched Files:

usr/lib/libant.so.1	(ANT device driver)
usr/lib/libant.so (->libant.so.1)	
usr/lib/libhrm.so.1	(HRM profile library)
usr/lib/libhrm.so (->libhrm.so.1)	
usr/bin/NewANT	(programma di test)

ArchivedAs:

rootfs-kx-0004-add-antdriver.tar

Source:

Progetto EGit in collaborazione con Univ. Tor Vergata (ing. Francesco Zampognaro)

HowToApply:

Scaricare il sorgente da bitbucket come *.tar e ricompilare con
make clean; make; make arm

Oppure da Eclipse clonare in locale tramite EGit.

<https://bitbucket.org/fzamps/ant-hrm>

17.31 Patch rootfs-kx-0004-add-antdriver (Provvisorio)

Questa patch:

- aggiunge il supporto driver ANTC7 Receiver al KX

Patched Files:

todo

ArchivedAs:

ToDo

Source:

ToDo

HowToApply:

ToDo

Ricompile per KX (arm)

```
host $ cd <usermodedrv>/ant-epygenesys/src/common  
host $ ln -snf config-kx.mk config.mk  
host $ make clean  
host $ make  
host $ make install
```

(<- non richiesto in quanto il link punta già al giusto makefile)

Ricompile per host (x86, 32/64b)

```
host $ cd <usermodedrv>/ant-epygenesys/src/common  
host $ ln -snf config-host.mk config.mk  
host $ make clean  
host $ make  
host $ make install
```

HowToTest:

ToDo

17.32 Patch rootfs-kx-000x-add-iwrapdriver (Provvisorio)

Questa patch:

- aggiunge il supporto driver iWRAP al KX

Patched Files:

usr/bin/iwrap	(iWRAP Server daemon)
usr/bin/iwrapclient	(programma di test)
etc/config.xml	(config file)
etc/inittab	

usr/bin/arm-iWrap	(iWRAP Server daemon)
usr/bin/arm-iWrapClient	(programma di test)
etc/iwrap-config.xml	(config file)
etc/inittab	

Source:

Progetto in collaborazione con Univ. Tor Vergata (ing. Francesco Zampognaro)

HowToApply:

Estrarre il pacchetto dei sorgenti in <tools>

```
host $ cd <tools>
host $ unzip iWrap_v1.0.zip
host $ cd iWrap/iWrap
host $ vim sdk_path           (<- editare e specificare la path del sdk)
host $ make arm-iWrap         (compilazione)
host $ cp arm-iWrap <rootfs>/usr/bin
host $ cp iwrap-config.xml <rootfs>/etc
host $ cd ..iWrapClient/
host $ vim sdk_path           (<- editare e specificare la path del sdk)
host $ make arm-iWrapClient   (compilazione)
host $ cp arm-iWrapClient <rootfs>/usr/bin
```

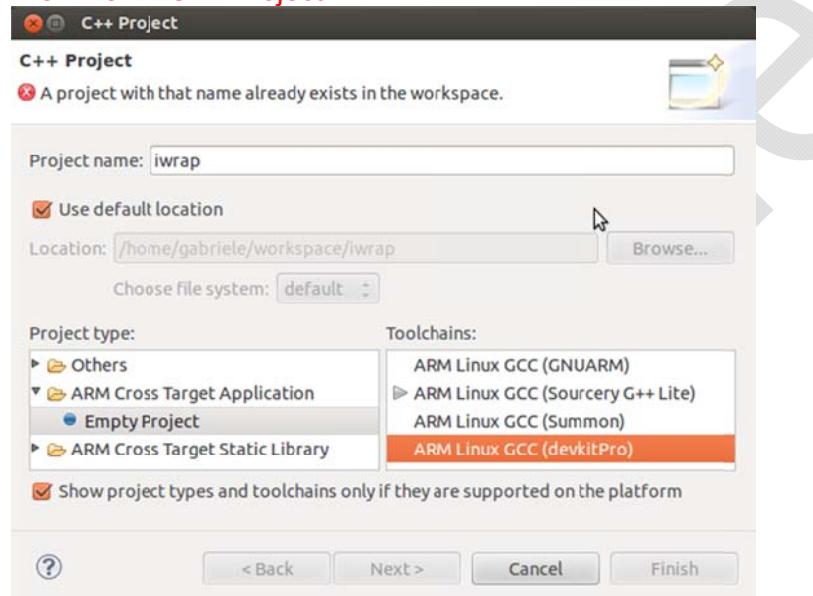
HowToApply:

Estrarre il pacchetto dei sorgenti in <tools>

```
host $ cd <tools>
host $ tar zxf iwrap_v0.2.tar.gz
```

Importare i progetti server (*iwrap*) e client (*iwrapclient*) in Eclipse per ricompilazione

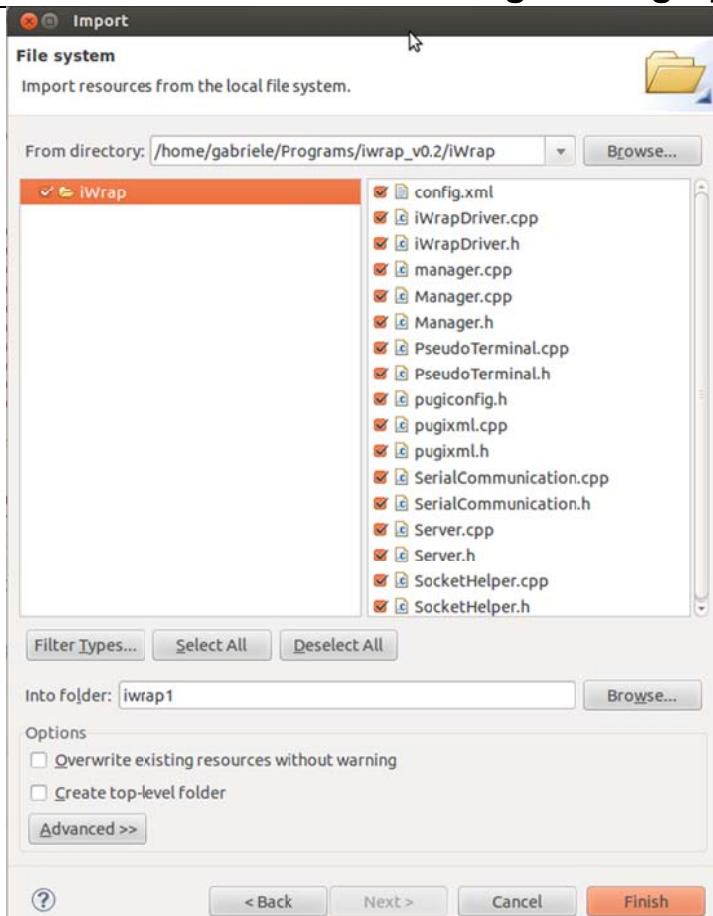
- 1) File->New->C++ Project



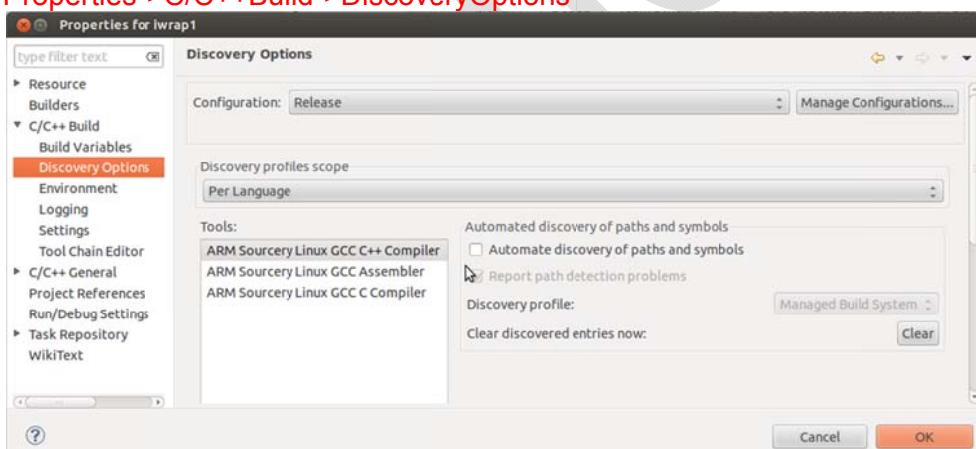
Next->Finish

- 2) Project Explorer-> Right Click sul progetto
Import->General->FileSystem ..Browse to <tools>/iwrap_v0.2/iwrap

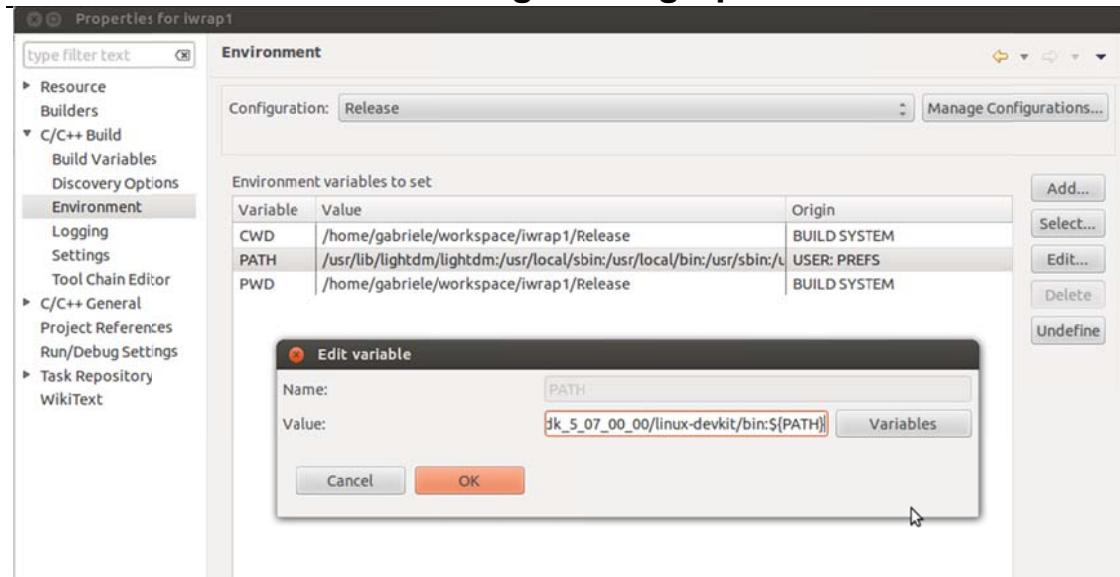
Engineering Specification - confidential



- 3) Project Explorer-> Right Click sul progetto
Properties->C/C++Build->BuildVariables, aggiungere __USE_GNU (to All configurations)
Chiudere Eclipse e riavviarlo
- 4) Project Explorer-> Right Click sul progetto
Properties->C/C++Build->DiscoveryOptions



- 5) Properties->C/C++Build->Environment, aggiungere a PATH (to All configurations)
`/home/gabriele/sdk_5_07_00_00/linux-devkit/bin`



- 6) Properties->C/C++Build->Settings->ToolSettings (set to All configurations)
 Target Processor: arm926ej-s
 Thumb: disabled
 Endianess: Little Endian
 ARM devkitPro Linux GCC <tool> command: *arm-arago-linux-gnueabi-<tool>*
 ARM devkitPro Linux GCC C++ Linker: add *-fexceptions* to **Other flags**
- 7) Properties->C/C++Build->Settings->BuildArtifact (set to All configurations)
 Artifact extension: *None*
- 8) Properties->C/C++General->PathsAndSymbols->Includes->GNU C++ (add to All configurations)
/home/Gabriele/sdk_5_07_00_00/linux-devkit/arm-arago-linux-gnueabi/usr/include
/home/Gabriele/sdk_5_07_00_00/linux-devkit/arm-arago-linux-gnueabi/include/c++/4.5.3
- 9) Properties->C/C++General->PathsAndSymbols->Libraries (add to All configurations)
pthread
util

Ricompilare.

Copiare gli eseguibili sul rootfs

```
host $ cd workspace
host $ cp iwrap/config.xml <rootfs>/etc
host $ cp iwrap/Release/iwrap <rootfs>/usr/bin/
host $ cp iwrap/iwrapclient <rootfs>/usr/bin/
```

Editare <rootfs>/etc/inittab e aggiungere

```
bt:5:respawn:/usr/bin/iwrap /etc/iwrapconfig.xml
```

HowToTest:

Da terminale esterno ssh, lanciare iWrapClient

```
host $ ssh root@<ipaddr>
target $ arm-iWrapClient
```

Lato host è possibile utilizzare un adattatore USB-Bluetooth, p.es. Parani UD-100. Inserire l'adattatore sul PC host

```
host $ hcitool dev          (<- mostra i devices bluetooth presenti del sistema hci<x>)
Devices:
      hci0      00:01:95:14:AB:44
host $ sudo rfcomm connect hci<x> <WT-41_btaddr>      (<- effettua una connessione SPP)
```

A questo punto si possono inviare e ricevere dati tramite */dev/rfcomm0*

Esiste uno script per inviare dati randomici:

```
host $ sudo python bluetoothRW.py /dev/rfcomm<y>      (<- invia e riceve dati dalla seriale)
```

17.33 Patch SpO2

Questa patch:

- aggiunge il supporto Bluetooth Pulse Oximeter, modello NONIN WristOx2 3150 (<http://www.nonin.com/OEMSolutions/WristOx23150-OEM>)

Patched Files:

```
<iwrap>/
<iwrap>/nonin-spo2/*
..
```

(programma di test da host)

Source:

Progetto in collaborazione con Epygenesis (E.Betti)

HowToTest:

Leggiamo l'indirizzo BT e il PIN sul retro del NONIN, nel seguente caso

<SPO2_btaddr> = 00:1C:05:01:45:2A

<SPO2_pin> = 087991



Lato host è possibile utilizzare un adattatore USB-Bluetooth, p.es. Parani UD-100

Inserire l'adattatore sul PC host

```
host $ hcitool dev
Devices:
  hci0      00:01:95:14:AB:44
```

(<- mostra i devices bluetooth presenti del sistema hci-<x>)
 (<- è il Parani)

Accendere e indossare il device SPO2.

Attivare il Bluetooth manager su host, eseguire discovery e pairing utilizzando il PIN.

A questo punto si possono inviare e ricevere dati tramite /dev/rfcomm0

```
host $ sudo rfcomm connect hci-<x> <SPO2_btaddr>
Connected /dev/rfcomm0 to 00:01:95:14:AB:44
Press CTRL-C for hangup
```

Compilare per host programma di test

```
host $ cd <kx-dev>/usermode-drivers/nonin-spo2/wriston2
host $ make
```

Lanciare programma di test

```
host $ sudo ./epywristox2 /dev/rfcomm0
```

A questo punto si dovrebbero leggere in tempo reale i valori

Status1

Status2

Heart Rate

SpO2

Lo stato della batteria del device è

0: battery okay. Ci sono 2 o 3 tacche sul display del device.

1: battery low. 1 tacca sul display del device. Da un momento all'altro il device smetterà di fornire valori validi.

17.34 Patch rootfs-kx-000x-add-poweroff-at-halt (Provvisorio)

Questa patch:

- Spegne l'alimentazione del sistema KX al comando di shell halt

Patched Files on Target:

etc/init.d/halt

Source:

Aggiungere subito prima del comando *halt* il seguente gruppo di comandi

```
#Power off the KX system
mount -t sysfs none /sys
echo "Power off the KX system .."
cd /sys/class/gpio
echo 2 > export
cd gpio2
echo 'out' > direction
echo 0 > value
cd /sys/class/gpio
echo 2 > unexport
```

In pratica viene portata a 0 la linea SPB_HOLD, il che spegne il DC/DC converter di alimentazione del KX

17.35 Patch rootfs-kx-000x-add-reset-at-reboot (Provvisorio)

Questa patch:

- Attiva un hardware system reset del KX al comando di shell reboot

Patched Files:

etc/init.d/reboot

Source:

Aggiungere subito prima del comando *reboot* il seguente gruppo di comandi

```
#Resetting the KX system
mount -t sysfs none /sys
echo "Resetting the KX system .."
cd /sys/class/gpio
echo 145 > export
cd gpio145
echo 'out' > direction
```

```
echo 1 > value  
cd /sys/class/gpio  
echo 145 > unexport
```

In pratica viene portata a 1 la linea SW_RST, il che porta a 0 la linea MRESETn_VIO

17.36 Patch rootfs-kx-000x-add-network-cfg (Provvisorio)

Questa patch:

- Configura la rete nel caso in cui non sia stato U-Boot a farlo tramite stringa bootargs. Questo avviene quando il boot del kernel non richiede la presenza della rete

Patched Files:

etc/network/interfaces

Source:

Aggiungere il seguente gruppo di comandi

```
# Wired or wireless interfaces  
auto eth0  
iface eth0 inet static  
    address 192.168.0.149  
    netmask 255.255.255.0  
    pre-up /bin/grep -v -e "ip=[0-9]\.+.[0-9]\.+.[0-9]\.+.[0-9]\+/" /proc/cmdline > /dev/null
```

17.37 Patch rootfs-kx-000x-add-kxsensor-driver (Provvisorio)

Questa patch:

- Crea un super-driver che legge tutti i sensori presenti sul bus I2C e SPI (ADC) e li presenta all'applicazione in userspace in maniera sincrona, attraverso il device file /dev/kxsensors
- Il driver gestisce sia canali “veloci”, p.es. i canali del ADC (campionati con un'interrupt periodica a 10ms) che canali “lenti”, p.es. sensori di pressione barometrica e umidità relativa ambiente (campionati a 1 s)
- Il driver passa l'array dei dati all'applicazione tramite un buffer circolare di 256 bytes
- Per installare il driver kxsensors è necessario prima installare il driver ad768x

Author: Emiliano Betti

Patched Files on Host:

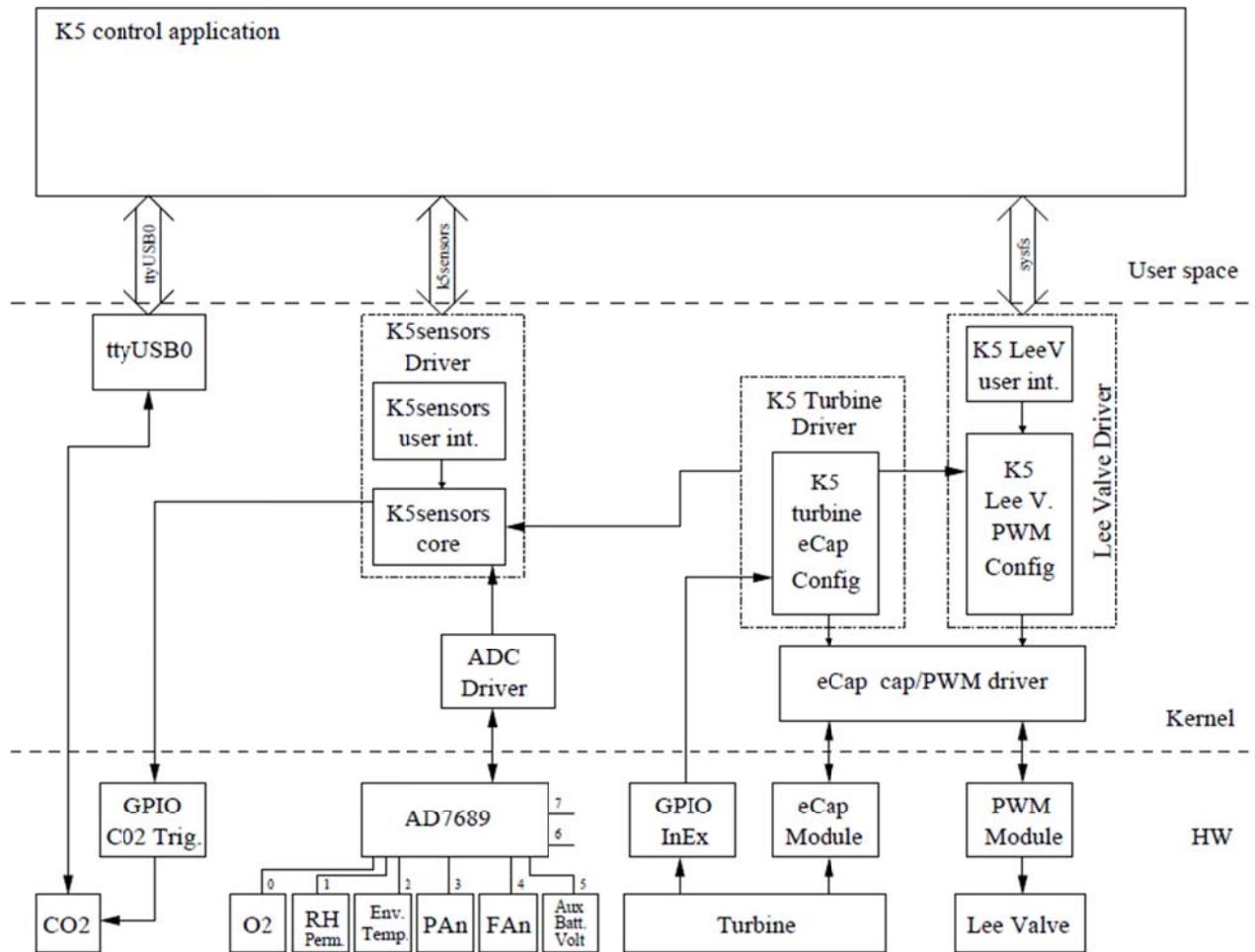
Todo

Patched Files on Target:

Todo

La seguente figura mostra l'attuale collocazione del driver nel contesto dell'architettura software KX

Engineering Specification - confidential



HowToTest:

```
target $ test kxsensors --continue /dev/kxsensors
```

target \$ test_rxsensor
Reading from /dev/kxsensors

```
[ 168.349572] ad768x 0.4: cannot record 8 channels (sampled channels are 6)
```

t14162061423825176000, id 1, st 0x0, adct 3389, adco 48768, adc1 0, adc2 2, adc3 48558, adc4 2704, adc5 53920, adc6 0, adc7 0, adcE 16266, ma 5307, rhtA 2144, alt 2002, pb 989482, intT 240, turbine (0, inex 00000000, io 0)
t1416206142392518991, id 2, st 0x0, adct 3389, adco 48752, adc1 0, adc2 1, adc3 48562, adc4 2702, adc5 53920, adc6 0, adc7 0, adcE 16266, ma 5307, rhtA 2144, alt 2002, pb 989482, intT 240, turbine (0, inex 00000000, io 0)
t1416206142392506000, id 3, st 0x0, adct 3389, adco 48747, adc1 0, adc2 4, adc3 48562, adc4 2701, adc5 53921, adc6 0, adc7 0, adcE 16266, ma 5307, rhtA 2144, alt 2002, pb 989482, intT 240, turbine (0, inex 00000000, io 0)

```
Reading from /dev/kxsensors

Tadc 3509 °Cx100
O2 40792 clk
RH SampL 10328 clk -> 787 mV
Tamb 9810 clk > 519 mV ->
Pan 50096clk -> 3822 mV ->
Fn1 10120clk -> 772 mV
AuxBatt 54840 clk -> 4183 mV
TRHSample 17419 clk -> 1329 mV
RHambi 3264 %x100
TRHambi 2605 %x100
Alt 1543 mx10
Ph 994572 Pa
Tdev 298 °Cx10
```

<i>t</i>	<i>nnnnnnnnnnnnnnnnnnnn</i>	- timestamp in nanosecondi
<i>id</i>	<i>n</i>	- numero progressivo dell'interrupt periodica a 10ms. Nota: se l'incremento è >1 allora sono stati persi dei campioni
<i>st</i>	<i>n</i>	- flag di stato. Riporta eventuali condizioni di errore
<i>adcT</i>	<i>n</i>	- Temperatura interna del ADC in #clk. Al momento non funzionante
<i>adc0</i>	<i>n</i>	- O2 in [#clk]
<i>adc1</i>	<i>n</i>	- RH_sampl, permapure sampler Relative Umidity in [#clk] (sensore analogico SHT21S). Valore aggiornato a 0.5 Hz
<i>adc2</i>	<i>n</i>	- T_amb, Ambient Temperature in [#clk] (sensore analogico TMP37 su cavo turbina)
<i>adc3</i>	<i>n</i>	- PA, Analyzer's pressure in [#clk] (sensore analogico)
<i>adc4</i>	<i>n</i>	- FA, Analyzer's flow in [#clk]
<i>adc5</i>	<i>n</i>	- AUXBATIN, Auxiliary Battery Voltage in [#clk]
<i>adc6</i>	<i>n</i>	- Not Used
<i>adc7</i>	<i>n</i>	- pull-down a GND. Questo canale viene utilizzato per rilevare transizioni spurie indotte da ESD sulla linea CS del bus SPI di collegamento ADC. Infatti questi eventi causano il cambiamento di ordine dei canali nell'array di output del driver.
<i>adcE</i>	<i>n</i>	- Temperature of SHT21S in [#clk] (sensore analogico SHT21S). Valore aggiornato a 0.5 Hz
<i>rha</i>	<i>n</i>	- RH_amb, Ambient Relative Umidity in [%x100] (sensore digitale su I2C, SHT21). Valore aggiornato a 1 Hz
<i>rhaT</i>	<i>n</i>	- Temperature of SHT21 in [°Cx100] (sensore digitale su I2C, SHT21). Valore aggiornato a 1 Hz
<i>alt</i>	<i>n</i>	- Altimeter in [m x10] (sensore digitale su I2C, MPL3115A2)
<i>pb</i>	<i>n</i>	- PB, Barometric Pressure (sensore digitale su I2C, MPL3115A2)
<i>intT</i>	<i>n</i>	- Device Internal Temperature in [°Cx10] (sensore digitale su I2C, MPI 3115A2)

Engineering Specification - confidential

turbine - Volume Turbina

Format: (i y, inex xxxxxxxx, i0 n)

y: numero di interrupts (0..7), cioè numero di impulsi della turbina rilevati negli ultimi 10 ms

xxxxxxx: ogni nibble rappresenta lo stato per tutti gli y impulsi

n: durata in [ns] del primo degli y impulsi

Nota1: Essendo il convertitore ADC a 16 –bit e il reference a 5V, per convertire il valore di un canale ADC da #clk a mV basta moltiplicare per 5000/65536

Esistono dei parametri in lettura o scrittura per interagire col driver

```
target $ cat /sys/class/kxsensors/buffer_overrun
0
target $ cat /sys/class/kxsensors/sampling_errors
0
target $ cat /sys/class/kxsensors/slow_sampling_errors
0
target $ cat /sys/class/kxsensors/missed_samples
0
target $ cat /sys/class/kxsensors/sampling_period_us
10000
target $ cat /sys/class/kxsensors/slow_sampling_period_ms
1000
target $ cat /sys/class/kxsensors/ap_calibration
695          (<- valore che al boot viene impostato dal driver come offset per PAn in maniera che sia uguale a PB)
target $ echo 1 > /sys/class/kxsensors/calibration (<-esegue calibrazione dell'offset per PAn in maniera che sia uguale a PB)
```

Il modulo ad768x ha un parametro che definisce il numero massimo di canali acquisiti (1..8, default: 6)

Nota2: Si noti che adc6 e adc7 hanno valore esattamente 0. In realtà ciò non è dovuto al fatto che il valore letto è 0 mV, ma perché il driver ad768x sta campionando i canali da 0 a 5. Infatti su KX quei canali non sono utilizzati e pertanto è inutile campionarli. Supponendo di voler provare a campionare 8 canali invece che 6, la sequenza dei comandi è la seguente

```
target $ lsmod
Module           Size  Used by
kxsensors        59017  0
ad768x          6433   1 kxsensors
mic3291         3135   0
g_serial        24283   0
mpl3115a2       5087   1 kxsensors
mag3110         17011   0
target $ rmmod kxsensors
target $ rmmod ad768x
target $ modprobe ad768x channels=8
[ 83.200279] ad768x 0.4: Sampling 8 channels
[ 83.207484] ad768x 0.4: found ad7689 (8 channels, 16 bits per word, max 3000000 HZ)
[ 83.220117] ad768x 0.4: device configured: f053
target $ modprobe kxsensors
target $ test_kxsensors /dev/kxsensors
Reading from /dev/kxsensors
t 1416206799591867844, id 1, st 0x0, adcT 3472, adc0 48604, adc1 0, adc2 1, adc3 48631, adc4 28609, adc5 54288, adc6 07, adc7 101, adcE 18146, rba 5110, rbaT 2223, alt 1990, pb 989685, intT 282, turbine (i 0, inex 00000000, i0 0)
t 1416206799601865094, id 2, st 0x0, adcT 3472, adc0 48492, adc1 0, adc2 1, adc3 48626, adc4 28607, adc5 54285, adc6 06, adc7 100, adcE 18146, rba 5110, rbaT 2223, alt 1990, pb 989685, intT 282, turbine (i 0, inex 00000000, i0 0)
t 1416206799611868969, id 3, st 0x0, adcT 3472, adc0 48512, adc1 0, adc2 1, adc3 48634, adc4 28611, adc5 54283, adc6 06, adc7 100, adcE 18146, rba 5110, rbaT 2223, alt 1990, pb 989685, intT 282, turbine (i 0, inex 00000000, i0 0)
```

17.37.1 kxsensors per KX

Quanto detto sopra vale per KX

17.37.2 ksampler per Spartacus

Su Spartacus il campionatore dei sensori si chiama ksampler.

```
target$ test_ksampler /dev/ksampler
[ 55.283505] random: nonblocking pool is initialized
#id  st  adcT  adc0  adc1  adc2  adc3  adc4  adc5  adc6  adc7  adcE  alt  pb  intT  ti  tinex  ti0  pflow  pvol  psv_acc  duty  status
1  0x0  3413  2348  29873  2899  11429  0  0  0  0  -20670  1288045  294  0  0  0  0  0  0  0  0  0  0
2  0x0  3417  2004  29907  2899  11436  0  0  0  0  -20670  1288045  294  0  0  0  10262  -10262  0  0  3
3  0x0  3419  3698  29890  2899  11436  0  0  0  0  -20670  1288045  294  0  0  0  10262  -20524  0  0  1
4  0x0  3415  3092  29904  2899  11432  0  0  0  0  -20670  1288045  294  0  0  0  10262  -30786  0  0  1
```

Engineering Specification - confidential

5	0x0	3423	1985	29878	2899	11436	0	0	0	0	0	-20670	1288045	294	0	0	0	10262	-41048	0	0	1
6	0x0	3419	1946	29863	2900	11441	0	0	0	0	0	-20670	1288045	294	0	0	0	10266	-51314	0	0	1
...																						

t nnnnnnnnnnnnnnnnnnnnnn - timestamp in nanosecondi

id n - numero progressivo dell'interrupt periodica a 10ms. Nota: se l'incremento è >1 allora sono stati persi dei campioni
st n - flag di stato. Riporta eventuali condizioni di errore

adcT n - Temperatura interna del ADC in #clk

adc0 n - O2 in [#clk]

adc1 n - PNT Flow in [#clk] (sensore analogico _____)

adc2 n - Not Used ?

adc3 n - PA, Analyzer's pressure in [#clk] (sensore analogico _____)

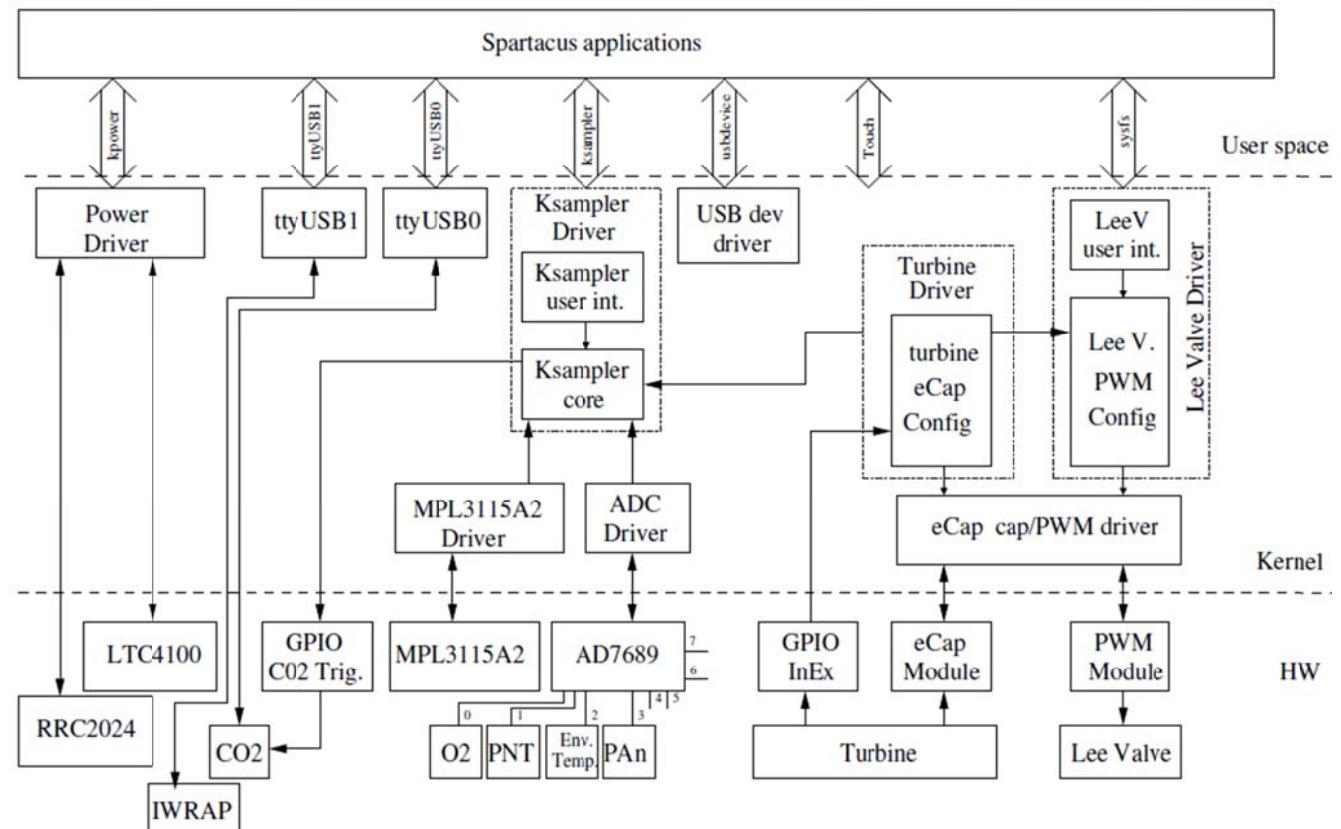
adc4 n - Not Used

adc5 n - Not Used

adc6 n - T_amb, in [#clk] (sensore analogico TMP37 su cavo Turbina Esterna)

adc7 n - pull-down a GND.

La seguente figura mostra l'attuale collocazione del driver nel contesto dell'architettura software XNRG



17.38 Patch rootfs-kx-000x-add-psplash (Provvisorio)

Questa patch:

- Aggiunge al filesystem degli eseguibili che servono a visualizzare il psplash al boot e allo shutdown

Author: Gianni Colarossi

Patched Files:

<rootfs>/home/root/kmain/content/images/kxpsplash.ong

<rootfs>/usr/bin/psplash-X

<rootfs>/usr/bin/psplash-write

17.39 Patch rootfs-kx-000x-add-console-logo (Provvisorio)

Questa patch:

- Aggiunge al filesystem il file che serve a visualizzare sulla console l'immagine testuale al boot

Author: Gianni Colarossi

Patched Files:

```
<rootfs>/etc/issue  
<rootfs>/etc/issue.net
```

17.40 Patch rootfs-kx-000x-add-mpl3115a2

Questa patch:

- Aggiunge al filesystem il file che serve a visualizzare sulla console l'immagine testuale al boot

Author: Emiliano Betti

Patched Files:

```
<rootfs>/etc/issue  
<rootfs>/etc/issue.net
```

```
root@kx:~# cd /sys/bus/i2c/drivers/mpl3115a2/1-0060
root@kx:/sys/bus/i2c/drivers/mpl3115a2/1-0060# ls
alt_offset      name      samples      temp_offset
driver          pb_offset   sea_level_pascal uevent
modalias        power      subsystem
```

18 Fundamentals on flash devices

18.1 NAND

- When it is erased, all bits are set to '1' (you will see 0xff on all bytes in a hexdump)
 - You can change as many bits as you want to '0'
 - You cannot set a bit back to '1' by using a regular write. You have to erase a whole erase block to do so

- The number of erase cycles per block is limited. Once you have reached the limit, some bits will not get back to 0xff
- In the case of the chip in the Neo1973, this is 100000 guaranteed per-block erase cycles.

19 Flash programming utilities

Le utility di programmazione flash disponibili sono di due tipi. Una è basata su CCS e sfrutta la connessione all'host tramite l'emulatore esterno, l'altra è un'applicazione basata su Mono e sfrutta il collegamento seriale.

19.1 Programmazione via seriale

Oltre al CCS, per programmare la flash è disponibile un programma a riga di comando che usa la seriale. Il programma di utility si chiama SFH (Serial Flasher Host). Essendo basato su Mono, SFH può essere eseguito sia sul PC che sull'host. Per lanciare gli eseguibili pre-compilati sul PC deve essere installato Microsoft .Net framework v4

10) Predisporre il target in Serial boot Mode (sulla EVM basta mettere a ON gli switch S7.7 e S7.8)

S7:8	S7:7	S7:6	S7:5
ON	ON	OFF	OFF

Sulla scheda KXDB occorre mettere a ON gli switch SW1.3 e SW1.4

Nota: per aggiornare invece UBL nella NAND sulla scheda KXDB occorre mettere a ON solo lo switch SW1.1

11) Collegare la porta seriale del target (UART2) al PC (p.es., COM1)

12) Scaricare e scompattare i sorgenti sul PC da

http://processors.wiki.ti.com/index.php/Serial_Boot_and_Flash>Loading_Utility_for_OMAP-L138

Notare che l'SDK già contiene una cartella /OMAP-L138_FlashAndBootUtils_x_xx, però in questa repository si possono trovare versioni più recenti

13) Copiare nella cartella

..../OMAP-L138_FlashAndBootUtils_x_xx/OMAP-L138/GNU

i files binari che si intende aggiornare, p.es. l'UBL valido per la flash SPI e U-Boot

-/OMAP-L138_FlashAndBootUtils_x_xx/OMAP-L138/ubl/ ubl_AM1808_SPI_MEM.bin
- u-boot.bin

Nota: per aggiornare invece UBL nella NAND occorrerà utilizzare il file

-/OMAP-L138_FlashAndBootUtils_x_xx/OMAP-L138/ubl/ ubl_AM1808_NAND.bin

14) Aprire una command window nella cartella

..../OMAP-L138_FlashAndBootUtils_x_xx/OMAP-L138/GNU

e inviare i comandi necessari.

15) In generale ci sono tre modi di usare SFH:

Engineering Specification - confidential

```
..\sfh_OMAP-L138.exe -erase  
(cancella tutta la flash)
```

```
..\sfh_OMAP-L138.exe -flash_noubl <binary application file>  
(scrive a 0x0 solo l'immagine applicazione)
```

```
..\sfh_OMAP-L138.exe -flash <UBL binary file> <binary application file>  
(scrive UBL in 0x0 e poi u-boot a un indirizzo predefinito, def: 0x10000)
```

Nel nostro caso:

```
> sfh_OMAP-L138.exe -flash ubl_AM1808_SPI_MEM.bin u-boot.bin -targetType AM1808 -flashType SPI_MEM -p COM1 -baud 115200
```

```
-targetType : Specifies exact target type within OMAP-L138 family  
-flashType : Specifies exact flash type (default SPI_MEM)  
-p <COM PORT NAME> : Allows specifying com port other than default 'COM1' or  
'/dev/ttyS0'.  
-h : Show help text.  
-v : See verbose output from target device  
-baud <BAUD RATE> : Allows specifying baud rate other than default (115200)  
-APPStartAddr : Changes entry point of application (default 0xC1080000)  
-APPLoadAddr : Changes load address of application (default 0xC1080000)  
-APPFlashBlock : Changes the block to flash the image into (only for no_ubl  
mode)
```

Nota: per aggiornare invece UBL nella NAND utilizzeremo invece il comando

```
> sfh_OMAP-L138.exe -flash ubl_AM1808_NAND.bin u-boot.bin -targetType AM1808 -flashType NAND -p COM1 -baud 115200
```

Nota: per ricompilare SFH

```
host $ mkdir -p sfh  
host $ cd sfh  
host $ wget http://sourceforge.net/projects/dvflashutils/files/OMAP-L138/v2.40/OMAP-L138_FlashAndBootUtils_2_40.tar.gz  
host $ tar xvf OMAP-L138_FlashAndBootUtils_2_40.tar.gz  
host $ cd OMAP-L138_FlashAndBootUtils_2_40/OMAP-L138  
host $ make clean  
host $ make DSP_CROSSCOMPILE=/opt/ti/ccsv5/tools/compiler/c6000_7.4.2/bin/ CROSSCOMPILE=/opt/ti/ccsv5/tools/compiler/gcc-arm-none-eabi-4_7-2012q4/bin/arm-none-eabi- DSP_LIB_PATH=/opt/ti/ccsv5/tools/compiler/c6000_7.4.2/lib
```

Ora *sfh_OMAP-L138.exe* si trova nella GNU directory

16) Switch-ON target

Engineering Specification - confidential

```
C:\WINDOWS\system32\cmd.exe -sfh OMAP-L138.exe -flash ubl_AM1808_SPI_MEM.bin u-boot.bin -targetType AM1808 -flashType SPI_MEM -p COM1 -baud 115200
The port 'COM5' does not exist.
>sfh OMAP-L138.exe -flash ubl_AM1808_SPI_MEM.bin u-boot.bin -targetType AM1808 -flashType SPI_MEM -p COM1 -baud 115200

TI Serial Flasher Host Program for OMAP-L138
<C> 2010, Texas Instruments, Inc.
Ver. 1.67

[TYPE] UBL and application image
[UBL] ubl_AM1808_SPI_MEM.bin
[APP IMAGE] u-boot.bin
[TARGET] AM1808
[DEVICE] SPI_MEM

Attempting to connect to device COM1...
Press any key to end this program at any time.

<AIS Parse>: Read magic word 0x41504954.
<AIS Parse>: Waiting for BOOTME... <power on or reset target now>
<AIS Parse>: E001ME received!
<AIS Parse>: Performing Start-Word Sync...
<AIS Parse>: Performing Ping Opcode Sync...
<AIS Parse>: Processing command 0: 0x58535901.
<AIS Parse>: Performing Opcode Sync...
<AIS Parse>: Loading section...
<AIS Parse>: Loaded 9332-Byte section to address 0x80000000.
<AIS Parse>: Processing command 1: 0x58535901.
<AIS Parse>: Performing Opcode Sync...
<AIS Parse>: Loading section...
<AIS Parse>: Loaded 808-Byte section to address 0x80002474.
<AIS Parse>: Processing command 2: 0x58535906.
<AIS Parse>: Performing Opcode Sync...
<AIS Parse>: Performing jump and close...
<AIS Parse>: AIS complete. Jump to address 0x80000000.
<AIS Parse>: Waiting for DONE...
<AIS Parse>: Boot completed successfully.

Waiting for SFT on the OMAP-L138...

Flashing UBL ubl_AM1808_SPI_MEM.bin <8792 bytes> at 0x80000000
100% [██████████] Image data transmitted over UART.

100% [██████████] UBL programming complete

Flashing application u-boot.bin <163500 bytes> at 0x00010000
100% [██████████] Image data transmitted over UART.

92% [██████████-----] Programming application into flash...
```

- 17) Sulla EVM rimettere a OFF gli switch S7.7 e S7.8
- 18) Reboot
- 19) **Nota.** Per un target custom è probabile si debba ricompilare SFH, ad esempio se vi sono differenze nella configurazione della ram DDR, della UART, della SPI o del PLL

19.1 Programmazione via u-boot

http://processors.wiki.ti.com/index.php/Booting_Linux_kernel_using_U-Boot

E' necessario portare in RAM il file da aggiornare. Qui si assume che sia disponibile un server TFTP in rete da dove scaricare il file. E' anche necessario conoscere la mappa del supporto di memoria non volatile

scrittura UBL su flash SPI

```
target # tftp c0700000 kx/ubl_AM1808_SPI_MEM.bin
target # sf probe 0
target # sf erase 0 10000
target # sf write c0700000 0 ${filesize}
```

scrittura u-boot su flash SPI

```
target # tftp c0700000 kx/u-boot-ubl.bin
target # sf probe 0
target # sf erase 10000 80000
target # sf write c0700000 10000 ${filesize}
```

scrittura kernel su flash SPI

```
target # tftp c0700000 kx/ulimage
target # sf probe 0
```

Engineering Specification - confidential

```
target # sf erase b0000 350000  
target # sf write c0700000 b0000 ${filesize}
```

scrittura ramdisk su flash SPI

```
target # tftp c1180000 kx/ramdisk.gz  
target # sf probe 0  
target # sf erase 400000 3e0000  
target # sf write c1180000 400000 ${filesize}
```

scrittura kernel su flash NAND

```
target # tftp c0700000 kx/uimage  
target # nand erase clean c0000 400000  
target # nand write.e c0700000 kernel ${filesize}
```

scrittura rootfs JFFS2 su flash NAND

```
target # tftp c2000000 kx/rootfs-sum.jffs2  
target # nand erase clean 4c0000 1fb40000  
target # nand write.jffs2 c2000000 rootfs ${filesize}  
target # nand write.trimjffs c2000000 rootfs ${filesize} (metodo alternativo di scrittura)
```

scrittura UBL su flash NAND

```
target # tftp c0700000 kx/ubl_AM1808_NAND.bin  
target # nand erase.part ubl  
target # nand write.e c0700000 ubl ${filesize}
```

scrittura u-boot su flash NAND

```
target # tftp c0700000 kx/u-boot-ubl.bin  
target # nand erase.part u-boot  
target # nand write.e c0700000 u-boot ${filesize}
```

scrittura rootfs JFFS2 su flash NOR

```
target # tftp c2000000 kx/rootfs-summed.jffs2  
target # protect off 0x60000000 +${filesize}  
target # erase 0x60000000 +${filesize}  
target # cp.b 0xc2000000 0x60000000 ${filesize}  
target # protect on 0x60000000 +${filesize}
```

19.2 Programmazione via CCS

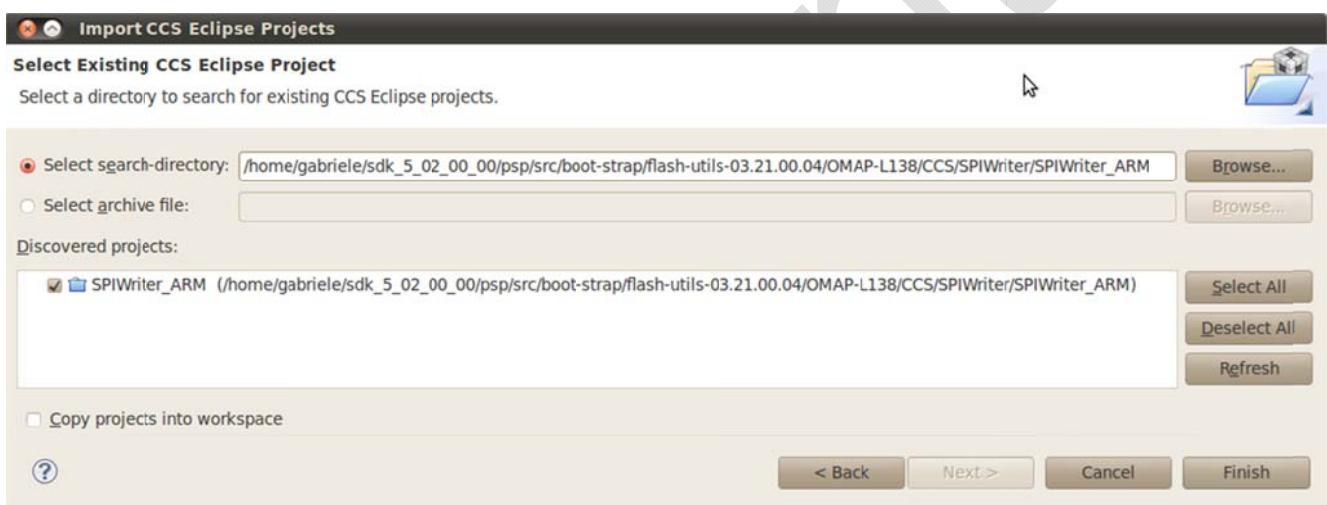
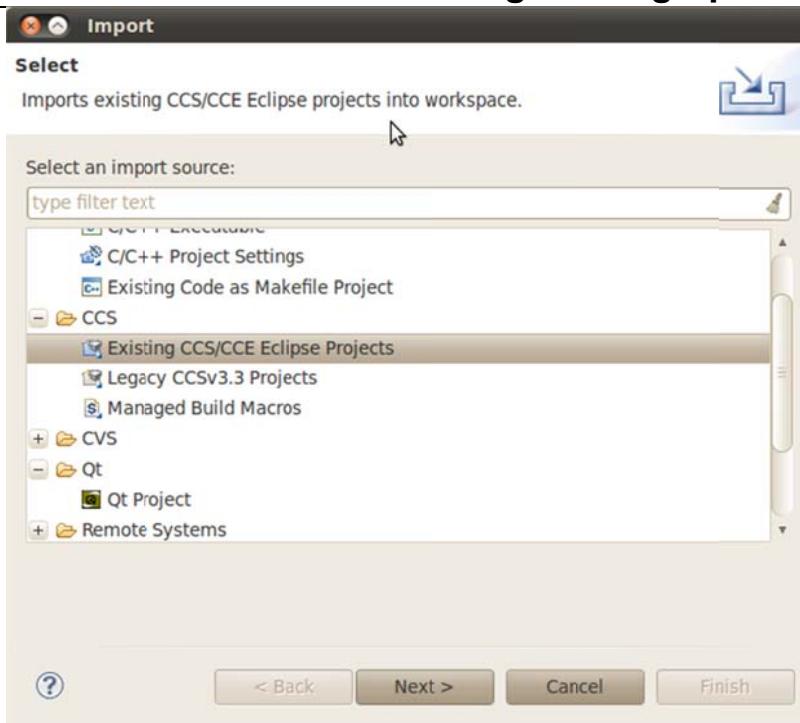
I sorgenti delle utility di programmazione Flash si trovano all'interno del PSP. Essi possono essere ricompilati tramite CCS. Si assume che SDKv5 sia installato e che il PSP sia in <psp> (vedi paragrafi relativi).

Tra i vari progetti ci sono l'SPIWriter_ARM, NANDWriter_ARM e NORWrite_ARM, che permettono di programmare UBL e u-boot rispettivamente nella flash SPI, NAND e NOR.

48) Per importare un progetto, ad esempio SPI Writer

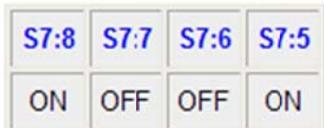
Da **File->Import...**

Engineering Specification - confidential



Da **Window->Preferences** aggiungere la versione Qt da associare al progetto (**Add**)
Ricompile con **Project->Clean, Build**.

- 15) Per procedere alla programmazione collegare l'emulatore XDS100v2 al PC e al target EVM
- 16) Predisporre il target in Emulation boot Mode (sulla EVM basta mettere a ON gli switch S7.5 e S7.8)



http://processors.wiki.ti.com/index.php/GSG:_OMAP-L138_DVEVM_Additional_Procedures#Flashing_images_to_NAND_Flash

20 Init e inittab

Il primo processo in user-space che viene lanciato dal kernel è init, tutti gli altri processi sono figli di init, compresa l'applicazione KX.

/etc/inittab contiene direttive per init.

In /etc/rc.d/init.d ci sono tutti gli script per lanciare o terminare i vari servizi

Un runlevel è uno stato di sistema definito dai suoi servizi e processi attivi

System V Init ha 6 runlevels (0:halt,...,6:reboot)

/etc/rc.d/rcN.d contiene i link simbolici agli script di startup (S..) e shutdown (K..) dei servizi del runlevel N

21 Qt Framework

Il KX utilizza Qt per creare l'applicazione e la Graphical User Interface. La GUI permette di interagire con il KX attraverso il touchscreen. Questo capitolo descrive il setup di Qt SDK e delle librerie Qt Embedded che dovranno essere linkate con l'applicazione.

Questi tools sono forniti con la licenza Open Source GPLv3.

Il primo passo è l'installazione nel target NFS delle librerie dinamiche necessarie a Qt.

Se richiesto utilizzare sudo su ciascuno dei comandi che seguono.

Anche la GUI del XNRG è basata su librerie Qt.

Su XNRG passiamo a Qt 5.9. Su KX si resta su Qt 4.8, in quanto la versione 5.x richiede l'acceleratore grafico, che il KX non ha.

Sia su KX che su XNRG le librerie Qt sono linkate staticamente.

Su XNRG, per mitigare gli obblighi relativi alla licenza, conviene linkare dinamicamente.

Quindi le applicazioni progress e xnrgfw devono linkare Qt dinamicamente.

Questo passaggio verrà fatto a settembre 2017

21.1 DBus

<http://blog.csdn.net/chanuei/archive/2011/03/05/6225689.aspx>

<http://zhxt.blogbus.com/logs/67299348.html>

DBus è una libreria generica per IPC a basso livello. Sfrutta due bus di comunicazione, "system" e "session". Matrix GUI ha bisogno di QtDBus. Per compilare QtDBus dobbiamo prima compilare DBus. Per compilare DBus è necessario expat. Expat è una libreria C per il parsing XML stream-oriented Scaricare il pacchetto expat in /opt da <http://sourceforge.net/projects/expat/>

Estrarre, configurare, compilare, installare nel target filesystem

```
host $ sudo tar -xzf expat-2.0.1.tar.gz
host $ mkdir -p auxdir
host $ cd expat-2.0.1
host $ make clean
host $ ./configure --host=arm-linux --prefix=/expat CC=/var/local/CodeSourcery/arm-2009q1/bin/arm-none-linux-gnueabi-gcc
host $ make
host $ make install DESTDIR=/opt/auxdir
```

Engineering Specification - confidential

Scaricare il pacchetto DBus in /opt da

<http://www.freedesktop.org/wiki/Software/dbus#Download>

Estrarre, configurare, compilare, installare nel target filesystem

```
host $ sudo tar -xzf dbus-1.4.8.tar.gz
host $ cd dbus-1.4.8
host $ make clean
host $ ./configure --host=arm-linux --prefix=/usr --sysconfdir=/etc --localstatedir=/var NM=/var/local/CodeSourcery/arm-2009q1/bin/arm-none-linux-gnueabi-nm "CC=/var/local/CodeSourcery/arm-2009q1/bin/arm-none-linux-gnueabi-gcc -I/opt/auxdir/expat/include -L/opt/auxdir/expat/lib" --with-xml=expat --with-x=no ac_cv_lib_expat_XML_ParserCreate_MM=yes ac_cv_have_abstract_sockets=yes
host $ make
host $ make install DESTDIR=<rootpath>
```

Al termine di questo processo avremo le librerie nel target NFS, in particolare in

<rootpath>/usr/lib/libdbus-1.0.so.3.5.5	(libreria)
<rootpath>/usr/lib/libdbus-1.0.so.3	(link simbolico alla libreria)
<rootpath>/usr/lib/libdbus-1.0.so	(link simbolico alla libreria)
<rootpath>/usr/lib/libdbus.a	(current ar archive, non necessario, eliminare)
<rootpath>/usr/lib/libdbus.la	(libtool library file, non necessario, eliminare)
<rootpath>/usr/lib/dbus-1.0/include/	
<rootpath>/usr/include/dbus-1.0/	
<rootpath>/usr/share/doc/dbus	
<rootpath>/etc/dbus-1.0/..	
<rootpath>/var/lib/dbus/..	
<rootpath>/usr/lib/pkgconfig/dbus-1.pc	

21.2 Tslib

Tslib è la libreria per la gestione del touchscreen.

- 4) Installare i tools necessari

```
host $ sudo apt-get install autogen autoconf libtool
```

- 5) Scaricare in /opt il pacchetto da <http://github.com/kergoth/tslib>

- 6) Estrarre

```
host $ tar -xzf kergoth-tslib-1.0-96-g5243db5.tar.gz
host $ cd kergoth-tslib-g5243db5
```

- 7) esportare le seguenti variabili per la cross-compilazione sull'host

```
host $ export CC=/var/local/CodeSourcery/arm-2009q1/bin/arm-none-linux-gnueabi-gcc
host $ export CXX=/var/local/CodeSourcery/arm-2009q1/bin/arm-none-linux-gnueabi-g++
host $ export ac_cv_func_malloc_0_nonnull=yes
host $ export CONFIG_SITE=kx.autogen
```

- 8) creare il file configure

```
host $ ./autogen-clean.sh
host $ ./autogen.sh (uses autogen.ac to create configure)
```

- 9) configurare

```
host $ ./configure CC=/var/local/CodeSourcery/arm-2009q1/bin/arm-none-linux-gnueabi-gcc CXX=/var/local/CodeSourcery/arm-2009q1/bin/arm-none-linux-gnueabi-g++ --host=arm-none-linux-gnueabi --target=arm-none-linux-gnueabi --enable-static=no --enable-shared=yes --prefix=/usr --sysconfdir=/etc
```

- 10) compilare

```
host $ make
host $ make install DESTDIR=<rootpath>
dove <rootpath> = /home/<user>/workdir/kx/filesys
```

Al termine di questo processo avremo le librerie nel target NFS

```
<rootpath>/usr/lib/libts-1.0.so.0.0.0  
<rootpath>/usr/lib/libts-1.0.so.0 (link simbolico al libts-1.0.so.0.0.0)  
<rootpath>/usr/lib/libts-1.0.so (link simbolico al libts-1.0.so.0.0.0)  
<rootpath>/usr/lib/libts.la (non necessario, eliminare)  
<rootpath>/etc/ts.conf  
<rootpath>/usr/include/tslib.h (non strettamente necessario)  
<rootpath>/usr/lib/ts/  
<rootpath>/usr/bin/ts_calibrate  
<rootpath>/usr/bin/ts_test  
<rootpath>/usr/lib/pkgconfig/tslib.pc
```

- 11) Editare il file di configurazione <rootpath>/etc/ts.conf e scommentare “module_raw input”
- 12) Per provare la libreria sul file system demo del SDK non serve altro che cancellare <rootpath>/etc/pointercal. Altrimenti..

13) Aggiungere sul target le seguenti variabili di ambiente in <rootpath>/etc/profile:

```
target $ export QWS_MOUSE_PROTO='Tslib:/dev/input/touchscreen0'  
target $ export TSLIB_CALIBFILE='/etc/pointercal'  
target $ export TSLIB_CONFFILE='/etc/ts.conf'  
target $ export TSLIB_CONSOLEDEVICE='none'  
target $ export TSLIB_FBDEVICE='/dev/fb0'  
target $ export TSLIB_PLUGINDIR='/usr/lib/ts'  
target $ export TSLIB_TSDEVICE='/dev/input/touchscreen0'
```

- 14) La prima volta che parte il target con la nuova libreria, eseguire routine di calibrazione (crea il file di calibrazione <rootpath>/etc/pointercal)

```
target $ ts_calibrate
```

- 15) Per verificare funzionamento, arrestare tutte le applicazioni grafiche e lanciare

```
target $ ts_test
```

21.2.1 tslib per KX

Il package dei sorgenti è
<dev>/fs/extpkg/tslib/tslib-1.1.tar.gz
Lo script di compilazione è
<dev>/fs/extpkg/tslib/gen-pkg.sh

21.2.2 tslib per Spartacus

Il package dei sorgenti è
<dev>/fs/extpkg/tslib/tslib-1.1.tar.gz
Lo script di compilazione è
<dev>/fs/extpkg/tslib/gen-pkg.sh

21.3 Qt Libraries for Embedded Linux

<http://doc.qt.nokia.com/4.7-snapshot/qt-embedded-install.html>

- 1) Assicurarsi di avere installato il g++, altrimenti

```
host $ sudo apt-get install g++
```
- 2) Da <http://qt.nokia.com/downloads> scaricare e copiare in /home

qt-everywhere-opensource-src-4.7.x.tar.gz

Attualmente la versione corrente è 4.7.3

3) Estrarre

```
host $ gunzip qt-everywhere-opensource-src-4.7.x.tar.gz  
host $ tar -xvf qt-everywhere-opensource-src-4.7.x.tar
```

4) A questo punto occorre informare Qt della nostra toolchain. Editiamo quindi il file qmake.conf relativo al sistema operativo per cui vogliamo cross-compilare

```
host $ gedit qt-everywhere-opensource-src-4.7.x/mkspecs/qws/linux-arm-g++/qmake.conf
```

Commentare le linee

```
#QMAKE_CC = arm-linux-gcc  
#QMAKE_CXX = arm-linux-g++  
#QMAKE_LINK = arm-linux-g++  
#QMAKE_LINK_SHLIB = arm-linux-g++  
#QMAKE_AR = arm-linux-ar cqs  
#QMAKE_OBJCOPY = arm-linux-objcopy  
#QMAKE_STRIP = arm-linux-strip
```

Riscrivere

```
QMAKE_CC = /var/local/CodeSourcery/arm-2009q1/bin/arm-none-linux-gnueabi-gcc  
QMAKE_CXX = /var/local/CodeSourcery/arm-2009q1/bin/arm-none-linux-gnueabi-g++  
QMAKE_LINK = /var/local/CodeSourcery/arm-2009q1/bin/arm-none-linux-gnueabi-g++  
QMAKE_LINK_SHLIB = /var/local/CodeSourcery/arm-2009q1/bin/arm-none-linux-gnueabi-g++  
QMAKE_AR = /var/local/CodeSourcery/arm-2009q1/bin/arm-none-linux-gnueabi-ar cqs  
QMAKE_OBJCOPY = /var/local/CodeSourcery/arm-2009q1/bin/arm-none-linux-gnueabi-objcopy  
QMAKE_RANLIB = /var/local/CodeSourcery/arm-2009q1/bin/arm-none-linux-gnueabi-ranlib  
QMAKE_STRIP = /var/local/CodeSourcery/arm-2009q1/bin/arm-none-linux-gnueabi-strip
```

Aggiungere le linee

```
TSLIB_INCDIR = /home/<user>/workdir/kx/filesys/usr/include  
LIBS_LIBDIR = /home/<user>/workdir/kx/filesys/usr/lib  
  
#paths to dbus.h  
QMAKE_INCDIR += /home/<user>/workdir/kx/filesys/usr/include/dbus-1.0  
QMAKE_INCDIR += /home/<user>/workdir/kx/filesys/usr/lib/dbus-1.0/include/dbus  
QMAKE_INCDIR += /home/<user>/workdir/kx/filesys/usr/lib/dbus-1.0/include  
QMAKE_INCDIR += /home/<user>/workdir/kx/filesys/usr/lib/dbus-1.0  
  
QMAKE_INCDIR += /home/<user>/sdk_4_01_00_00/linux-devkit/arm-none-linux-gnueabi/usr/include  
QMAKE_INCDIR += /home/<user>/sdk_4_01_00_00/linux-devkit/arm-none-linux-  
gnueabi/usr/include/libxml2  
  
QMAKE_INCDIR += $$TSLIB_INCDIR  
QMAKE_LIBDIR += $$LIBS_LIBDIR  
  
#flags for tslib  
QMAKE_CFLAGS += -I/home/<user>/workdir/kx/filesys/usr/include  
QMAKE_LFLAGS += -L/home/<user>/workdir/kx/filesys/usr/lib -Wl,-rpath-  
link=/home/<user>/workdir/kx/filesys/usr/lib  
  
#flags for dbus  
QMAKE_CFLAGS += -I/home/<user>/workdir/kx/filesys/usr/lib/dbus-1.0/include  
QMAKE_CFLAGS += -I/home/<user>/workdir/kx/filesys/usr/include/dbus-1.0  
QMAKE_LFLAGS += -L/home/<user>/workdir/kx/filesys/usr/lib  
  
QMAKE_CFLAGS += -I/home/<user>/sdk_4_01_00_00/linux-devkit/arm-none-linux-  
gnueabi/usr/include  
QMAKE_LFLAGS += -L/home/<user>/sdk_4_01_00_00/linux-devkit/arm-none-linux-gnueabi/usr/lib
```

Nota. Queste linee aggiuntive servono per configurare Qt con il supporto TSLib e DBus

Engineering Specification - confidential

- 5) Aggiorniamo la variabile di ambiente

```
host $ export QMAKESPEC=/home/gabriele/qt-everywhere-opensource-src-4.7.x/mkspecs/qws/linux-arm-g++
```

- 6) Assicurarsi che non siano più definite le variabili di ambiente CC e CXX

- 7) Configurare

```
host $ cd qt-everywhere-opensource-src-4.7.x
host $ ./configure -opensource -embedded arm -platform qws/linux-x86-g++ -xplatform qws/linux-arm-g++ -no-armfpa -little-endian -qt-gfx-transformed -qt-gfx-linuxfb -nomake demos -nomake examples -no-phonon -no-qt3support -no-feature-CURSOR -qt-mouse-tslib
```

buone di Gianni per 4.8.0 (al 13 nov 2012)

```
host$ ./configure -embedded arm -prefix /usr/local/TrollTech/Qt4.8.0arm -opensource -nomake examples -nomake docs -xplatform qws/linux-arm-arago-linux-gnueabi-g++ -qt-gfx-transformed -qt-mouse-tslib -depths 16,24,32 -little-endian -optimized-qmake -no-feature-CURSOR -no-armfpa -confirm-license
```

```
host $ cd qt-everywhere-opensource-src-4.7.x
host$ export PKG_CONFIG_PATH=<rootpath>/usr/lib/pkgconfig
host$ ./configure -opensource --confirm-license -embedded arm -platform qws/linux-x86-g++ -xplatform qws/linux-arm-g++ -no-armfpa -little-endian -qt-gfx-transformed -qt-gfx-linuxfb -nomake demos -nomake examples -no-qt3support -plugin-kbd-linuxinput -plugin-gfx-linuxfb -plugin-gfx-transformed -qt-freetype -depths 16,24,32 -optimized-qmake -qt-libjpeg -qt-libmng -qt-libpng -qt-libtiff -qt-zlib -qt-sql-lite -plugin-sql-lite -fast -no-rpath -no-feature-CURSOR -qt-mouse-tslib -DQT_QWS_CLIENTBLIT -DQT_NO_QWS_CURSOR -dbus -force-pkg-config
```

```
host $ popd
host$ export PKG_CONFIG_ALLOW_SYSTEM_LIBS=yes
host$ export PKG_CONFIG_SYSROOT_DIR=<rootpath>
host$ export PKG_CONFIG_PATH=<rootpath>/usr/lib/pkgconfig
host$ ./configure -opensource -confirm-license -embedded arm -platform qws/linux-x86-g++ xplatform qws/linux-arm-g++ -no-armfpa -little-endian -qt-gfx-transformed -qt-gfx-linuxfb -nomake demos -nomake examples -nomake docs -nomake translations -no-qt3support -plugin-mouse-tslib -plugin-kbd-linuxinput -plugin-gfx-linuxfb -plugin-gfx-transformed -plugin-gfx-powervr -plugin-gfx-qvfb -qt-gfx-multiscreen -qt-freetype -depths 16,24,32 -optimized-qmake -qt-libjpeg -qt-libmng -qt-libpng -qt-libtiff -qt-zlib -qt-sql-lite -plugin-sql-lite -fast -no-rpath -no-feature-CURSOR -qt-mouse-tslib -D QT_QWS_CLIENTBLIT -D QT_NO_QWS_CURSOR -dbus -force-pkg-config -continue
```

Con questa scelta le librerie generate saranno:

libQtCore.so.4.7.x
libQtDBus.so.4.7.x
libQtDeclarative.so.4.7.x
libQtGui.so.4.7.x
libQtMultimedia.so.4.7.x
libQtNetwork.so.4.7.x
libQtScript.so.4.7.x
libQtScriptTools.so.4.7.x
libQtSql.so.4.7.x
libQtSvg.so.4.7.x
libQtTest.so.4.7.x
libQtWebKit.so.4.7.x
libQtXml.so.4.7.x

libQt3SupportE.so.4.6.3
libQtAssistantClientE.so.4.6.3
libQtCLuceneE.so.4.6.3
libQtDesignerComponentsE.so.4.6.3
libQtDesignerE.so.4.6.3
libQtHelpE.so.4.6.3

Nota. Per quanto riguarda le opzioni di configurazione:

-embedded <arch> This will enable the embedded build
-no-armfpa Do not use the ARM-FPA floating point format
-little-endian Target platform is little endian (LSB first)
-qt-gfx-<driver> Enable a graphics <driver> in the QtGui library
-nomake <part> Exclude part from the list of parts to be built

Engineering Specification - confidential

Nota. L'opzione qt-mouse-tslib significa che il touchscreen è il primary mouse device

Nota. Se si vuole avere anche il display virtuale VNC, aggiungere -qt-gfx-vnc

Nota. L'installazione, di default, avviene in

<qlibarm>/=/usr/local/Trolltech/QtEmbedded-4.7.x-arm

Se si desidera modificare questa path basta usare –prefix <qlibarm>

Nota. -L<rootpath>/lib e -I<rootpath>/include sono le path a libts.so e libts.h

Nota. Dopo la prima volta, se necessario riconfigurare, usare prima

host \$ make confclean

- 8) Compilazione e installazione sull'host (il make richiede più di 3 ore)

host \$ make

Nota. Se l'host è un multi-core con N core, è possibile velocizzare la compilazione del kernel usando l'opzione -j#, dove # è 2N

- 9) Installazione sull'host

host \$ sudo make install

Nota. Di default L'installazione avviene sull'host in <qlibarm>

- 10) Installazione sul target

L'applicazione Qt che dovrà girare sul target si aspetterà di trovare in <qlibarm> le librerie Qt e i font.

Per installare in <rootpath>/<qlibarm>

host \$./configure --prefix <qlibarm> ...

host \$ make

host \$ sudo make install INSTALL_ROOT=<rootpath>

Nota. La variabile opzionale INSTALL_ROOT serve per appendere una path davanti a quella specificata nel configure con –prefix.

Il comando make install copia sul NFS anche altre cartelle oltre a lib. Queste altre cartelle non sono necessarie e vanno cancellate. L'installazione si può anche fare a mano, creando le cartelle e copiandoci dentro librerie e fonts

host \$ mkdir -p <rootpath>/<qlibarm>/lib/fonts

Copiare i file *.so.4.7.x

host \$ cp <qlibarm>/lib/*.so.4.7.x <rootpath>/<qlibarm>/lib/

e rinominarli *.so.4

In alternativa, creare dei link simbolici *.so.4 a *.so.4.7.x

Copiare i fonts

host \$ cp <qlibarm>/lib/fonts/* <rootpath>/<qlibarm>/lib/fonts/

- 11) Aggiornamento variabili di sistema sul target. Editare <rootpath>/etc/profile e aggiungere

export PATH=<qlibarm>/lib:\$PATH

- 12) Per poter lanciare il qmake per ARM da shell sull'host (cosa sempre utile), creare il link di sistema

home \$ sudo ln -s <qlibarm>/bin/qmake /usr/bin/qmake-arm

Verificare con

home \$ qmake-arm -v

- 13) Final Integration

host \$ echo "export LD_LIBRARY_PATH=<qlibarm>/lib" >> <rootpath>/etc/profile

XNRG:

Il file di configurazione delle Qt per XNRG si trova nello script
sdk/static-libs/src/qt/build-static.sh

Lo script applica delle correzioni alla configuazione dei default EGLS

21.4 Ricompilare demo matrix_gui

Ricompilare matrix_gui con le nuove le librerie Qt4.7.3

```
host $ cd <sdk>/example-applications/matrix_gui-e-1.3
host $ gedit matrix-gui.pro (aggiungere QT += dbus)
host $ gedit Makefile.build
commentare la linea che comincia con qmake2 <etc.> e sostituirla con
/usr/local/Trolltech/QtEmbedded-4.7.3-arm/bin/qmake -spec qws/linux-arm-g++ <etc.>
host $ cd <sdk>
host $ make clean
host $ make all
```

21.5 Set the toolchain paths

Per settare le path del SDK con la toolchain nelle variabili di ambiente della shell corrente

```
host $ cd <dev>
host $ source sdk/current/_profile.sh
```

21.6 Ricompilare XNRG qt-helloX

Nel XNRG il file di configurazione qmake.conf si trova in

```
host $ gedit qt-everywhere-opensource-src-4.7.x/mkspecs/qws/linux-arm-g++/qmake.conf
host $ <dev>/sdk/static-libs/built/qt-5.7-arm-epy-linux-gnueabihf-on-x86_64/mkspecs/linux-arm-epy-linux-gnueabihf-g++/qmake.conf
```

Ricompilare helloX con le librerie Qt5

```
host $ cd <dev>
host $ source sdk/current/_profile.sh
host $ cd apps/qt-helloX
host $ qmake-xnrg helloX.pro
host $ make clean
host $ make
```

Testare il programma in NFS

```
host $ sudo cp -r helloX qml /var/lib/nfs-Rootfs/root/
```

Lanciare la scheda in modalità NFS.

21.7 Qt Creator (standalone)

Da www.qt.io/download-open-source scaricare e copiare in /home/Download

qt-unified-linux-x64-x.x.x-x-online.run

Attualmente la versione corrente è 2.0.3-1

- 1) Rendere eseguibile e lanciare l'installazione

```
host $ chmod u+x qt-unified-linux-x64-2.0.3-1-online.run
host $ ./qt-unified-linux-x64-2.0.3-1-online.run
```

Come directory di installazione scegliere la cartella di default <qt>=/home/<user>/qt

21.8 Qt SDK

<http://doc.qt.nokia.com/qtcreator-snapshot/index.html>

Qt SDK comprende Qt Creator e Qt Developement Libraries

- 2) Installare i tools necessari

```
home $ sudo apt-get install libglib2.0-dev libSM-dev libXrender-dev libfontconfig1-dev libXext-dev
```

- 3) Da <http://qt.nokia.com/downloads> scaricare e copiare in /home

Qt_SDK_Lin32_offline_vx_x_x_en.run

Attualmente la versione corrente è 1.1.1

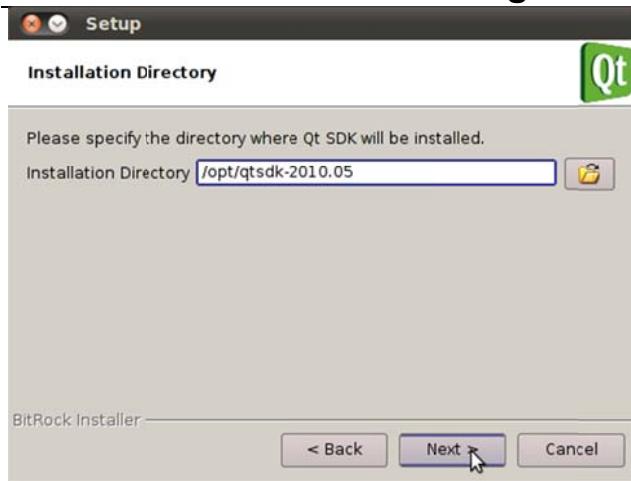
- 4) Rendere eseguibile e lanciare l'installazione

```
home $ chmod u+x qt-sdk-linux-x86-opensource-2010.05.1.bin  
home $ sudo ./qt-sdk-linux-x86-opensource-2010.05.1.bin
```

Nota. Se non si utilizza sudo, QT sarà installato in /home



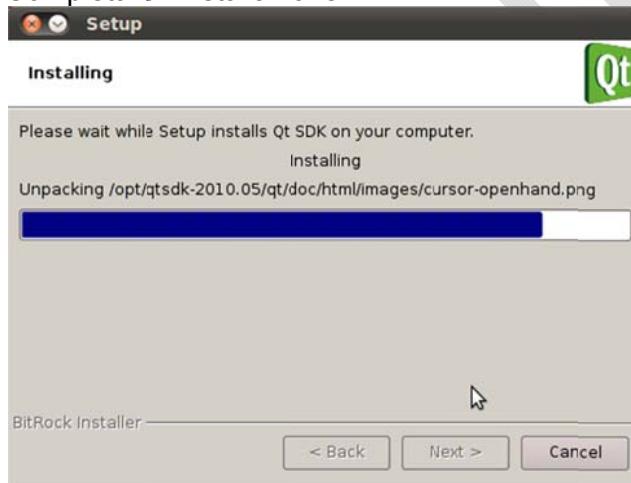
Come directory di installazione scegliere la cartella di default <qt sdk>=/opt/qt sdk-2010.05

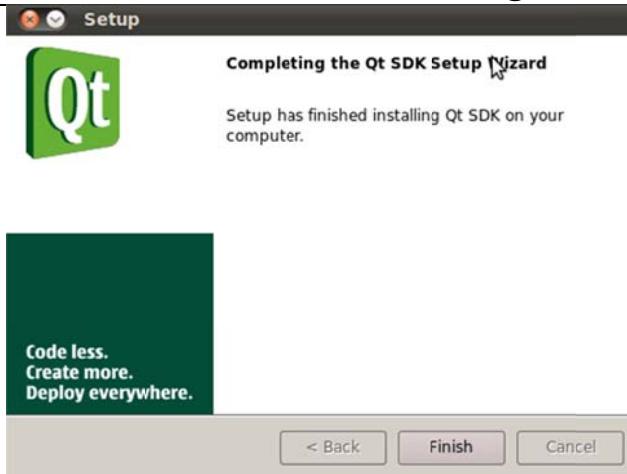


Come componenti da installare selezionare sia Qt Creator che Librerie (per host x86)



Completare l'installazione





Nota. Per disinstallare

```
home $ cd <qtdir>/bin  
home $ sudo ./uninstall
```

- 5) Creare il link di sistema per qmake (sarà visto da Qt Creator come versione “Qt in PATH”)

```
home $ sudo ln -s <qtdir>/bin/qmake /usr/bin/qmake
```

Verificare con

```
home $ qmake -v
```

21.9 Qt Host Libraries

Il pacchetto Qt SDK contiene già le librerie per l'host (x86), ma potrebbero essere vecchie. Quindi, se necessario, le si può sostituire con le stesse che utilizziamo per l'ARM

- 1) Generare le librerie per l'host (x86) e installarle nella stessa directory di installazione del Qt Creator

```
host $ cd qt-everywhere-opensource-src-4.7.x  
host $ make confclean  
host $ ./configure -opensource --prefix <qlib>  
host $ make  
host $ sudo make install
```

Nota. Subito dopo il ./configure dovrebbe essere notificato che le librerie, una volta compilate, verranno installate in <qlib>

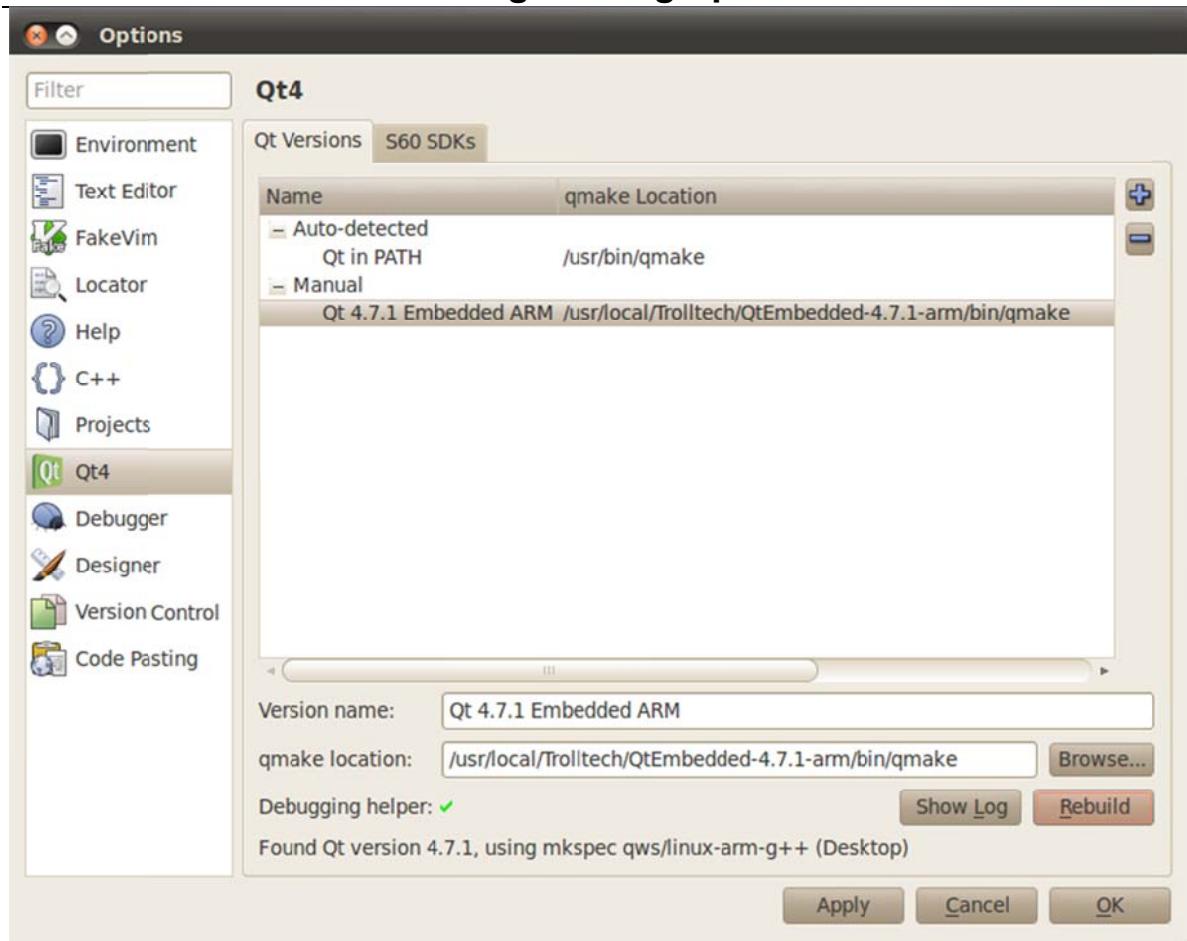
21.10 Configurare versioni in Qt Creator

- 2) Lanciare QtCreator. Sotto **Tools->Options->Qt4** sono visibili tutte le versioni di Qt disponibili. La versione “Qt in PATH” è quella di default, compilata per l'host (x86), e viene agganciata automaticamente al link /usr/bin/qmake, se esiste.

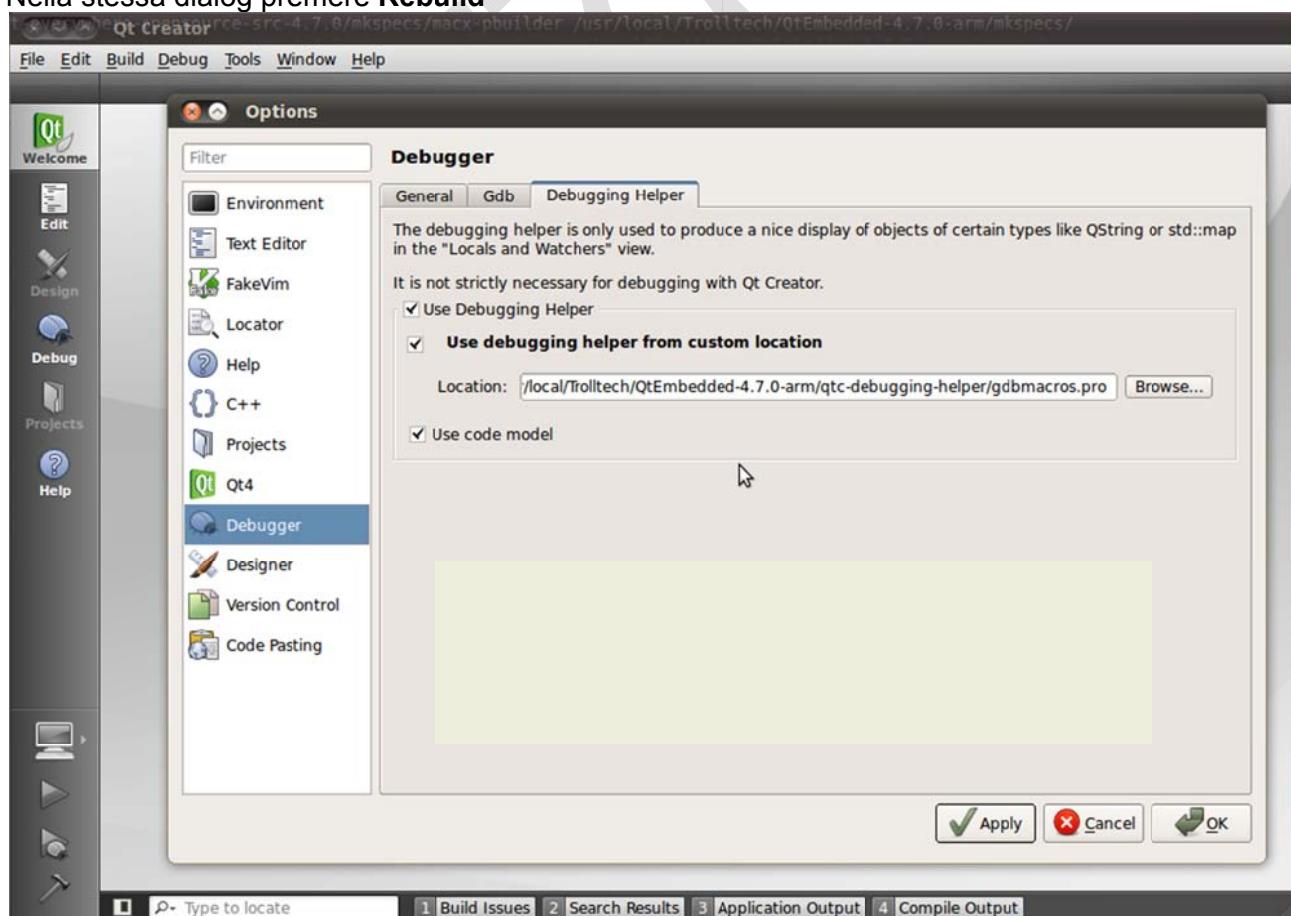
Nella sezione Manual aggiungere la versione di qmake precedentemente compilata per arm. Il version name può essere qualsiasi cosa (Es. **Qt 4.7.x Embedded**), mentre la path deve puntare al qmake compilato insieme alle librerie Qt Embedded

```
<qlibarm>/bin/qmake
```

Engineering Specification - confidential



Nella stessa dialog premere Rebuild



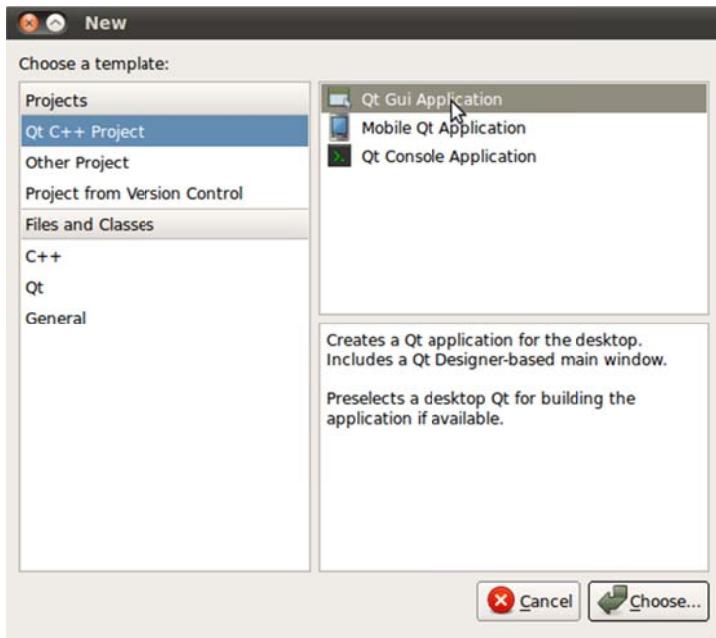
X Sri,

File: Linux_Development_Guide_rev21.docx

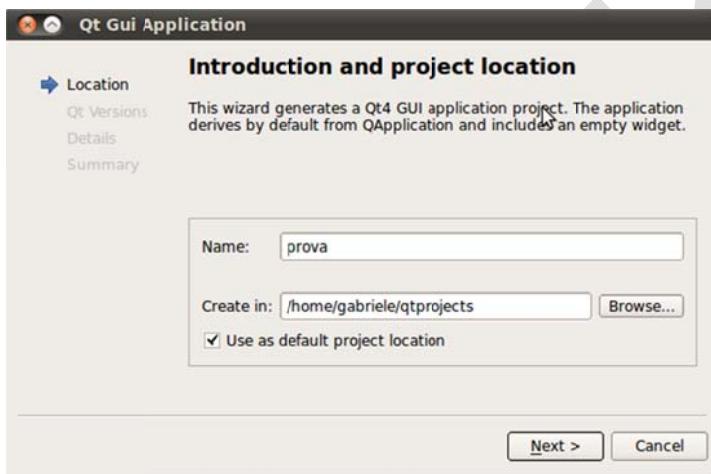
page 144/299

3) Creare un progetto

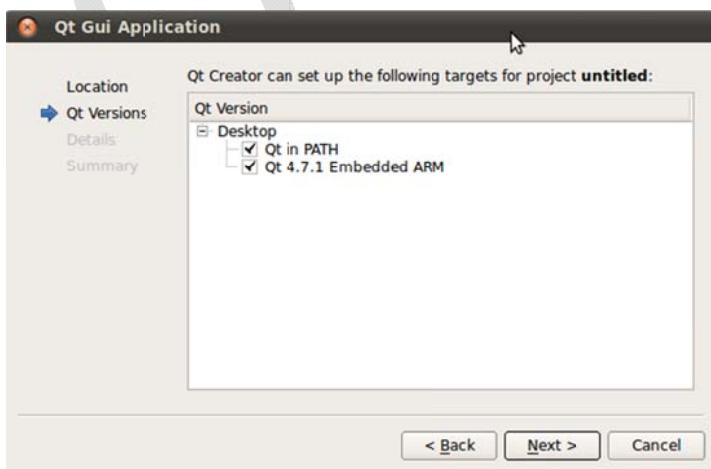
File->New File or Project->Qt C++ Project->Qt Gui Application->Choose

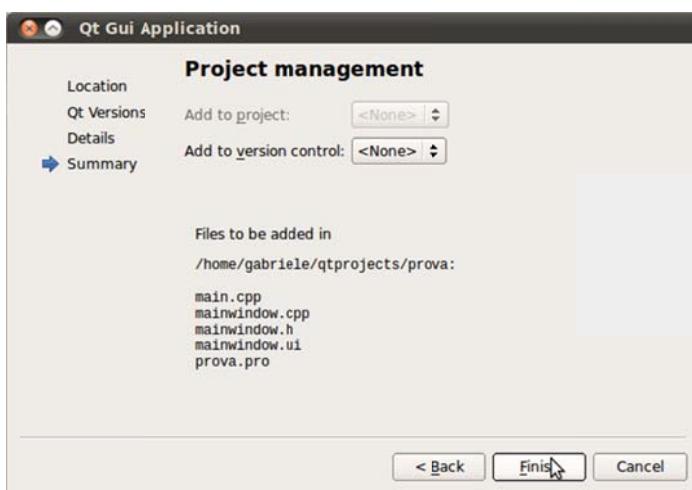
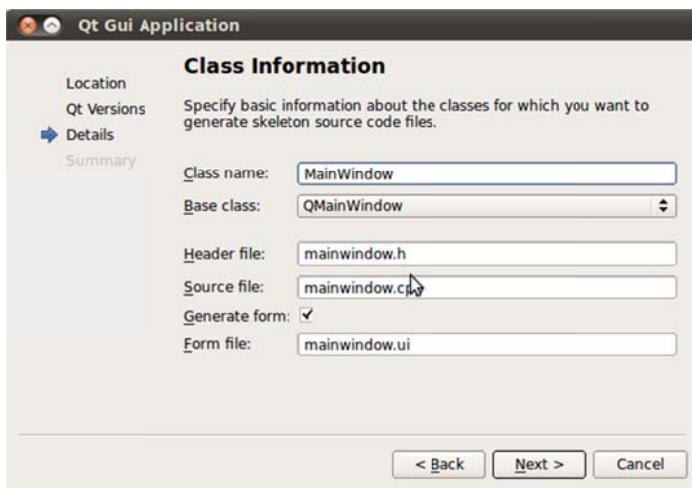


Assegnare un nome al progetto e una repository per i progetti



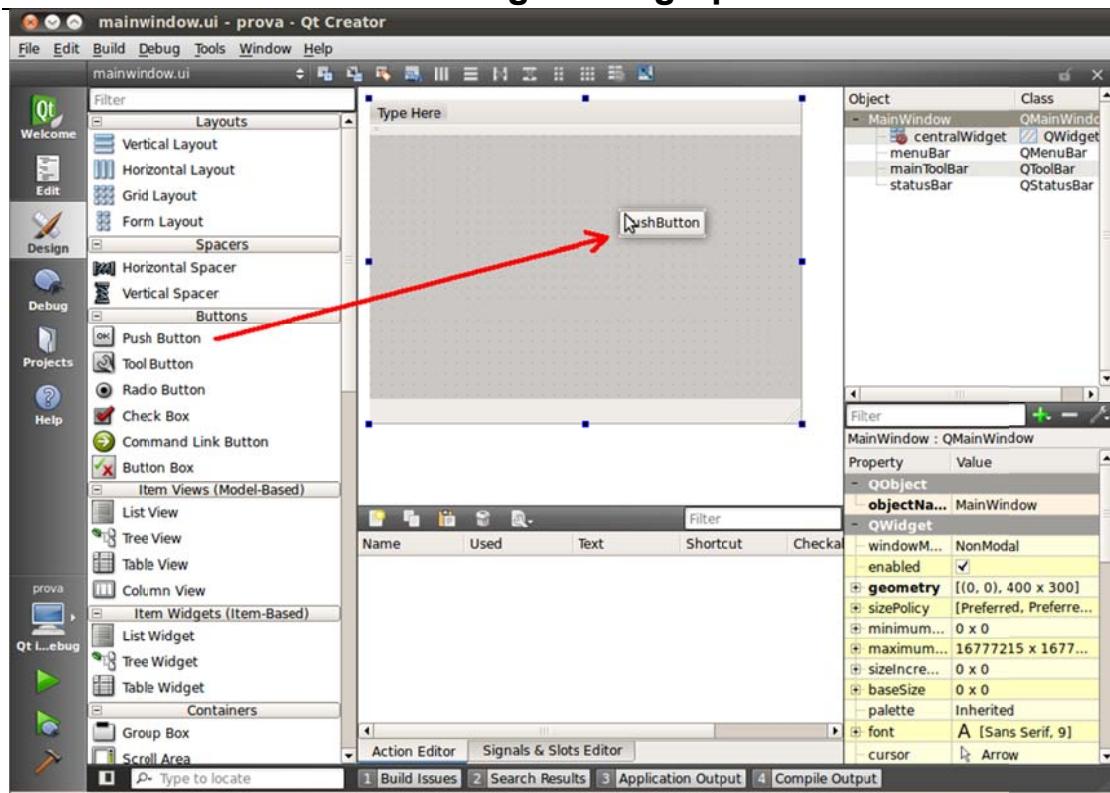
Verranno visualizzate tutte le versioni disponibili di qmake per compilare il progetto





Sulla finestra Design, importare nell'area attiva un Push Button (basta trascinarlo)

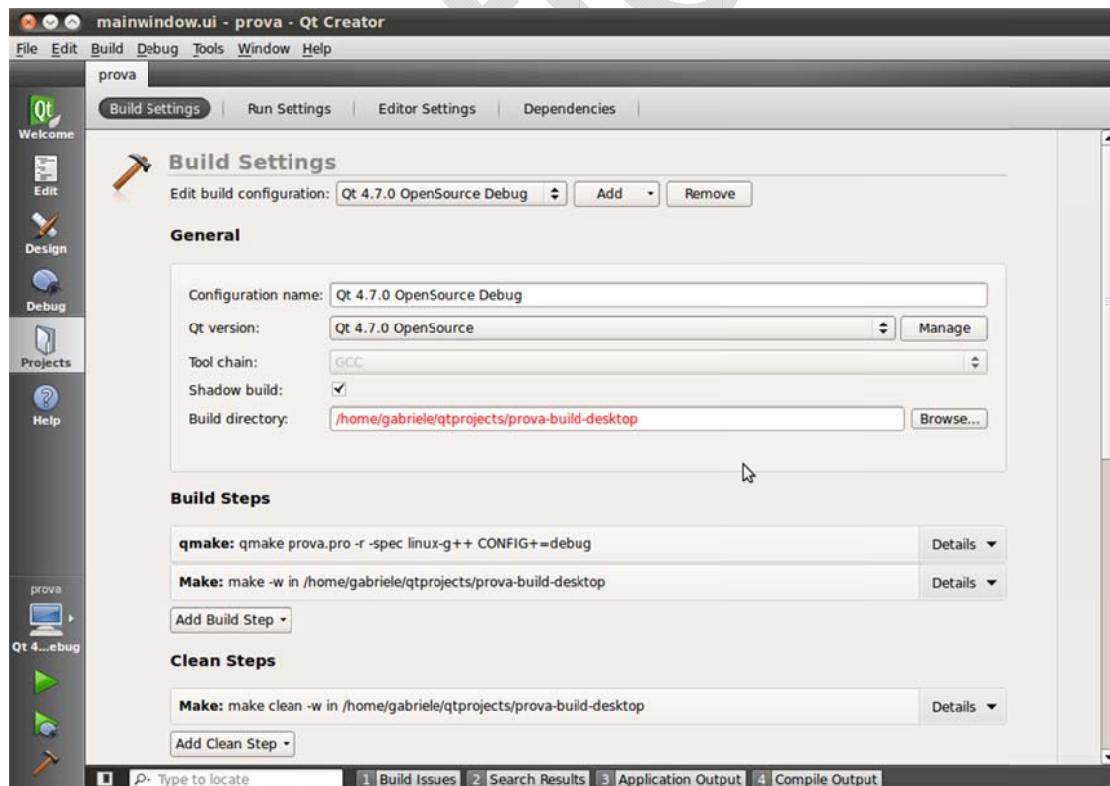
Engineering Specification - confidential



- 16) Normalmente un progetto viene anche provato sull'host. Creare le sottocartelle di progetto debug_arm e debug_x86 (e/o release_arm, release_x86)

```
home $ mkdir /home/<user>/qtproject/prova/debug_arm  
home $ mkdir /home/<user>/qtproject/prova/debug_x86
```

Nel prospetto Project, selezionare la versione Qt desiderata e la directory di build



Engineering Specification - confidential

Ricompile cliccando sull'icona martello o da **Build->RebuildAll**.

La freccia verde ricompila e lancia l'applicazione (se compilata per host).

La freccia verde con la lente lancia il debugger.

Prima di eseguire una nuova compilazione è bene eseguire **Build-> CleanAll**.

Dopo aver ricompilato per IARM copiare l'eseguibile sul target

```
host $ cp /home/<user>/qtprojects/prova/debug_arm/prova <rootpath>/home/root
```

Sul target lanciare l'applicazione

```
target $ cd ~/home/root  
target $ ./prova -qws
```

Nota. -qws serve solo per applicazioni Embedded. Per una applicazione Qt che fosse compilata per x86 non servirebbe

21.11 Progetto Qt MIDAS demo

[http://processors.wiki.ti.com/index.php/Medical_Imaging_Demo_Application_Starter_\(MIDAS\)](http://processors.wiki.ti.com/index.php/Medical_Imaging_Demo_Application_Starter_(MIDAS))

To do

21.12 Progetto Qt Arago demo

http://processors.wiki.ti.com/index.php/Qt_Framework_Demo_Setup

Qt based UI example source code

I sorgenti del progetto UI basato su Qt si trovano in

<https://qforge.ti.com/gf/project/qleslayer/scmsvn/?action=AccessInfo>

<http://arago-project.org/git/?p=arago.git;a=summary>

Aggiungere sul target le seguenti variabili di ambiente:

target \$ export LD_LIBRARY_PATH='/usr/lib'

target \$ export TSLIB_TSDEVICE='/dev/input/touchscreen0'

target \$ export QWS_MOUSE_PROTO='Tslib:/dev/input/touchscreen0'

target \$ export TSLIB_CALIBFILE='/etc/pointercal'

target \$ export TSLIB_CONFFILE='/etc/ts.conf'

target \$ export TSLIB_PLUGINDIR='/lib/ts'

Verificare funzionamento

target \$ /usr/tests/am1x-demo

21.13 Uso di Qt con VNC

Durante lo sviluppo dell'applicazione grafica Qt Embedded è utile il protocollo VNC. La libreria Qt Embedded deve essere configurata e compilata con l'opzione -qt-gfx-vnc.

To do

<http://web.mit.edu/qt-dynamic/www/qt-embedded-vnc.html>

21.14 Uso di qmake da shell

```
host $ cd projectname\
```

- 1) Per creare il file di progetto projectname.pro

```
host $ qmake -project
```

- 2) Per creare il Makefile

```
host $ qmake projectname.pro
```

- 3) Build

```
host $ make clean  
host $ make
```

- 4) Usare il prodotto della compilazione

Nota. Attenzione. Verificare che il file compilato ha usato il qmake per la piattaforma corretta

21.15 Qt - Tips

http://web.njit.edu/all_topics/Prog_Lang_Docs/html/qt/index.html

To do

- 1) Per creare un thread in Qt è necessario creare una classe che eredita da QThread. Si popola il metodo run() con il proprio codice. Si usa il metodo start() per lanciare il thread
- 2) Per leggere dai devices in Qt ci sono due modi. Il primo nodo sfrutta un socket IP. Il secondo modo prevede di lanciare uno script che carica il modulo driver (insmod) e crea in /dev un nodo collegato al driver (mknod), in modo che l'applicazione Qt possa accedere a quel nodo come se fosse un file.
- 3) QObject è la classe Qt che genera tutte le altre, come QThread, QWidget, ecc.. Sta in libQtCore
- 4) QWidget è la classe che genera tutti i controlli grafici

22 GDB

- 1) Ricompilare il GDB client

```
host $ cd <tools>/gdb-tools/gdb-7.2  
host $ ./configure --target=arm-linux --build=i686 --prefix=<tools>/gdb-tools  
host $ make  
host $ make install
```

Il client da usare è /gdb-tools/bin/arm-linux-gdb

- 2) Ricompilare il GDB server

```
host $ cd gdb/gdbserver  
host $ ./configure --build=i686 --host=arm-none-linux-gnueabi --target=arm-linux --prefix=<tools>/gdb-tools  
host $ make  
host $ make install
```

Il client da usare è /gdb-tools/bin/arm-linux-gdbserver

- 3) Copiare il server nel target filesystem

```
host $ cd <tools>/gdb-tools/bin  
host $ cp arm-linux-gdbserver <rootpath>/usr/bin/gdbserver
```

- 4) Nell'ambiente grafico desiderato (Qt, CCS, ecc.) configurare il GDB client

- 5) Lanciare il gdbserver sul target

23 SSH

SSH (Secure Shell) è un modo di instaurare una connessione sicura da un Client a un Server remoto, e su quest'ultimo eseguire comandi.

La porta della connessione è TCP port 22, ma può essere cambiata.

23.1 Principi della comunicazione sicura su una rete pubblica

SSH sfrutta un problema di alta complessità, come la fattorizzazione di un numero semiprimo molto lungo (RSA) oppure il calcolo del logaritmo discreto (DSA), o Diffie Ellmann.

Quando A e B devono autenticarsi con SSH:

A e B generano separatamente una chiave pubblica (*pub*) e una chiave privata (*pri*).

In RSA-4096, *pri* contiene due numeri primi da 2048 bit, mentre *pub* contiene il prodotto dei due (4096 bit).

In DSA, *pri* contiene due numeri (a,b), mentre *pub* contiene a^b .

La comunicazione tra A e B avviene in tre passaggi:

- 1) A e B generano le chiavi e poi si comunicano la pubblica
- 2) A codifica "pippo" con *pri_A* e lo invia a B, B decodifica con *pub_A* e legge "pippo". In questo modo B è sicuro dell'identità di A. Lo stesso avviene nell'altro verso
- 3) A genera e codifica una "chiave di sessione" con *pub_B* e la invia a B, B la decodifica con *pri_B*
- 4) A e B iniziano a usare la chiave di sessione comune per criptare i messaggi secondo un algoritmo simmetrico (p.es. AES-128), avente un costo computazionale minore.

Per generare una coppia di chiavi RSA-4096

```
host $ ssh-keygen -o -t rsa -b 4096
```

La chiave privata viene salvata (in base 64) in /home/<user>/.ssh/id_rsa

La chiave pubblica viene salvata (in base 64) in /home/<user>/.ssh/id_rsa.pub

RSA-1024 non può essere considerata assolutamente sicura

RSA-4096 può essere considerata assolutamente sicura

23.2 SSH login su Server

To do

Il Server è una VM Ubuntu presente in rete che ha le seguenti impostazioni

hostname: serverfirmware
serverip: <fwserverip> (statico)
password: firmware

Avendo un profilo utente sul Server con permesso di accesso si può fare il login dall'host remoto tramite il protocollo SSH

- 6) Per creare la connessione

```
host $ ssh -l <user> <serverip>
```

Ad esempio

```
host $ ssh -l gabriele 192.168.0.232
```

Il prompt diventa

```
user@serverfirmware $
```

- 7) Per uscire

```
<user>@serverfirmware $ exit
```

Nota: in /home/<user>/.ssh/known_hosts vi è la lista corrente dei terminali abilitati

- 8) Per eseguire un comando sul Server, p.es. un ls -l

```
host $ ssh -i /home/<user>/.ssh/id_rsa <user>@<serverip> 'sh -c "ls -l"'
```

....

- 9) Per generare una nuova chiave e accedere al server come utente git

```
host $ ssh-keygen
```

Quando richiesto inserire password oppure non inserire nulla se non si vuole

A questo punto appendere la chiave pubblica generata id-rsa.pub a /home/git/.ssh/authorized_keys

23.3 SSH login su Target

SSH è basato su RSA

- 10) Per creare la connessione

```
host $ ssh root@<ipaddr>
```

Ad esempio

```
host $ ssh root@192.168.0.149
```

- 11) Nel caso in cui la chiave RSA sia cambiata (perché ad esempio è cambiato il target) occorre aggiungere la nuova chiave in /home/<user>/.ssh/known_hosts.

Per rigenerare la chiave

```
host $ ssh-keygen -R <ipaddr>
```

Quindi provare nuovamente a connettersi

```
host $ ssh root@192.168.0.149
```

A connessione avvenuta la chiave del target sarà stata registrata in /home/<user>/.ssh/known_hosts

- 12) Nel caso in cui la chiave RSA non sia mai stata generata occorre generarla

```
host $ ssh-keygen -t rsa
```

questa operazione aggiunge i file id_rsa e id_rsa.pub in /home/<user>/.ssh/.

- 13) Nel KX si è deciso di generare una chiave, unica per tutti i KX, e memorizzarla nel target filesystem.

In questo modo, una volta che con un PC host mi sono collegato a un KX, quella chiave è salvata sull'host e non sarà più necessario farlo in seguito, dato che non cambia.

23.4 Usare Teraterm SSH da Windows

Teraterm, installato su PC Windows, può aprire connessioni SSH su un host remoto.

Esempio: per connettere

24 SCP

SCP è basato su SSH. Permette ad esempio di trasferire files dall'host al Server

```
host $ scp <localdir>/filename <serverip>:<remotedir>
```

Esempio

```
host $ scp <tftpbootdir>/<target>/ulimage <fwserverip>:/tftpboot/<target>
```

Esempio

```
host $ cd <uboot>/../  
host $ scp u-boot-qpatches-01.00.tar.gz <fwserverip>:<fwrepo>/u-boot/v01/
```

Esempio

```
host $ cd <linux>/../  
host $ scp linux-qpatches-01.00.tar.gz <fwserverip>:<fwrepo>/kernel/v01/
```

Esempio

```
host $ cd <rootfs>/../  
host $ scp rootfs-qpatches-01.00.tar.gz <fwserverip>:<fwrepo>/rootfs/
```

Esempio (trasferimento file dal target all'host)

```
target $ scp /etc/init.d/X gabriele@192.168.0.142:/home/gabriele
```

Host '192.168.0.142' is not in the trusted hosts file.

(ecdsa-sha2-nistp256 fingerprint md5 c8:42:7f:58:64:1a:74:aa:02:6d:f7:db:ba:44:f4:df)

Do you want to continue connecting? (y/n) y

gabriele@192.168.0.142's password:

X 100% 5579 5.5KB/s 00:00

(<- immettere <password>)

25 http server

To do

26 CVS

CVS è un VCS (Version Control System). Tramite CVS si può tenere traccia dei sorgenti e dei documenti. Si raccomanda di usare CVS per salvare sorgenti e makefiles.

Ad esempio Qt Creator ha due metodi di build, qmake e CMake. I makefile sono *.pro e *.pri per qmake e *.cmake per CMake.

Login ssh e installazione server CVS

```
host $ ssh <user>@<fwserverip>  
<user>@serverfirmware $ sudo apt-get install cvs
```

Verrà creata la repository /srv/cvs (<cvsrepo>)

creare un utente cvs (senza password) e un gruppo cvs e assegnare loro i permessi owner e group

```
<user>@serverfirmware $ sudo adduser cvs  
<user>@serverfirmware $ sudo addgroup cvs  
<user>@serverfirmware $ sudo chown -R cvs:cvs /srv/cvs
```

aggiungere gli sviluppatori desiderati al gruppo cvs

```
<user>@serverfirmware $ sudo usermod -a -G cvs gianni  
<user>@serverfirmware $ sudo usermod -a -G cvs mimmo  
<user>@serverfirmware $ sudo usermod -a -G cvs gabriele
```

A questo punto il server è stato predisposto e si può cominciare a lavorare.

26.1 Creare una nuova repository

La repository /srv/cvs è stata creata automaticamente.

Per creare una ulteriore repository occorre creare la cartella, cambiarvi i permessi ed inizializzarla.
P.es creiamo una repository in /var/local/mycvs

```
<user>@serverfirmware $ cd /var/local  
<user>@serverfirmware $ sudo mkdir mycvs  
<user>@serverfirmware $ sudo chown -R cvs:cvs mycvs  
<user>@serverfirmware $ sudo chmod -R g+w mycvs  
<user>@serverfirmware $ sudo cvs -d /var/local/mycvs init
```

a questo punto all'interno di /var/local/mycvs verrà creata la cartella CVSROOT

Ovviamente si possono creare delle repository ad uso del singolo utente. P.es

```
<user>@serverfirmware $ cd /home/gabriele  
<user>@serverfirmware $ mkdir cvsrepo  
<user>@serverfirmware $ sudo chown -R gabriele:gabriele cvsrepo  
<user>@serverfirmware $ sudo chmod -R g+w cvsrepo  
<user>@serverfirmware $ sudo cvs -d /home/gabriele/cvsrepo init
```

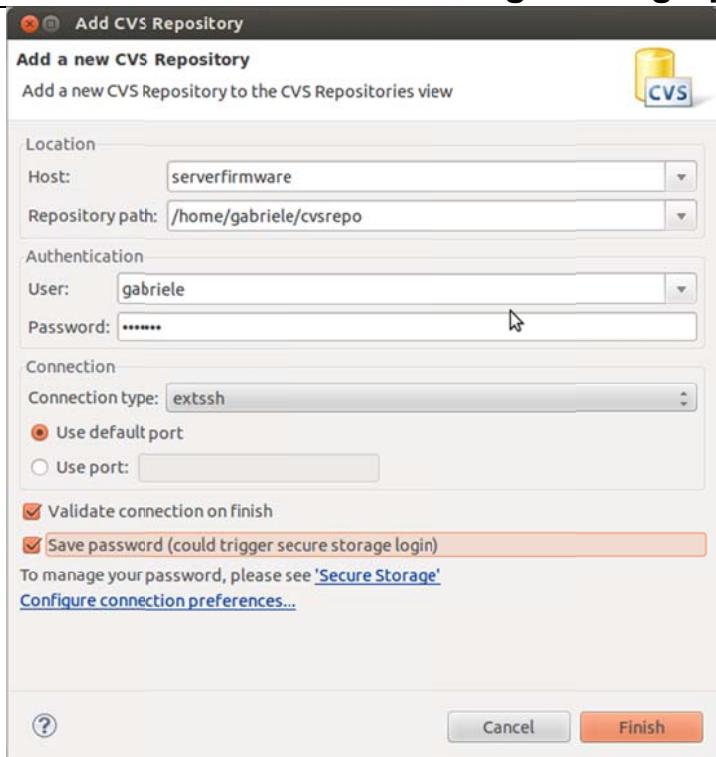
26.2 CVS in Eclipse

http://wiki.eclipse.org/CVS_FAQ

Da **Window->Show View->Other** scegliere **CVS** e poi **CVS Repositories**. Comparirà una nuova tab in basso intitolata CVS Repositories che mostrerà le repositories utilizzate in precedenza. Se è la prima volta che si usa CVS in Eclipse la tab sarà vuota.

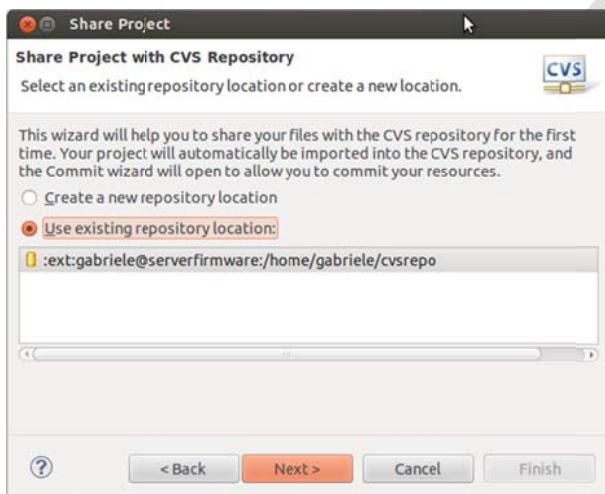
Per aggiungere nuove repositories fare click con il tasto destro in un'area libera della tab e scegliere **New->Repository Location**.

Engineering Specification - confidential

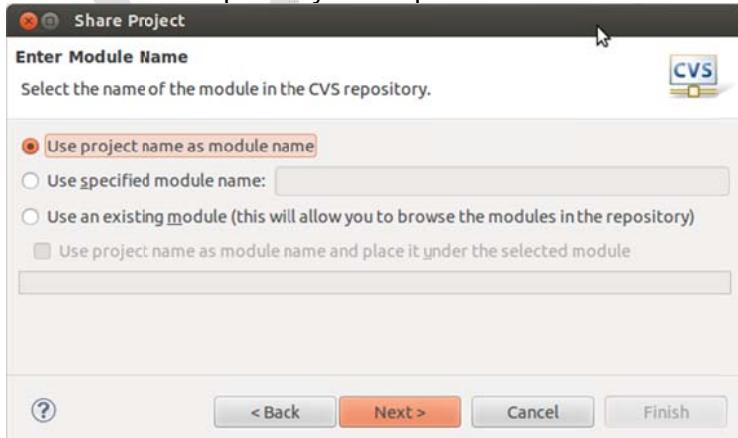


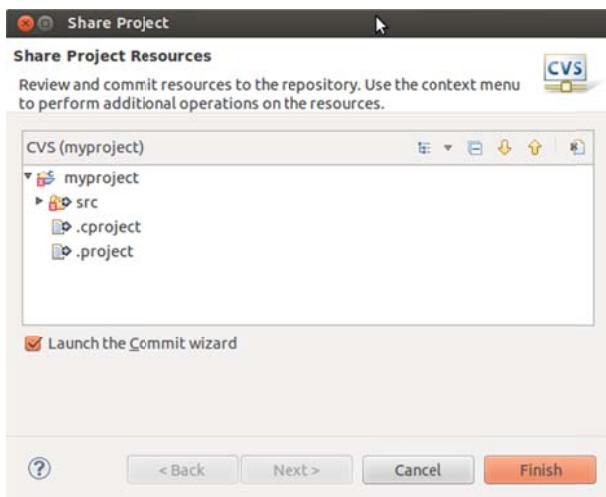
Riempire i vari campi come mostrato in figura.

Per aggiungere un nuovo progetto ad una repository esistente, nella tab **Project Explorer** aprire il progetto che si deve aggiungere, selezionarlo e con il tasto destro scegliere **Team->Share Project....**
Selezionare CVS

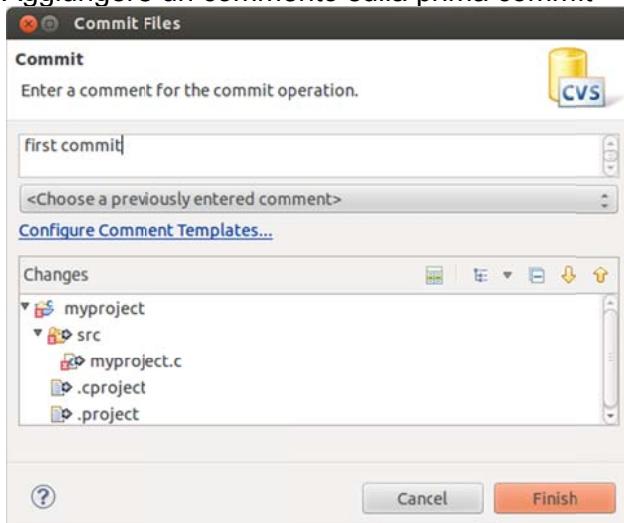


Selezionare la repository nella quale si intende inserire il progetto e premere Next >





Aggiungere un commento sulla prima commit

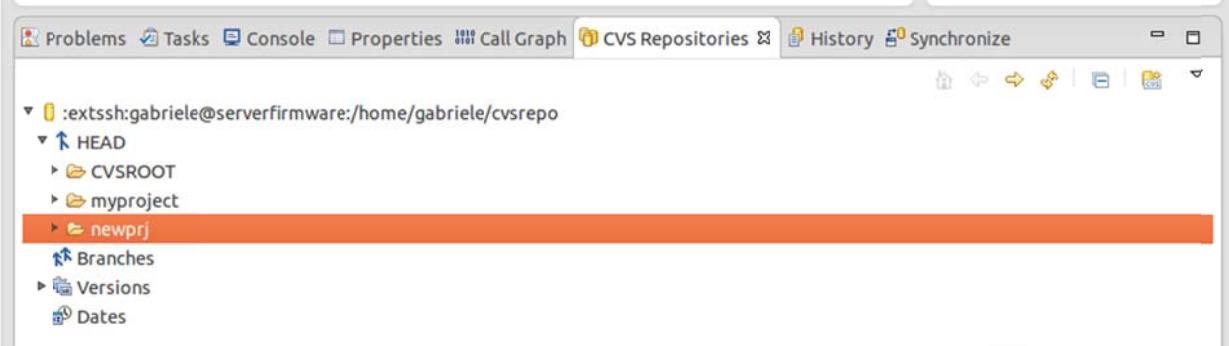


A questo punto si inizia a lavorare sul progetto.

Quando si vuole fare il check-in delle modifiche, selezionare il progetto e con il tasto destro scegliere **Team->Commit...** e inserire un commento per il check-in.

Per visualizzare informazioni sui file del progetto, da **Window->Show View->Other** selezionare **Team** e **History**.

Per creare una copia locale di un progetto che è già in una repository, selezionare la repository dalla tab **CVS Repositories**, espanderla e selezionare il progetto da **HEAD** e con il tasto destro selezionare **Check Out**.



Dato che è possibile fare il checkout di un progetto contemporaneamente da più postazioni è consigliabile eseguire spesso il comando **Team->Update** per mantenere il più possibile aggiornata la copia locale. Qualora vi fosse un problema di concorrenza sullo stesso file questo viene segnalato durante l'esecuzione del comando **Team->Update**.

27 SVN

SVN è un VCS (Version Control System). Tramite SVN si può tenere traccia dei sorgenti e dei documenti.

Login ssh e installazione server SVN

```
host $ ssh <user>@<fwserverip>
<user>@serverfirmware $ sudo apt-get install subversion
```

creiamo repository <svnrepo>

```
<user>@serverfirmware $ svnadmin create <svnrepo>
```

configuriamo il file svnserver.conf

```
<user>@serverfirmware $ nano <svnrepo>/conf/svnserver.conf
```

togliere alcuni commenti:

```
anon-access=read
auth-access = write
password-db=passwd
```

per salvare Ctrl+O, invio
per uscire Ctrl+X

inserire le password

```
<user>@serverfirmware $ nano <svnrepo>/conf/passwd
```

```
[users]
```

```
gabriele = <password>
```

per salvare Ctrl+O, invio

per uscire Ctrl+X e invio

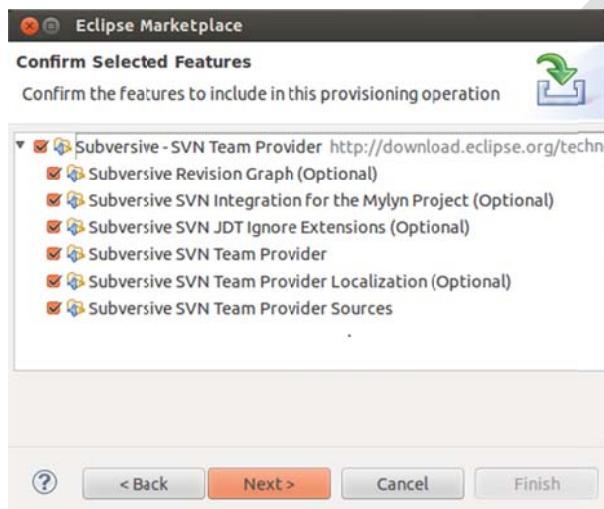
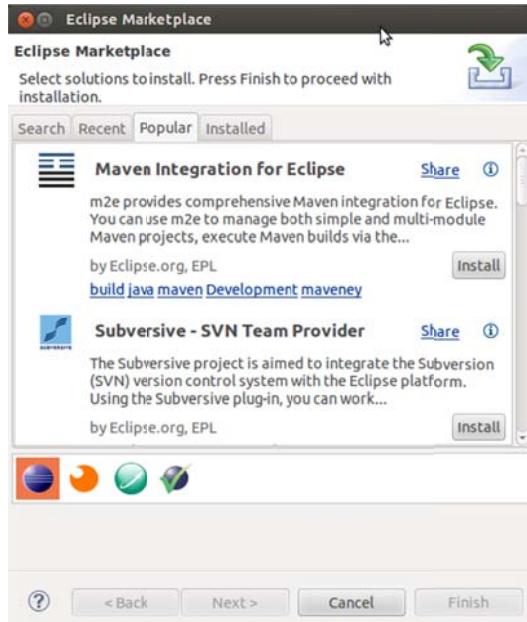
Avviare il server se non è già in uso

```
<user>@serverfirmware $ svnserve -d
```

27.1 SVN in Eclipse

http://wiki.eclipse.org/SVN_Howto

Da **Help->Eclipse Marketplace..** scegliere Collaboration tra le Categories e selezionare **Subversive – SVN Team Provider**



Completare l'installazione.

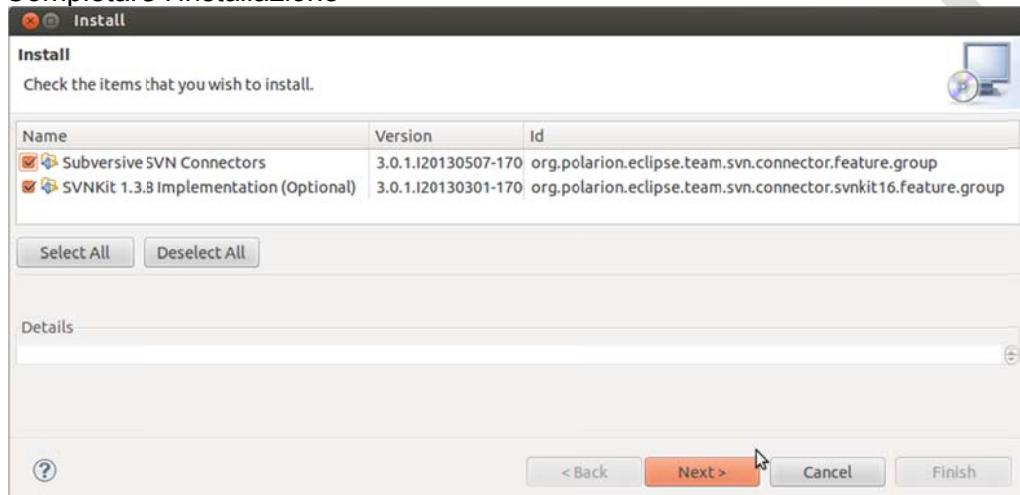
Da **Window->Show View->Other** scegliere **SVN** e poi **SVN Repositories**. Questa operazione non può essere completata se non si installa un Connector.

Selezionare SVN Kit 1.3.8

Engineering Specification - confidential



Completere l'installazione

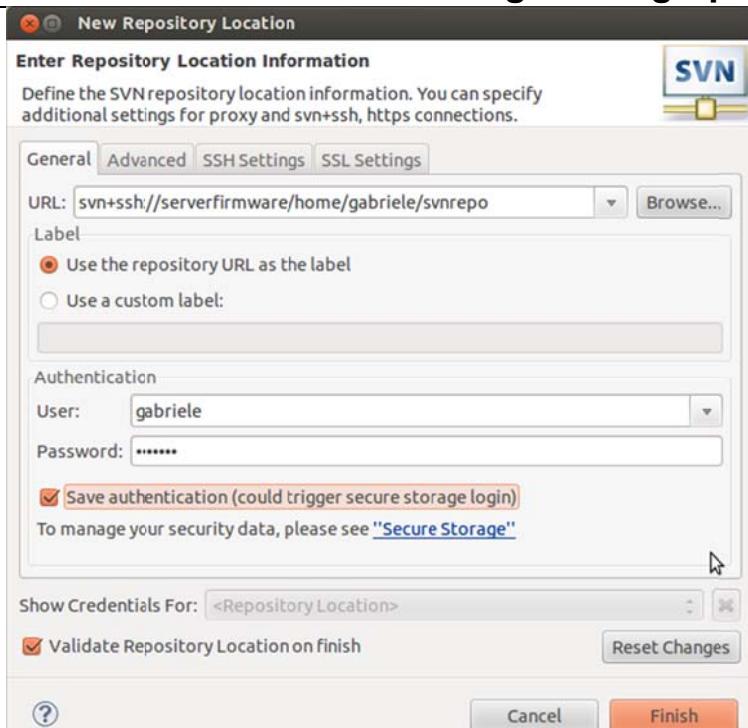


Da **Window->Show View->Other** scegliere **SVN** e poi **SVN Repositories**.

Comparirà una nuova tab in basso intitolata **SVN Repositories** che mostrerà le repositories utilizzate in precedenza. Se è la prima volta che si usa SVN in Eclipse la tab sarà vuota.

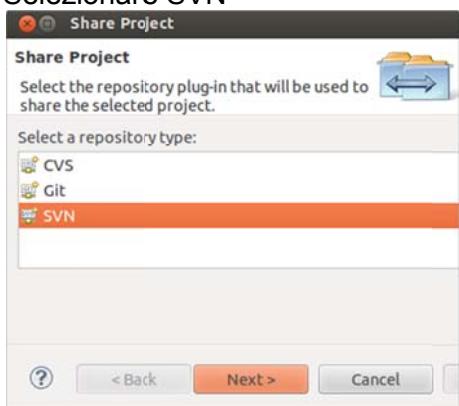
Per aggiungere nuove repositories fare click con il tasto destro in un'area libera della tab e scegliere **New->Repository Location**

Engineering Specification - confidential

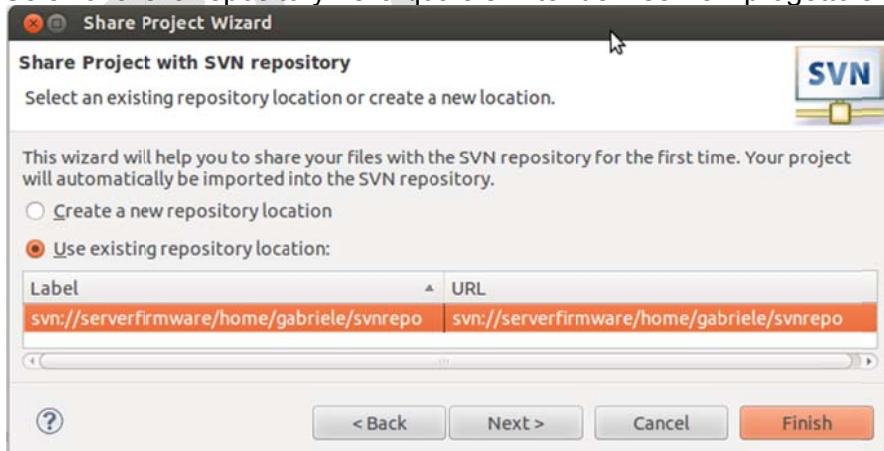


Riempire i vari campi come mostrato in figura.

Per aggiungere un nuovo progetto ad una repository esistente, nella tab **Project Explorer** aprire il progetto che si deve aggiungere, selezionarlo e con il tasto destro scegliere **Team->Share Project...**
Selezionare SVN

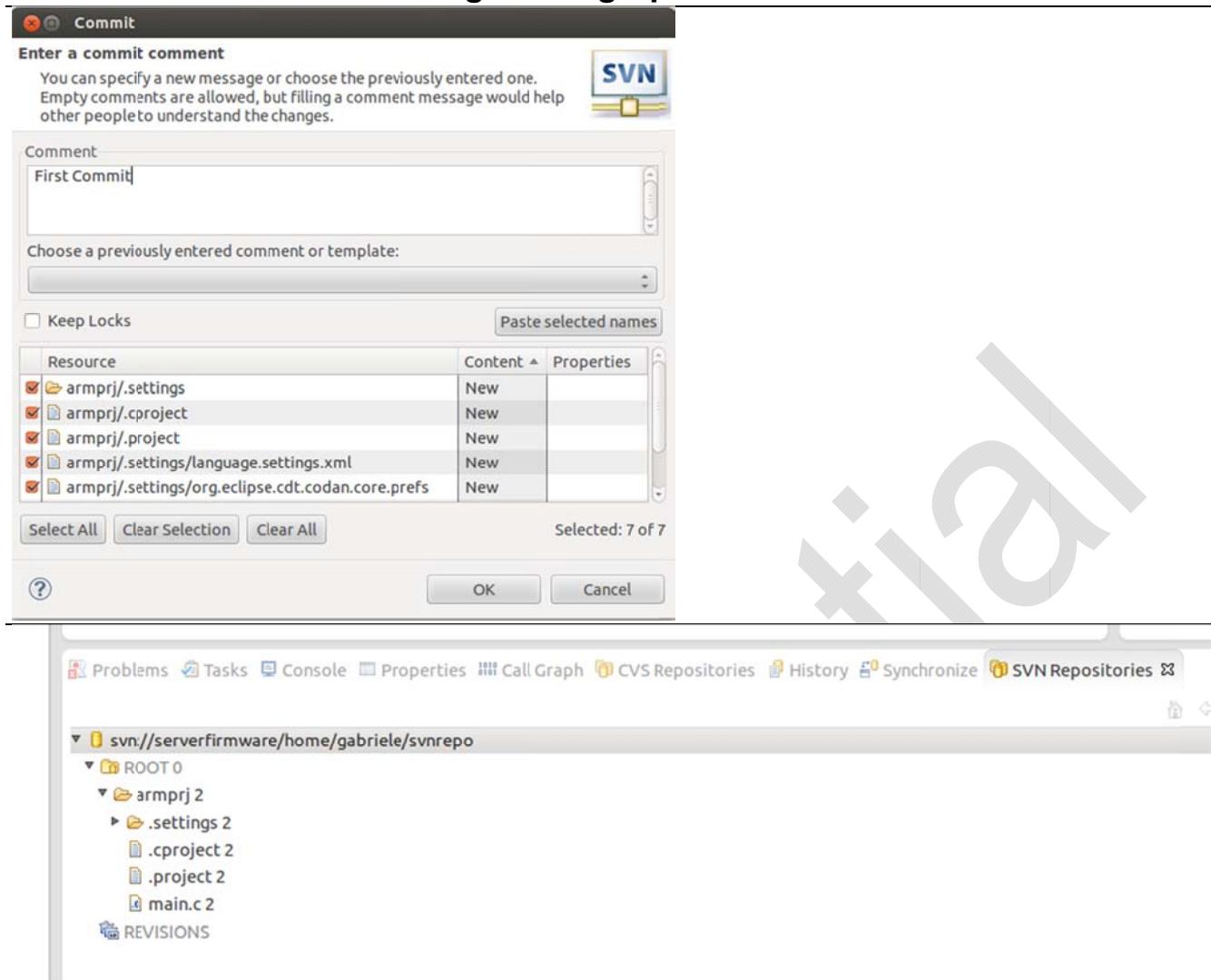


Selezionare la repository nella quale si intende inserire il progetto e premere Next >



Inserire un commento

Engineering Specification - confidential



A questo punto si inizia a lavorare sul progetto.
Quando si vuole fare il check-in delle modifiche, selezionare il progetto e con il tasto destro scegliere **Team->Commit...** e inserire un commento per il check-in.

Utilizzo

Ho un progetto CCS qtguiproject
Project Clean

New -> Project Structure -> qtguiproject

Seleziona trunk

Import->path->seleziona cartella progetto qtguiproject

Il progetto sta in trunk.

Trunk n significa che stiamo alla release n (un progressivo)

Per creare una copia locale di lavoro

Da trunk .. FindCheckout as...-> qtguiproject
Next .. specifica la path

Quando il lavoro è ok lo congelo con un nome (tag)
Ma prima devo fare un Commit

Per essere sicuro che le modifiche fatte in locale stanno aggiornate sul trunk
Basta assicurarsi che non c'è la freccetta.
Se c'è la freccia fare un Commit
Dopo il commit, per rinfrescare lo stato del progetto
Punta su progetto .. Team->Update

Commit è sinonimo di checkin
(allineamento del locale col remoto)

Punta su progetto .. Team->Commit.. inserisco un commento sotto Comment
La freccia scompare perché non ho modifiche

A ogni Commit/Update la revisione n incrementa di 1
(allinea il remoto col locale)

Adesso facciamo il tag vero e proprio
Punta su progetto .. Team->Tag.. gli do un nome p.es. "Tag Iniziale"

Lo sviluppo continua sul trunk mentre il tag creato resta fermo

Adesso rinomino la cartella qtuioproject
Cancello il progetto in CCS (Eclipse)

Checkout è ripristinare in locale qualche cosa che ho in remoto (un trunk, tag oppure branch)

Posizionarsi su trunk, tasto destro -> FindCheckoutAs..

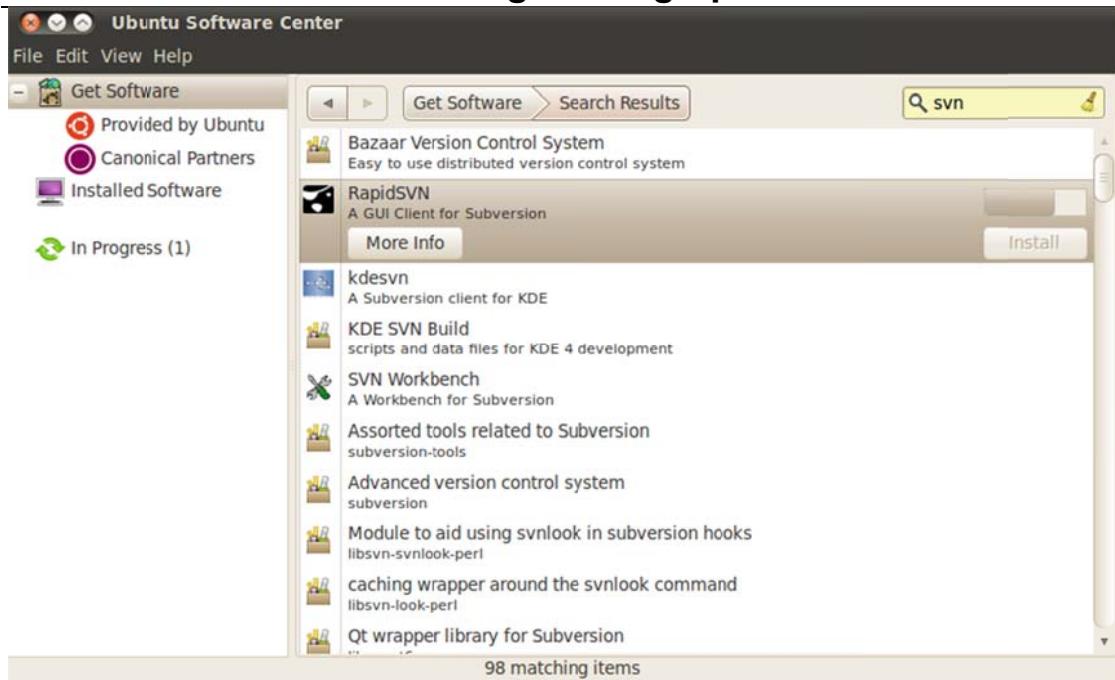
Per aggiungere un file, una volta creato
Punta sul file -> Team->Add to Version Control

Per provare una modifica di cui non sono sicuro creo un branch
Prima fare un Commit per essere sicuro che siamo allineati
Punta su trunk .. New->Branch
Punta sul progetto .. Team->Switch selezioni il branch

Per tornare a lavorare sul trunk
Punta sul progetto .. Team->Switch selezioni il trunk

Da Trunk, se sono soddisfatto delle modifiche sviluppate nel branch faccio un merge col branch
Punta sul progetto .. Team->Merge .. URL .. branch

Per usare SVN fuori da CCS (o Eclipse)
da Ubuntu Software Centre:



28 Prontuario dei comandi vim

Per aprire un file

```
host $ vim <filename>
```

Per aprire due file

```
host $ vim -o <filename1> <filename2>
```

Per aprire un file alla linea <row>

```
host $ vim <filename> +<row>
```

Per andare alla linea <row>

```
:<row>
```

Per entrare nella modalità editing spostandosi con le frecce

```
i
```

Per uscire

```
ESC
```

Per rendere permanenti le modifiche

```
:w
```

Per uscire senza rendere permanenti le modifiche

```
:q!
```

Per uscire rendendo permanenti le modifiche

```
:x
```

oppure

```
:wq
```

Per conoscere il nome del file che si sta editando

```
:f
```

Engineering Specification - confidential

Per cercare tutte le occorrenze di <item>

/<item>

E poi procedere con *n*

Per cercare tutte le linee che iniziano per <item>

/^<item>

E poi procedere con *n*

Per selezionare una parte di riga

v

Per selezionare un settore rettangolare

Ctrl+ v

Per copiare la selezione

y

Per indentare automaticamente una sezione di codice già selezionata

=

Per tagliare una parte di riga

v

Per selezionare <n> e copiare righe a partire da quella puntata dal cursore

<n>yy

oppure

<n>Y

Per incollare le righe selezionate a partire dalla riga sotto a quella puntata dal cursore

p

Per incollare le righe selezionate a partire dalla riga sopra a quella puntata dal cursore

Shift+ p

Per cancellare <n> righe a partire da quella puntata dal cursore

<n>dd

Per sostituire <item1> con <item2> su tutto il file chiedendo conferma ogni volta

:%s/<item1>/<item2>/gc

Se <item1> contiene alcuni caratteri particolari come “_” occorre precederli con “\”

Per impostare i tags /<path_to_tags>/tags

:set tags=/<path_to_tags>/tags

Quindi posizinarsi su un item e premere Ctrl+]

Per cercare altre opzioni oltre la prima mostrata

:ts

Undo

u

Per aprire <file> dall'interno di vi con allineamento orizzontale

:sp <file>

Per aprire <file> dall'interno di vi con allineamento verticale

:vs <file>

per passare da un file all'altro dall'interno di vi

Ctrl+ w

Per sostituire <item> ai caratteri selezionati

c

ESC

<down>

Per tornare alla shell mentre vim resta in gbr.rt4

gbr

Ctrl+z

per poi tornare all'editor

%

Enter

Per invocare un programma <prog> perché agisca su un file <file> e ottenere il file trasformato,

```
host $ vim <file>                                (<- apre il file)
.:%!<prog>                                         (<- trasforma il file secondo <prog>)
```

Esempio: per visualizzare un file binario in modo hex, con offsets sulla sinistra e ascii a destra, esiste il programma *xxd*. Supponiamo di voler cambiare qualcosa di questo file binario

```
host $ vim <file>
.:%!xxd                                              (<- attenzione, questo comando trasforma effettivamente il file)
... modificare, ecc.
.:%!xxd -r                                           (<- l'opzione -r di xxd esegue il reverse, cioè riporta file al formato originario)
:x                                                 (<- salva e chiude)
```

28.1 Creare e lavorare con i tags in vi

Installiamo ctags

```
host $ sudo apt-get install exuberant-ctags
```

Per creare i tags nella cartella dei sorgenti kernel

```
host $ cd <linux>
host $ make tags
```

Per creare i tags in una cartella <srcdir> contenente i sorgenti dell'applicazione

```
host $ cd <srcdir>
host $ ctags *.cpp *.h                               (indicare qui tutti i files dei sorgenti)
```

A questo punto è stato creato il file <srcdir>/tags

Editare con vi uno dei file sorgenti. Essendo il file dei tags nella stessa cartella del file, vi lo carica automaticamente

```
host $ vi <file>.cpp
```

Nel caso in cui il file dei tags si trovi in una cartella differente allora

```
host $ vi <file>
:set tags=</path_to_tags>/tags
```

Una volta posizionato il cursore sul nome di una funzione di cui si vuole vedere la definizione

Ctrl+]

Per tornare indietro

Ctrl+t

28.2 Editare files binari su Host con xxd, dd e cat

Installiamo xxd

```
host $ sudo apt-get install xxd
```

Per aprire il file binario <inputfile> e visualizzarlo in formato canonico (offset + hex + ASCII)

```
host $ xxd <inputfile>
```

Per aprire il file e visualizzarne <l> bytes a partire dall'offset <s>

```
host $ xxd -s <s> -l <l> <inputfile>
```

Con il programma dd è possibile copiare un file binario, convertirlo o formattarlo.

Engineering Specification - confidential

Per copiare su <outputfile> i primi blocchi da <n> bytes di <inputfile>

```
host $ dd if=<inputfile> of=<outputfile> bs=<n> count=<b>
```

Per creare il file <outputfile> come concatenazione dei file <file1> e <file2>

```
host $ cat <file1> <file2> > <outputfile>
```

29 GIT

<http://git-scm.com/documentation>

GIT è un DVCS (Distributed Version Control System).

Installazione

```
host $ sudo apt-get install git --
```

Configurazione username

```
host $ git config --global user.name "FIRST_NAME LAST_NAME"
```

Configurazione e-mail address

```
git config --global user.email "MY_NAME@example.com"
```

```
host $ ssh-keygen -t rsa
```

per farsi una copia in locale

```
host $ git clone git@serverfirmware:testproj
```

si modifica testproj

```
host $ git commit -a -m "modificato da gabriele ecc.."
```

```
host $ git push
```

29.1 Configurare GIT

Configurare git (va fatto solo la prima volta)

```
host $ git config --edit          (per cambiare editor aggiungendo la linea "editor = vim". Va fatto solo la prima volta)
```

```
host $ git config --global user.name "Gabriele Filosofi"      (per cambiare nome autore)
```

```
host $ git config --global user.email "gabrielef@X.it"        (per cambiare e-mail autore)
```

29.2 Creare repo remoto (su server)

Un repository git è distribuito su uno o più server, o “remoti”.

Dapprima si deve creare un utente e un gruppo sul server, per es. chiamati “git” e “git”

Procedura per creare una repository <reponame>.git su server come utente git

Su Server:

```
server $ sudo su
server $ cd /home/git
server $ mkdir <reponame>.git
server $ cd <reponame>.git
server $ git --bare init          (l'opzione --bare crea solamente lo storico e non la copia di lavoro)
server $ cd..
server $ chown -R git.git <reponame>.git
```

poi, su PC locale:

```
host $ cd myproject          (be myproject the folder with your source files, if any)
host $ git init
host $ touch filenew         (only if the folder is empty you need to create a dummy file)
host $ git add .
host $ git commit -m 'initial commit'
host $ git remote add origin git+ssh://git@192.168.0.232/~/<reponame>.git
host $ git push -u origin master
host $ rm filenew
```

Nota: *origin* è il nome di default con cui git battezza il primo remoto di un repository quando vi si accede da PC locale. Per i successivi remoti che uno volesse creare bisognerà specificare un nome differente.

Per visualizzare l'URL di un remoto <remotename> (p.es. *origin*)

```
host $ git remote show <remotename>
```

Nel nostro caso, per il KX, è

git+ssh://git@192.168.0.203/~/kx-dev.git

e per XNRG

git+ssh://git@192.168.0.203/~/xnrg-dev.git

29.3 Creare repo locale

Prima di tutto occorre generare una chiave pubblica per accedere al server come utente git

Procedura per creare una copia locale della repository *kx-uboot.git* esistente su server

```
host $ git clone git+ssh://git@192.168.0.203/~/kx-dev.git  (creo una copia della repo esistente su server in locale, autom. ho il branch master)
```

29.4 Creare un branch

Quando occorre fare un certo lavoro partendo da una repository master esistente, è utile generare un branch locale della repository e lavorare su quella. Opzionalmente si può anche fare una copia del branch sul server (se è bene che questo lavoro resti sotto backup) e, sempre opzionalmente, si può fare il merge sul master del lavoro svolto sul branch.

```
host $ git branch <branchname>          (creo un branch locale chiamato <branchname>)
host $ git checkout <branchname>          (passo al branch locale)
host $ git push -u origin <branchname>    (opzionale: crea una copia del branch locale anche sul server)

lavoro sui file e ogni tanto, facendo
  host $ git add <src>                   (aggiunge la cartella src e tutto il contenuto a quello che sarà il prossimo commit)
  host $ git commit                         (registra le modifiche su branch locale)
  host $ git push                           (opzionale: riporta branch locale anche su server)

host $ git checkout master                 (passo a branch locale master)
host $ git merge <branchname>             (opzionale: faccio il merge su master dell'opera svolta sul branch)
host $ git push                           (opzionale: riporto le modifiche master locale su master su server)
```

Nota:

i due comandi

```
host $ git branch <branchname>          (creo un branch locale chiamato <branchname>)
host $ git checkout <branchname>          (passo al branch locale)
```

si possono condensare in uno solo

```
host $ git branch -b <branchname>        (creo un branch locale chiamato <branchname> e ci passa dentro)
```

29.5 Utilità e comandi vari

In qualsiasi momento posso verificare lo stato della situazione corrente in vari modi:

```
host $ git status
host $ git log
```

Engineering Specification - confidential

host \$ gitk

(modalità grafica)

Utilizzando gitk si ha una interfaccia grafica, dove la situazione corrente del tuo repo locale è indicata dal tondino giallo



Per vedere le differenze introdotte tra commit <hash1> e commit <hash2> (precedente)

host \$ git diff <hash1> <hash2>

host \$ git diff --HEAD~1

Per vedere le differenze tra commit N-esimo e commit M-esimo (precedente)

host \$ git diff --HEAD~N --HEAD~M

Per le differenze tra due branch <branchname1> e <branchname2>

host \$ git diff <branchname1> <branchname2>

Per vedere le differenze su <path/filename> tra branch <branchname1> e <branchname2>

host \$ git difftool <branchname1> <branchname2> -- <path/filename>

Prima di lanciare il comando suddetto si dovrebbe poter selezionare il tool da usare, p.es.

host \$ git .gitconfig --global diff.tool <toolname>

Es. <toolname> = meld

Per vedere, nella modalità grafica, tutti i branch in dettaglio

host \$ gitk --all

Per vedere, nella modalità grafica, solo i commit relativi ai file presenti nella directory <dirname>

host \$ gitk <dirname>

Per cancellare una repo locale basta cancellare la cartella

host \$ rmkdir kx-uboot/

Per push-are tutte le modifiche fatte su uno stack

host \$ git stash -u

Per pop-are le modifiche

host \$ git stash pop

Per ri-applicare le modifiche ma senza rimuoverlo dallo stack (cioè come un pop senza pop)

host \$ git stash apply

Per visualizzare l'elenco dei file modificati nello stash

host \$ git stash show

Per avere la lista degli stash effettuati finora

host \$ git stash list

Per cancellare i dati pushati sul top dello stash

host \$ git stash drop

Notare che lo stack stash è trasversale a tutti i branch. Quindi, p.es., potrei fare il push su un branch, cambiare branch e poi fare il pop.

Per avere info su <item>

host \$ git help <item>

Operazione inversa di *git add <filename>*

host \$ git reset HEAD <filename>

Per cancellare forzatamente tutte le cartelle e i file untracked, cioè non riconosciuti da git, a partire dalla directory corrente

host \$ git clean -dfx .

Importante: Per annullare una o più operazioni involontarie di git rm che hanno cancellato alcuni file localmente, oppure per recuperare da un merge o stash pop involontario,

host \$ git reset --hard HEAD

Per cancellare gli ultimi <n> commit locali

```
host $ git reset --soft HEAD~<n>
```

oppure

```
host $ git reset --hard HEAD~<n>
```

Per escludere da git uno o più files in tutta la directory corrente e sottodirectory editare il file .gitignore. Il kernel e u-boot già hanno .gitignore

Per cancellare forzatamente un branch anche se non hai fatto un merge

```
host $ git checkout master
```

```
host $ git branch -D <branchname>
```

Per cancellare un branch, ma solo se hai fatto un merge

```
host $ git branch -d <branchname>
```

Per cancellare il branch remoto

```
host $ git push origin <branchname>
```

Per vedere tutti i branch disponibili su repository remoto

```
host $ git pull
```

```
host $ git branch -r
```

Per vedere tutti i branch disponibili su repository locale

```
host $ git branch -l
```

Per passare al branch <branchname>

```
host $ git checkout <branchname>
```

Per riportare le modifiche fatte su <branchsrc> su <branchdst>

```
host $ git checkout <branchdst>
```

```
host $ git merge <branchsrc>
```

Per riportare le modifiche fatte su <branchname> remoto su <branchname> locale

```
host $ git pull <branchname>
```

Per riportare le modifiche fatte su <branchname> locale su <branchname> remoto

```
host $ git push <branchname>
```

Per rivedere le differenze prima del git add

```
host $ git diff
```

Per vedere quali sono i file differenti tra <branchname1> e <branchname2>

```
host $ git diff <branchname1>..<branchname2> | grep "^\diff"
```

Per applicare una patch esterna

```
host $ cd <reponame>
```

```
host $ git apply -p<n> --index <patch>
```

(<n>=1,2 in dipendenza del livello della parent directory comune)

```
host $ git commit
```

```
host $ git push
```

Esempio:

```
host $ cd kx-uboot/src/u-boot-2011.12-rc1/
```

```
host $ git apply -p2 --index ../../patches/0001-kx-uboot-basic.patch
```

```
host $ git commit
```

```
host $ git push
```

Per integrare solo alcune delle modifiche sul file <filename> a un commit non ancora push-ato

```
host $ git add -p <filename>
```

Nota: premendo successivamente s la modifica multipla viene spezzata in due o più

Per scartare le modifiche fatte su <filename> sovrascrivendolo con quello in repo

```
host $ git checkout <filename>
```

Per integrare le modifiche sul file <filename> a un commit non ancora push-ato

```
host $ git add <filename>
```

```
host $ git commit --amend
```

```
host $ git push
```

Per integrare le modifiche sul file <filename> a un commit già push-ato (sconsigliato)

```
host $ git add <filename>
```

```
host $ git commit --amend
```

```
host $ git push --force
```

Engineering Specification - confidential

Per integrare le modifiche fatte su tutti i file modificati

```
host $ git commit -a -m "commento relativo al commit"  
host $ git push
```

Per integrare le modifiche fatte su un certo numero di files modificati

```
host $ git commit -m "commento relativo al commit" <filename1> <filename2>..  
host $ git push
```

Per spostare o rinominare un file o una cartella utilizzare il prefisso git, altrimenti si dovrà risolvere successivamente con git add e git rm

```
host $ git mv ...
```

Talvolta git prova a risolvere i conflitti durante un merge.

Se non ci riesce si ha un messaggio “Automatic merge failed; fix conflicts and then commit the result”

In tal caso ..?

```
host $
```

Dopo avere fatto delle modifiche in locale, può essere conveniente creare un nuovo branch dove inserirle per avere un backup, senza essere costretti a modificare il branch master

```
host $ git checkout -b <new-branch-name>           (creo il nuovo branch localmente)  
host $ git add <file-modificato-1>  
host $ git add <file-modificato-2>  
host $ git add <file-modificato-3>  
..  
host $ git commit  
host $ git push -u origin <new-branch-name>         (creo anche il branch remoto)
```

Ogni tanto si può dare il seguente comando per ottimizzare le performance di git

```
host $ git gc
```

Talvolta serve dare una ripulita al repository remoto buttando alcuni vecchi branch ormai inutili

```
host $ git push origin:<branchname>
```

Ovviamente poi è il caso di rimuovere anche le copie locali

```
host $ git prune origin
```

Di questi branch rimossi rimarrà comunque traccia sulle copie locali degli altri sviluppatori, finché questi non decideranno di fare un prune, prima con

```
host $ git remote prune -n origin
```

Questo comando mostra ciò che verrà rimosso senza però fare niente (è un dry run).

A quel punto, se ciò che propone sta bene,

```
host $ git prune origin
```

Per vedere il log di tutte le modifiche effettuate su uno specifico file

```
host $ git log -p <filename>
```

Nota: l'opzione -p indica che si intende visualizzare soltanto le patch.

Avendo due branch, ciascuno con la sua storia di commit, e il branch

```
host $ git log -p <filename>
```

Per visualizzare tutti i remoti di una repository

```
host $ git remote show
```

Nel nostro caso, per il KX, potremmo avere un unico <remotename>

origin

e, anche per XNRG,

origin

Per visualizzare l'URL <remoteurl> di un remoto <remotename> (p.es. **origin**)

```
host $ git remote show <remotename>
```

Nel nostro caso, per il KX, è

git+ssh://git@192.168.0.203/~/kx-dev.git

e per XNRG

git+ssh://git@192.168.0.203/~/xnrg-dev.git

Per clonare un remoto avente URL <remoteurl> su un nuovo server

```
nuovoserver $ git clone --mirror <remoteurl>
```

Poi, da un repository locale, per agganciarsi al remoto clonato su nuovoserver avente IP <nuovoserverip>

```
host $ git remote set-url <remotename> git+ssh://git@<nuovoserverip>/~/<reponame>.git  
host $ git remote set-url --push <remotename> git+ssh://git@<nuovoserverip>/~/<reponame>.git
```

Esempio

Nel nostro caso, avendo spostato il repository remoto su 192.168.0.203, per il KX

```
host $ git remote set-url origin git+ssh://git@192.168.0.203/~kx-dev.git  
host $ git remote set-url --push origin git+ssh://git@192.168.0.203/~kx-dev.git  
e per XNRG
```

```
host $ git remote set-url origin git+ssh://git@192.168.0.203/~xnrg-dev.git  
host $ git remote set-url --push origin git+ssh://git@192.168.0.203/~xnrg-dev.git
```

29.6 Utilizzare repo locale kx-uboot

La repository git dei sorgenti u-boot contiene uno script X-build.sh

```
host $ cd kx-uboot/src/u-boot
```

Per fare il clean e ricompilare u-boot e installarlo in <tftpbootdir>

```
host $ ./X-build.sh --rebuild <version>
```

Se il file di configurazione opzionale <version>, del tipo major.minor è presente, questa va a ridefinire la costante KX_FW_VERSION presente nel file di configurazione

Per compilare u-boot senza passare per clean

```
host $ ./X-build.sh --build <conf>
```

Se il file di configurazione opzionale <config> è assente, viene utilizzato quello di default

Nota: non usare sudo per questo script

Per generare i tags di u-boot

```
host $ ./X-build.sh --tags
```

Al posto dell'opzione build, si può sempre utilizzare build-single, che sfrutta un solo core. Questo permette di leggere meglio l'output del compilatore.

29.7 Creare git tags

Per creare un tag in git

```
host $ git tag <tagname> -m <comment>
```

Al tag è associato un hash (un numero del tipo a47ef67ed90736dbf4ccb14e74f7490fc785f1a4).

Avendo il tag si può ricaricare in locale la situazione così come era nella versione marcata dal tag

```
host $ git checkout <tagname>
```

Oppure anche con l'hash

```
host $ git checkout <hash>
```

E' chiaro che una volta entrati in questa condizione non si devono eseguire commit. Si possono comunque creare branch e ripartire da lì.

Per ritornare alla situazione aggiornata basta eseguire

```
host $ git checkout master
```

Il tag è di default visibile solo localmente. Per rendere il tag visibile a tutti

```
host $ git push --tags
```

Per cancellare il tag creato localmente

```
host $ git tag -d <tagname>
```

Per cancellare un tag già creato sul remoto

1. Ogni stazione locale (che ha già fatto un pull o un git remote update) deve eseguire

```
host $ git tag -d <tagname>
```

```
host $ git push origin :refs/tags/<tagname>
```

Per cancellare un commit già creato sul remoto

2. Ogni stazione locale deve cancellare il commit eseguendo

```
host $ git reset --hard HEAD~1
```

3. Da una sola stazione eseguire l'amend del commit e la cancellazione del commit remoto

```
host $ TODO..
```

Nel KX il meccanismo dei tag è utilizzato per marcare e identificare una versione FW, infatti il comando di creazione del root filesystem `<fs>/rootfs/gen-fs.sh` si preoccupa anche di creare il file `<fs>/rootfs/fs/X.build` inserendovi all'interno queste informazioni, che potranno essere recuperate dall'applicazione ed eventualmente visualizzate nella GUI

```
host $ cat <fs>/rootfs/fs/X.build
Mon Sep 29 17:33:37 CEST 2014
0.0-beta
a47ef67ed90736dbf4ccb14e74f7490fc785f1a4
root@fub1204
```

(← data del build del fs)
(← tag)
(← hash)
(← hostname)

Quindi, per creare un build del FW con versione

```
host $ git tag 0.1-beta -m "nuova versione fw per Alberto"
host $ git push --tags
host $ cd <fs>/rootfs
host $ sudo ./gen-fs.sh
```

Al tag è associato un hash (un numero del tipo `a47ef67ed90736dbf4ccb14e74f7490fc785f1a4`).

30 KX How To

30.1 Firmware versioning

Il file `<fs>/rootfs/fs/X.build` contiene le informazioni della versione FW

La versione FW del KX è del tipo

X.Y.W.Z

dove

X = major (0,1,...)

Y = minor (0,1,...)

W = alpha (0), beta (1), rc (2), release (3)

Z = build number (0,1,..)

La versione FW può anche essere stampata in questo modo

X.Y-betaZ finchè si debugga inserendo le features pianificate per X.Y, con Z che incrementa

Engineering Specification - confidential

```
target $ mount  
...  
ubi1:archive on /mnt/archive type ubifs (rw,relatime)  
...
```

30.3 Settings

Le applicazioni del KX e anche del WNRG hanno alcuni file di settings nella cartella `/usr/share/X/kmain/settings`

Essi sono

`Cfg.mmap`
`Settings.mmap`

Quando si lavora in NFS e l'applicazione viene ricompilata toccando queste strutture, oppure si vuole semplicemente ripristinare i settings di default del KX, è sufficiente cancellare i suddetti files sul target filesystem

```
target # /etc/init.d/kmain stop  
target # cd /usr/share/X/kmain/settings-default/Signals.conf /usr/share/X/kmain  
target # rm settings/*.mmap  
target # ./g
```

(<- per fermare l'applicazione sul target)
(<- per cancellare i vecchi files mmap)
(<- per lanciare nuovamente l'applicazione)

Se si va modificare anche il file Signals.conf, è necessario copiare a mano `Signals.conf` dalla cartella `settings-default`

```
target # cp /usr/share/X/kmain/settings-default/Signals.conf /usr/share/X/kmain/settings-default/
```

30.4 QML

QML utilizza una sintassi simile al JSON.

I sorgenti QML del Qt servono a descrivere gli oggetti grafici.

Per esempio per definire un rettangolo, le sue dimensioni esterne, il colore e gli anchors, si utilizzano i cotratti qml.

Le espressioni, i metodi, i signal handlers, ecc. che implementano la logica di applicazione sono scritti in JavaScript. Vi è una forte integrazione fra QML e JavaScript in Qt, analoga a quella tra HTML e JavaScript usata in un web browser.

Molti file di applicazione del KX e del XNRG sono scritti in QML. Essi hanno estensione `.qml`.

Quando si lavora in NFS e vengono modificati soltanto dei file qml è possibile vedere subito il risultato eseguendo

```
host $ cd /kx-dev/apps/kmain  
host $ ./m
```

e poi, sul KX, selezionando una diversa impostazione di Setting>System>User Function

30.5 KX – How to get a screenshot

Per catturare uno o più screenshot del display KX

- 1) Inserire SD card compatibile con dentro una cartella `_X_screenshots`
- 2) automaticamente il FW monta la SD su una cartella

```
target # mount -t vfat /dev/mmcblk0p1 /mnt/card
```

- 3) Premere tast REC per catturare screenshots

- 4) Estrarre SD card e spostare immagini su PC Windows, in formato `.raw`, nella cartella condivisa `\192.168.0.232\convert`

Automaticamente le immagini verranno trasformate in formato `.png` (il tool utilizzato è anche disponibile per Linux su FW repository)

Engineering Specification - confidential

- 5) Estrarre SD card e spostare immagini su PC Windows, in formato .raw, nella cartella condivisa \\192.168.0.203\BitmapConverter
Automaticamente le immagini verranno trasformate in formato .png (il tool utilizzato è anche disponibile per Linux su FW repository)
- 6) Togliere le immagini dalla cartella (in quanto risorsa condivisa)

La conversione si può anche fare a mano su PC host, dove si trova l'eseguibile *r2p.sh*
Spostare i file .raw nella cartella host dove si trova l'eseguibile *r2p.sh*

```
host # cd <path-to>/raw2png  
host # ./r2p.sh
```

Ora i files .raw sono stati convertiti in altrettanti file .png

Per progetto Spartacus (XNRG):

Su XNRG non ci sono problemi in quanto gli screenshots possono essere direttamente esportati in PNG sulla pennetta usb. A eseguire la conversione ci pensa direttamente una funzione della libreria Qt.

30.6 How to get access to linux shell with password

La shell di linux su KX è protetta da password

La password può essere recuperata nel file

```
host $ vim <dev>/scripts/open_console
```

In questo momento è

b3c6d787acb

Nel funzionamento in NFS la password non è necessaria. È stata disabilitata.

Per progetto Spartacus (XNRG):

TODO

30.7 How to get event logs

Per memorizzare i log files del KX nella SD card

- 1) Inserire SD card compatibile con dentro una cartella _X_log
- 2) Accendere KX
- 3) Lavorare col KX attivando le funzioni che interessano
- 4) Spegnere KX
- 5) Estrarre SD card e consultare i vari files di log memorizzati nelle cartelle (ogni cartella è una sessione di power-on to power-off)

Per progetto Spartacus (XNRG):

Nel XNRG i log vengono memorizzati automaticamente in una partizione di memoria peristente rispetto a aggiornamenti FW (archive).

- 1) Accendere XNRG
- 2) Inserire USB memory drive
- 3) Sulla GUI esiste una funzione "Export Log to Pendrive"
- 4) Estrarre USB drive e estrarre file *xnrg-log.zip* (password: *xnrg0987654321*)
- 5) Consultare i vari files di log memorizzati nelle cartelle

30.8 KX – How to activate hw modules via SD card

Per attivare uno dei moduli HW del KX

- 1) inserire SD card compatibile con dentro il file di attivazione xml generato da programma delle licenze *NGActivator*. Il nome del file è tipo *KX_<serial number>_HW.lic*
- 2) navigare su menu display *Utility/Licence* e confermare
- 3) Il KX si riavvia automaticamente

30.9 Bitbucket

Bitbucket è un server remoto che supporta due DVCS tools, Git e Mercurial.

Creare un account:

Signup for a free account to <http://bitbucket.org/>

Team Name: X

Username: X

Password: paolob

members: gabrielef@X.it (Administrator); domenicoc@X.it; giannic@X.it

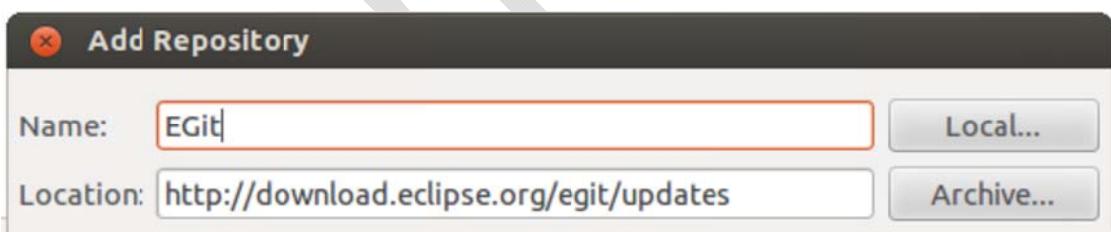
Dopodichè occorre installare un tool DVCS, tipo EGit in Eclipse, per creare clonare progetti ecc.

30.10 EGit

EGit (Eclipse Git) è un DVCS, ideale per open source.

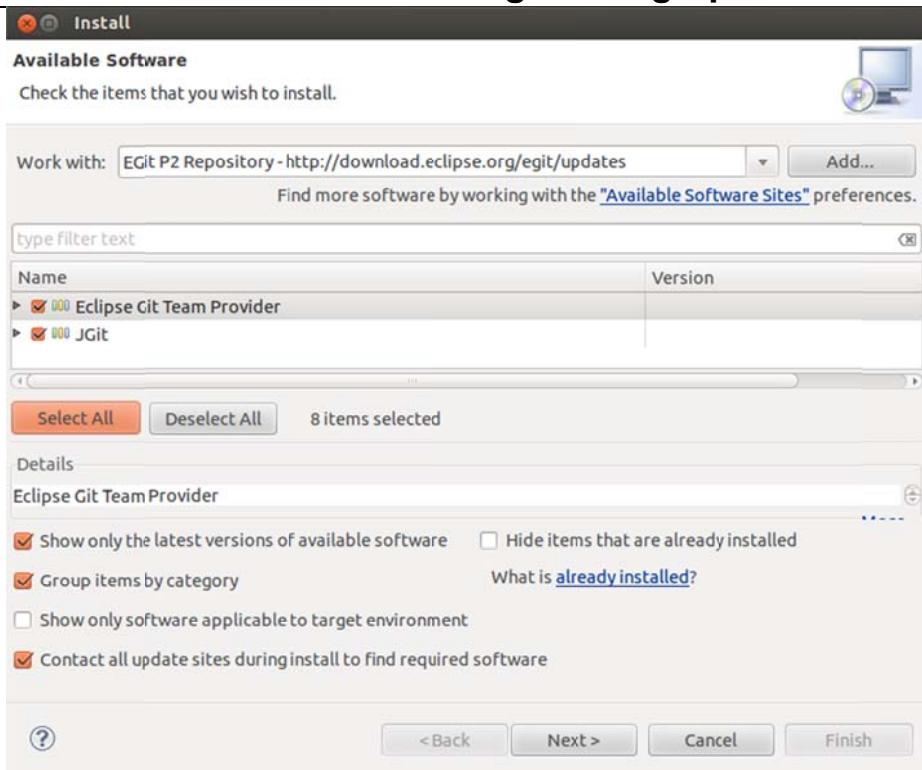
In Eclipse, **Help->Install New Software->Add...**

<http://download.eclipse.org/egit/updates>

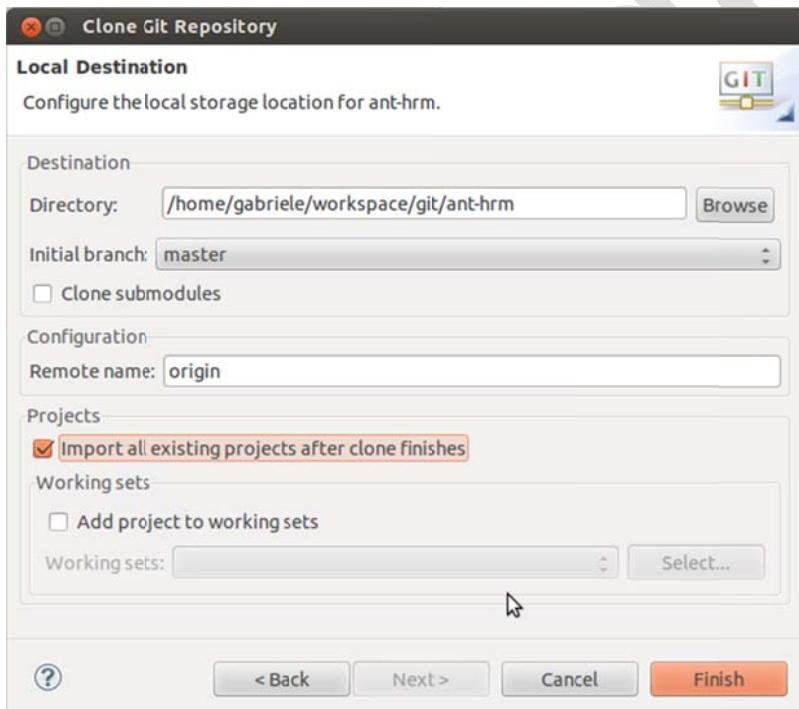


Completare l'installazione

Engineering Specification - confidential



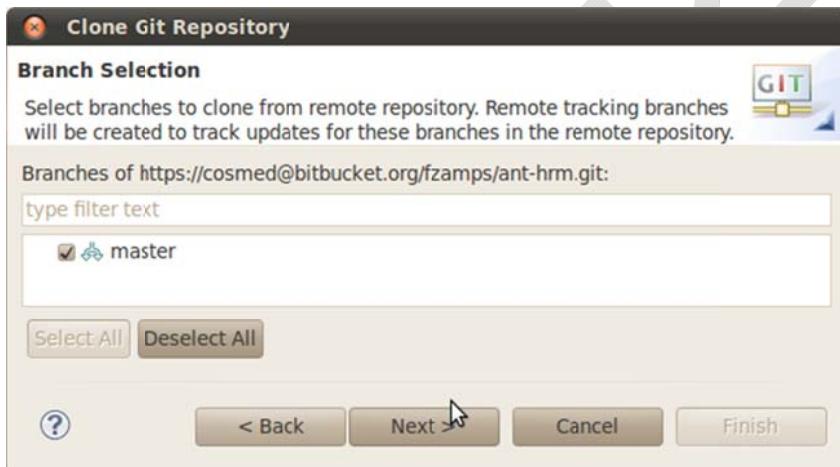
Per clonare in locale un Git Repository esistente, da **Window->Open Perspective->Other.. ->Git Repository Exploring** selezionare il comando “Clone a Git Repository”, inserire i dati di account bitbucket e della repository che interessa clonare

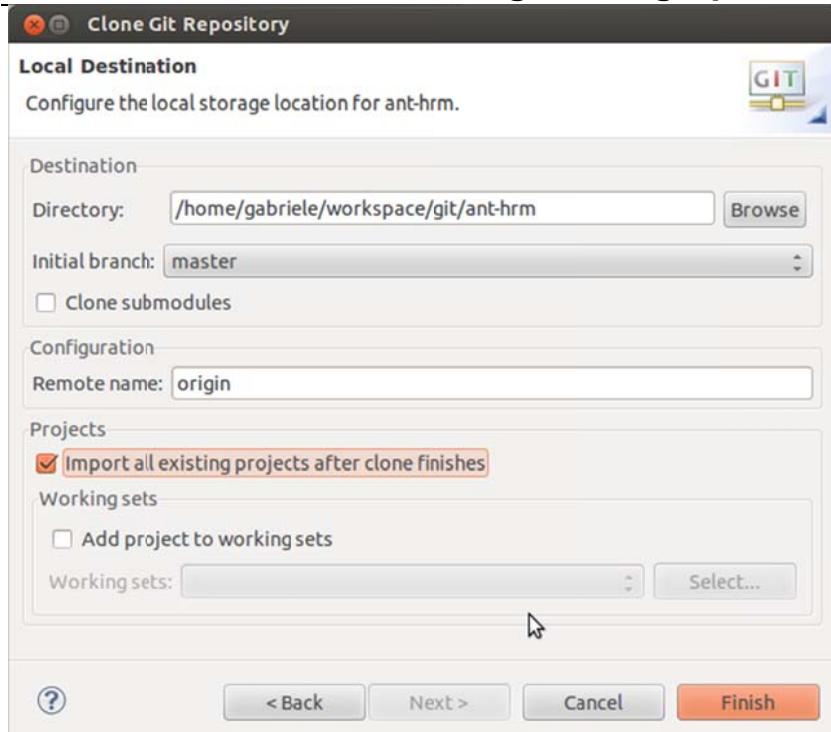


Engineering Specification - confidential



La password è quella dell'account bitbucket (paolob).





31 Librerie esterne

In generale per compilare una applicazione c'è bisogno di librerie esterne, siano esse statiche o dinamiche.

Le librerie statiche sono in genere più efficienti, in quanto gli indirizzi vengono risolti al tempo di compilazione, quindi una function call è un semplice jump. Invece per le librerie dinamiche i simboli devono essere risolti a runtime, cosa che richiede tempo.

Includere una libreria esterna su XNRG è semplificato da alcuni script presenti nella repository git del progetto. In KX la cosa è leggermente meno sofisticata.

Per progetto Spartacus (XNRG):

Sull'host il compilatore gcc per cross-compilare le applicazioni ARM35xx si porta dietro una cartella chiamata sysroot. Essa va distinta dal filesystem del target che si chiama rootfs, dove invece programmi vengono eseguiti.

In particolare sysroot contiene tutte le librerie statiche e gli header file, che servono per compilare e linkare le applicazioni, ma non servono per eseguirle a runtime sul target. Invece le librerie dinamiche e gli eseguibili devono essere presenti sia su sysroot che su rootfs. Gli eseguibili devono stare solo su rootfs.

Nei Makefile le variabili sono:

```
BUILDFS = sdk/current/sysroot  
INSTALL_ROOT = /var/lib/nfs-XNRG-rootfs
```

Host

```
sysroot
libiwrapctrl.a
libiwrapctrl.so
iwrapctrl.h
```

Target

```
rootfs
libiwrapctrl.so
iwrap-client-test
```

La cartella *sysroot* è

```
<dev>/crosstool-ng-1.22.0/arm-epey-linux-gnueabihf/arm-epey-linux-gnueabihf/sysroot
e la si trova sia nella variabile di ambiente $SYSROOT esportata dallo script
<dev>/sdk/current/sysroot
che nel link simbolico
host $ ls- la /sdk/current/sysroot
lrwxrwxrwx 1 gabriele gabriele 31 ago 1 09:33 sdk/current/sysroot -> arm-epey-linux-gnueabihf/sysroot
```

Quando è necessario installare una nuova libreria esterna si procede in questo modo

Copiare pacchetto scaricato dal web (p.es. *tslib-1.1.tar.gz*) in

```
host $ <dev>/fs/extpkg/tslib/
```

All'interno della cartella avremo anche uno script

```
host $ <dev>/fs/extpkg/tslib/gen-pkg.sh
```

che va preparato e infine eseguito.

Lo script estrae l'archivio compresso, lo configura come serve (p.es. specificando se la libreria va compilata statica o dinamica), lo cross-compila e poi lo installa.

Se la libreria è dinamica l'installazione avviene sia nella cartella *sysroot* (in forma espansa) sia nella cartella

```
host $ <dev>/fs/rootfs/packages/
```

in forma compressa (Es. *tslib-pkg.tar.gz*).

Se la libreria è statica, come detto in precedenza, l'installazione avviene solo in *sysroot*.

Quando verrà eseguito lo script che genera il *rootfs*

```
host $ cd <dev>/fs/rootfs/
```

```
host $ sudo ./gen-fs.sh
```

Lo script non farà altro che prendere tutti i pacchetti presenti in *packages* e copiarli nella immagine finale del *rootfs*

```
host $ <dev>/fs/rootfs/fs/
```

Se si vuole aggiornare il FW nella eMMC è necessario anche generare l'immagine con

```
host $ cd <dev>/fs/rootfs/
```

```
host $ sudo ./gen-img.sh
```

Dopo di questo si procede ad aggiornare il FW sul target

31.1 qwt

Libreria per creare grafici

E' una libreria esterna a Qt per le applicazioni Qt Embedded

31.1.1 Qwt per KX

Rimuovere la libreria e i relative link, se presenti, in <qtlibarm>/lib

```
host $ cd <qtlibarm>/lib
```

```
host $ rm libqwt.so*
```

Compilare e installare libreria per arm

```
host $ cd /opt/qwt-6.0.0-rc5/src
```

Engineering Specification - confidential

```
host $ sudo -s
host $ make clean
host $ qmake-arm src.pro
host $ make
host $ sudo cp ..//lib/libqwt.so.6.0.0 <qlibarm>/lib
host $ cd <qlibarm>/lib
host $ sudo ln -s libqwt.so.6.0.0 libqwt.so.6.0
host $ sudo ln -s libqwt.so.6.0.0 libqwt.so.6
host $ sudo ln -s libqwt.so.6.0.0 libqwt.so
```

Installare anche nel target root filesystem

```
host $ cd /opt/qwt-6.0.0-rc5/lib
host $ cp libqwt.so.6.0.0 <rootpath>/<qlibarm>/lib
host $ cd <rootpath>/<qlibarm>/lib
host $ sudo ln -s libqwt.so.6.0.0 libqwt.so.6.0
host $ sudo ln -s libqwt.so.6.0.0 libqwt.so.6
host $ sudo ln -s libqwt.so.6.0.0 libqwt.so
```

Per le applicazioni Qt host (x86), analogamente,

```
host $ cd /opt/qwt-6.0.0-rc5/src
host $ sudo -s
host $ make clean
host $ qmake
host $ make
host $ cp lib/libqwt.so.6.0.0 <qlib>/lib
host $ cd <qlib>/lib
host $ ln -s libqwt.so.6.0.0 libqwt.so.6.0
host $ ln -s libqwt.so.6.0.0 libqwt.so.6
host $ ln -s libqwt.so.6.0.0 libqwt.so
```

Passaggi analoghi vanno fatti per tutte le librerie esterne che devono essere importate in progetti Qt.

Adesso, per utilizzare lqwt in una applicazione Qt, si può prendere un progetto tra quelli di esempio presenti nella cartella dei sorgenti, sottocartella examples.

Da QtCreator aprire il progetto

/opt/qwt-6.0.0-rc5/examples/dials/dials.pro

In dials.pro

Commentare la riga

```
#include (..//examples.pri)
```

Aggiungere

```
LIBS += -lqwt
```

```
INCLUDEPATH += /opt/qwt-6.0.0-rc5/src
```

Ricompile il progetto (supponiamo per ARM)

Assicurarsi di avere nel target filesystem la libreria libQtSvg.so e i relativi link simbolici

Copiare l'eseguibile dials nel target filesystem

```
host $ cp /opt/qwt-6.0.0-rc5/examples/dials/dials <rootpath>/home/root
```

provare

```
target $ cd /home/root
```

```
target $ ./dials -qws
```

31.1.2 Spartacus

Anche il Progetto Spartacus richiede la libreria qwt.

Compilare e installare libreria per arm

```
host $ cp <dev>/sdk/qwt/qwt-6.1.3
```

Engineering Specification - confidential

```
host $ source ../../current/_profile.sh  
host $ vim qwtconfig.pri  
host $ qmake-xnrg qwt.pro  
host $ make  
host $ sudo make install
```

<- seleziona toolchain giusta per ARM
<- modificare (vedi NOTA)

Nota: le modifiche al config sono le seguenti

QWT_INSTALL_PREFIX = \$(SYSROOT)/usr

Commentare le successive assegnazioni di QWT_INSTALL_PREFIX

#QWT_CONFIG += QwtDII

(la libreria deve essere statica)

#QWT_CONFIG += QwtOpenGL

#QWT_CONFIG += QwtExamples

31.2 libusb

Libreria per permettere di accedere da user space alla porta USB Host.

Download site:

<http://sourceforge.net/projects/libusb/>

Estrarre il pacchetto in /opt

```
host $ tar xjf libusb-1.0.8.tar.bz2
```

Compilare per arm

```
host $ cd /opt/libusb-1.0.8  
host $ sudo -s  
host $ make clean  
host $ ./configure --host=arm-none-linux-gnueabi  
host $ make
```

Installare

```
host $ cp libusb/.libs/libusb-1.0.so.0.0.0 <qlibarm>/lib  
host $ cp libusb/.libs/libusb-1.0.so.0.0.0 <rootpath>/<qlibarm>/lib
```

Sia in <qlibarm>/lib che in <rootpath>/<qlibarm>/lib creare i relativi links

```
host $ ln -s libusb-1.0.so.0.0.0 libusb-1.0.so  
host $ ln -s libusb-1.0.so.0.0.0 libusb-1.0.so.0  
host $ ln -s libusb-1.0.so.0.0.0 libusb-1.0.so.0.0
```

Copiare anche nel target root filesystem, creando anche lì i link simbolici.

Per le applicazioni Qt host, procedere analogamente a sopra, ma usare

```
host $ ./configure
```

e installare in <qlib>/lib

Per utilizzare la libusb in una applicazione Qt, editare projectname.pro e aggiungere

LIBS += -lusb-1.0

31.3 QExtSerialPort

Download site:

<http://sourceforge.net/projects/qextserialport/files/qextserialport/1.1/qextserialport-1.1.tar.gz/download>

Libreria per interfacciarsi a una porta seriale

Compilare libreria per arm

Estrarre il pacchetto in /opt

```
host $ tar zxf qextserialport-1.1.tar.gz
```

Compilare per arm

```
host $ cd /opt/qextserialport  
host $ sudo -s  
host $ make clean  
host $ qmake-arm  
host $ make
```

Installare

```
host $ cp build/libqextserialport.so.1.0.0 <qlibarm>/lib  
host $ cp build/libqextserialport.so.1.0.0 <rootpath>/<qlibarm>/lib
```

Sia in <qlibarm>/lib che in <rootpath>/<qlibarm>/lib creare i relativi links

```
host $ ln -s libqextserialport.so.1.0.0 libqextserialport.so.1.0  
host $ ln -s libqextserialport.so.1.0.0 libqextserialport.so.1  
host $ ln -s libqextserialport.so.1.0.0 libqextserialport.so
```

Copiare anche nel target root filesystem, creando anche lì i link simbolici.

Per le applicazioni Qt host, procedere analogamente a sopra, ma usare

```
host $ qmake
```

e installare in <qlib>/lib

Per utilizzarla in una applicazione Qt, editare projectname.pro e aggiungere

LIBS += -lqextserialport

INCLUDEPATH += /opt/qextserialport

31.4 Dropbear

Download site:

<http://matt.ucc.asn.au/dropbear/releases/>

dropbear è una versione di SSH 2 server e client, più adatto per applicazioni embedded.

La versione full-featured richiederebbe zlib. Qui ne facciamo a meno

Scaricare pacchetto dropbear. P.es. dropbear-0.51 (da <http://matt.ucc.asn.au/dropbear/releases/>)

```
host$ wget http://matt.ucc.asn.au/dropbear-0.51.tar.gz
```

Copiarlo in /home/<user>/Program/extlibs, estrarlo

```
host $ tar zxf dropbear-x.xx.tar.gz
```

Configurare per arm una versione “minimale” senza zlib

```
host $ cd /dropbear-0.51  
host $ export CC=arm-linux-uclibcgnueabi-gcc  
host $ ./configure --host=<armtoolch> --disable-zlib --disable-syslog
```

Conmpilare

```
host $ make PROGRAMS="dbclient dropbear dropbearkey dropbearconvert scp ssh" STATIC=0 MULTI=1  
prima di ri-configurare utilizzare make clean
```

Comprimere

```
host $ <armtoolch>-strip dropbearmulti
```

Installare sul target

```
host $ cp dropbearmulti <rootfs>/usr/sbin
```

Creare link simbolici sul target

```
host $ cd <rootfs>/usr/sbin  
host $ ln -s dropbearmulti dbclient  
host $ ln -s dropbearmulti dropbear  
host $ ln -s dropbearmulti dropbearkey  
host $ ln -s dropbearmulti dropbearconvert
```

```
host $ ln -s dropbearmulti scp  
host $ ln -s dropbearmulti ssh
```

Sul target creare le chiavi rsa e dss (tipicamente questo viene fatto da uno script di init presente in /etc/init.d)

```
target $ mkdir /etc/dropbear  
target $ dropbearkey -t dss -f /etc/dropbear/dropbear_dss_host_key  
target $ dropbearkey -t rsa -f /etc/dropbear/dropbear_rsa_host_key
```

lanciare dropbear (tipicamente questo viene fatto da uno script di init presente in /etc/init.d)

```
target $ dropbear -a
```

Da host aprire una connessione SSH con il target (cfr. paragrafo *SSH login su Target*)

```
host $ ssh root@<ipaddr>
```

Known problem:

Questa soluzione presenta un problema in quanto all'atto di aprire la connessione SSH richiede la password (?), mentre il file */usr/sbin/dropbearmulti* preso dal filesys TI funziona e non chiede password (CR). Pertanto decidiamo di copiarlo da lì.

31.5 Util-linux

Util-linux è un pacchetto di utilità generali.

31.5.1 Util-linux per KX

Il package dei sorgenti è

<dev>/fs/extpkg/util-linux/util-linux-2.24.tar.gz

Lo script di compilazione è

<dev>/fs/extpkg/util-linux/gen-pkg.sh

31.5.2 Util-linux per Spartacus

Il package dei sorgenti è

<dev>/fs/extpkg/util-linux/util-linux-2.24.tar.gz

Lo script di compilazione è

<dev>/fs/extpkg/util-linux/gen-pkg.sh

31.6 fbset

fbset è un pacchetto di utilità generali.

31.6.1 fbset per Spartacus

Il package dei sorgenti è

<dev>/fs/extpkg/fbset/fbset-2.1.tar.gz

Lo script di compilazione è

<dev>/fs/extpkg/fbset/Makefile

32 Database

32.1 SQL Lite

SQL Lite è una implementazione di database SQL, leggero e locale (nel senso che non permette l'accesso remoto).

SQL Lite è sia una libreria che tool che, da riga di comando, permette di creare le tabelle del database.

LibQtSqlLite è la libreria di Qt che permette di usare il database

```
host $ cd /opt/sqlite-3.7.3/
host $ ./configure
host $ make
host $ ./sqlite3 mydb
      (creo database)
>> create table mytable(nome varchar(30), cognomen varchar(30), statura int);
      (creo una tabella)
>> select * from mytable;
      (visualizzo contenuto tabella)
>> select nome from mytable;
      (visualizzo solo i nomi della tabella)
>> select * from mytable where nome="gabry";
      (visualizzo solo il nome se uguale a "gabry")
>> select * from mytable where nome in {"gabry", "gigi"};
      (per uscire)
>>.exit
```

Creiamo un database mydb nel modo appena descritto

Copiamo mydb nella cartella di progetto Qt

```
host $ cp mydb /home/<user>/qtprojects/provadb/
```

32.1.1 Sqlite per KX

Il package dei sorgenti è

```
<dev>/fs/extpkg/sqlite3/sqlite-autoconf-3080500.tar.gz
```

Lo script di compilazione è

```
<dev>/fs/extpkg/sqlite3/gen-pkg.sh
```

32.1.2 Sqlite per Spartacus

Il package dei sorgenti è

```
<dev>/fs/extpkg/sqlite3/sqlite-autoconf-3080500.tar.gz
```

Lo script di compilazione è

```
<dev>/fs/extpkg/sqlite3/gen-pkg.sh
```

32.2 KX database

Sul KX esiste una implementazione SQLite per il database dell'archivio shede anagrafiche.

Si trova in

```
target $ /mnt/archive/database/archive.sqlite
```

E' formato da 4 tabelle, la prima delle quali contiene le schede anagrafiche dei soggetti.

E' possibile interagire a basso livello con il database con dei comandi

```
target $ cd /usr/share/X/kmain
./sql databases/archive.sqlite "select * from suubjects"
ARENA FRANCESCO    1970/12/31      0     170.0    75.0    9      b6c2ceb6-963f-40e9-a326-acfc34c7fed4
Lastname ARIANNA   1990/06/27      4     150.0    50.0    9      fc0984fe-bfec-48ca-95f2-d4dc20c81b50
lastnameY firstname 1999/02/31      0     150.0    50.0    9      498d9246-ee06-4c92-9768-2a85720fb693
```

L'ultima colonna indica l'UUID del soggetto.

33 Arago matrix GUI

http://processors.wiki.ti.com/index.php/Matrix_Users_Guide

Matrix GUI è un'applicazione QT/WebKit/C++ (un tipo di web browser) basata su HTML e Cascading Style Sheets (CSS) per lanciare applicazioni qualsiasi da un menu strutturato a pulsanti facilmente personalizzabile. Ciascun pulsante può lanciare un'applicazione, aprire o chiudere un sub menu, o visualizzare una pagina web presa da internet.

Per aggiungere nuovi pulsanti non serve ricompilare il progetto, basta editare file HTML e immagini.

I file html si trovano in

HOST: <sdk>/example-applications/matrix-gui-e-1.3/am180x/html

TARGET: <rootpath>/usr/share/matrix/html

Mentre le immagini sono in

HOST: <sdk>/example-applications/matrix-gui-e-1.3/images

TARGET: <rootpath>/usr/share/matrix/images

33.1 Aggiungere il submenu kx a matrix_gui

- 1) Editare <sdk>/example-applications/matrix-gui-e-1.3/am180x/html/menu_main_2.html
Copiare la sezione "Ethernet" e chiamarla "KX Mobile CPET"
File del submenu: menu_kx.html
Icona: 3d1-icon.png

Fare una copia di <sdk>/example-applications/matrix-gui-e-1.3/am180x/html/menu_etherne.html e chiamarla menu_kx.html

Aggiungere icone spiro-icon e spiro-dead in <sdk>/example-applications/matrix-gui-e-1.3/images

Editare menu_kx.html

Title: "KX Mobile CPET"

Header: "X KX Mobile CPET"

Definire nuovo oggetto con

appName: "Spirometry"

iconName: spiro-icon.png

appDesc:desc_comingsoon.html

Installare nel target filesystem

host \$ cd <sdk>

host \$ make install

34 Licenze

Le licenze più usate sono

GPLv2, LGPLv2, GPLv3, LGPLv3, BSD, APACHEv2, MPLv1.1, TI_TSPA, openssl, MIT, ECLIPSEv1

34.1 GPL

Usando un sorgente GPL (GNU General Public Licence), originale o modificato, si ha l'obbligo, se richiesto, di fornire il sorgente modificato come GPL. Attualmente esistono due versioni, v2 e v3. Linux è GPLv2. Sicuramente si deve fornire il sorgente delle patch applicate. Per quanto riguarda i moduli (compresi quindi i drivers) bisogna vedere.

34.2 LGPL

Usando un sorgente LGPL (GNU Lesser General Public Licence) si ha l'obbligo, se richiesto, di fornire il sorgente come LGPL, ma solo se lo si è modificato. L'obbligo esiste solo se si linka staticamente.

Ad esempio la libreria grafica Qt può essere rilasciata come LGPL v3.

Per il progetto KX, che linka la libreria Qt 4.7 staticamente, questo significa never rendere disponibile i programmi *qmlrun*, le librerie *plug* e *Iplug*, il *main* e la cartella *qml* (in pratica la GUI dell'applicazione).

Per quanto riguarda XNRG la Qt 5.9 verrà utilizzata dinamica, questo significa rendere disponibili solamente i binari e gli script di linking.

34.3 Tivoization

Il termine proviene in riferimento all'utilizzo che TiVo fa del software rilasciato sotto licenza GNU GPL nel proprio videoregistratore digitale. Il software TiVo incorpora il kernel Linux e del software GNU, entrambi i quali sono rilasciati sotto GPLv2. Tale GPLv2 richiede ai distributori di rendere il corrispondente codice sorgente disponibile a ogni persona che riceve il software. L'obiettivo di questa richiesta è consentire agli utenti del software rilasciato sotto GPL di modificare il software per meglio adattarlo ai propri scopi. Tuttavia, Stallman crede che TiVo abbia aggirato questa norma, facendo sì che il prodotto esegua solo programmi la cui **firma digitale** è fra quelle autorizzate dal produttore di TiVo. Così, mentre TiVo aderisce alle richieste della GPLv2 e permette la modifica del codice sorgente, ogni software modificato non sarà eseguibile sull'hardware TiVo. Secondo Torvalds, l'utilizzo di firme digitali private è necessario per migliorare la sicurezza dei sistemi.

Nella GPLv3 vi è una clausola contro la tivoization, ma questa è comunque permessa nel software per sistemi elettromedicali. Comunque Linux NON è GPLv3.

35 Device Drivers

Todo

35.1 Creare e caricare un driver module

Apriamo il template driver

```
host $ cd <mymodules>/template
host $ cat template.c
/*
 * Kernel template module
 *
 * Description: TODO
 *
 * Copyright (c) 2012 by Gabriele Filosofi <gabrielef@X.it>
```

Engineering Specification - confidential

```
*  
* This program is free software; you can redistribute it and/or  
* modify it under the terms of the GNU General Public License as  
* published by the Free Software Foundation version 2.  
*  
* This program is distributed .as is. WITHOUT ANY WARRANTY of any  
* kind, whether express or implied; without even the implied warranty  
* of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the  
* GNU General Public License for more details.  
*/  
  
#include <linux/kernel.h>  
#include <linux/version.h> /* info about the kernel version */  
#include <linux/vermagic.h> /* info about the kernel version */  
#include <linux/sched.h> /* defs for a lot of kernel API used by the driver */  
#include <linux/init.h>  
#include <linux/module.h>  
#include <linux/interrupt.h>  
#include <linux/platform_device.h>  
#include <linux/spinlock.h>  
#include <linux/compiler.h>  
#include <linux/device.h>  
#include <linux/io.h>  
#include <linux/clk.h>  
#include <linux/delay.h>  
#include <linux/err.h> /* error codes */  
#include <linux/errno.h> /* error codes */  
#include <linux/slab.h>  
  
#include <linux/fs.h>  
#include <linux/types.h>  
#include <linux/cdev.h>  
  
/* char device driver stuff */  
static dev_t dev; /* device structure to make entry in device. It holds the major and minor numbers */  
#define NUM_DEVICES 4 /* total num of contiguous dev number to be requested */  
#define DEV_NAME "mydevice" /* the name that will appear in /proc/devices and sysfs */  
#define DEV_MAJOR 271 /* static major number associated with this device (not already assigned in  
Documentation/devices.txt) */  
  
//these parms can be passed at insmod time (e.g. $ insmod template.ko howmany=10 whom="Mom") or at boot time (for built-in modules)  
static char *whom = "world";  
static int howmany = 1;  
//syntax: module_param(param_name, param_type, param_permission_for_sys_entry);  
module_param(howmany, int, S_IRUGO); //perm is S_IRUGO (read only) so you can read only it (e.g. $ cat  
/sys/module/template/parameters/howmany)  
module_param(whom, charp, 0); //perm is 0 so param is not representend at all in /sys  
  
/**  
 * template_func  
 */  
void template_func(void)  
{  
//TODO  
}  
EXPORT_SYMBOL(template_func); /* exports symbos to the kernel */  
//EXPORT_SYMBOL_GPL(template_func); /* limits use of the symbol to GPL-licensed modules */  
  
/**  
 * template_init: Module init  
 */  
static int __init template_init(void)  
{  
    int err=0;  
    int i;  
  
    for(i=0;i<howmany;i++)  
        printk(KERN_ALERT "Hello %s!\n",whom);  
  
    printk(KERN_INFO "The kernel version is " UTS_RELEASE "\n"); //prints Linux version like "2.6.38"  
  
    err = alloc_chrdev_region(&dev, 0, NUM_DEVICES, DEV_NAME); /* the kernel assign you a device major number,  
dynamically (preferred). */  
    // The assigned number is visible in /proc/devices  
    /* use this if you already know static device  
numbers */  
}
```

Engineering Specification - confidential

```
//err = register_chrdev_region(dev, NUM_DEVICES, DEV_NAME); /* request of device numbers to kernel */
if (err < 0) {
    printk(KERN_WARNING "mydevice: unable to get major %d\n", DEV_MAJOR);
    return err;
}

/*
 * init and register to the kernel any facility offered by the module */
//err = register_this(ptr1, "name");
if(err) goto error;

return 0;

error:
    return err;
}

/***
 * template_exit: Module exit
 */
static void __exit template_exit(void)
{
//    dev_t dev = MKDEV(DEV_MAJOR, 0);
//    unregister_chrdev_region(dev, NUM_DEVICES);

    printk(KERN_ALERT "Goodbye template!\n");
}

module_init(template_init);
module_exit(template_exit);

MODULE_LICENSE("GPL"); /* The license governing this module */
MODULE_AUTHOR("Gabriele Filosofi <gabrielef@X.it>");
MODULE_DESCRIPTION("Kernel template module");
MODULE_ALIAS("KTM"); /* An alternate name for this module */
MODULE_VERSION("1.0"); /* A module version. Appears in /sys (e.g. cat /sys/module/template/version) */
```

Per compilare il modulo è utile creare un Makefile. Il seguente esempio di Makefile permette di compilare il modulo per il target o per l'host

```
host $ cat Makefile
#
# Makefile for Kernel template module
# Copyright (c) 2012 by Gabriele Filosofi <gabrielef@X.it>
#
ifeq ($(KERNELRELEASE),)
obj-m := template.o
else
    KERNELDIR = /home/gabriele/sdk_5_03_02_00/board-support/src/kx/kernel/linux-01.00
#    KERNELDIR = /usr/src/linux-headers-2.6.32-38-generic
    PWD := $(shell pwd)
default:
    @echo =====
    @echo Building Kernel template module
    @echo =====
    $(MAKE) -C $(KERNELDIR) ARCH=arm CROSS_COMPILE=arm-arago-linux-gnueabi- M=$(PWD) modules
#    $(MAKE) -C $(KERNELDIR) M=$(PWD) modules
endif
```

Compilazione del programma

```
host $ make template
```

Copia nel filesystem

```
host $ cp template.ko <rootpath>/home
```

Prova sul target

```
target $ insmod template.ko
Hello world!
The kernel version is 2.6.37-01.00
```

Per caricare un modulo <module>.ko è meglio utilizzare *modprobe*, che carica anche tutti gli eventuali altri moduli da cui <module> dipende

```
target $ modprobe <module>
```

Per scaricare un modulo

```
target $ rmmod <module>
```

Per fare in maniera che ci pensi il sistema a caricare il modulo <module>.ko si deve editare il file /etc/modules e aggiungere il nome <module> all'interno

35.2 Integrare un driver module nel kernel tree

Creare una directory

```
host $ cd <linux>
host $ mkdir -p drivers/mymod
```

Copiarci i sorgenti

```
host $ cp mymod.c drivers/mymod
```

Creare Kconfig e Makefile

```
host $ cd drivers/mymod
host $ cd drivers/mymod
```

```
host $ cat drivers/mymod/Kconfig
config MYMOD
    tristate "Mymod support"
    default y
    help
        If you say yes here you get support for the Mymod hardware.

        This driver can also be built as a module. If so, the module
        will be called mymod.

host $ cat drivers/mymod/Makefile
#
# Makefile for the Mymod-specific device drivers.
#
# Each configuration option enables a list of files.
obj-$(CONFIG_MYMOD)          += mymod.o

host $ cat drivers/Kconfig
config MYMOD
    tristate "Mymod support"
    default y
    help
        If you say yes here you get support for the Mymod hardware.

        This driver can also be built as a module. If so, the module
        will be called mymod.

host $ cat drivers/Makefile
#
...
source "drivers/mymod/Kconfig"
endmenu
```

Per implementare meccanismi di sincronizzazione e di mutua esclusione lato kernel linux mette a disposizione una serie di strumenti

35.3 Programming techniques - mutex

Se occorre evitare l'accesso concorrente di più processi su una stessa parte di codice, conviene proteggerla con un mutex

```
#include <linux/mutex.h>
struct mutex amutex;
...
mutex_init(amutex);
...
mutex_lock(amutex);
<.. code to be locked ..>
mutex_unlock(amutex);
```

35.4 Programming techniques - spinlock

Patch RT

Con il kernel del KX (versione classica) anche se in teoria si può cambiare la priorità di un interrupt rispetto a un altro, qualsiasi interrupt ha priorità maggiore su qualsiasi processo. Questo crea latenze e impredictibilità.

Sul progetto XNRG la versione di linux utilizzata contempla anche la patch RT (Real Time).

La patch RT cerca di portare tutto nel contesto di un processo in maniera tale che sia lo scheduler a guidare tutto quanto, anche assegnare le priorità degli interrupt handler.

P.es. l'interrupt handler del bus I2C gira in un kernel thread, e lo scheduler è in grado di cambiare la priorità di quel thread.

Patch RT e spinlock

Senza la patch RT, prendendo uno spinlock in contesto interruzione le interrupt venivano disabilitate (per evitare deadlock).

La patch RT ridefinisce alcune funzioni di base, tra cui lo spinlock. I nuovi spinlock in realtà prendono un mutex (semaforo), perché tanto tutto è stato portato nel contesto di un processo.

Quindi anche la *spinlock_irq_save()* non le disabilita più le interruzioni. Cioè *spinlock_irq_save()* espande a *RT_spinlock* che in realtà è un mutex.

I vecchi spinlock esistono ancora ma sono stati chiamati *raw_spinlock*.

La patch in definitiva va a toccare solo il file spinlock.h. In questo modo tutte le chiamate del kernel a spinlock espanderanno a *RT_spinlock*. Solo per le interruzioni hard la patch usa *raw_spinlock*.

Problema che avevamo su XNRG a causa della patch:

Nell'interrupt handler della turbina leggevamo un gpio (inex) con la chiamata *gpio_get_value()*, implementata nella gpiolib del kernel. Gpiolib prende uno *spinlock_irq_save* (per ricavarsi un descrittore del gpio) che con la patch non disabilita più le interruzioni. Certamente anche gli altri che prendevano lo spinlock non le disabilitavano (come l'heartbit LED) e quindi poteva capitare un deadlock: l'heartbit gpio (in un contesto processo) prendeva lo spinlock, poi arrivava il driver turbina (in contesto interruzione) che prendeva lo stesso spinlock, e tutto si bloccava.

Una possibile soluzione poteva essere cambiare tutti i spinlock dei gpio in *raw_spin_lock*. Ma questa strada andrebbe a vanificare i benefici della patch RT.

La soluzione è stata abilitare un altro interrupt attivo su tutti e due i fronti del gpio dell'inex. Questo gpio (grazie al FF) cambia stato solo quando la paletta cambia verso (raramente rispetto al caso KX, dove non c'è il FF). Inoltre si può usare un interrupt "threaded" (cioè in contesto processo) in quanto se viene servito con un po' di ritardo non succede niente dato che in quel momento il flusso è quasi zero. Poi, nell'interrupt handler si aggiorna una variabile con lo stato dell'inex. Nell'interrupt handler della linea veloce (hard) si prende il periodo, si legge la variabile e li si mette nel buffer.

Differenza di implementazione del driver turbina tra KX e XNRG:

Nel KX non vi è una patch RT, ma vi sono le interrupt FIQ che risolvono.

Nel XNRG non vi sono le FIQ ma vi è la patch RT che risolve.

Le uniche interruzioni hard rimaste nel XNRG sono quattro

1 per la driver turbina

1 per il timer (il tick di sistema)

2 per la ethernet (npn usata)

35.5 Programming techniques - tasklet

Todo

35.6 Programming techniques - atomic operations

Todo

35.7 Programming techniques - lists

Todo

35.8 Programming techniques - delay

Aggiungere un delay in msec. Può andare in sleep

```
msleep(<ms>);
```

Aggiungere un delay in usec. Può comunque essere interrotto quindi il tempo può essere maggiore

```
udelay(<us>);
```

Aggiungere un delay in usec che sarà compreso tra un minimo e un massimo

```
usleep_range(<usmin>, <usmax>);
```

35.9 Programming techniques - interrupt

Eseguire una parte di codice con le interrupt disabilitate

```
unsigned long flags;
```

```
..
```

```
local_irq_save(flags);
```

```
<.. code ..>
```

```
local_irq_restore(flags);
```

36 Creare una distribuzione from scratch

I prodotti embedded possono avere CPU anche molto performanti, che supportano tranquillamente distribuzioni desktop come Ubuntu, Debian, Fedora, Red Hat, ma difficilmente ciò viene fatto. Si preferisce preparare il filesystem da installando solo ciò che serve.

36.1 Buildroot

Buildroot (<https://buildroot.org/>) è un progetto open source che fornisce un modo semplice ed efficiente per generare toolchain, filesystem, kernel e bootloader per sistemi embedded attraverso cross-compilazione.

Partiamo da un target hardware bare metal (cioè vuoto o senza SO) come la BeagleBone Black, basata su TI am3358.

Sull'host accertiamoci di avere ncurses development

```
host $ apt-get install ncurses-dev
```

Inoltre possiamo decidere di fare un clone del kernel ufficiale della beaglebone (versione 4.1.20) con

```
host $ git clone git://github.com/beagleboard/linux.git
```

Inoltre possiamo decidere di fare un clone dell'ultima versione di u-boot con

```
host $ git clone git://git.denx.de/u-boot.git
```

Dal sito di Buildroot verificare quale è l'ultima versione disponibile di buildroot. Sia 2016.02.

Scarichiamo il file di archivio ed estraiamone il contenuto

```
host $ tar xf buildroot-2016.02.tar.gz
```

In alternativa si può anche scaricare la git repository da <git://git.buildroot.net/buildroot>

Entriamo nella cartella

```
host $ cd buildroot-2016.02
```

Cerchiamo il file di configurazione del target specifico in nostro possesso,

configs/beaglebone_defconfig, quindi

```
host $ make beaglebone_defconfig
```

Lanciamo il configuratore

```
host $ make menuconfig
```

Per la nostra Beaglebone Black effettuiamo le seguenti configurazioni.

Target options

- ARM little endian
- ELF
- cortex-A8
- EABIhf
- VFPv3-D16
- ARM

Toolchain

In 'Toolchain' menu we will customize some options:

- Kernel headers: we will use Kernel version 4.1.x, so in 'Kernel Headers' menu choice 'Linux 4.1.x kernel headers'

Engineering Specification - confidential

- C library: instead of 'uClibc' we will use the GNU library. Select 'glibc' as 'C library' and '2.22' under 'glibc version'
- Binutils: select 'binutils 2.25.1' under 'Binutils Version'
- GCC: we will use the latest gcc version, select 'gcc 5.x' as 'GCC compiler Version'
- G++: we will need G++ in the toolchain for some packages we are going to install, mark 'Enable C++ support' for that

System

Under 'System configuration' set system wide options:

- Set a 'System hostname' (eg. Idroma15)
- Set a 'System banner' (eg. Welcome to Linux Day 2015)
- Mark 'Enable root login with password' and set a 'Root password'
- Enter 'Run a getty (login prompt) after boot' menu and set 'ttyS0' in 'TTY port' instead of 'ttyO0'. This because BeagleBone defconfig we are going to use for Kernel configuration replace ttyO with ttyS (config 'CONFIG_SERIAL_8250_OMAP_TTYO_FIXUP=y')
- Mark 'remount root filesystem read-write during boot'
- Leave 'board/beaglebone/post-image.sh' as 'Custom scripts to run after creating filesystem images'

Under 'Filesystem images' set 'ext4' under 'ext2/3/4 variant'.

Kernel

Default Kernel git repository, 'git://git.ti.com/ti-linux-kernel/ti-linux-kernel.git', is too old, we will use official BeagleBoard repository.

- Leave as 'Kernel version' the value 'Custom Git repository'
- Set as 'URL of custom repository' the URL 'git://github.com/beagleboard/linux.git'
- Set as 'Custom repository version' the commit 'ef6d5016630d120d503cb5a85be3bdd4e691d288' that is the latest version of bb.org_defconfig for Kernel version 4.1.10
- Empty 'Custom kernel patches' field, we don't need any patch for the Kernel
- This repository has an in-tree defconfig, so set 'Using an in-tree defconfig file' in 'Kernel configuration' field and write 'bb.org' in 'Defconfig name'
- We will use 'zImage' as 'Kernel binary format'
- In 'Device tree source' set 'Use a device tree present in the kernel' and then 'am335x-boneblack' in 'Device Tree Source file names'

Nota:

Da www.kernel.org scarichiamo l'ultima versione di linux longterm (**linux-4.4.7.tar.xz**) ed estrainamo il contenuto sull'host.

Su XNRG, al 11/01/2017 abbiamo la versione 4.4.23 (4.4.23-ti-rt-r51-X)

Da <http://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/tree/arch/arm/configs/?id=v4.4>

si può vedere qual è il nome della configurazione valida per la beaglebone (**omap2plus_defconfig**)

Quindi il nome da specificare nel configuratore è **omap2plus**

Da <http://git.kernel.org/cgit/linux/kernel/git/torvalds/linux.git/tree/arch/arm/boot/dts/?id=v4.4>

si può vedere il nome del file DTS per il target desiderato (**am335x-boneblack.dts**)

Quindi il nome da specificare nel configuratore è **am335x-boneblack**

U-Boot

Default version of U-Boot is too old to be compiled with the GCC 5.x version we selected before.

- As 'U-Boot board name' set 'am335x_boneblack'. This is the defconfig for BeagleBone Black in U-Boot
- Set as 'U-Boot Version' the preset value '2015.07', that is the latest stable version as of today
- We will need MLO file as SPL and u-boot.img as U-Boot binary, so keep the values 'u-boot.img' under 'U-Boot binary format', mark 'Install U-Boot SPL binary image' and then set 'MLO' in 'U-Boot SPL binary image name'

Other stuff

The most important settings are now ready. You could compile as it is, but we would like to add some third party software.

- Under 'Target packages' mark 'Show packages that are also provided by busybox'
- Go back to 'System configuration' menu and set 'bash' as '/bin/sh'
- Under 'Target packages' again mark:
 - 'Development tools'
 - 'git'
 - 'grep'
 - 'sed'
 - 'Hardware handling'
 - 'i2c-tools'
 - 'Networking applications'
 - 'dhpcd'
 - 'openssh'
 - 'wget'
 - 'Shell and utilities'
 - 'dialog'
 - 'file'
 - 'screen'
 - 'sudo'
 - 'time'
 - 'which'
 - 'System tools'
 - 'htop'
 - 'login utilities'
 - 'tar'
 - 'util-linux'
 - 'install utilities'
 - 'Text editors and viewers'
 - 'vim'
 - 'install runtime'

Compiliamo,

```
host $ make
```

(Il processo è spiegato per bene su <https://www.youtube.com/watch?v=oqM3VcJYXas>)

Tutti i file immagine prodotti si troveranno nella sottocartella *output/images*.

Creare una SD card con due partizioni, una FAT32 bootable (almeno 400MB) chiamata "boot" e il resto ext4 per il root filesystem chiamata "rootfs".

Copiamo tutti i file ad eccezione di rootfs.ext4 (o rootfs.ext2) nella prima partizione FAT32.

Usiamo dd sulla seconda partizione EXT4 per copiarvi rootfs.ext4.

```
host $ sudo dd if=output/images/rootfs.ext4 of=/dev/sdb2 bs=1M
```

dove /dev/sdc2 è la partizione "rootfs"

Inserire la SD card nel target e riavviare.

36.2 Yocto

Yocto, che ora include OpenEmbedded, è il sistema utilizzato dai moduli Freescale iMX6 (adesso NXP).

Per scaricare il BSP della Freescale

<http://freescale.github.io/>

Creare, configurare e compilare un nuovo progetto (p.es. "test")

```
. setup_environment test
```

Editare il file

test/conf/local.conf

Specificando il tipo di cpu (IMX6UL)

Editare il file

test/conf/bblayer.conf

Specificando i meta-..

Compilare e installare usando una delle "ricette" disponibili (p.es. core-image-minimal) con
`bitbake core-image-minimal`

I file generati si troveranno nella cartella

tmp/deploy/image/test-geb6ul/

Copiare questi files nella SD card, utilizzando il comando *dd*

Aggiungere dei pacchetti alla distribuzione (p.es. apache)

Vedere dove si trova il pacchetto apache con

```
bitbake -s | grep apache*
```

Si vedrà che apache è nel meta-webserver.

Editare il file

test/conf/bblayer.conf

aggiungendo la linea

`$(BSPDIR)/sources/meta-openembedded/meta-webserver`

Editare il file

test/conf/local.conf

aggiungendo apache2

Nota: se il pacchetto desiderato non si trova allora cercare il meta su internet e poi scaricarlo con git clone git://..

Compilare il kernel aggiungendo qualche driver alla configurazione

Andare nella cartella dei sorgenti linux con il comando

```
bitbake -c devshell linux-imx
```

Esportare le variabili di ambiente necessario per cross-compilare

Le si trova al capitolo 10 del manuale, ma con la seguente differenza

```
export LOADADDR=0x80008000
```

Configurare il kernel con

```
make menuconfig
```

Scegliere il driver che si vuole aggiungere

Ricompile la "ricetta" con bitbake

Compilare il singolo pacchetto (p.es. php)

```
bitbake php
```

Nota: in questo caso il pacchetto viene sì compilato, ma non installato nel file di immagine del filesystem.

37 KX remote debug

Per poter connettersi in modo sicuro dall'azienda (X) a un prodotto (KX) situato presso un cliente vi sono diverse soluzioni.

37.1 Dynamic DNS

Dyndns è un servizio che permette a una macchina (p.es. il KX con S/N 2014100012) di registrare il suo IP (statico o dinamico che sia, **ma necessariamente pubblico**) presso un server Dynamic DNS, associandolo a un URL (p.es. *kx-2014100012.dyndns.org*)

A quel punto il KX può essere chiamato in SSH da X facendo riferimento all'URL.

Il servizio Dynamic DNS è gratuito.

Il problema è che l'IP del KX deve essere necessariamente pubblico.

37.2 SSH tunneling

Si può sfruttare un SSH tunneling. SSH tunneling è la "VPN dei poveri".

X deve attivare

- Una VM, con indirizzo privato <VM-IP> e un utente <VM-user>. La VM avrà p.es. una distribuzione debian e installato openSSH server configurato per il tunneling. Abilitiamo anche l'opzione di aggiornamenti automatici, per tenerlo aggiornato in termini di sicurezza
- Un indirizzo IP pubblico sul Firewall, <public-IP>
- Una *destination NAT* sul Firewall, che esegue il forwarding dei messaggi destinati a <public-IP>:60000 su <VM-IP>:22

1) Il KX, avente utente "root", una volta ottenuto un indirizzo <KX-IP> da un server DHCP, esegue il comando

```
target $ ssh -p 60000 -R 61000:localhost:22 <VM-user>@<public-IP>
```

In realtà, dato che su KX abbiamo dropbear e non openSSH, il comando sarà un po' diverso

```
target $ dbclient -NT 60000 -R 61000:localhost:22 remotedebug@<public-IP>
```

La prima parte del comando (ssh -p 60000) crea una connessione SSH su <public-IP>:60000, che verrà inoltrata a VM a causa della *destination NAT*.

La seconda parte (-R 61000:localhost:22) crea un tunnel, cioè istruisce la VM a ridirigere su localhost@<KX-IP>:22 tutto quello che riceve su <VM-IP>:61000

2) Dalla macchina sviluppatore si apre un terminale SSH sul KX col comando

```
host $ ssh -p 60000 root@<VM-IP>
```

Su KX vi è installato un client SSH minimale (Dropbear) ma sufficiente (la versione più completa è openSSH).

Le modifiche apportate al FW KX riguardano i seguenti files

```
<rootfs>/etc/init.d/dropbear
<rootfs>/etc/network/interfaces
<rootfs>/etc/home/root/.ssh/id_rsa      (<- chiave privata)
<rootfs>/etc/home/root/.ssh/id_rsa.pub   (<- chiave pubblica)
<rootfs>/etc/home/root/.ssh/known_hosts  (<-definisce VM un known host)
```

37.3 UMTS

Per controllare molte macchine da remoto può essere conveniente utilizzare sulle macchine stesse un modulo UMTS con una SIM. Si ha così una copertura globale (3G o 4G) con un throughput minimo garantito (p.es. 300kbytes/s). Si attiva poi un contratto M2M con l'operatore che prevede una tariffa mensile (p.es. 30 euro) per poter trasferire un certo ammontare di dati (p.es. 1Gbytes) complessivi su un lotto di dispositivi (p.es. 150).

Questa soluzione può essere utile per aggiornamento FW, log, assistenza tecnica, ecc.

38 Creare ed eseguire un'applicazione

Todo

38.1 Accedere da userspace a un char device driver

Il seguente programma legge e scrive i dati sulla porta virtuale /dev/ttyUSB2.

Apriamo il programma

```
host $ cd <myapp>/ftdi
host $ cat ftdi.c
#include <stdio.h>      //Standard io definitions
#include <string.h>      //String function definitions
#include <unistd.h>      //Unix standard function definitions
#include <fcntl.h>        //File control definitions
#include <errno.h>        //Error number definitions
#include <termios.h>       //POSIX terminal control definitions

struct termios options;

int open_port(void)
{
    int fd;           //File descriptor for the port
    fd = open("/dev/ttyUSB0", O_RDWR | O_NOCTTY | O_NDELAY);

    tcgetattr(fd, &options);
    cfsetspeed(&options, B9600);
    cfsetospeed(&options, B9600);
    options.c_cflag |= (CLOCAL | CREAD);
    tcsetattr(fd, TCSANOW, &options);
```

Engineering Specification - confidential

```
if(fd == -1)
{
    perror("open_port: Unable to open /dev/ttyUSB0 - \n");
}
else
fcntl(fd, F_SETFL, 0);

return (fd);
}

int write_port(int fd)
{
    int n = 0;
    n = write(fd, "RESET", 5);
    if(n < 0)
        fputs("write() of 5 byte failed!\n", stderr);
    return(n);
}

int read_port(int fd, char* buf)
{
    char b[1];
    int i=0;
    while(i<20) {
        int n = read(fd, b, 1); // read a char at a time
        if( n==0 ) {
            usleep( 8000/9.6 ); /* wait exactly this time and then try again */
            buf[i] = b[0];
            i++;
            continue;
        }
        buf[i] = 0; // null terminate the string
        return 0;
    }
}

int main()
{
    int c,k;
    char buf[40];
    c = open_port();
    printf("The port status is: %i\n", c);
    k = write_port(c);
    printf("Write of 5 byte successful\n", k);
//    read_port(c, buf);
    read(c, buf, 20);
    printf("%s\n",buf);
    close(c);
    return(0);
}
```

Compilazione del programma

```
host $ cd <myapp>/ftdi
host $ arm-arago-linux-gnueabi-gcc -O2 -o ftdi ftdi.c
```

oppure, per aggiungere la libreria <libpath>/usr/lib

```
host $ arm-arago-linux-gnueabi-gcc -O2 -o ftdi ftdi.c -L <libpath>
```

Esempio:

```
host $ arm-arago-linux-gnueabi-gcc -O2 -o ftdi ftdi.c -L <rootfs>/usr/lib
```

Copia nel filesystem

```
host $ cp ftdi <rootpath>/home
```

Prova sul target

```
target $ cd /home
target $ ./ftdi
```

Todo

38.2 Accedere da userspace al MAG3110 device driver (eCompass)

Il seguente programma C/POSIX implementa due thread che leggono in tempo reale i dati del magnetometro (mag3110) e dell'accelerometro (mma9550) dai rispettivi drivers e implementa la bussola elettronica (eCompass) con relativa calibrazione Hard e Soft Iron e Tilt-compensation.

Apriamo il programma sorgente

```
host $ cd <myapp>/eCompass  
host $ cat eCompass.c  
#include <stdio.h>      //Standard io definitions  
...
```

Compilazione

```
host $ arm-arago-linux-gnueabi-gcc -lm -lpthread -O2 -o eCompass eCompass.c
```

Nota: l'opzione -lm consente di linkare le funzioni matematiche sqrt, atan2, ecc

Nota: l'opzione -lpthread consente di linkare le funzioni di gestione dei threads

Copia nel filesystem

```
host $ cp eCompass <rootpath>/opt
```

Prova sul target

```
target $ cd /opt  
target $ ./eCompass  
0: Enter geomagnetic field B (uT) and inclination delta (deg)  
1: Enter simulation hard / soft iron model V[] and invW[]  
2: Display simulation and calibrated parameters  
3: Run eCompass with 7 or 4 element calibration  
4: Calibrate magnetometer offset  
99: Quit  
Enter command and hit enter: 3  
  
Enter calibration model (7 or 4 element): 7  
Enter number of eCompass iterations: 800  
  
..... (ruotare la scheda KXMB un paio di volte su ciascun asse)  
  
Iteration: 799  
f6DOFRawG: Gpx -0.29 Gpy 0.21 Gpz -0.95  
f6DOFRawB: Bpx 271.00 Bpy -261.00 Bpz -76.00  
f6DOFCalB: Bcx 11.10 Bcy 15.85 Bcz -44.96  
f6DOFOutp: Phi 167.57 The 16.46 Psi 13.56 delta 59.64  
f6DOFOutp: LPPhi 149.77 LPThe 31.47 LPPsi -0.51 LPdelta 45.48  
  
Calculating 7 element SVD calibration at iteration 800 with 19 in Smart FIFO  
Calibration Fit Error (%)= 3.389  
Calibration hard iron (in uT) Vx= 260.983 Vy= -276.578 Vz= -34.772 (stima del campo Hard iron)  
  
Calibration geomagnetic field (uT) B= 47.037 (intensità stimata del vettore B)  
  
Calibration inverse soft iron matrix invW (normalized)  
Row 0 1.06004 0.00000 0.00000  
Row 1 0.00000 0.90610 0.00000  
Row 2 0.00000 0.00000 1.04111  
For comparison: Simulation inverse soft iron matrix invW (normalized)  
Row 0 1.00000 0.00000 0.00000  
Row 1 0.00000 1.00000 0.00000  
Row 2 0.00000 0.00000 1.00000
```

Determinazione dell'offset di calibrazione del sensore

```
0: Enter geomagnetic field B (uT) and inclination delta (deg)  
1: Enter simulation hard / soft iron model V[] and invW[]  
2: Display simulation and calibrated parameters  
3: Run eCompass with 7 or 4 element calibration  
4: Calibrate magnetometer offset  
99: Quit  
Enter command and hit enter: 4
```

Rotate PCB in each direction a few and then press ESC

..... (ruotare la scheda KXMB un paio di volte su ciascun asse)

```
read: BpxMax 2999; BpxMin 2179 (uT/10)
read: BpyMax -2297; BpyMin -3182 (uT/10)
read: BpzMax -30; BpzMin -783 (uT/10)
```

```
target $ cat /sys/class/mag/mag/calibration_offset
2589,-2739,-406
```

(verifica)

Todo

38.3 Accedere da userspace al MMA9553 device driver (Pedometer)

Il seguente programma C/POSIX implementa un thread che legge in tempo reale i dati del dell'accelerometro-pedometro (mma9553) dal rispettivo driver.

Apriamo il programma sorgente

```
host $ cd <myapp>/pedometer
host $ cat pedometer.c
#include <stdio.h>      //Standard io definitions
...
```

Compilazione

```
host $ arm-arago-linux-gnueabi-gcc -lpthread -O2 -o pedometer pedometer.c
```

Nota: l'opzione -lpthread consente di linkare le funzioni di gestione del thread

Copia nel filesystem

```
host $ cp pedometer <rootpath>/opt
```

Prova sul target

```
target $ cd /opt
target $ ./pedometer
1: Set Filter Step Mode
2: Set Activity Threshold
3: Set Sleep Mode
4: Pedmeter output (press ESC to return)
5: Accelerometer continous readings
6: Get Accelerometer offset
99: Quit
Enter command and hit enter: 4
```

```
stepCnt = 00012, activity = 2, speed = 06821 m/h, distance = 00012 m, sleepCnt = 00100, suspended = 1
```

stepCnt è il numero degli step contati fino ad ora.

Il rilevamento di uno step si ha quando la dispersione (max - min) della media mobile di $|G|$ su 0.19 sec (@fs=30Hz) è maggiore di 0.13 g per almeno 0.07 sec.

Activity è l'indice del livello di attività del soggetto (1:REST,2:WALKING,3:JOGGING,4:RUNNING)

Distance è la distanza percorsa in m. Viene calcolata moltiplicando l'ampiezza del passo per il numero di step. L'ampiezza del passo può essere stimata in base all'attività o in base ai dati di sesso, altezza e peso.

Speed è la velocità media e viene calcolata come distance/time

Vi sarebbe la possibilità di misurare anche le calorie

Calories = MetabolicFactor * 0.00029 / StepRate * Weight

39 Creazione ed esecuzione di scripts

Todo

39.1 Creazione ed esecuzione script da Init runlevel

Verifica del runlevel di default

```
target $ cat /etc/inittab
```

Supponiamo sia 5.

Creazione dello script

```
target $ cd /etc/init.d
target $ vi myinitscript
...
target $ cat myinitscript
#!/bin/sh

case "$1" in
  start)
    echo -n "Starting myapp application"
    /home/myscript
    ;;
  stop)
    echo -n "Stopping myapp application"
    ;;
  *)
    echo "Usage: /etc/init.d/myinitscript {start|stop}"
    exit 1
esac

exit 0
```

Verifica dello script

```
target $ /bin/sh myinitscript start
```

..

```
target $ /bin/sh myinitscript stop
```

..

```
target $ update-rc.d myinitscript defaults
```

Registrazione dello script di start sul runlevel di default (5)

```
target $ update-rc.d myinitscript defaults
```

Posso anche specificare un diverso runlevel ecc.

Verifica

```
target $ ls ./rc5.d/S20myinitscript
```

Per rimuovere la registrazione dello script

```
target $ rm /etc/rc5.d/myinitscript
```

```
target $ update-rc.d myinitscript remove
```

Todo

39.2 Creazione ed esecuzione di script da udev

Supponiamo di volere eseguire uno script myscrip1 quando inserisco una SD card, e uno script myscrip2 quando la disinserisco

Inseriamo la SD card

Controlliamo quali attributi ha

```
target $ udevadm info -a -n /dev/mmcblk0 | more
```

Concentriamoci su un nodo in particolare del sysfs e copiamo quali sono gli attributi della periferica che debbono sincronizzare le operazioni

Ad esempio

```
KERNEL=="mmcblk0"  
SUBSYSTEM=="block"
```

Creazione del rules file

```
target $ cd /etc/udev/rules.d/  
target $ vi 99-fwupdate.rules  
...  
target $ cat 99-fwupdate.rules  
# Run fw update every time a mmc device appears  
# On remove actions, check the fw update is correct, otherwise do a backup  
ACTION=="add", KERNEL=="mmcblk0", SUBSYSTEM=="block", RUN+=""/home/fwupd"  
ACTION=="remove", KERNEL=="mmcblk0", SUBSYSTEM=="block", RUN+=""/home/fwupdcheck"
```

Ricarichiamo le rules di udev (oppure resettare il target)

```
target $ udevadm control --reload-rules
```

Creazione dello script da lanciare all'inserimento della SD card

```
target $ cd /home  
target $ vi fwupd  
...  
target $ cat fwupd  
#!/bin/sh  
echo heartbeat > /sys/class/leds/STATLED_EN/trigger
```

Creazione dello script da lanciare all'estrazione della SD card

```
target $ cd /home  
target $ vi fwupdcheck  
...  
target $ cat fwupdcheck  
#!/bin/sh  
echo none > /sys/class/leds/STATLED_EN/trigger
```

Verifica

Inseriamo la SD card -> Il LED lampeggia

Disinseriamo la SD card -> Il LED smette di lampeggiare

39.3 Creazione ed esecuzione di script makefile su host

Supponiamo di volere eseguire uno script makefile sull'host che esegue una serie di comandi separati da ;

Si può andare a capo con \. L'ultimo comando della sequenza non necessita di ;

Se un comando è preceduto da @ allora sarà "silent", cioè senza stampa a terminale

Esempio 1:

In questo esempio si testa l'esistenza di un certo file e, se esiste, lo si elimina.

```
@if [ -f ${ROOTFS_DESTDIR}/etc/dropbear/dropbear_rsa_host_key ]; then \  
echo Deleting existing etc/dropbear/dropbear_rsa_host_key; \  
sudo rm ${ROOTFS_DESTDIR}/etc/dropbear/dropbear_rsa_host_key; \  
fi
```

Esempio 2:

In questo esempio un ciclo for viene applicato a tutti i file di un certo tipo presenti nella directory corrente. Per selezionare i file si utilizza il comando find con varie opzioni. L'opzione –type f specifica

Engineering Specification - confidential

che si cercano dei file regolari; l'opzione `-maxdepth 1` specifica che interessano solo i file presenti nel primo livello della directory corrente, ecc..

```
@for file in `find . -maxdepth 1 -type f -name "rootfs-kx-*tar"'; do \
if [ -f $$file ] ; \
then echo $$file; \
tar xf $$file; \
fi; \
done
```

Esempio 3:

In questo esempio un ciclo for viene applicato a tutte le directory presenti nella directory corrente (`-type d`). Con l'opzione `-mindepth 1` possiamo escludere dal ciclo la directory `"."`

```
@tempdir=`mktemp -d`; cd $$tempdir; \
scp $(FWSERVERIP):$(ROOTFSREPO_DIR)/rootfs-kx-0003-bundle.tar.gz .; tar zxf rootfs-kx-0003-bundle.tar.gz; rm rootfs-kx-0003-
bundle.tar.gz; \
for dir in `find . -maxdepth 1 -mindepth 1 -type d`; do \
if [ -d $$dir ] ; \
then echo $$dir; \
cp -Rfa $$dir/* $(ROOTFS_DESTDIR); \
fi; \
done; \
rm -Rf $$tempdir
```

Esempio 4:

In questo esempio si crea una directory temporanea, vi sientra, ci si fa qualcosa, e poi la si elimina

```
@tempdir=`mktemp -d`; cd $$tempdir; \
...
rm -Rf $$tempdir
```

Esempio 4:

Questo è un esempio di makefile per il build di una libreria (formato da un file di config e altri di Makefile)

File config.mk

```
CROSS_PATH = ~/sdk_5_07_00_00/linux-devkit/
ROOTFS=/var/lib/nfs-kx-rootfs
LIBNAME=kxgpio
```

File Makefile

```
# Copyright (C) 2013, X s.r.l. <http://www.X.com/>
# Authors: Gabriele Filosofi <gabrielef@X.it>,
#          Emiliano Betti <betti@epigenesys.com>
# Date: November 26th, 2013

include ./config.mk

LIBNAME=kxgpio

LIB=libs$(LIBNAME).so      (<- la cartella corrente ha una sottocartella libsrc con i sorgenti della libreria)
TEST=libtest$(LIBNAME)-test (<- la cartella corrente ha una sottocartella libtest con i sorgenti del progr di test)

all: $(LIB) $(TEST)        (<- col comando 'make all' si eseguirà sia 'make -C libsrc' che 'make -C libtest')
    @echo "Build completed"

$(LIB):
    @make -C libsrc        (<- con questo comando è come se si entrasse in libsrc e si eseguisse make)

$(TEST): $(LIB)
    @make -C libtest       (<- con questo comando è come se si entrasse in libtest e si eseguisse make)

install: $(LIB)           (<- prima di installare ha senso ricompilare)
    @install -D $(LIB) $(ROOTFS)/usr/lib/   (<- i comandi preceduti di @ sono silent, cioè senza stampe a
terminale)
    @echo ""
```

Engineering Specification - confidential

```
@echo "Installation completed"  
@echo "Note that you may need to run ldconfig on your system"  
@echo ""  
  
install-test: $(TEST) install      (<- prima di installare ha senso ricompilare. Anche la libreria usata dal test program)  
        @install -D $(TEST) $(ROOTFS)/usr/bin/  
  
clean:  
        @make -C libsrc clean  
        @make -C libtest clean  
  
distclean: clean  
        @make -C libsrc distclean
```

Esempio 4:

Possiamo scaricare il Makefile di E.Betti in questo modo

```
host $ git clone git://github.com/ebetti/Makefile.git
```

Oppure possiamo scaricare la libreria GitHub-CSP di G.Filosofi in questo modo

```
host $ git clone git://github.com/Gfilosofi/GitHub-CSP.git
```

40 Debuggare in kernel space

40.1 Dindbg

Il kernel può essere compilato con un supporto di debug attivabile nel config tramite la macro CONFIG_DYNAMIC_DEBUG. Si chiama dynamic debug (dyndbg).

Ogni pr_debug()/dev_dbg() presente nel kernel espande a una struttura che riporta molte informazioni, tra cui il nome del modulo, il numero di linea, il nome della funzione, ecc.

A run time si potranno attivare tutte e sole le print desiderate con le informazioni desiderate, attraverso delle semplici query su un file di controllo che deve essere montato (debugfs).

41 Debuggare un'applicazione

42 Modificare la mappa di memoria del target

43 Aspetti da affrontare

43.1 Porting al Kernel 2.6.37-rc6

<http://www.kernel.org/pub/linux/kernel/v2.6/testing/linux-2.6.37-rc3.tar.bz2>

Prendere le patch di TI nel SDK e cercare di applicarle a questo kernel

To do

43.2 zlib

Scaricare pacchetto zlib. P.es. zlib-1.2.6 (da <http://zlib.net/>)

```
host $ wget http://zlib.net/zlib-1.2.6.tar.gz
```

Copiarlo in /home/<user>/Program/extlibs, estrarlo

```
host $ tar zxf zlib-x.x.x.tar.gz
```

Configurare zlib

```
host $ cd zlib-x.x.x  
host $ ./configure --prefix=/usr
```

Per cross-compilare per ARM, editare il Makefile e inserire

```
CROSS = arm-none-linux-gnueabi-  
CC = $(CROSS)gcc  
LDSHARED = $(CROSS)gcc -shared ...  
CPP = $(CROSS)gcc -E  
AR = $(CROSS)ar rc  
RANLIB = $(CROSS)ranlib
```

```
host $ make  
host $ make install DESTDIR=<tools>/...
```

43.3 Developing with Graphics

43.4 How to Connect to an EVM via Telnet

43.5 How to Setup a Samba Server

43.6 Understanding the Boot Sequence

43.7 How to install a debian package

Il gestore dei pacchetti *apt-get* è facile da usare

Per cercare un particolare pacchetto

```
host $ sudo apt-cache search <packagename>
```

Oppure

```
host $ sudo aptitude search <packagename>
```

Per installare il pacchetto

```
host $ sudo apt-get <packagename>
```

Per conoscere tutti i pacchetti installati

```
host $ apt list --installed
```

Il gestore dei pacchetti *apt-get* è basato su *dpkg*. Se abbiamo un pacchetto .deb, possiamo utilizzare direttamente *dpkg*.

Per installare

```
host $ sudo dpkg -i <packagename>.deb
```

Se *dpkg* riporta un errore di dipendenze, si può utilizzare

```
host $ apt-get install -f
```

per scaricare le dipendenze e configurare

Per disinstallare

```
host $ sudo dpkg -r <packagename>.deb
```

43.8 Pin Setup Utility

http://processors.wiki.ti.com/index.php/Pin_Setup_Tool_for_AM18xx_ARM_Microprocessors

44 PRU

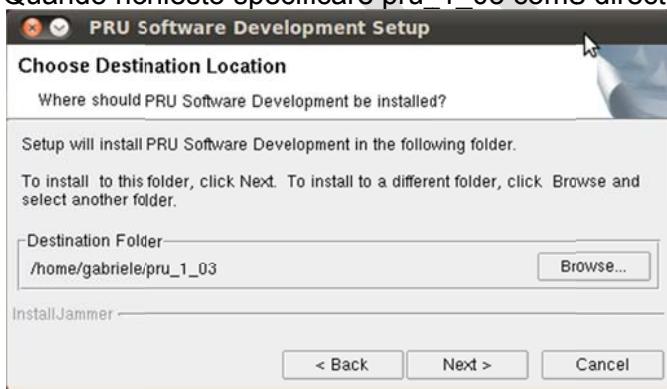
http://processors.wiki.ti.com/index.php/Programmable_Realtime_Unit_Subsystem

Scaricare in /home/<user> il pacchetto sprc941a.tar.gz.

```
host $ cd /home/<user>
host $ tar zxf sprc941a.tar.gz
host $ rm sprc941a.tar.gz
host $ ./PRU_Development_1_03_Linux-x86_Setup.bin
```



Quando richiesto specificare pru_1_03 come directory di installazione.



```
host $ cd pru_1_03
```

45 KX application

La seguente tabella mostra solo alcuni dei moduli del programma applicativo KX sviluppati da X.

Modules	Description
application modules	
FastSign.c	gas signal seed-up module
calibgasapp.c	gas calibration module
turbine.c	turbine driver module
sampler.c	signal sampling module
kxarc.c	test archive module
sqldb.c	subject database module
tcsignal.c	signal conditioning module
tcontrols.c	controllers management module
/blegw/main.c	bluetooth low-energy driver main program module
pctrlp.c	remote control panel module
ccfg.c	configuration parameters module
testdown.c	pc test download module
kevent.c	system events module
gpshared.c	gps information sharing module
keyevent.c	keyboard event management module
cumlport.c	pc communication protocol module
pergo.c	remote ergo test module
ergo.c	ergo test program module
dmc.c	dmc library module
licence.c	licence management module
gpscpl.cpp	gps user interface module
nmeacommand.cpp	NMEA message decoder module
qmlloader.cpp	qml file loader module
/qmlrun/main.cpp	application main program module
taracpl.cpp	tarature programs interface module
appctrlp.cpp	control panel user interface module
ergocpl.cpp	ergo test user interface module
Mutility.qml	utility menu component module
Menulitem.qml	basic menu component module
AppsModel.qml	executable applications descriptor module
Compass.qml	eCompass user interface module
DBStatus.qml	database utility user interface module
SetInternational.qml	international settings module
Navbar.qml	navigation bar module
RadiolistEx.qml	generic radio button list module
Pagearray.qml	qml pages sequencer module
SubjFind.qml	subject selection interface module
Mwelcome.qml	main menu interface module
Testing.qml	ergo test user interface module

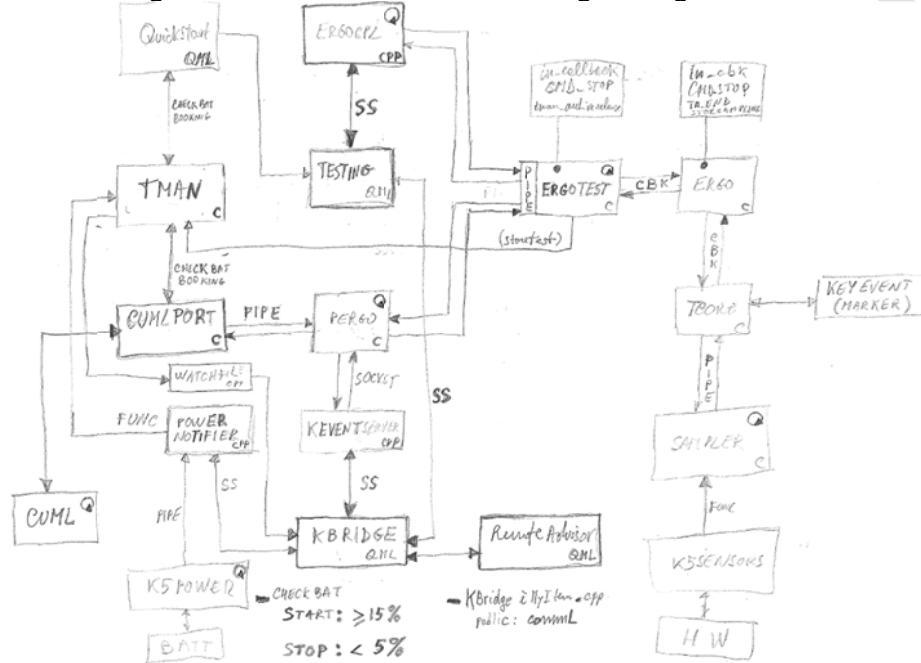
Engineering Specification - confidential

Mdbsearch.qml	database search user interface module
UIPanel.qml	control panel user interface module
Msettings.qml	settings menu module
RemoteAd.qml	remote advisor user interface module
Systemsets.qml	system settings user interface module
Gpscpl.qml	gps user interface module
Statbar.qml	status bar user interface module
Turbcal.qml	turbine calibration user interface module (message popup)
Setdate.qml	date setting user interface module
Mcalibrate.qml	calibration menu module
MessageBox.qml	generic message box user interface module
Keybnums.qml	on-screen numerical keyboard module
BlueInfo.qml	bluetooth user interface module
KBridge.qml	system event manager module
GListView.qml	generic list view user interface module
Erasedb.qml	database erase user interface module
FloatKeyb.qml	alphanumeric on-screen keyboard module
DialogBar.qml	generic dialog bar user interface module
DBExport.qml	database export user interface module
AntInfo.qml	ant sensors user interface module
Settimezone.qml	time zone setting user interface module
splash.qml	splash screen module
funcs.js	javascript function library module
kernel modules	
mma955x_sysfs.c	accelerometer/pedometer sys module
kxsensors_core.c	kx sensor driver core module
kxsensors_sysfs.c	kx sensor driver sys module
kxsensors_co2trigger.c	kx sensor driver co2 trigger module
kxsensors_sampling.c	kx sensor sampler module
kxpower_battery.c	battery power management driver module
kxpower_ltc4100.c	charger power management driver module
hrecg_core.c	hr/ecg interface driver core module
mpl3115a2_core.c	pressure sensor driver core module
mic3291_core.c	lcd backlight driver core module
mag3110_core.c	magnetometer driver core module
ad768x_core.c	adc driver core module
uboot modules	
cmd_fw_upd.c	fw update program module
cmd_ioexp.c	expander io control module
da850kx.c	uboot da850 init module
linux modules	
mach-davinci/board-da850-kx.c	kernel da850 init module

Engineering Specification - confidential

userspace modules	
bike_compute.c	ant bike profile module
hr.c	ant hr profile module
antctrl.c	ant control module
kxctrl.c	kx control library
/iwrap/daemon/main.c	iwrap bluetooth driver main program
iwrap-daemon.c	iwrap daemon module

L'architettura del test di ergometria è schematizzata nella figura seguente



L'applicazione userspace del KX è un insieme di moduli c/c++/qt/qml connessi da meccanismi di comunicazione interprocesso

SS: signal-slot

CBK: callback

PIPE: pipe o fifo

SOCKET: unix sockets

La maggior parte dei sensori vengono letti da un driver di alto livello in kernel space (kxsensors). I dati sono poi trasmessi ai moduli sampler.c e tcore.c in userspace. I programmi del KX (siano essi calibrazioni, test, pannello di controllo, ecc.) ricevono i dati da tcore.c (tramite una fifo) e generano l'output o verso il display (tramite una wrapper in C++ verso un modulo qml che poi viene interpretato dall'interfaccia grafica in Qt mediante un plugin) oppure verso il PC (tramite una wrapper in C verso il modulo cuml che è l'interfaccia verso i driver usb/bluetooth). Il Pannello di Controllo ha entrambe le interfacce, dato che può essere lanciato sia da PC che localmente, mentre le calibrazioni hanno solo il canale grafico.

Vi è poi un server qws ..

46 Comandi bash e shell

Per vedere tutti i file in modo ricorsivo anche nelle sottodirectory

```
host $ ls -IR
```

Per copiare un file (cp conserva i permessi, install li cambia in *rwxr-xr-x*)

```
host $ cp filename destdir  
host $ install filename destdir
```

Per visualizzare il contenuto di un file (cat visualizza tutto insieme, less un po' per volta)

```
host $ cat filename  
host $ less filename
```

Per convertire e copiare un file

```
host $ dd if=<inputfile> of=<outputfile>
```

Per conoscere bene le opzioni dell'utility dd consultare man

Per conoscere le dimensioni di una directory (in kbyte)

```
host $ du -s <dirname>
```

Per conoscere le dimensioni di una directory con anche il dettaglio delle sottocartelle ecc.

```
host $ du -h <dirname>
```

Per conoscere le dimensioni del filesystem

```
host $ df -h
```

Per stampare la struttura ad albero dei file di una directory

```
host $ tree <dirname>
```

Per conoscere la quantità di memoria libera e occupata

```
host $ free -m
```

Con il parametro -s si può fornire il periodo (in sec) di ripetizione automatica

```
host $ free -s <N> (<- ripete il comando free ogni <N> sec)
```

Per vedere i processi attivi

```
host $ ps -aux
```

VSZ contiene la memoria virtuale allocata dal processo

RSS contiene la memoria resident (cioè RAM) utilizzata dal processo

Per estrarre un file .tar in modo verbose

```
host $ tar xvf filename.tar
```

Per estrarre un file .tar conservando i permessi dei file

```
host $ tar xpf filename.tar
```

Per esplorare il contenuto di un file .tar senza estrarlo

```
host $ tar xvf filename.tar
```

Per esplorare il contenuto di un file .tar.gz senza estrarlo

```
host $ tar -tf filename.tar.gz
```

Per estrarre un file .tar.gz in modo verbose

```
host $ tar xvzf filename.tar.gz
```

Per estrarre un file .tbz2

```
host $ tar xpf filename.tbz2
```

Per estrarre un file .tar.bz2 in modo verbose

```
host $ tar xvjf filename.tar.bz2
```

Engineering Specification - confidential

Per creare un file .tar

```
host $ tar cf dirname.tar dirname/
```

Per creare un file .tar.gz

```
host $ tar czf dirname.tar.gz dirname/
```

Per creare un file

```
host $ touch filename
```

Per creare una cartella

```
host $ mkdir -p filename
```

Per acquisire diritti di amministratore

```
host $ sudo -s
```

Per trovare in dirname e sottocartelle il file filename

```
host $ find dirname -name filename
```

Per trovare in dirname e sottocartelle i file che contengono la stringa "string"

```
host $ find dirname | xargs grep "string" -sl
```

oppure

```
host $ grep "string" -rin *
```

Per trovare in un file filename tutti gli item "string" evidenziandoli in rosso

```
host $ cat filename | grep "string" -e
```

Per installare un pacchetto SW una volta scompattata la directory di installazione

```
host $ ./configure (esegue lo script configure)
```

```
host $ make (esegue Makefile che crea eseguibili)
```

```
host $ make install (copia eseguibili nelle cartelle appropriate del filesystem)
```

```
host $ make install DESTDIR=<dir> (copia eseguibili nelle cartelle con prefisso <dir>)
```

Per configurare la compilazione su arm

```
host $ ./configure --host=arm-none-linux-gnueabi
```

Per copiare la cartella dirname e chiamarla dirname_copy

```
host $ cp -R dirname/ dirname_copy/
```

Per copiare la cartella dirname e chiamarla dirname_copy, preservando anche i link e gli attributi files (archive mode)

```
host $ cp -a dirname/ dirname_copy/
```

Per cambiare owner e group a un link simbolico

```
host $ chown -h newowner:nwegrup linkname
```

Per cambiare owner e group a una directory

```
host $ chown -R newowner:nwegrup dirname
```

Per aggiungere il permesso di esecuzione per tutti a un file

```
host $ chmod a+x filename
```

Per vedere tutti i processi attivi

```
host $ ps ax
```

oppure

```
host $ ps aux
```

Per vedere i processi lanciati dal terminale corrente

```
host $ fg
```

oppure

```
host $ jobs
```

Per fermare un processo in foreground

Engineering Specification - confidential

host \$ Ctrl+ z
e per riattivarlo
host \$ fg

Per arrestare forzatamente un processo con id <id>

host \$ kill -9 <id>

Per trasmettere da shell su /dev/ttys0

host \$ echo "testo" > /dev/ttys0

Per ricevere caratteri da un chardev (p.es. /dev/ttys0)

host \$ cat < /dev/ttys0

oppure

host \$ od -x < /dev/ttys0 (in formato hex)

host \$ od -d < /dev/ttys0 (in formato decimale)

Per configurare da shell /dev/ttys0

host \$ stty 115200 </dev/ttys0

Per configurare da shell /dev/ttys1 in modo 9600,N,8,1

host \$ stty -F /dev/ttys1 raw speed 9600 -crtcts cs8 -parenb cstopb

Per cercare un pacchetto software packetname

host \$ sudo apt-get search packetname

Per installare un pacchetto software packetname

host \$ sudo apt-get install packetname

Per rimuovere un pacchetto software packetname

host \$ sudo apt-get remove packetname

Per rimuovere un pacchetto software packetname

host \$ sudo apt-get purge packetname

Per creare un link simbolico a un file

host \$ ln -s filename linkname

Per ri-definire un link simbolico esistente

host \$ ln -snf newfilename linkname

Per sincronizzare due cartelle, vale a dire sovrascrivere il contenuto di <srcdir>/ in <destdir>/ ma lasciando inalterati i file che sono identici (anche come data)

host \$ rsync --avc --delete <srcdir> <destdir>

host \$ rsync --avc --delete <srcdir> <destdir>/

Nota: l'opzione "n" serve nel primo comando per simulare l'operazione senza però eseguirla effettivamente

Nota: l'opzione "c" serve a utilizzare anche il checksum per verificare l'identità dei files

Per disattivare e riattivare la rete

host \$ ifconfig eth0 down

host \$ ifconfig eth0 up

Per configurare l'indirizzo di rete <ipaddr> con subnet mask 255.255.255.0

host \$ ifconfig eth0 <ipaddr> netmask 255.255.255.0 up

Per selezionare il default gateway alla quale eth0 consegnerà i pacchetti verso host esterni (non appartenenti alla stessa LAN)

host \$ route add default gw <gwipaddr>

Per aggiungere server DNS e altro

host \$ vim /etc/resolv.conf

aggiungere p.es.

Engineering Specification - confidential

nameserver 192.168.0.240
nameserver 192.168.0.241
search intranet.X.it

Per lanciare il demone che chiede a un DHCP server l'assegnazione di un indirizzo ip
host \$ sudo dhclient

Nota: normalmente l'indirizzo ottenuto dal server è temporaneo

Per abilitare la funzionalità di router alla macchina (cioè di ritrasmettere su una interfaccia di rete pacchetti ricevuti un'altra interfaccia di rete)

host \$ sysctl -w net.ipv4.ip_forward=1

oppure

host \$ echo 1 > /proc/sys/net/ipv4/ip_forward

Per aggiungere un percorso a un router (che ha quindi almeno due interfacce di rete) perché ritrasmetta verso <ipaddr2> tutti i pacchetti che riceve destinati alla sottorete 192.168.1.x

host \$ rout add -net 192.168.1.0/24 gw <ipaddr2>

Per verificare se un file eseguibile è stato compilato per ARM oppure per x86

host \$ file filename

Per cambiare directory senza perdere la directory corrente

host \$ pushd newdir

e tornare indietro

host \$ popd

Per visualizzare le porte IP aperte

host \$ netstat -tulp

host \$ netstat -tuln

host \$ netstat -s

Per direzionare il terminale sulla porta tcp <ipaddr>:<port>

host \$ nc <ipaddr> <port>

Per verificare se la macchina <ipaddr> è raggiungibile inviando 4 pacchetti ICMP

host \$ ping <ipaddr> -c 4

Per spegnere la macchina

host \$ shutdown -h now

oppure

host \$ halt

oppure

host \$ init 0

Per riavviare la macchina

host \$ reboot

Per rivedere il log del boot

host \$ dmesg

Per vedere stampati a terminale i messaggi generati da *printk(KERN_DEBUG ..)*

host \$ dmesg -n 8

In pratica viene abbassata la soglia di accesso dei messaggi alla console.

Per scaricare un pacchetto da URL

host \$ wget URL

Per disabilitare il livello <N> di print sulla console

host \$ dmesg -n<N>

Per vedere le opzioni di configurazione kernel

target \$ cat /proc/config.gz

Se in particolare si cercano le stringhe con p.es. "SPI", ma di queste non quelle con "spin"

Engineering Specification - confidential

```
target $ zcat /proc/config.gz | grep -i SPI | grep -iv spin
```

Per conoscere la versione di linux e della cpu

```
target $ cat /proc/version
Linux version 3.3.0-kx-006 (gabriele@fub1204) (gcc version 4.5.3 20110311 (prerelease) (GCC) ) #1 PREEMPT Mon Jun 22 09:11:01
CEST 2015

target $ cat /proc/cpuinfo
Processor      : ARM926EJ-S rev 5 (v5l)
BogoMIPS       : 227.32
Features       : swp half thumb fastmult edsp java
CPU implementer : 0x41
CPU architecture: 5TEJ
CPU variant    : 0x0
CPU part       : 0x926
CPU revision   : 5

Hardware      : X KX
Revision      : 12110003
Serial        : 0000000780cba23
```

Per conoscere la string bootargs

```
target $ cat /proc/cmdline
```

Per isolare di una stringa una particolare parte

```
target $ c="abcdefg"
target $ echo "${c}"
abcdefg
target $ echo "${c%*efg}"
abcd
target $ echo "${c%*c*}"
ab
target $ echo "${c##abc}"
defg
target $ echo "${c##*b}"
cdefg
```

Per conoscere la versione del kernel

```
target $ uname -r
```

Per scaricare nella cartella linus-v33-rc1 una copia del kernel con git

```
host $ git clone git://git.kernel.org/pub/scm/linux/kernel/git/stable/linux_stable.git linus_v33-rc1
```

Per riavviare in caso di blocco tastier e mouse

```
Alt + Stamp + digitare "REISUB"
```

Per sapere quai librerie servono a una applicazione appname

```
host $ ldd appname
host $ <armtoolch>-ldd appname
```

Per stampare la data in un determinato formato

```
host $ date +"%m-%d-%Y"
```

Per utilizzare la data in uno script

```
#!/bin/bash
NOW=$(date +"%m-%d-%Y")
FILE="backup.$NOW.tar.gz"
# rest of script
```

Per settare la data e l'ora di sistema a, diciamo, mese Novembre, giorno 2, ore 12:57, anno 2014

```
target $ date 110212572014
```

L'ora di sistema viene mantenuta fino al prossimo riavvio. Per renderla permanente occorre riportarla anche nel real-time clock (HW clock, o RTC). Per farlo in modalità UTC (Universal Time Coordinate)

```
target $ hwclock --utc --systohc
```

Se lo si vuole fare tenendo conto della propria area geografica

```
target $ hwclock --localtime --systohc
```

dove ovviamente deve esistere /etc/localtime che è un link simbolico al file in /usr/share/zoneinfo/ che indica la propria area geografica. Inoltre deve essere stata esportata la variabile TZ con

```
export TZ="UTC"
```

Engineering Specification - confidential

Per visualizzare il tempo dell'HW clock

```
target $ hwclock -r  
2000-01-01 01:11:26.505034+1:00
```

Per settare il clock di sistema sul valore del HW clock

```
target $ hwclock -s
```

Esempio Nello script di avvio del XNRG vengono eseguite queste due istruzioni

```
/sbin/hwclock -s -f /dev/rtc1      (<- setta il clock di sistema sul valore dell'HW clock RTC DS3107)  
/sbin/hwclock -w -f /dev/rtc0      (<- setta il clock della BBB sul valore del clock di sistema)
```

-s, --hctosys set the system time from the hardware clock
-w, --systohc set the hardware clock from the current system time
-f special /dev/... file to use instead of default

Per maggiori dettagli vedi sezione corrispondente.

Per avere più informazioni dal comando *hwclock*, aggiungere l'opzione *--debug*

```
root@kx:~# /bin/date 2015.05.22-11:00:00  
Fri May 22 11:00:00 GMT+7 2015  
root@kx:~# hwclock -w  
root@kx:~# date  
Fri May 22 11:00:18 GMT+7 2015  
root@kx:~# /bin/date 2015.05.22-10:38:00  
Fri May 22 10:38:00 GMT+2 2015  
root@kx:~# hwclock -w  
root@kx:~# date  
Fri May 22 10:38:05 GMT+2 2015  
root@kx:~#
```

Per terminare e ri-lanciare il servizio NFS

```
host $ sudo /etc/init.d/nfs-kernel-server reload
```

(su Ubuntu host)

Per vedere la memoria free e quella allocata

```
host $ free
```

Per vedere la directory corrente

```
host $ pwd
```

oppure

```
host $ echo $(PWD)
```

Per eseguire un comando shell (p.es. *pwd*) in un Makefile, inserirlo tra apici inversi (*AltGr + ?*)

Per eseguire in background un comando in uno script bash

```
#!/bin/bash  
top &          (<- il simbolo & dirotta l'output del command fuori dalla shell, in modo che si può usare l'input)  
..
```

Per conoscere il MAJOR number di un modulo

```
target $ cat /proc/devices
```

Per sapere se un modulo è presente nel kernel

```
target $ modinfo modname
```

Per sapere quali moduli sono caricati in memoria

```
target $ lsmod
```

Per consentire al loader dinamico del kernel di sapere dove si trovano tutte le librerie dinamiche di cui ha bisogno, esiste un comando che può anche essere invocato da shell

```
target $ ldconfig
```

Questo comando va a leggere */etc/ld.so.conf*, che è un file con elencate le directory. Altre volte */etc/ld.so.conf* è un link simbolico alla cartella */etc/ld.so.conf.d/*, contenente molti file separati, per una migliore gestione delle installazioni. Il risultato di *ldconfig* viene scritto su */etc/ld.so.cache*.

Nel sistema kx si è deciso di usare */usr/share/X/lib* come directory di installazione librerie userspace, quindi */etc/ld.so.conf* contiene sicuramente

```
target $ cat /etc/ld.so.conf  
/usr/share/X/lib
```

Engineering Specification - confidential

Per creare il nodo da associare al char driver module kxsensors.ko con MAJOR number 249

```
target $ mknod -m 0666 /dev/kxsensor c 249 0
```

Per creare il nodo da associare al char driver ttyUSB0 con MAJOR number 189

```
target $ mknod -m 0666 /dev/ttyUSB0 c 189 0  
target $ mknod -m 0666 /dev/ttyUSB1 c 189 1  
target $ mknod -m 0666 /dev/ttyUSB2 c 189 2  
target $ mknod -m 0666 /dev/ttyUSB3 c 189 3
```

Nota: il MAJOR number da utilizzare è quello riportato nel file /proc/devices in corrispondenza del driver tty

Uno script bash inizia sempre con

```
#!/bin/sh
```

e deve avere abilitato il permesso di esecuzione (x)

Per vedere un file in formato esadecimale

```
host $ cat filename | hexdump -C  
P.es. per leggere la partizione /dev/mtd11  
host $ cat /dev/mtd11 | hexdump -C
```

Per lanciare matrix-gui ruotando l'immagine di 270 gradi (sfrutta una libreria Qt)

```
target $ matrix_browser -qws -display transformed:Rot270
```

Per aprire un nuovo terminale

```
Ctrl+T
```

Per chiudere il terminale corrente

```
Ctrl+d
```

oppure

```
exit
```

Per cercare una stringa già usata nella shell che contiene la stringa <string>

```
Ctrl+r, <string>
```

E per scorrere eventuali altre ricorrenze

```
Ctrl+r
```

Per debuggare uno script (quando viene eseguito si hanno delle print sulla shell)

```
#!/bin/bash  
set -x  
...
```

Per isolare il nome del file eliminando la path e un suffisso

```
host$ basename <path><filename> <suffix>
```

Per isolare un item (p.es. tipo di console) di un file (p.es. /proc/cmdline) in uno script

```
#!/bin/bash  
read CMDLINE </proc/cmdline  
for x in $CMDLINE; do  
    [[ $x = console=ttyGS0* ]] || continue  
    /sbin/getty -L /dev/ttys0 115200  
done
```

(<- carica il contenuto del file)
(<- se l'item "console=ttyGS0" è presente)
(<- fai qualcosa)

Per creare una funzione che ritorna un valore e poi testarla in uno script

```
#!/bin/bash  
function check_kxlbtst()  
{  
    read CMDLINE </proc/cmdline  
    for x in $CMDLINE; do  
        [[ $x = cmd=* ]] || continue  
        local cmd=${x:4:4}  
    done  
    if [[ $cmd -eq 0005 ]]; then  
        return 1  
    else  
        return 0  
    fi  
}
```

(<- carica il contenuto del file)
(<- se l'item "cmd=" è presente)
(<- definisco una var locale col valore che segue)
(<- se il valore di cmd è 0005)

Engineering Specification - confidential

```
...
check_kxlbtst
if [ $? -ne 0 ]; then
...
fi
```

(-> chiamo la funzione)
(-> se non torna zero allora)

Per rendere silent un comando <command> in uno script

```
@<command>
```

Per impostare la tastiera in italiano

```
host $ loadkeys it
```

Nota: non funziona su terminali virtuali ma solo su terminali su seriale fisica

Per analizzare le differenze tra <file1> e <file2>

```
host $ diff -pruN <file1> <file2> | less
```

oppure, dopo avere installato il tool grafico kompare con *apt-get install kompare*

```
host $ kompare <file1> <file2>
```

Per vedere tutte le chiamate userspace di un programma <prog>, in fase di debug

```
target $ strace <prog>
```

strace può anche essere agganciato a un thread già in esecuzione con PID <pid>

```
target $ strace -p <pid>
```

l'output può anche essere caricato su un file di uscita

```
target $ strace -p <pid> &> pippo
```

Per conoscere il massimo PID gestito dal sistema

```
target $ cat /proc/sys/kernel/pid_max
```

Note. Questo parametro può essere cambiato dinamicamente

Per impostare il livello di "verbosità" <N> (da 1 a 7 dove 7 è il più "verboso") delle printk sul terminale

```
target $ dmesg -n <N>
```

Per cancellare il buffer dmesg

```
target $ dmesg -c
```

Per vedere dove si è bloccato un processo con PID <pid>, ecc

```
target $ cat /proc/<pid>/wchan
```

```
target $ cat /proc/<pid>/task/<pid>/status
```

magari il processo ha generato altri kernel thread, per vedere tutti i pid

```
target $ ls /proc/<pid>/task/
```

alcuni di questi thread possono avere acceduto a dei file, per sapere i loro descrittori <fd>

```
target $ ls -l /proc/<pid>/task/<pid1>/fd/
```

In questo modo vedo la funzione dove il programma si è bloccato

Ci sono anche

```
target $ cat /proc/<threadnum>/task/<threadnum2>/status
```

e

```
target $ strace -p <pid>
```

Per vedere se un processo con PID <pid> ha un memory leak

```
target $ cat /proc/<pid>/statm
```

```
13325 5274 3031 13 0 6851 0
```

Se, ripetendo questa operazione a distanza di tempo, il secondo numero aumenta, allora c'è un memory leak

Per controllare in fase di compilazione che una condizione <cond> sia vera si può usare

```
#include <assert.h>
```

```
..
```

Engineering Specification - confidential

```
assert(<cond>);
```

```
...
```

Se la condizione non è vera a runtime il programma termina indicando file e riga.

Per utilizzare asser la macro di preprocessore NDEBUG deve essere definita. Si può definirla nel Makefile con il FLAG di compilazione -DNDEBUG

Per eseguire un comando da shell in maniera periodica. Per es per vedere come varia nel tempo l'utilizzo della memoria

```
target $ while true ; do clear ; free ; sleep 1; done
```

Oppure per vedere il tempo che passa

```
target $ while true ; do clear ; date ; sleep 1; done
```

Oppure per monitorare i processi kworker

```
target $ while true ; do clear ; ps -aux | grep kworker ; sleep 1 ; done
```

Per evitare che le print scatenate dal comando di shell <cmd> impegnino la seriale (oberando il sistema con interrupts, ecc.)

```
host $ <cmd> &> /dev/null
```

Per lanciare l'interfaccia grafica (solo per l'utente corrente)

```
host $ startx
```

Per installare un file <file> in una cartella <dirpath> che andrebbe creata se non esiste

```
host $ install -D <dirpath>/<file>
```

E' un command molto usato negli script makefile e ha molte opzioni

Per vedere il map file di una libreria statica (.a) oppure dinamica (.so)

```
host $ nm <libname>
```

Questo comando permette di vedere tutti i simboli della libreria.

Per vedere le periferiche pci

```
host $ lspci
```

Con l'opzione -v ho un output verbose, con -vv ho un verbose più approfondito, ecc..

Con l'opzione -x ho anche una stampa della memoria

Per vedere le periferiche usb

```
host $ lsusb
```

Sul target KX, per vedere le periferiche usb

```
Target $ usb-devices
```

```
T: Bus=01 Lev=00 Prnt=00 Port=00 Cnt=00 Dev#= 1 Spd=12 MxCh= 1          (<- DA8xx OHCI)
D: Ver= 1.10 Cls=09(hub ) Sub=00 Prot=00 MxPS=64 #Cfgs= 1
P: Vendor=1d6b ProdID=0001 Rev=03.03
S: Manufacturer=Linux 3.3.0-kx-006 ohci_hcd
S: Product=DA8xx OHCI
S: SerialNumber=ohci.0
C: #Ifs= 1 Cfg#= 1 Atr=e0 MxPwr=0mA
I: If#= 0 Alt= 0 #EPs= 1 Cls=09(hub ) Sub=00 Prot=00 Driver=hub

T: Bus=01 Lev=01 Prnt=01 Port=00 Cnt=01 Dev#= 2 Spd=12 MxCh= 4          (<- USB2514BI-AEZG, USB Hub)
D: Ver= 2.00 Cls=09(hub ) Sub=00 Prot=00 MxPS=64 #Cfgs= 1
P: Vendor=0424 ProdID=2514 Rev=0b.b3
C: #Ifs= 1 Cfg#= 1 Atr=e0 MxPwr=2mA
I: If#= 0 Alt= 0 #EPs= 1 Cls=09(hub ) Sub=00 Prot=00 Driver=hub

T: Bus=01 Lev=02 Prnt=02 Port=00 Cnt=01 Dev#= 3 Spd=12 MxCh= 0          (<- FT4232HQ, USB-QUART)
D: Ver= 2.00 Cls=00(ifc ) Sub=00 Prot=00 MxPS=64 #Cfgs= 1
P: Vendor=0403 ProdID=6011 Rev=08.00
S: Manufacturer=FTDI
S: Product=Quad RS232-HS
C: #Ifs= 4 Cfg#= 1 Atr=80 MxPwr=500mA
I: If#= 0 Alt= 0 #EPs= 2 Cls=ff(vend.) Sub=ff Prot=ff Driver=ftdi_sio
I: If#= 1 Alt= 0 #EPs= 2 Cls=ff(vend.) Sub=ff Prot=ff Driver=ftdi_sio
I: If#= 2 Alt= 0 #EPs= 2 Cls=ff(vend.) Sub=ff Prot=ff Driver=ftdi_sio
I: If#= 3 Alt= 0 #EPs= 2 Cls=ff(vend.) Sub=ff Prot=ff Driver=ftdi_sio

T: Bus=01 Lev=02 Prnt=02 Port=03 Cnt=02 Dev#= 8 Spd=12 MxCh= 0          (<- BLE112, Bluetooth Low Energy)
D: Ver= 2.00 Cls=02(commc) Sub=00 Prot=00 MxPS=32 #Cfgs= 1
P: Vendor=2458 ProdID=0001 Rev=00.01
S: Manufacturer=Bluegiga
```

Engineering Specification - confidential

```
S: Product=Low Energy Dongle  
S: SerialNumber=1  
C: #Ifs= 2 Cfg#= 1 Atr=80 MxPwr=74mA  
I: If#= 0 Alt= 0 #EPs= 1 Cls=02(commc) Sub=02 Prot=01 Driver=cdc_acm  
I: If#= 1 Alt= 0 #EPs= 2 Cls=0a(data ) Sub=00 Prot=00 Driver=cdc_acm
```

Per freezare la cpu per <N> secondi

```
target $ rtcwake -d /dev/rtc0 -s <N> -m mem
```

Nota: Suspend-to-RAM allows the system to save power by freezing all tasks, asking all drivers to move the devices to a low power state (by disabling the peripheral clock using its LPSC), cutting the clock to external RAM and finally moving the SoC to DeepSleep mode.

Per sapere i parametri interni di una sd card

```
host $ dmesg  
host $ sudo udevadm info -a -n /dev/sdb
```

(← per conoscere il nodo della card, sia sdb)

Per aprire un documento pdf

```
host $ evince <filename>.pdf
```

Per aprire un file immagine png

```
host $ eog <filename>.png
```

Per generare un flusso di dati randomici e magari dirigerlo sullo schermo

```
host $ cat /dev/urandom > /dev/fb0
```

Per fare in modo che un demone o programma parta all'avvio dell'host, è necessario scrivere in **/etc/rc.local**. Ad esempio, se volessimo far partire il demone di sincronizzazione della cartella Dropbox

```
exec "/home/gabriele/.dropbox-dist/dropbox-Lnx.x86_64-21.4.25/dropboxd"
```

Questo Sistema vale per Ubuntu. Altri sistemi potrebbero avere un file differente.

Per nascondere cursore lampeggiante

```
host $ setterm --cursor off
```

Per riattivare cursore lampeggiante

```
host $ setterm --cursor on
```

Per nascondere cursore lampeggiante su interfaccia GUI

```
host $ echo 0 > /sys/class/graphics/fbcon/cursor_blink
```

Per fare un controllo sui settori danneggiati di una USB pen drive. Supponendo /dev/sdb1 il punto di mount,

```
host $ sudo badblocks /dev/sdb1
```

Per conoscere la path assoluta canonica di un file

```
host $ realpath <filename>
```

Funziona anche con i link simbolici. E' utilizzato nel makefile di build delle applicazioni XNRG

Per installare questo comando

```
host $ sudo apt-get install realpath
```

47 System Calls

Le *system calls* sono dei comandi per consentire a un processo un userspace di richiedere servizi dal kernel del SO. Linux ha circa 300 system calls. Le system call sono tutte quelle descritte alla pagina 2 di **man**.

Le system call sono eseguite dal kernel del SO, ma le chiamate passano attraverso la libc. Per attivare una system call bisogna fornire al SO un numero (associato a quella specifica system call), gli eventuali argomenti, e una istruzione che triggeri la system call. Questa istruzione può variare a seconda dell'architettura e inizialmente per i386 era **int 0x80**. Tutte queste varie operazioni vengono rese trasparenti al programmatore grazie a funzioni di wrapping presenti nella libc. Il fatto che le system call siano wrappate dalla libc implica che le versione del kernel e la versione della libc non possono essere troppo distanti tra di loro, altrimenti possono esserci incompatibilità.

Quando si cerca una system call su man, è bene stare attenti a quale pagina di man viene visualizzata. Il comando di shell **man** è organizzato a pagine. La pagina 1 è dedicata ai comandi di shell, la pagina 2 è dedicata alle system call, la pagina 3 è dedicata alle API della libc, come ad esempio **malloc**. Vi sono poi altre pagine, da 4 a 9.

Quindi, per esempio, se cerchiamo informazioni sull' system call **kill** eseguiamo

man 2 kill

se invece siamo interessati al comando di shell **kill**, eseguiamo

man 1 kill

47.1 open

ToDo

47.2 read

ToDo

47.3 write

ToDo

47.4 close

ToDo

47.5 wait

ToDo

47.6 fork

la system call **fork** crea un nuovo processo (child) duplicando il processo chiamante (parent).

47.7 exit

ToDo

47.8 brk

La system call *brk* serve a spostare la posizion del program break, che definisce la fine del segmento dati non inizializzati del processo.

Brk è alla base delle implementazioni delle funzioni di allocazione e deallocazione della memoria nello heap implementate dalla glibc.

Per maggiori informazioni

[man 2 brk](#)

47.9 kill

La system call *kill* può essere usata per inviare un segnale a un processo attivo identificato da un PID.

Se la kill ritorna 0 allora il processo ha ricevuto il segnale.

Se la kill ritorna -1 allora il processo non era attivo, oppure non si avevano i permessi per inviare quel segnale, oppure l'identificativo del segnale utilizzato era non valido.

Nel FW, in *apps/kxapps/chelper/chelper.c*, la funzione kill the Control Panel

```
int kill_ext(int pid, int signal, int attempts, int waitms){
    int status,endpid,i;
    const char* signame = strsignal(signal);

    for(i=0; i<attempts; i++){
        __gDebug("sending signal %s to pid:%i ...",signame,pid );
        kill(pid, signal);
        usleep(waitms*1000);

        if( kill(pid,0) != 0 ){ //controlla se ancora esiste il pid
__gDebug("pid %i is closed!",pid );
            No__gDebug("...closed");
            return 0;                                //OK: chiuso
        }
        __gDebug("...still alive");
        endpid = waitpid(pid, &status, WNOHANG|WUNTRACED);
    }
}
```

All'interno di uno script si può usare "kill -0" per determinare se un processo esiste oppure no.
In questo caso ci stiamo riferendo a kill comando di shell, non system call.

47.10 mmap

La system call *mmap* serve a mappare una porzione di un file (oppure un device) in un'area di memoria RAM. Essa restituisce il puntatore all'area di memoria, e a quel punto si può lavorare sul file come fosse un array. L'operazione inversa, *munmap*, salva le modifiche sul file. Del file rimappato con *mmap* non si può modificare la dimensione. Esistono opzioni per fare in modo che le modifiche eseguite siano immediatamente visibili nel file (anche ad altri processi che aprono il file), ovvero solo al momento del *munmap*.

Per maggiori informazioni

[man 2 mmap](#)

47.11 stat

stat serve a ottenere informazioni su un file, come tempo di accesso, dimensione, ecc. Si tratta di informazioni presenti nella directory

Per maggiori informazioni

man 2 stat

47.12 select

La system call *select* serve a monitorare più file nello stesso processo userspace, in attesa che uno di essi diventi pronto per poterci effettuare delle operazioni di I/O. L'alternativa sarebbe quella di usare un processo indipendente per ogni file che si voglia controllare.

Per maggiori informazioni

man 2 select

47.13 pipe

La system call *pipe* serve a creare una pipe. Una *pipe* è un canale di comunicazione tra due processi, che sono stati generati da uno stesso processo padre.

Una *fifo* è una named pipe, una pipe a cui è associato un nome, quello di uno pseudo file presente nel filesystem.

Un esempio di fifo nel KX è quella che serve a inviare comandi al modulo iWRAP

target \$ echo <cmd> > /dev/shm/iwrapcmd fifo

Esempio:

root@kx:~# echo "CLOSE 0" > /dev/shm/iwrapcmd fifo

48 Library Calls (API)

Le API sono chiamate di libreria libc, e non vanno confuse con le *system calls*

Le library call sono tutte quelle descritte alla pagina 1 di *man*.

48.1 malloc e free

malloc e **free** sono API della libreria libc che servono a allocare e deallocare memoria dinamicamente nello heap. Esse sono implementate utilizzando la system call **brk**

48.2 htop

Per avere informazioni sui processi attivi

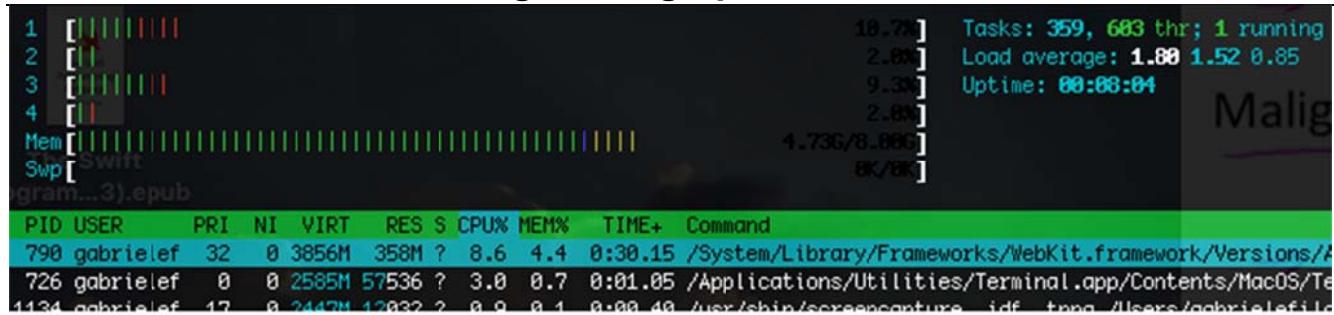
host \$ htop

htop è simile a top, ma ha il vantaggi di poter scrollare nelle varie direzioni. Inoltre si possono inviare comandi ai processi (p.es. kill)

Per avere informazioni sul singolo processo <pid>

host \$ htop -p <pid>

La parte alta del prospetto di htop mostra un numero di barre uguale al numero di *threads* della CPU.



La figura è il caso di un Intel Core i5 dual-core a 2,6GHz.

Ci si potrebbe chiedere perché ho 4 threads essendo il processore dual-core.

La risposta è l'Hyper-Threading (HT), dove per uno stesso core fisico, in hardware vengono replicate alcune risorse, come i registri, la cache L1, ecc, ma sicuramente l'accesso alla RAM è condiviso.

Questi sono chiamati core logici o *threads*. Sulle architetture CPU convenzionali i thread sono al massimo 2 per ogni core fisico, ma in altre architetture (p.es. GPU, possono essere di più).

Dal punto di vista della programmazione il *thread* (di un processo) è definito come l'unità minima di schedulazione. Vi è una corrispondenza tra questi due concetti, uno HW e uno SW. Quindi su ogni core logico posso avere massimo 2 threads schedulati in esecuzione.

Per conoscere esattamente l'organizzazione della CPU non basta *htop*, è necessario guardare il seguente file

```
host $ cat /proc/cpuinfo
```

Ogni processore ha un *physical_id*. Per ogni *physical_id* vi è uno o più *core_id* (sono i core fisici). Per ogni *core_id* vi è uno o due *siblings* (sono i *threads*).

Tipicamente un core fisico è più efficiente di un core logico, ma meno efficiente di due core logici che lavorano in parallelo. Ma questo può non essere più vero se sono richiesti molti accessi alla RAM.

48.3 inotify

La API *inotify* permette di monitorare facilmente eventi che accadono nel filesystem, cioè cambiamenti su singoli file o directory.

Esempio:

<http://www.thegeekstuff.com/2010/04/inotify-c-program-example/>

```
/*This is the sample program to notify us for the file creation and file deletion takes place in "/tmp" directory*/
#include <stdio.h>
#include <stdlib.h>
#include <errno.h>
#include <sys/types.h>
#include <linux/inotify.h>

#define EVENT_SIZE ( sizeof (struct inotify_event) )
#define EVENT_BUF_LEN   ( 1024 * ( EVENT_SIZE + 16 ) )

int main( )
{
    int length, i = 0;
    int fd;
    int wd;
    char buffer[EVENT_BUF_LEN];

    /*creating the INOTIFY instance*/
    fd = inotify_init();                                         (<- esiste la più recente inotify_init1() che ha funzioni extra)

    /*checking for error*/
    if ( fd < 0 ) {
        perror( "inotify_init" );
    }

    /*adding the "/tmp" directory into watch list. Here, the suggestion is to validate the existence of the directory before adding into monitoring list.*/
    wd = inotify_add_watch( fd, "/tmp", IN_CREATE | IN_DELETE );
```

```
/*read to determine the event change happens on "/tmp" directory. Actually this read blocks until the change event occurs*/
length = read( fd, buffer, EVENT_BUF_LEN );

/*checking for error*/
if ( length < 0 ) {
    perror( "read" );
}

/*actually read return the list of change events happens. Here, read the change event one by one and process it accordingly.*/
while ( i < length ) {   struct inotify_event *event = ( struct inotify_event * ) &buffer[ i ];   if ( event->len ) {
    if ( event->mask & IN_CREATE ) {
        if ( event->mask & IN_ISDIR ) {
            printf( "New directory %s created.\n", event->name );
        }
        else {
            printf( "New file %s created.\n", event->name );
        }
    }
    else if ( event->mask & IN_DELETE ) {
        if ( event->mask & IN_ISDIR ) {
            printf( "Directory %s deleted.\n", event->name );
        }
        else {
            printf( "File %s deleted.\n", event->name );
        }
    }
}
i += EVENT_SIZE + event->len;
}
/*removing the "/tmp" directory from the watch list.*/
inotify_rm_watch( fd, wd );

/*closing the INOTIFY instance*/
close( fd );
```

49 Comandi di System administration

I comandi di sistema sono tutti quelli descritti alla pagina 8 di **man**.

49.1 vmstat

Per avere statistiche sulla occupazione di memoria virtuale

host \$ vmstat <s> <n>

Il parametro intero <s> dice ogni quanti secondi deve eseguire il comando

Il parametro intero <n> dice quante volte in tutto deve eseguire il comando

Molte altre informazioni sono disponibili, e le si può vedere in una tabella con

host \$ vmstat -s

50 Lavorare con Taskjuggler

Taskjuggler è un software di project management a riga di commando.

50.1 Creazione ed esecuzione script da Init runlevel

Installazione Ruby 2.0

```
host $ sudo apt-get install ruby2.0
```

Installazione Taskjuggler

```
host $ sudo gem2.0 install taskjuggler
```

Compilare progetto

TODO

Compilare

```
host $ t3 main.tjp
```

I report generati si trovano nella sottocartella html. Consultarli con il browser

51 Kernel RT Preemp

Oltre al classico kernel linux vanilla è possibile utilizzare una versione patchata che è completamente preemtible.

https://rt.wiki.kernel.org/index.php/RT_PREEMPT_HOWTO

La patch trasforma tutti gli spinlock in mutex, e quindi diventano interrompibili.

Inoltre le IRQ sono gestite tutte come interrupt software (come thread) e quindi stanno sotto lo scheduler.

Si riducono un po' le performance nel caso medio ma migliora la situazione nei casi in cui è richiesta maggiore predicitività.

Queste avranno un po' più di latenza (touch screen, ecc), ma quelli che avranno priorità maggiore saranno più veloci.

Se la periodica la metto sotto interruzione

Tutto il resto non ci ferma più

Conta la priorità assegnata al processo

Con il Progetto Spartacus decidiamo di utilizzare la BeagleBone Black e la versione di linux 4.4 con la patch RT.

Una volta scaricato il kernel 4.4, gli si applica la patch e si attiva l'opzione di configurazione RT prima di ricompilare.

Se a un processo usermode diamo priorità RT il kernel lo interromperà poco rispetto a una soluzione vanilla non RT.

Ma anche in kernel si avranno miglioramenti, in quanto le interruzioni verranno disabilitate per minore tempo e quindi le latenze saranno più basse.

52 Opzioni di compilazione gcc

http://www.csbi.mit.edu/technology/intel_fce/doc/main_for/mergedProjects/copts_for/common_content/options_ref_cross_refs_for.htm

Quando si invoca il compilatore gcc o g++ si aggiungono delle opzioni che è importante conoscere
Ad esempio compilando per la configurazione Release

```
host $ <armtoolch>-g++ -Wall -O2 objectfilename sourcefilename.cpp
```

-Wall (This enables all the warnings about constructions that some users consider questionable, and that are easy to avoid)

-O2 Ottimizzazione per la velocità
-O3 Ottimizzazione ancora maggiore

Ad esempio compilando per la configurazione Debug

```
host $ <armtoolch>-g++ -Wall -g objectfilename sourcefilename.cpp
```

-g Chiede al compilatore di includere nel file oggetto tutti i simboli di debug

Ad esempio compilando una libreria

```
host $ <armtoolch>-g++ -shared -f PIC -WI,-soname,libkxgpi.so -o libkxgpi.so -L/home/gabriele/sdk_5_07_00_00/linux-devkit/arm-arago-linux-gnueabi/usr/lib
```

-shared	Specifica che la libreria è dinamica
-f PIC	Flag che abilita Positioning Independent Code, importante per librerie
<internal_name>.so	Specifica il nome della libreria internamente al kernel
<file_name>.so	Specifica il nome del file della libreria
-L<libs_path>	Specifica la path per librerie esterne eventualmente usate

53 Other

53.1 make clean, mrproper, distclean

Tipicamente quando lanciamo il make abbiamo a disposizione tre diversi livelli di cancellazione dei file prodotti da una precedente compilazione.

Per cancellare solo gli obj (.o)

```
host $ make clean
```

Per cancellare anche eventuali file di configurazione (p.es. .config nel caso del kernel)

```
host $ make mrproper
```

Per cancellare anche tutti i prodotti che possono essere installati (p.es. .a e .lib)

```
host $ make distclean
```

53.2 Redirezione

Per ridirezionare lo standard output di un processo <program> su <file>

```
host $ <program> 1> <file>
```

per ridirezionare lo standard error

```
host $ <program> 2> <file>
```

per ridirezionare entrambe

```
host $ <program> &> <file>
```

53.3 Link simbolici e hard

Un soft-link (link simbolico) è un file, che contiene il path del file puntato.

Per creare un link simbolico <linkname> a <path>

```
host $ ln -s <path> <linkname>
```

Per ridefinire un link simbolico che già esiste

```
host $ ln -snf <path> <linkname>
```

Un hard-link è una entry della di una directory, contenente il puntatore all'inode del file puntato.

Uno stesso inode può essere puntato da più hard-link.

Digressione sugli hard-link. Qualsiasi tipo di filesystem (FAT, ext4, sysfs, ecc.) deve interfacciarsi a userspace mediante il VFS (*Virtual File System*), una sorta di astrazione di tutti i possibili filesystem di linux.

In realtà

```
host $ rm <filename>
```

non rimuove <filename>, ma rimuove l'hard-link all'inode di <filename>, presente nella entry della directory che contiene <filename>

```
host $ unlink <filename>
```

L'utilità degli hard-link si ha, ad esempio, nella seguente situazione: supponiamo di avere una cartella molto grande, p.es. i sorgenti di una particolare versione di kernel, <kernelsource>. Vogliamo farne una copia e applicarvi delle patches.

Invece di utilizzare

```
host $ cp <kernelsource> <kernelsource_b>
```

che esegue una copia di tutta la cartella (occupando una marea di spazio) conviene utilizzare

```
host $ cp -l <kerneldir> <kernelsource_b>
```

che in realtà crea solamente nella entry di <kernelsource_b> un hard-link all'inode di ciascuno dei file di <kernelsource>.

Quando si applica una patch a un file di <kernelsource_b>

- 1) Viene creato il file patchato <filename_b> all'interno di <kernelsource_b>
- 2) l'hard-link all'inode di <filename>, nella entry di <kernelsource_b>, viene sostituito con l'hard-link all'inode <filename_b>

53.1 Integrare un RTC

Integrare un RTC in una applicazione linux è molto semplice.

Supponiamo di volere utilizzare un DS1307.

Nella configurazione di default del kernel, **am3355x-xnrg_defconfig**, abilitiamo il suddetto device.

```
#  
# I2C RTC drivers  
#  
# CONFIG_RTC_DRV_ABB5ZES3 is not set  
# CONFIG_RTC_DRV_ABX80X is not set  
CONFIG_RTC_DRV_DS1307=y  
...
```

Nel file DTS, <linux>/arch/arm/boot/dts/am3355x-xnrg.dts, sotto la sezione del bus I2C dove abbiamo collegato il dispositivo

```
&i2c2 {  
...  
    rtc-X: ds1307@68 {
```

Engineering Specification - confidential

```
compatible = "dallas,ds1307";
reg = <0x68>;
};
```

dove rtc-X è il nome scelto per il driver, mentre 0x68 è l'indirizzo I2C

In uno degli script di avvio del target, **/etc/init.d/S00X**, definiamo e chiamiamo le due funzioni di init e stop

```
function clock_init()
{
    echo "Creating RTC DS1307"
#    /bin/echo ds1307 0x68 > /sys/bus/i2c/devices/i2c-2/new_device
    if [ ! -c /dev/rtc1 ];then
        echo "Device rtc1 not found!"
        ln -snf /dev/rtc0 /dev/rtc-X  (l'applicazione farà riferimento al RTC della BBB)
        return
    fi
    ln -snf /dev/rtc1 /dev/rtc-X  (altrimenti l'applicazione farà riferimento al RTC DS1307)
    /sbin/hwclock -s -f /dev/rtc1  (<- setta il clock di sistema sul valore dell'HW clock RTC DS1307)
    /sbin/hwclock -w -f /dev/rtc0  (<- setta il clock della BBB sul valore del clock di sistema)
}

function clock_stop()
{
    ln -snf /dev/rtc0 /dev/rtc-X
#    /bin/echo 0x68 > /sys/bus/i2c/devices/i2c-2/delete_device
}

function X_init()
{
    printf "X init script: "
...
    load_modules
    clock_init
    echo "OK"
}

function X_stop()
{
    printf "X stop script: "
    clock_stop
    unload_modules
...
    echo "OK"
}
```

Questo RTC 1307 è stato mappato su **/dev/rtc1** in quanto **/dev/rtc0** è già riservato all'RTC della BBB, che però è inutilizzabile in quanto non ha un power supply di backup.

Si poneva anche il problema di possibili schede che non avevano ancora montato il DS1307. Quindi a runtime l'applicazione deve fare sempre riferimento a **dev/rtc-X**, link simbolico al vero device.

Se il dispositivo DS1307 è presente allora avremo

```
# ls -la /dev/rtc-X
lrwxrwxrwx 1 root root 9 Jan 1 01:00 /dev/rtc-X -> /dev/rtc1
```

altrimenti avremo

```
# ls -la /dev/rtc-X
lrwxrwxrwx 1 root root 9 Jan 1 01:00 /dev/rtc-X -> /dev/rtc0
```

che è l'RTC della BBB.

53.2 Thread e processi

ToDo

Differenze tra thread e processi

I thread hanno memoria condivisa, quindi se uno di essi scrive una flag, l'altro la vede.

I processi no.

Stato di un processo

Un processo può trovarsi in diversi stati.

Lo stato di un processo viene evidenziato da una lettera nella colonna STAT quando si esegue il comando di shell ps

'R': RUNNING

'S' : INTERRUPTIBLE SLEEP

'D': UNINTERRUPTIBLE SLEEP

'Z': ZOMBIE

'X': DEAD

Processi Zombie

Quando un processo viene terminato per qualche motivo tutta la memoria e le risorse ad esso associate vengono liberate così da poter essere utilizzate da altri processi.

Tuttavia, prima di diventare DEAD, il processo transita in uno stato chiamato Zombie, in attesa che il suo valore di uscita venga letto dal processo parent nella *control block* (*task struct* in linux) mediante la chiamata di sistema *wait()*.

Spetta al processo parent andare a leggere la *process table* del processo zombie perché questo venga effettivamente rimosso.

La chiamata *wait()* può essere eseguita in normale codice sequenziale ma, più comunemente, viene invocata all'interno di un gestore del segnale SIGCHLD, il quale è inviato al processo padre ogni volta un suo processo figlio muore.

Se la *wait* non viene chiamata, il processo zombie continua a rimanere nella tabella dei processi. In alcuni casi ciò viene fatto di proposito, ad esempio quando il processo padre, creando un nuovo processo figlio, vuole assicurarsi che questo abbia un PID diverso da un figlio creato in precedenza e appena terminato.

Processi zombie che persistono più a lungo di un breve periodo di tempo tipicamente indicano un bug del programma padre. Come capita per altri tipi di memory leak, la presenza di pochi processi zombie non è preoccupante, ma può evidenziare un problema che sotto maggiori carichi può mandare in crash il sistema. Dato che non vi è memoria allocata per i processi zombie a parte il *control block* stesso, il problema principale non è il consumo di memoria quanto l'esaurimento dei PID disponibili e il riempimento della tabella dei processi, con l'impossibilità di creare nuovi processi.

53.3 Garbage Collection e Reference Counting

Il garbage collector (GC) annota le aree di memoria non più referenziate, cioè allocate da un processo attivo, e le libera automaticamente. La garbage collection è stata inventata nel 1959 da John McCarthy per il linguaggio di Lisp.

Il JavaScript e il C# hanno il GC, il C e il C++ no.

Un tipo di GC è il Reference Counting, una tecnica di memorizzazione del numero di riferimenti, puntatori o handle a un oggetto.

Qt non usa GC ma il Reference Counting. Un oggetto QObject può avere una serie di children, e se viene distrutto allora distrugge tutti i suoi figli.

In Objective-C esiste Automatic Reference Counting (ARC), introdotto da Apple nel 2011 in iOS e OS X, dove il reference counting viene spostato dal programmatore al compilatore. In precedenza il programmatore doveva mandare dei messaggi di retain e release agli oggetti per deallocazione o prevenire la deallocazione. Con ARC, il compilatore lo fa automaticamente esaminando il codice sorgente e aggiungendo i messaggi di retain e release nel codice compilato.

ARC non è uguale al GC, visto che non ci sono thread in background che agiscono per deallocare gli oggetti non più referenziati. Al contrario del GC, ARC non gestisce automaticamente i riferimenti nei cicli; è compito del programmatore evitare i cicli utilizzando weak references.

53.4 Sicurezza informatica (Cybersecurity)

Dal punto di vista della sicurezza, Linux è rinomato perché una applicazione può piantare un processo, ma il kernel resta sempre in piedi. Un processo con privilegi di amministratore (root) può fare tutto. Le applicazioni XNRG e KX girano come utente root.

Se un'applicazione qualsiasi crasha possiamo solo resettare il prodotto. Sarebbe possibile fare una applicazione "home" robusta che gira come un utente privilegiato che lancia altre cose. Quando qualcosa crasha la palla ritorna all'applicazione home che fa qualcosa per gestire la situazione.

53.5 Memory Swap

Le applicazioni XNRG e KX non hanno lo swap della memoria. Si potrebbe attivare, ma al momento non c'è. Quanto significa che se la RAM fisica viene saturata, la macchina si pianta.

ToDo

53.6 Isolamento tra HW e SO

Un SO come Linux ha il controllo totale dell'HW.

Ci sono dei sistemi differenti in cui tra l'HW e Linux c'è un hypervisor, una sorta di SO minimale il cui lavoro principale è creare un isolamento. P.es. l'HW ha 4GB di RAM, ma l'hypervisor a Linux ne fa vedere solo 1GB.

53.7 Prontuario dei comandi di U-Boot (KX)

```
?      - alias for 'help'
askenv - get environment variables from stdin
base   - print or set address offset
bdinfo - print Board Info structure
boot   - boot default, i.e., run 'bootcmd'
bootd  - boot default, i.e., run 'bootcmd'
bootm  - boot application image from memory
bootp  - boot image via network using BOOTP/TFTP protocol
cmp    - memory compare
coninfo - print console devices and information
cp     - memory copy
crc32  - checksum calculation
dhcp   - boot image via network using DHCP/TFTP protocol
echo   - echo args to console
editenv - edit environment variable
env    - environment handling commands
exit   - exit script
ext2load- load binary file from a Ext2 filesystem
ext2ls - list files in a directory (default '/')
false  - do nothing, unsuccessfully
fatinfo - print information about filesystem
fatload - load binary file from a dos filesystem
fatls  - list files in a directory (default '/')
go    - start application at address 'addr'
help   - print command description/usage
iminfo - print header information for application image
imxtract- extract a part of a multi-image
itest  - return true/false on integer compare
```

loadb - load binary file over serial line (kermit mode)
loads - load S-Record file over serial line
loady - load binary file over serial line (ymodem mode)
loop - infinite loop on address range
md - memory display
mdc - memory display cyclic
mii - MII utility commands
mm - memory modify (auto-incrementing address)
mmc - MMC sub system
mmcinfo - display MMC info
mtest - simple RAM read/write test
mw - memory write (fill)
mwc - memory write cyclic
nfs - boot image via network using NFS protocol
nm - memory modify (constant address)
ping - send ICMP ECHO_REQUEST to network host
printenv - print environment variables
reset - Perform RESET of the CPU
run - run commands in an environment variable
saveenv - save environment variables to persistent storage
saves - save S-Record file over serial line
setenv - set environment variables
sf - SPI flash sub-system
showvar - print local hushshell variables
sleep - delay execution for some time
source - run script from memory
sspi - SPI utility command
test - minimal test like /bin/sh
tftpboot - boot image via network using TFTP protocol
true - do nothing, successfully
version - print monitor version

53.8 Prontuario dei comandi di U-Boot (BBB)

? - alias for 'help'
askenv - get environment variables from stdin
base - print or set address offset
bdinfo - print Board Info structure
boot - boot default, i.e., run 'bootcmd'
bootd - boot default, i.e., run 'bootcmd'
bootm - boot application image from memory
bootp - boot image via network using BOOTP/TFTP protocol
bootz - boot Linux zImage image from memory
cmp - memory compare
coninfo - print console devices and information
cp - memory copy
crc32 - checksum calculation
dfu - Device Firmware Upgrade
dhcp - boot image via network using DHCP/TFTP protocol
dm - Driver model low level access
echo - echo args to console
editenv - edit environment variable
eeprom - EEPROM sub-system
env - environment handling commands
exit - exit script
ext2load - load binary file from a Ext2 filesystem
ext2ls - list files in a directory (default /)
ext4load - load binary file from a Ext4 filesystem
ext4ls - list files in a directory (default /)
ext4size - determine a file's size
false - do nothing, unsuccessfully
fastboot - use USB Fastboot protocol
fatinfo - print information about filesystem
fatload - load binary file from a dos filesystem
fatls - list files in a directory (default /)
fatsize - determine a file's size
fatwrite - write file into a dos filesystem
fdt - flattened device tree utility commands
go - start application at address 'addr'
gpio - query and control gpio pins
gpt - GUID Partition Table
help - print command description/usage
i2c - I2C sub-system
iminfo - print header information for application image

imxtract- extract a part of a multi-image
 itest - return true/false on integer compare
load - load binary file from a filesystem
 loadb - load binary file over serial line (kermit mode)
 loads - load S-Record file over serial line
loadx - load binary file over serial line (xmodem mode)
 loady - load binary file over serial line (ymodem mode)
 loop - infinite loop on address range
ls - list files in a directory (default /)
 md - memory display
 mdio - MDIO utility commands
 mii - MII utility commands
 mm - memory modify (auto-incrementing address)
 mmc - MMC sub system
 mmcinfo - display MMC info
 mw - memory write (fill)
 nfs - boot image via network using NFS protocol
 nm - memory modify (constant address)
part - disk partition related commands
 ping - send ICMP ECHO_REQUEST to network host
 printenv - print environment variables
 reset - Perform RESET of the CPU
 run - run commands in an environment variable
 save - save file to a filesystem
 setenv - set environment variables
 sf - SPI flash sub-system
 showvar - print local hushshell variables
size - determine a file's size
 sleep - delay execution for some time
 source - run script from memory
spl - SPL configuration
 sspi - SPI utility command
 test - minimal test like /bin/sh
 tftpboot - boot image via network using TFTP protocol
 true - do nothing, successfully
ums - Use the UMS [User Mass Storage]
usb - USB sub-system
usbboot - boot from USB device
 version - print monitor, compiler and linker version

53.9 Opzioni di Boot

Sul Target, dal prompt di u-boot

```

target # setenv ipaddr 192.168.0.149 <ipaddr>
target # setenv ethaddr 00:08:ee:04:34:d2
target # setenv netmask 255.255.255.0
target # setenv gatewayip 192.168.0.221
target # setenv serverip 192.168.0.142
target # setenv nfsserverip 192.168.0.142
target # setenv rootpath /home/gabriele/workdir/kx/filesys
  
```

Se si lavora con NFS dal Server:

```

target # setenv serverip 192.168.0.203
target # setenv nfsserverip 192.168.0.203
target # setenv rootpath /home/X/common/sw/workdir/kx/filesys
  
```

static IP, nfs, tftp:

```

target # setenv bootargs console=ttyS2,115200n8 nointrd rw ip=${ipaddr}:${serverip}:255.255.255.0::off,root=dev/nfs
nfsroot=${nfsserverip}:${rootpath},nolock mem=128M rootdelay=3 panic=5
target # setenv bootcmd 'tftp;bootm'
  
```

dhcp, nfs, tftp:

```

target # setenv bootargs console=ttyS2,115200n8 nointrd rw ip=dhcp root=dev/nfs nfsroot=${nfsserverip}:${rootpath},nolock mem=128M
rootdelay=3
target # setenv bootcmd 'dhcp;bootm'
  
```

root fs su SD, kernel su SPI flash

```

target # setenv bootargs console=ttyS2,115200n8 root=/dev/mmcblk0p1 rw rootwait ip=off mem=32M
target # setenv bootcmd 'sf probe 0;sf read 0xc0700000 0x80000 0x220000;bootm 0xc0700000'
  
```

Nota: la rete viene disabilitata. Sarà il kernel a configurarla.

Engineering Specification - confidential

dhcp, boot kernel e mount ramdisk da flash SPI:

```
target # setenv bootargs mem=32M console=ttyS2,115200n8 root=/dev/ram0 rw initrd=0xc1180000,4M ip=dhcp eth=${ethaddr}
target # setenv bootcmd 'sf probe 0;sf read 0xc0700000 0xa0000 0x360000;sf 0xc1180000 0x400000 0x3e0000;bootm 0xc0700000'
```

dhcp, boot kernel e mount ramdisk da tftp:

```
target # setenv bootargs mem=32M console=ttyS2,115200n8 root=/dev/ram0 rw initrd=0xc1180000,4M ip=dhcp eth=${ethaddr}
target # setenv bootcmd 'tftp 0xc0700000 ulimage;tftp 0xc1180000 ramdisk-base.gz;bootm 0xc0700000'
```

frequenza massima per definizione punti di lavoro (456/408/372/300/200/96 MHz):

```
target # setenv maxcpuclk 456000000
```

opzione mem. Può servire per fare sì che Linux usi blocchi discontinui di memoria. P.Es.

```
target # setenv bootargs console=ttyS2,115200n8 nointrd rw ip=dhcp root=dev/nfs nfsroot=${nfsserverip}:${rootpath},nolock
mem=32M@0xc0000000 mem=64M@0xc4000000
```

opzione rootdelay. Può servire per ritardare il mount del root fs in modo che la rete sia sicuramente operativa. P.Es.

```
target # setenv bootargs console=ttyS2,115200n8 nointrd rw ip=dhcp root=dev/nfs nfsroot=${nfsserverip}:${rootpath},nolock mem=128M
rootdelay=3
```

La variabile bootargs è la concatenazione di alcune sottostringhe:

aggiunta di una com port di console

```
console=ttyS2,115200n8
```

aggiunta del ip address

```
ip=${ipaddr}:${serverip}:${gatewayip}:${netmask}:${hostname}:eth0:off
```

aggiunta della opzione dhcp

```
ip=dhcp mac_addr=${ethaddr}
```

aggiunta di un ramdisk

```
root=/dev/ram0 rw initrd=0xc1180000,18M
```

aggiunta di un JFFS2 (l'indice del nodo mtdblock su cui va montato dipende dalla partizione)

```
root=/dev/mtdblock4 rw rootfstype=jffs2 nointrd rootwait
```

aggiunta di un mmc (l'indice del nodo mmcblk su cui va montato dipende dalla partizione)

```
root=/dev/mmcblk0p2 rw rootwait ip=off
```

aggiunta di un NFS

```
root=/dev/nfs rw nfsroot=${serverip}:${rootpath} rootdelay=3
```

aggiunta size della memoria per linux e altre opzioni

```
mem=128M panic=1 davinci_mmc.use_dma=0 mac_addr=${ethaddr} sn=${serial#}
```

aggiunta della/e partizione/i

```
mtdparts=davinci_nand.1:128k(u-boot_env),128k(ubl),512k(u-boot),4M(kernel),-(rootfs)
```

specificazione del primo processo eseguito (di solito non usato in quanto già esiste il System V Init)

```
init=/sbin/myinit
```

La variabile bootcmd è la concatenazione di alcuni comandi da eseguire (con "run"):

boot kernel da TFTP

```
tftp c0700000 kx/ulimage; bootm c0700000
```

boot kernel da flash SPI

```
sf probe 0; sf read c0700000 b0000 350000; bootm c0700000
```

boot kernel da flash NAND (in questo esempio "kernel" è il nome della partizione)

```
nand read.e c0700000 kernel 400000; bootm c0700000
```

```
boot kernel da mmc  
mmc rescan 0;if fatload mmc 0 0xc0600000 boot.scr
```

53.10 Ripristinare le variabili di ambiente di default in u-boot

Per ripristinare le variabili di ambiente di default occorre corrompere il settore nella flash dove queste variabili vengono memorizzate, in modo tale che al boot, u-boot si accorga del problema e ripristini la situazione di default.

Quindi, occorre conoscere il supporto flash (es. flash SPI), l'offset dell'area (es. 576KB) e il size (es. 64KB). Poi si procede alla cancellazione del settore (i valori si intendono in formato hex)

```
target # sf probe 0  
target # sf erase 90000 10000
```

La stessa cosa si deve ottenere con un solo comando, avendolo correttamente definito in CONFIG_EXTRA_ENV_SETTINGS.

```
target # run er_sf_env
```

53.11 Leggere e scrivere la Flash SPI da u-boot

Lettura da flash SPI dei primi 10 bytes dell'ultimo settore da 64KB

```
target # sf probe 0  
target # sf read c0000000 7f0000 10000  
target # md.b c0000000 10
```

Modificare i 10 byte e riscrivelerli nella flash SPI

```
target # mm.b c0000000  
... (modificare 10 bytes)  
target # sf write c0000000 7f0000 10
```

53.12 Leggere e scrivere una periferica su I2C da u-boot

Scansione dei dispositivi presenti sul bus

```
target # i2c probe  
Valid chip addresses: 0E 21 48 4C
```

Lettura e scrittura su dispositivo MAG3110 (ADDR:0x0E)

```
target # i2c md 0x0e 0 0x11 (lettura registri interni)  
target # i2c mw 0x0e 0x11 0x80 (scrittura del registro 11 con 0x80)  
target # i2c mw 0x0e 0x10 0x01 (scrittura del registro 10 con 0x01)  
target # i2c md 0x0e 0 0x11 (lettura registri interni)  
0000: ff 0d e8 f2 b8 f8 5a c4 02 00 00 00 00 00 00 03 .....Z.....
```

53.13 Leggere e scrivere la Flash SPI da kernel

ToDo

53.14 Leggere e scrivere una periferica su I2C da kernel

Scansione dei dispositivi presenti sul bus del KX

```
# i2cdetect -y -a -r 2          (← scansion all bus i2c-2)
 0 1 2 3 4 5 6 7 8 9 a b c d e f
00: -----
10: -----
20: -----
30: ----- UU -----
40: -----
50: 50 ----- 54 -----
60: -----
70: -----
```



```
# i2cdetect -y -a -r 1          (← scansion all bus i2c-1)
 0 1 2 3 4 5 6 7 8 9 a b c d e f
00: ----- UU --
10: -----
20: -- UU -----
30: -----
40: UU ----- UU ----- UU -----
50: -----
60: 60 ----- 60 -----
70: -----
```

La sigla “UU” significa che la periferica i2c con quel determinato indirizzo è stata rilevata E esiste un driver per accedere ad essa. Ad esempio, il sensore di umidità e temperatura SHT21 presente sul KX ha indirizzo 0x40. In corrispondenza di questo leggiamo “UU”. Quindi

```
target $ cd /sys/bus/i2c/devices/1-0040
target $:/sys/bus/i2c/devices/1-0040# ls
driver      modalias      subsystem
humidity1_input name      temp1_input
hwmon       power       uevent
target $:/sys/bus/i2c/devices/1-0040# cat humidity1_input
36350
target $:/sys/bus/i2c/devices/1-0040# cat temp1_input
25555
root@am180x-evm:/sys/bus/i2c/devices/1-0040#
```

Per progetto Spartacus (XNRG):

Scansione dei dispositivi presenti sul bus del XNRG

```
# i2cdetect -y -a -r 0          (← scansion all bus i2c-0)
 0 1 2 3 4 5 6 7 8 9 a b c d e f
00: -----
10: -----
20: ----- UU -----
30: ----- 34 -----
40: -----
50: UU -----
60: -----
70: -----
```



```
# i2cdetect -y -a -r 1          (← scansion all bus i2c-1)
 0 1 2 3 4 5 6 7 8 9 a b c d e f
00: -----
10: -----
20: -- UU -----
30: -----
40: -----
50: -----
60: UU -----
70: -----
```



```
# i2cdetect -y -a -r 2          (← scansion all bus i2c-2)
 0 1 2 3 4 5 6 7 8 9 a b c d e f
00: ----- 09 - 0b 0c -----      (← battery charger (09,0c) and battery (0b))
10: -----
20: -----
30: ----- UU -----
40: -----
50: 50 ----- 54 -----          (<- pinmuxE2P (54), ..)
60: ----- UU -----
```

70: -----

53.15 Inviare comandi iWRAP al modulo BT da kernel

Per inviare comando <cmd> a iWRAP

```
target $ echo <cmd> > /dev/shm/iwrapcmd fifo
```

Esempio: per resettare

```
target $ echo RESET > /dev/shm/iwrapcmd fifo
```

Esempio: per interrompere attività RF

```
target $ echo 'TEST PAUSE' > /dev/shm/iwrapcmd fifo
```

Per visualizzare messaggi di log del driver iWRAP

```
target $ cat /var/log/iwrapd.log
Opening serial port '/dev/ttyUSB2'
Device reset (try 1)
New connection from device 84:A6:C8:1F:EB:FD (link 0, channel 1, profile RFCOMM)
Warning: Connection from device not yet paired (MAC=84:A6:C8:1F:EB:FD)
Warning: command time out
Warning: Not sure command was received, but I keep going anyway...
Warning: command time out
Warning: Not sure command was received, but I keep going anyway...
Warning: command time out
Warning: Not sure command was received, but I keep going anyway...
:: Module MAC is 00:07:80:1B:3A:11
:: Friendly name is KX-1234509876
Pairing MAC=84:A6:C8:1F:EB:FD
Pairing MAC=70:DE:E2:73:1E:DC
Pairing MAC=00:01:95:14:AB:55
Pairing MAC=00:01:95:1F:6F:9B
Pairing MAC=3C:15:C2:E0:B1:D7
Pairing MAC=6C:40:08:BD:EB:BB
Warning: Repeating configuration sequence...
:: Module MAC is 00:07:80:1B:3A:11
:: Friendly name is KX-1234509876
Device 84:A6:C8:1F:EB:FD paired again, resetting device info
Device 70:DE:E2:73:1E:DC paired again, resetting device info
Device 00:01:95:14:AB:55 paired again, resetting device info
Device 00:01:95:1F:6F:9B paired again, resetting device info
Device 3C:15:C2:E0:B1:D7 paired again, resetting device info
Device 6C:40:08:BD:EB:BB paired again, resetting device info
iWrap Daemon initialized
...
```

Per visualizzare anche le risposte del WT41 ai comandi, è necessario ricompilare il driver *iwrap* con l'opzione *DEBUG=y*

```
host $ cd kx-dev/usermode-drivers/iwrap
host $ make clean
host $ DEBUG=y make
host $ make install
```

dal terminale si potrà usare il seguente comando per vedere in tempo reale il logging dei messaggi

```
target $ tail -f /var/log/iwrapd.log
DEBUG[__iwrap_main:538]: select returned 1
DEBUG[__iwrap_main:571]: CMD FIFO: TEST PAUSE
DEBUG[my_iwrap_busy:29]: IWRAP BUSY
DEBUG[__iwrap_command:50]: Sent CMD: TEST PAUSE
<= RX FF, 4:      OK\r\n
<= RX FF, 5:      OK.\r\n
DEBUG[my_iwrap_idle:24]: IWRAP IDLE (lcr 0)
DEBUG[__iwrap_main:493]: Entering select loop...
...
```

Il driver KX del WT41 invia i parametri di inizializzazione al WT41, ma solo se non esiste il file */etc/iwrap.init* che viene creato una volta che i parametri sono stati inviati. Quindi i parametri vengono effettivamente inviati solo dopo un aggiornamento FW (quando il file non esiste).

Quindi, per forzare il reinvio dei parametri

```
target $ /etc/init.d/iwrapd stop
target $ rm /etc/iwrap.init
```

(-> termina il processo)

```
target $ /etc/init.d/iwrapd start
```

(-> riavvia il processo)

In realtà il file `/etc/iwrap.init` contiene quelle informazioni usate dal driver proprio in fase di inizializzazione del WT41 che, se differenti da quelle presenti nel settore `KX_env` della flash SPI, devono essere ritrasmesse al modulo.

53.16 Usare iWRAP con i programmi di test

Nel caso in cui si voglia far funzionare iwrap sganciato da `cuml` e `kmain`, tramite i suoi programmi di test

```
target $ /etc/init.d/iwrapd restart      (<- sgancia iwrap da kmain e riavvia iwrap. WT41 riparte spento)  
target $ iwrap-client-test              (<- programma che riavvia il WT41 e fa l'echo dei bytes ricevuti x100)  
target $ iwrap-monitor                 (<- stampa solo il numero di bytes ricevuti e trasmessi)
```

Per usare questi programmi non serve ricompilare in modo *DEBUG*.

53.17 Rispristinare il MAC Address nella Flash SPI da u-boot

Nel PSP corrente linux legge il MAC address dalla flash SPI, se è valido, altrimenti ne genera uno randomico.

Cancellare il settore (il penultimo della flash SPI) e riscrivere i primi 6 byte con l'indirizzo MAC desiderato

```
target # sf probe  
target # sf erase 7f0000 10000  
target # mm.b c0000000  
c0000000: fb ? 00  
c0000001: ef ? 08  
c0000002: fd ? ee  
c0000003: df ? 03  
c0000004: ff ? 6a  
c0000005: ff ? c4  
c0000006: bf ? q  
target # sf write c0000000 7f0000 6  
target # reset
```

From linux shell it would be possible only if you have a MTD partition:

```
target $ flash_eraseall /dev/mtd<n>  
target $ echo -n 'x00\x08\xee\x03\x6a\xc4' > /dev/mtd3
```

53.18 Ripristinare il settore kx config. nella Flash SPI da u-boot

Cancellare il settore ed eseguire la procedura di inizializzazione. Infine salvare

```
target # sf erase a0000 10000  
target # kxconfigset mm.b c0000000  
target # kxconfig set  
SerialNumber : 2012070007  
New SerialNumber: 2012070007  
MAC Address : 00:08:ee:06:01:ee  
New Address : 00:08:ee:06:01:ee  
LCD Configuration  
LCD Enable : 1  
Model Name : URT UMSH-8065MD-19  
Hor. Resolution : 320  
Ver. Resolution : 240  
MMC 0 Configuration  
MMC 0 Enable : 0
```

```
target # kxconfig save  
SF: Detected M25P64 with page size 64 KiB, total 8 MiB  
Configuration Saved  
target #
```

53.19 Inizializzare o modificare il serial number da u-boot

Ogni scheda KXDB ha un serial number che viene inserito da u-boot tramite l'utility *kxconfig*, è memorizzato nella variabile di ambiente *serial#*, viene passato al kernel nella stringa di bootargs e lo si ritrova in */proc/cpuinfo*

```
target # setenv serial#          (cancellare la variabile con il precedente serial number)  
target # saveenv  
target # kxconfig set  
SerialNumber : 2013090001      (cambiare il serial number)  
...  
target # kxconfig save  
target # reset
```

53.20 Leggere e scrivere la Flash NAND da u-boot

La lettura/scrittura da/su una flash comporta tipicamente il trasferimento dei dati alla/dalla ram.
Lettura da flash nand dei primi 10 bytes del secondo settore 128KB

```
target # nand read c0000000 20000 10  
target # md.b c0000000 10
```

Modificare i 10 byte e riscriverli nella flash

```
target # mm.b c0000000  
... (modificare 10 bytes)  
target # nand write.e c0000000 20000 10
```

Cancellare nella flash 10 byte

```
target # nand erase 20000 10
```

Se sulla flash sono state definite delle partizioni (verificare con *mtdparts*), si può usare il nome della partizione invece degli offset di indirizzo.

P.es. avendo definito la variabile di ambiente *mtdparts* con valore *mtdparts=nand.0:4M(ramdisk),-(rootfs)* potremo leggere dalla nand in entrambe i seguenti modi

```
target # nand read c0000000 400000 10
```

```
target # nand read c0000000 rootfs 10
```

Anche in scrittura si può usare il nome della partizione. Si può in tal caso omettere anche il size (u-boot assume filesize = dimensione della partizione)

```
target # nand write.e c0000000 ramdisk
```

53.21 Leggere e scrivere la Flash NAND da linux

Per fare il dump del contenuto della NAND da shell linux si può usare *nanddump*.

Esempio. Per copiare nel file <filename> il contenuto della partizione *mtd<n>* in formato binario

```
target $ nanddump -f <filename> /dev/mtd<n>
```

Engineering Specification - confidential

Esempio. Per copiare nel file <filename> il contenuto della partizione mtd<n> in formato canonico (offset + hex + ASCII)

```
target $ nanddump -f <filename> -c /dev/mtd<n>
```

Per copiare il file binario <filename> nella partizione mtd<n>

```
target $ flash_erase /dev/mtd<n> 0 0  
target $ nandwrite -p /dev/mtd<n> <filename>
```

Per scrivere pochi bytes:

```
target $ flash_erase /dev/mtd<n> 0 0  
target $ echo -n 'x00\x08\xee\x03\x6a\xc4' > /dev/mtd<n>
```

In qualche caso può essere necessario usare delle opzioni per leggere/scrivere anche i dati OOB (-o) e magari evitare di scrivere ECC (-n), come nel seguente esempio

```
target $ nanddump -f u-boot-nand.bin -o /dev/mtd3  
target $ flash_erase /dev/mtd3 0 0  
target $ nandwrite -o -n /dev/mtd3 u-boot-nand.bin
```

(<- leggo il contenuto di /dev/mtd3)
(<- lo riscrivo in /dev/mtd3)

53.22 Debuggare comunicazione cuml con OMNIA

Dal terminale seriale, interrompere applicazione *kmain*

```
target # /etc/init.d/kmain stop
```

Da terminale remoto

```
target # killall -1 cuml
```

Dal terminale seriale

```
target # cd /usr/share/X/kmain/  
target # ./g
```

Dal periferica Bluetooth (Android/iPad) inviare comandi cuml

Per esempio si possono chiedere i seguenti valori

BXB:

O2 Gain: 0x34B
O2 Baseline: 0x34D
O2 Delay: 0x34A
O2 Resp. time: 0x844 (fall)
O2 Trimmer: 0x837
CO2 Gain: 0x30B
CO2 Baseline: 0x30D
CO2 Delay: 0x30A
CO2 Resp. time: 0x81F (rise)
CO2 Trimmer: 0x839
Analyzers Pressure: 0x829
Sampling Flow: 0x830

DMC:

O2 Gain: 0x44B
O2 Baseline: 0x44D
O2 Delay: 0x44A
O2 Resp. time: 0x844
O2 Trimmer: 0x838
CO2 Gain: 0x48B
CO2 Baseline: 0x48D
CO2 Delay: 0x48A
CO2 Resp. time: 0x81F (rise)

CO2 Trimmer: 0x838
Analyzers Pressure: 0x82A
Sampling Flow: 0x830

53.23 Comunicazione cuml tramite SSH

Dal terminale seriale, interrompere applicazione *kmain*

```
target # /etc/init.d/kmain stop
```

Da terminale remoto

```
target # killall -1 cuml
```

Dal terminale seriale

```
target # cd /usr/share/X/kmain/  
target # ./g
```

Dal periferica Bluetooth (Android/iPad) inviare comandi cuml

53.24 Testare USB Host interface da u-boot (XNRG)

La Digital Board del XNRG ha una versione di U-Boot che ha delle funzioni di utility per testare le interfacce USB Host. Inseriamo una o due chiavette USB. Una delle due contenga un file *prova.txt*. Quindi digitiamo i seguenti comandi

```
target # usb reset  
resetting USB...  
USB0: scanning bus 0 for devices... Error: Cannot find high speed parent of usb-1 device  
unable to get device descriptor (error=-71)  
2 USB Device(s) found  
scanning usb for storage devices... 1 Storage Device(s) found  
target # fatload usb 0 0x82000000 prova.txt  
reading prova.txt  
5 bytes read in 19 ms (0 Bytes/s)
```

53.25 Testare I2C da u-boot (XNRG)

```
target # i2c dev 0  
Setting bus to 0  
target # i2c probe  
Valid chip addresses: 24 34 50  
target # i2c dev 1  
Setting bus to 1  
target # i2c probe  
Valid chip addresses: 21 60
```

53.26 Debuggare comunicazione cuml con host bluetooth

Accendere il target KX.

Engineering Specification - confidential

Inserire un adattatore USB-Bluetooth nel host (p.es. Parani UD100)

Cambiare il nome del host in quello memorizzato in <iwrap>/scripts/open_debug_console.sh
Sull'host è necessario aprire un programma di emulazione terminale su una BT virtual COM.
Il FW del KX, a partire dalla versione 1.3, crea uno pseudoterminale sempre attivo. Quando il KX riceve una richiesta di connessione da un host con nome **bnDyG721MAD** il driver iwrap connette la console linux al WT41 mediante lo pseudoterminale.

La password di root per accedere al filesystem si trova in <iwrap>/scripts/open_debug_console.sh

```
host $ cd kx-dev/usermode-drivers/iwrap
host $ ./open_debug_console.sh
```

Lo script

- 1) Imposta il nome del Parani (p.es. **bnDyG721MAD**)
- 2) Eseguire pairing (sudo bluetooth-wizard)
- 3) col KX associato al nome utente dell'host e vi si connette
- 4) Crea /dev/rfcomm0
- 5) Apre minicom su /dev/rfcomm0
- 6) Esegue login con password (p.es. **b3c6d787acb**, la si trova in <dev>/scripts/open_console.sh)

Nota: per generare la password si è ricorso al metodo generale descritto a paragrafo *Generare facilmente una password sicura su host*

Nel caso si conosca il KX non sia uno di quelli di sviluppo predefiniti ma KX-<serialNumber>

```
host $ TARGET=KX-<serialNumber> ./open_debug_console.sh
```

per uscire, da minicom

```
exit
Ctrl A
X
```

Quando segue è obsoleto:

Per le versioni precedenti del FW KX si utilizza un metodo differente.

Tra le applicazioni presenti nella repo di sviluppo del KX vi è la cartella *ucuml*

```
host # cd <dev>/apps/appskx/ucuml
```

Ricompile *ucuml* per host (necessario soltanto la prima volta)

```
host # make clean
host # make
```

Editare il file bth

```
host # cat bth
#!/bin/sh
sudo /etc/init.d/bluetooth restart
sudo rfcomm release hci0
hcitool dev
sudo hciconfig -a hci0 reset
sudo hciconfig -a hci0 name OMNIA1          (<- specificare Friendly Name che si vuole assegnare al Bluetooth device host)
sudo hciconfig -a hci0 name
sudo hciconfig hci0 -a piscan
hcitool scan
hcitool con
sudo rfcomm connect hci0 00:07:80:50:11:82      (<- specificare l'indirizzo BT del KX che si vuole chiamare)
```

Creare la connessione

```
host # sudo ./bth
bluetooth start/running, process 24695
Can't release device: No such device
Devices:
```

Engineering Specification - confidential

```
hci0: Type: BR/EDR Bus: USB
      BD Address: 00:01:95:1F:6F:9E ACL MTU: 310:10 SCO MTU: 64:8
      Name: 'OMNIA1'
Scanning ...
  00:07:80:50:11:82 KX-2014040008
  00:07:80:5A:51:50 KX-2013120006
  00:07:80:F1:F2:F3 KX-2013110004
Connections:
Connected /dev/rfcomm0 to 00:07:80:50:11:82 on channel 1
Press CTRL-C for hangup
```

Aprire un altro terminale nella stessa cartella e lanciare *ucuml*

```
host # cd <dev>/apps/appskx/ucuml
host # ./ucuml
ucuml v:1.0 - (GC-10.2014)
USAGE: - press 'enter' to gain hex keyboard
      - press 'ctrl+c' to exit

Insert hex sequence:                               (<- premere invio)
ff 79
H:> FF 79 15                                     (<- editare comando)
                                                (<- risposta dal KX)
                                                (<- premere invio)

Insert hex sequence:                               (<- editare comando)
ff 79
H:> FF 79 15                                     (<- risposta dal KX)
...
...
```

In alternativa lanciare eseguibile host *serialkx*

53.27 Inviare o ricevere file al/dal KX con zmodem

Collegarsi al KX tramite programma di emulazione seriale che supporta zmodem (su UART oppure su COM virtuale Bluetooth, USB, ecc).

Da minicom, KX in ricezione

Aprire terminale di emulazione seriale con supporto zmodem (minicom, TeraTerm)

```
target$ lrz -b
lrz waiting to receive.**B0100000023be50                                (<- il KX si mette in ascolto)
```

Ctrl + A

S

Scegliere opzione zmodem

Selezionare il file su host

Inviare

Da minicom, KX in trasmissione

Aprime terminale di emulazione seriale con supporto zmodem (minicom, TeraTerm)

```
target$ lsz -b <filename>
```

Per definire su minicom la cartella di download

Ctrl + A

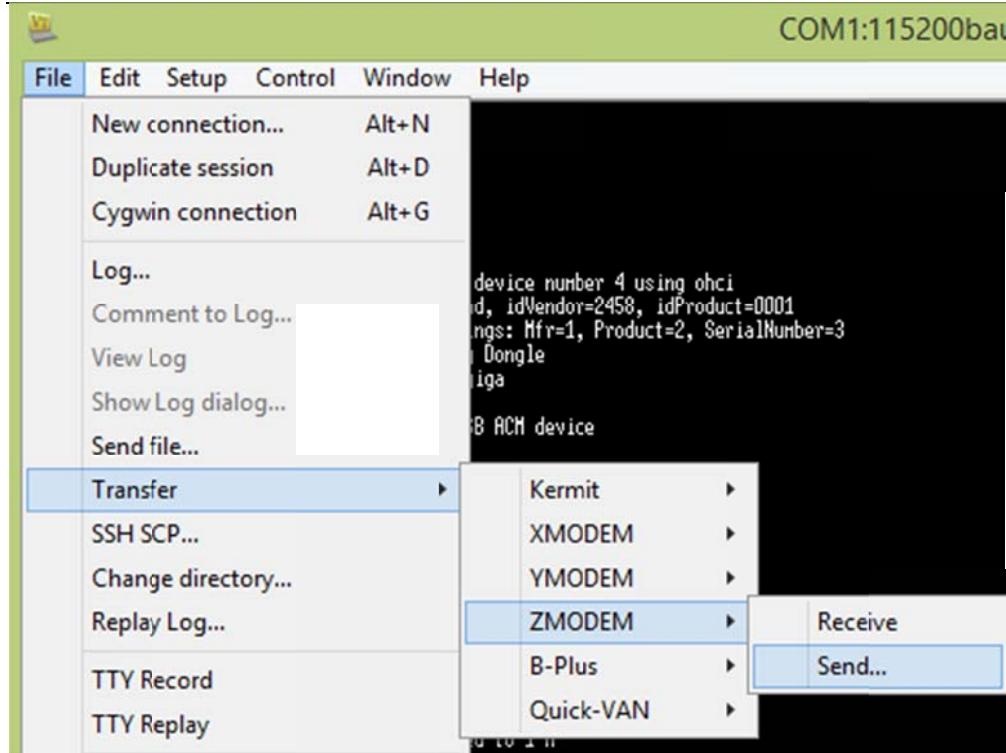
O

Selezionare il path

Da TeraTerm, KX in ricezione

Aprire terminale di emulazione seriale con supporto zmodem (minicom, TeraTerm)

```
target$ lrz -b                                         (<- si mette in ascolto)
```



Da TeraTerm, KX in trasmissione

Aprime terminale di emulazione seriale con supporto zmodem (minicom, TeraTerm)

53.28 Debuggare GPS

Per visualizzare le stringhe NMEA su teminale:

Attivare debug messages in gpss.c

```
host # vim <appskx>/gpss/gpss.c
#ifndef __DEBUG
#define __DEBUG(f,...)
#endif
```

(← mettere a 0 per attivare debug messages)

Ricompile e lanciare kmain in NFS su target in modalità debug

```
target # /etc/init.d/kmain stop
target # cd /usr/share/X/kmain/
target # ./g
```

Poi da un terminale remoto

```
target # killall -1 gpss
```

53.29 Definire un nuovo comando in u-boot

ToDo

54 Other

54.1 Tecniche di ottimizzazione (todo)

Per velocizzare l'accesso sul file system

- aggiungere alle opzioni di boot (bootargs) elevator=noop
 - aggiungere in /etc/fstab
- | | |
|----------------------------|---|
| noatime invece di relatime | (evita di memorizzare l'istante dell'ultimo accesso a i file) |
| nodiratime | |
| discard | |
| commit | |
| barrier=0 | |

Per vedere le opzioni correnti usare il comando

```
target $ cat /proc/mounts
```

- root@am180x-evm:~# echo 0 > /proc/sys/vm/swappiness

54.2 Tecniche di programmazione parallela

Oggi i sistemi multicore stanno diventando sempre più diffusi. La programmazione parallela permette di sfruttare questa possibilità.

OpenMP è una semplice estensione dei compilatori C/C++/Fortran che permette di aggiungere il parallelismo implicito a un programma senza necessità di doverlo riscrivere.

OpenMP è supportato da GCC, Clang++, Microsoft Visual C++.

OpenMP consiste di una serie di direttive di precompilazione (#pragmas). I compilatori che non riconoscono quelle direttive semplicemente le ignorano e tutto procede normalmente.

Esempio:

Consideriamo un semplice esempio in cui vogliamo inizializzare una tabella in parallelo (multiple threads)

```
target $ cat /proc/mounts
#include <cmath>

int main()
{
    const int size = 256;
    double sinTable[size];

    #pragma omp parallel for
    for(int n=0; n<size; ++n)
        sinTable[n] = std::sin(2 * M_PI * n / size);

    // the table is now initialized
}
```

Ciascun thread inizializzerà un aportzione della tabella.

Può essere compilato in questo modo

```
host $ g++ tmp.cpp -fopenmp
```

Esempio:

Calculating the Mandelbrot fractal in parallel (host computer)

This program calculates the classic Mandelbrot fractal at a low resolution and renders it with ASCII characters, calculating multiple pixels in parallel.

```
#include <complex>
#include <cstdio>

typedef std::complex<double> complex;

int MandelbrotCalculate(complex c, int maxiter)
{
    // iterates z = z + c until |z| >= 2 or maxiter is reached,
    // returns the number of iterations.
    complex z = c;
    int n=0;
    for(; n<maxiter; ++n)
    {
        if( std::abs(z) >= 2.0) break;
        z = z*z + c;
    }
    return n;
}
int main()
{
    const int width = 78, height = 44, num_pixels = width*height;

    const complex center(-.7, 0), span(2.7, -(4/3.0)*2.7*height/width);
    const complex begin = center-span/2.0;//, end = center+span/2.0;
    const int maxiter = 100000;

#pragma omp parallel for ordered schedule(dynamic)
    for(int pix=0; pix<num_pixels; ++pix)
    {
        const int x = pix%width, y = pix/width;

        complex c = begin + complex(x * span.real() / (width +1.0),
                                     y * span.imag() / (height+1.0));

        int n = MandelbrotCalculate(c, maxiter);
        if(n == maxiter) n = 0;

#pragma omp ordered
        {
            char c = '';
            if(n > 0)
            {
                static const char charset[] = ".c8M@jawrpogOQEPEGJ";
                c = charset[n % (sizeof(charset)-1)];
            }
            std::putchar(c);
            if(x+1 == width) std::puts("|");
        }
    }
}
```

Un'altra direttiva (#pragma shared) permette di rendere visibile una certa area di memoria a tutti i processi

54.3 Coding Style

Il kernel linux segue delle regole di coding, descritte in

<linux>/Documentation/CodingStyle

Un aspetto è ad esempio l'indentazione (linux utilizza 8 caratteri). Un altro aspetto è il numero di colonne (80 per linux).

Indent è un tool che si può installare con

```
host $ apt-get install indent
```

che prende in ingresso un file sorgente <sourcefile> e vi applica le corrette intentazioni e spaziature.

Le opzioni di indent che corrispondono al coding style di Use Kernighan & Ritchie sono

```
host $ indent -kr <sourcefile>
```

e corrisponde a

```
host $ indent -nbad -bap -bbo -nbc -br -brs -c33 -cd33 -ncdb -ce -ci4 -cli0 -cp33 -cs -d0 -di1 -nfcl -nfca -hnl -i4 -ip0 -l75 -lp -npcs -nprs -npsl -saf -sai -saw -nsc -nsob -nss <sourcefile>
```

Le opzioni di indent che corrispondono al coding style di linux sono

```
host $ indent -npro -kr -i8 -ts8 -sob -l80 -ss -nsc -cp1 <sourcefile>
```

Tutte le opzioni sono

--blank-lines-after-commas	-bc
--blank-lines-after-declarations	-bad
--blank-lines-after-procedures	-bap
--blank-lines-before-block-comments	-bbb
--braces-after-if-line	-bl
--braces-after-func-def-line	-blf
--brace-indent	-bli
--braces-after-struct-decl-line	-bls
--braces-on-if-line	-br
--braces-on-func-def-line	-brf
--braces-on-struct-decl-line	-brs
--break-after-boolean-operator	-nbbo
--break-before-boolean-operator	-bbo
--break-function-decl-args	-bfda
--break-function-decl-args-end	-bfde
--case-indentation	-clin
--case-brace-indentation	-cbin
--comment-delimiters-on-blank-lines	-cdb
--comment-indentation	-cn
--continuation-indentation	-cin
--continue-at-parentheses	-lp
--cuddle-do-while	-cdw
--cuddle-else	-ce
--declaration-comment-column	-cdn
--declaration-indentation	-din
--dont-break-function-decl-args	-nbfda
--dont-break-function-decl-args-end	-nbfde
--dont-break-procedure-type	-npsl
--dont-cuddle-do-while	-ncdw
--dont-cuddle-else	-nce
--dont-format-comments	-nfca
--dont-format-first-column-comments	-nfcl
--dont-line-up-parentheses	-nlp
--dont-left-justify-declarations	-ndj
--dont-space-special-separator	-nss
--dont-star-comments	-nsc
--else-endif-column	-cpn
--format-all-comments	-fca
--format-first-column-comments	-fc1
--gnu-style	-gnu
--honour-newlines	-hnl
--ignore-newlines	-nhnl
--ignore-profile	-npro
--indent-label	-iln

--indent-level	-in
--k-and-r-style	-kr
--leave-optional-blank-lines	-nsob
--leave-preprocessor-space	-lps
--left-justify-declarations	-dj
--line-comments-indentation	-dn
--line-length	-ln
--linux-style	-linux
--no-blank-lines-after-commas	-nbc
--no-blank-lines-after-declarations	-nbaf
--no-blank-lines-after-procedures	-nbap
--no-blank-lines-before-block-comments	-nbbb
--no-comment-delimiters-on-blank-lines	-ncdb
--no-space-after-casts	-ncs
--no-parameter-indentation	-nip
--no-space-after-for	-nsaf
--no-space-after-function-call-names	-npes
--no-space-after-if	-nsai
--no-space-after-parentheses	-nprs
--no-space-after-while	-nsaw
--no-tabs	-nut
--no-verbosity	-nv
--original	-orig
--parameter-indentation	-ipn
--paren-indentation	-pin
--preserve-mtime	-pmt
--preprocessor-indentation	-ppin
--procnames-start-lines	-psl
--space-after-cast	-cs
--space-after-for	-saf
--space-after-if	-sai
--space-after-parentheses	-prs
--space-after-procedure-calls	-pcs
--space-after-while	-saw
--space-special-semicolon	-ss
--standard-output	-st
--start-left-side-of-comments	-sc
--struct-brace-indentation	-sbin
--swallow-optional-blank-lines	-sob
--tab-size	-tsn
--use-tabs	-ut
--verbose	-v

54.4 Ridirigere la shell linux su PC tramite /dev/ttyGS0

La porta USB device del KX si presenta sul PC Windows come Serial Gadget 2.4 e in definitiva come una porta COM viruale.

Per ridirigere il terminale di linux su questa interfaccia

```
target $ /sbin/getty -L /dev/ttyGS0 115200
```

E' possibile impostare la cosa in /etc/inittab con la seguente linea

```
S:2345:respawn:/sbin/getty 115200 ttyGS0
```

Oppure, come è al momento implementato, basta mettere SW2.3 a ON e riavviare. U-boot inserirà in botargs "console=ttyGS0" e lo script /etc/rcS.d/S75kxboot.sh effettuerà lo switch della console

```
if test -c /dev/ttyGS0; then  
    chown root:root /dev/ttyGS0  
    chmod 666 /dev/ttyGS0  
    stty -F /dev/ttyGS0 -echo  
  
# If required from bootargs, redirect the linux console to USB serial device port  
read CMDLINE < /proc/cmdline  
for x in $CMDLINE; do  
    [[ $x = console=ttyGS0* ]] || continue  
    /sbin/getty -L /dev/ttyGS0 115200  
done  
fi
```

54.5 Cambiare il MAC address e IP address da shell linux

Modificare il MAC address da linux

```
target $ ifconfig eth0 down  
target $ ifconfig eth0 hw ether XX:XX:XX:XX:XX:XX  
target $ ifconfig eth0 up
```

Modificare IP address da linux

```
target $ sudo ifconfig eth0 x.x.x.x
```

Per settare un IP address statico si usa oppure /etc/network/interfaces

Per accedere al livello fisico dell'interfaccia di rete da linux

```
target $ ethtool eth0
```

54.6 Utilizzare il supporto Frequency Scaling nel kernel

Si suppone che il supporto Frequency Scaling sia stato abilitato nel kernel tree

Per vedere i punti di lavoro disponibili e impostare quello desiderato:

```
target $ cat /sys/devices/system/cpu/cpu0/cpufreq/scaling_available_frequencies  
456000 408000 372000 300000 200000 96000  
target $ echo 300000 > /sys/devices/system/cpu/cpu0/cpufreq/scaling_setspeed
```

(in KHz)
(set 300 MHz)

Nota: Nel sistema KX il Frequency Scaling deve essere disabilitato in quanto influenzerebbe le periferiche eCap/PWM che utilizzano il system clock (da verificare)

54.7 Utilizzare il supporto GPIO nel kernel

Si suppone che il supporto GPIO sia stato abilitato nel kernel tree

Device Drivers -> GPIO Support -> /sys/class/gpio/... (sys interface)

```
target $ cd /sys/class/gpio
```

Per controllare un GPIO occorre innanzitutto esportarlo con (p.es. GPIO6[0])

```
target $ echo 6 > export
```

Nota: questa operazione non va a buon fine se il GPIO in questione è già stato richiesto dal kernel con gpio_request();

Nota: in generale, il numero da associare a gpioN[M] è gpios_per_bank*N+M. Per AM1808 è 16*N+M

Impostare direzione e verificare stato logico

```
target $ cd gpio6  
target $ echo 'in' > direction  
target $ cat direction  
in
```

Engineering Specification - confidential

```
target $ cat value  
1
```

Vi è anche un supporto debug per GPIO

```
target $ mount -t debugfs debugfs /sys/kernel/debug  
target $ cat /sys/kernel/debug  
GPIOs 0-31, DaVinci:  
gpio-1 (SPB_OUTn      ) in hi  
gpio-2 (SPB_PSHOLD    ) out hi  
gpio-5 (USER_SW2      ) in lo  
gpio-6 (USER_SW3      ) in lo  
gpio-9 (TURB_SIG2     ) in hi  
gpio-10 (USER_SW1     ) in hi  
gpio-11 (SD0_DISn     ) out hi  
gpio-20 (sda          ) in hi  
gpio-21 (scl          ) in hi  
gpio-31 (ON_BD_USB_OVC ) in hi irq-132 edge-both
```

```
GPIOs 32-63, DaVinci:  
gpio-32 (LCD_TEST     ) out hi  
gpio-36 (AUXBAT_EN    ) out hi  
gpio-38 (MDIO_CLK_EN   ) out lo  
gpio-39 (STATLED_EN   ) out hi  
gpio-40 (TTLIN_VIO    ) in lo  
gpio-44 (LCD_PWR      ) out hi  
gpio-57 (SPI_LCDEN    ) out hi
```

```
GPIOs 64-95, DaVinci:  
gpio-64 (MMC_CD       ) in hi  
gpio-65 (MMC_WP       ) in hi
```

```
GPIOs 96-127, DaVinci:  
gpio-97 (BUZZ_EN      ) out lo  
gpio-98 (KXAB_IO9     ) out lo  
gpio-99 (KXAB_IO10    ) out lo  
gpio-100 (KXAB_IO11   ) out lo  
gpio-101 (MC_BBn      ) out lo  
gpio-104 (ANB_OE      ) out hi  
gpio-106 (USB_BYPASSn ) in hi  
gpio-107 (SPB_INTn    ) in hi  
gpio-108 (IOEXP_INTn  ) in hi  
gpio-109 (USB_RSTn    ) out hi  
gpio-110 (LEE2_EN     ) out lo
```

```
GPIOs 128-143, DaVinci:  
gpio-138 (SAMPL_EN    ) out lo  
gpio-140 (LEE1_EN     ) out lo  
gpio-141 (KXAB_IO12   ) out lo  
gpio-142 (KXAB_IO13   ) out lo  
gpio-143 (KXAB_IO8    ) out lo
```

```
GPIOs 144-159, i2c/1-0021, tca6416, can sleep:  
gpio-144 (deep_sleep_en ) out lo  
gpio-146 (cdc_muxsel   ) out hi  
gpio-150 (turb_ifsel0  ) out hi  
gpio-151 (turb_ifsel1  ) out hi  
gpio-152 (gps_pwr      ) out lo  
gpio-153 (usb_enum     ) out hi  
gpio-154 (ant_sleep    ) out lo  
gpio-155 (ant_suspend  ) out hi  
gpio-156 (bt_RST        ) out hi  
gpio-157 (cdc_pwr      ) out hi  
gpio-159 (co2_pwr      ) out hi
```

Esempio. For switching-on/off the CO2 Sensor use the CO2_PWRn (gpio 159, line signal is active low)

```
target $ cd /sys/class/gpio/  
target $ echo 159 > export  
target $ cd co2_pwr/  
target $ echo 'out' > direction  
target $ echo '0' > value  
target $ echo '1' > value
```

(-> CO2 Sensor is switched-ON)
(-> CO2 Sensor is switched-OFF)

Una volta finite di utilizzare il GPIO effettuare l'unesport

```
target $ echo 159 > unexport
```

54.8 Utilizzare il supporto LEDS nel kernel

Si suppone che il supporto LEDS_GPIO sia stato abilitato nel kernel tree e che nel kernel sia stato inizializzato il platform_device /eds-gpio

Configurazione kernel:

Da menuconfig:

Abilitare DeviceDrivers->LED Support->LED Class Suppor
Abilitare DeviceDrivers->LED Support->LED Support for GPIO connectedLEDs
Abilitare DeviceDrivers->LED Support->Platform device bindings for GPIO LEDs
Abilitare DeviceDrivers->LED Support->LED Trigger Support-> all
Abilitare DeviceDrivers->LED Support->

```
CONFIG_LEDS=y
CONFIG_LEDS_CLASS=y
CONFIG_LEDS_GPIO=y
CONFIG_LEDS_GPIO_PLATFORM=y
CONFIG_LEDS_TRIGGERERS=y
CONFIG_LEDS_TRIGGER_TIMER=y
CONFIG_LEDS_TRIGGER_HEARTBEAT=y
CONFIG_LEDS_TRIGGER_BACKLIGHT=y
CONFIG_LEDS_TRIGGER_GPIO=y
CONFIG_LEDS_TRIGGER_DEFAULT_ON=y
```

Verifica:

```
target $ ls /sys/class/leds/
LED_CHECK LED_MARK LED_ONOFF LED_REC LED_STATUS
target $ ls /sys/class/leds/LED_MARK
brightness max_brightness subsystem uevent device power trigger
```

Verificare funzionamento

```
target $ echo 1 > /sys/class/leds/LED_MARK/brightness
(target $ echo 0 > /sys/class/leds/LED_MARK/brightness
(il LED si accende)
(il LED si spegne)
```

Sincronizzare il LED su diversi eventi HW

```
target $ cat /sys/class/leds/LED_MARK/trigger
[none] nand-disk mmc0 timer heartbeat backlight gpio default-on
target $ echo heartbeat > /sys/class/leds/LED_MARK/trigger
(target $ echo mmc0 > /sys/class/leds/LED_MARK/trigger
(<- il led lampeggia)
(lampeggia se la SD card è inserita)
```

54.8.1 LEDS per KX

Quanto detto sopra vale per KX

54.8.2 LEDS per Spartacus

Nel progetto XNRG vi è un solo pulsante che necessita del supporto, LED_ONOFF. Deve essere inserito nel DTS

```
<linux>/arch/arm/boot/dts/am335x-xnrg.dts
```

Engineering Specification - confidential

```
leds {  
    compatible = "gpio-leds";  
    led-onoff {  
        label = "led-onoff";  
        /* gpios = <&gpio2 3 GPIO_ACTIVE_HIGH>; /* final board */  
        gpios = <&gpio0 26 GPIO_ACTIVE_HIGH>; /* proto display board */  
        default-state = "on";  
    };
```

```
CONFIG_LEDS_CLASS=y  
CONFIG_LEDS_GPIO=y  
CONFIG_LEDS_SYSCON=y  
CONFIG_LEDS_TRIGGERERS=y  
CONFIG_LEDS_TRIGGER_TIMER=y  
CONFIG_LEDS_TRIGGER_ONESHOT=y  
CONFIG_LEDS_TRIGGER_HEARTBEAT=y  
CONFIG_LEDS_TRIGGER_BACKLIGHT=y  
CONFIG_LEDS_TRIGGER_GPIO=y  
CONFIG_LEDS_TRIGGER_DEFAULT_ON=y  
CONFIG_LEDS_TRIGGER_TRANSIENT=y
```

Verifica:

```
target $ cd /sys/class/leds/led-onoff  
target $ echo "heartbeat" > trigger  
target $ echo "default-on" > trigger
```

(<- il led lampeggia)
(<- il led torna acceso fisso)

54.9 Utilizzare il supporto KEYS nel kernel

In questo caso vogliamo utilizzare il sottosistema /dev/event/input per i pulsanti del KX. E' lo stesso utilizzato per il touchscreen e per il mouse.

Si suppone che il supporto KEYBOARD_GPIO_POLLED sia stato abilitato nel kernel tree e che nel kernel sia stato inizializzato il platform_device *gpio-keys-polled*

Configurazione kernel:

Da menuconfig:

Abilitare DeviceDrivers->InputDeviceSupport->Keyboards->Polled GPIO buttons
Abilitare DeviceDrivers->InputDeviceSupport->Keyboards->TCA6416 Keypad Support

```
CONFIG_KEYBOARD_GPIO_POLLED=y  
CONFIG_INPUT_KEYBOARD=y  
CONFIG_KEYBOARD_TCA6416=y  
CONFIG_KEYS=y
```

Verifica:

```
target $ od -x /dev/input/event0  
00000000 a6d3 552d 8ae6 0002 0001 0042 0001 0000 (<- premuto PB_REC)  
00000020 a6d3 552d b803 0002 0000 0000 0000 0000  
00000040 a6d3 552d a548 0008 0001 0042 0000 0000 (<- lasciato PB_REC)  
00000060 a6d3 552d a561 0008 0000 0000 0000 0000  
0000100 a6db 552d a69a 0008 0001 0041 0001 0000 (<- premuto PB_MARK)  
0000120 a6db 552d a6b4 0008 0000 0000 0000 0000  
0000140 a6dc 552d b3dd 000b 0001 0041 0000 0000 (<- lasciato PB_MARK)  
0000160 a6dc 552d b3f7 000b 0000 0000 0000 0000  
0000200 a6e0 552d c27a 000e 0001 0040 0001 0000 (<- premuto PB_CHECK)  
0000220 a6e0 552d c44e 000e 0000 0000 0000 0000  
0000240 a6e1 552d b4fe 000b 0001 0040 0000 0000 (<- lasciato PB_CHECK)  
0000260 a6e1 552d b6c4 000b 0000 0000 0000 0000  
0000200 a6e0 552d c27a 000e 0001 003f 0001 0000 (<- premuto PB_ONOFF)
```

Engineering Specification - confidential

```
0000220 a6e0 552d c44e 000e 0000 0000 0000 0000
0000240 a6e1 552d b4fe 000b 0001 003f 0000 0000 (<- lasciato PB_ONOFF)
0000260 a6e1 552d b6c4 000b 0000 0000 0000 0000
```

La terzultima colonna indica il codice del tasto, in questo caso KEY_F8 (0x42), KEY_F7 (0x41), KEY_F6 (0x40) e KEY_F5 (0x3f). La penultima colonna indica l'evento (0x0001 pressione, 0x0000 rilascio).

Attualmente la rilevazione dei pulsanti viene gestita in Qt. In Qt i codici dei tasti differiscono da quelli sopradetti a meno di una costante additiva. P.es. l'evento con codice KEY_F5 (0x3f) viene ricodificato da Qt in 0x45.

Per progetto Spartacus (XNRG):

Nel progetto XNRG vi è un solo pulsante che necessita del supporto, PB_ONOFF. Deve essere inserito nel DTS

```
<linux>/arch/arm/boot/dts/am335x-xnrg.dts
```

```
gpio_keys {
    compatible = "gpio-keys";
    #address-cells = <1>;
    #size-cells = <0>;
    /*autorepeat;*/
    button@014 {
        label = "XNRG onoff";
        linux.code = <53>; /* custom */
        /* PB_INTN <====> gpio0[14] */
        interrupt-parent = <&gpio0>;
        interrupts = <14 IRQ_TYPE_EDGE_FALLING>;
        debounce-interval = <5>; /* ms */
        linux,input-type = <1>; /* EV_KEY */
    };
};
```

```
CONFIG_KEYBOARD_GPIO=y
CONFIG_INPUT_KEYBOARD=y
CONFIG_KEYS=y
```

Verifica:

```
target $ od -x /dev/input/event2
00000000 a6d3 552d 8ae6 0002 0001 003f 0001 0000
                                                 (<- premuto tasto ONOFF)
```

Notare che il device è input2. Infatti input1 è riservato al touch-screen.

La terzultima colonna indica il codice del tasto, in questo caso 0x3f. La penultima colonna indica l'evento (0x0001 pressione, 0x0000 rilascio).

54.10 Utilizzare il supporto ECAP-PWM davinci nel kernel

Si suppone che il supporto ECAP/PWM davinci sia stato abilitato nel kernel tree e che nel kernel sia stato inizializzato.

Configurazione kernel:

Da menuconfig:

```
CONFIG_HAVE_PWM=y
CONFIG_GENERIC_PWM=y
CONFIG_DAVINCI_EHRPWM=y
CONFIG_ECAP_PWM=y
CONFIG_ECAP_PWM=y
```

CONFIG_ECAP_CAP=y

Verifica PWM:

```
target $ echo 1 > /sys/class/pwm/ecap.2/request          (impegno la risorsa)
target $ echo 1 > /sys/class/pwm/ecap.2/polarity        (configura polarità alta)
target $ echo 100 > /sys/class/pwm/ecap.2/period_freq   (configura freq a 100 Hz)
target $ echo 50 > /sys/class/pwm/ecap.2/duty_percent    (configura 50% duty-cycle)
target $ echo 1 > /sys/class/pwm/ecap.2/run              (start)
```

A questo punto la linea PWM trasmette un'onda quadra a 100Hz con 50% duty-cycle

Nota: prima di riconfigurare i parametri di funzionamento, fermare la periferica con

```
target $ echo 0 > /sys/class/pwm/ecap.2/run             (stop)
```

54.10.1 eCap per KX

Quanto detto sopra vale per KX

54.10.2 eCap per Spartacus

Su Spartacus è presente nella cartella
/sys/X/devices/

```
host$ ls /sys/X/devices/lee_valve_pwm
conf  driver_override  modalias  power  sysfreq_hz
driver  ecap_turbine  of_node  subsystem uevent
```

For example, to test the V3_PWM GAS, do the following

```
host$ cd /sys/X/devices/ lee_valve_pwm
host$ cat parameters/test                         (<- a 5 ms pulse is sent to the V3 valve )
```

54.11 Utilizzare il supporto ECAP-PWM Turbina nel kernel

Il driver ECAP-PWM è stato customizzato da X per leggere gli impulsi generati da una turbina (eCAP) e, sulla base di una marcia (gear) generare degli impulsi che pilotano una valvola LEE (PWM). Si suppone che il supporto ECAP/PWM Turbina sia stato abilitato nel kernel tree e che nel kernel sia stato inizializzato il platform

Configurazione kernel:

Da menuconfig:

```
CONFIG_HAVE_PWM=y
CONFIG_GENERIC_PWM=y
CONFIG_DAVINCI_EHRPWM=y
CONFIG_ECAP_PWM=y
CONFIG_ECAP_PWM=y
CONFIG_ECAP_CAP=y
...
...
```

Il driver esporta la seguente interfaccia userspace

```
target $ ls -la /sys/class/pwm/ecap.2/kx_lee_valve_pwm/
-r--r--r-- 1 root  root  4096 Jan 15 05:45 busy           (<- staus: "free","busy")
-r--r--r-- 1 root  root  4096 Jan 15 05:45 debug         (<- debug prints)
-rw-rw-rw- 1 root  root  4096 Jan 15 05:49 gear          (<- ascii gear: "0","1",...)
-rw-rw-rw- 1 root  root  4096 Jan 15 05:45 gearbin       (<- bin gear: 0,1,..)
-rw-rw-rw- 1 root  root  4096 Jan 15 05:45 initialgear   (<- default gear: "0","1",..)
-rw-rw-rw- 1 root  root  4096 Jan 15 05:45 overlap        (<- pwm failure count: "0","1",..)
-r--r--r-- 1 root  root  4096 Jan 15 05:45 test          (<- pwm one-shot pulse test)
root@am180x-evm:/sys/class/pwm/ecap.2#
```

gear è la marcia e indica ogni quanti fronti positivi del segnale turbina viene generato un impulso one-shot del PWM.

gear può essere modificata runtime, gearbin è la versione numerica di gear, mentre initialgear imposta il valore di default che viene assunto al momento di impegnare la risorsa.

Il valore 0 in gear significa che “sto in folle”, nessuna marcia.

Verifica:

```
target $ cat /sys/class/pwm/ecap.2/kx_lee_valve_pwm/test (LEE3 valve pulses, i.e. 0-1-0)
one shot PWM (period 1500000, duty 750000)
target $ cat /sys/class/pwm/ecap.2/kx_lee_valve_pwm/debug
tsc 750638, period 1500000, duty 750000, ecctl1 0xc0aa, ecctl2 0x2c0, eceint 0x0, ecflg 0x0
```

Per far funzionare il driver occorre installare il driver *kxsensors.ko* e aprire */dev/kxsensors*.

Questo viene fatto dal driver turbina di Mimmo.

Al momento le marce utilizzate dal driver turbina sono 2,4,6.

54.11.1 PWM per KX

Quanto detto sopra vale per KX

54.11.2 PWM per Spartacus

Su XNRG, sulla linea V3PWMGAS è attivo il PWM0_OUT. Per provarlo eseguire i seguenti comandi

```
target $ cd class/pwm/pwmchip0/
target $ echo 0 > export
target $ cd pwm0
target $ echo 100000 > period
target $ echo 50000 > duty_cycle
target $ echo 1 > enable
```

(<- a 50 us waveform is enabled onto the line)

Nota: Per attivare uno stato logico fisso è sufficiente mettere duty_cycle pari a period e cambiare enable

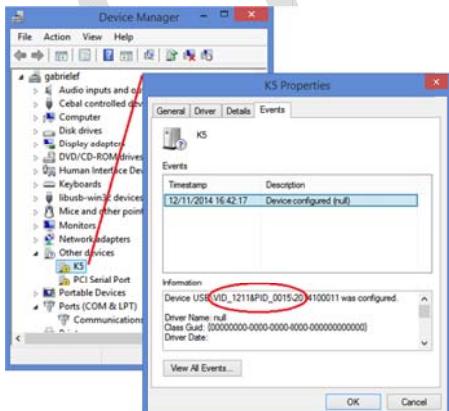
```
target $ echo 100000 > duty_cycle
target $ echo 0 > enable
target $ echo 1 > enable
```

(<- low level)
 (<- high level)

54.12 Utilizzare la connessione USB verso il PC

Il connettore esterno USB del KX rappresenta una interfaccia device verso il PC. Nel FW sarà implementato un driver compatibile con il driver USB host comune agli altri prodotti X (*cosmusb.inf*), basato su *gadget-zero*, con VID 1211 e PID 0015.

Questo è il modo standard di utilizzo della connessione.



Engineering Specification - confidential

Tuttavia per sviluppo e troubleshooting è possibile attivare altri tipi di interfaccia, attraverso il menu administrator settings del KX (da GUI Settings->System->Admin, password: 369).

USB Serial

In questa modalità il KX comunica con il PC attraverso un driver di tipo "Serial Gadget".

Lato KX viene eseguito

```
modprobe g_serial  
set_usb_serial           (<- vedere sorgente)
```

Il che significa che sul PC è necessario installare il driver *usbser.sys* (non firmato) utilizzando il file *linux-cdc-acm.inf*. Le applicazioni vedranno il device attraverso una COM virtuale. Vi è una limitazione del throughput in downstream che se violata determina il blocco della comunicazione. Quando si collega il KX a un PC windows, sul Pannello di Controllo la periferica viene riconosciuta. Ottenerne il driver *linux-cdc-acm.inf* e installarlo.

Collegare il cavo USB tra host PC e target

Aprire un terminale di emulazione seriale sulla COM port del PC host e comunicare col target

USB Ethernet (Windows)

In questa modalità il KX comunica con il PC (Windows) attraverso un driver di tipo "USB Ethernet/RNDIS Gadget", emulazione di rete ethernet.

Sul PC è necessario installare il driver utilizzando il file *linux-cdc-eth.inf*.

Lato KX viene eseguito

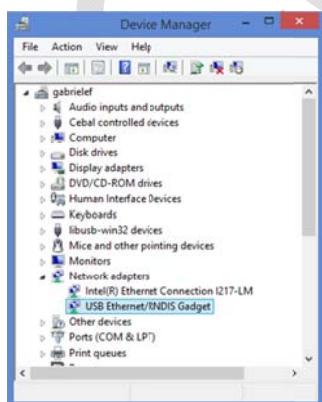
```
modprobe g_ether  
ifup usb0
```

e a runtime si ha

```
target $ ifconfig  
...  
usb0    Link encap:Ethernet HWaddr BE:14:DE:C9:B5:50  
        inet addr:10.0.0.1 Bcast:10.255.255.255 Mask:255.0.0.0  
        UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1  
        RX packets:328 errors:0 dropped:108 overruns:0 frame:0  
        TX packets:0 errors:0 dropped:0 overruns:0 carrier:0  
        collisions:0 txqueuelen:1000  
        RX bytes:33732 (32.9 Kib) TX bytes:0 (0.0 B)
```

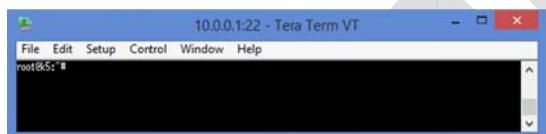
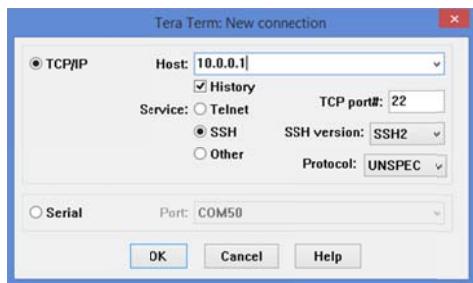
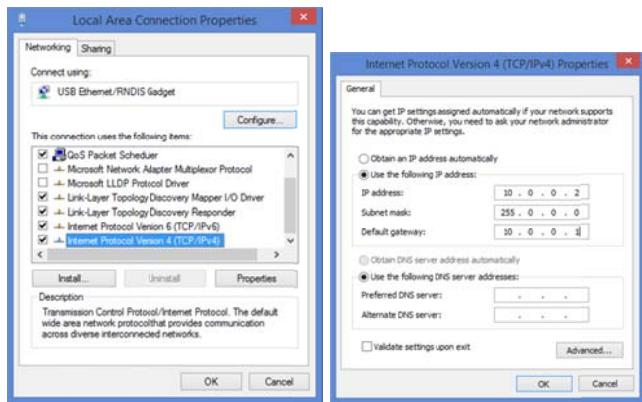
Quindi il KX ha indirizzo 10.0.0.1 (definito in */etc/network/interfaces*)

Lato PC si ha



Engineering Specification - confidential

Da Control Panel\All Control Panel Items\Network Connections



USB Ethernet + ACM (Linux)

Lato KX viene eseguito

```
modprobe g_cdc
ifconfig usb0 10.0.0.1 netmask 255.0.0.0 up
set_usb_serial
```

USB Mass Storage

Lato KX viene eseguito

```
umount -f /dev/mmcblk0p1
modprobe g_mass_storage file=/dev/mmcblk0p1 stall=0
```

54.13 Patch 0016-kx-kernel-add-usb-support

Questa patch introduce il supporto peripheral per USB

Git Branch Name:

usb-gadget

Patched Files:

```
<kernel>/arch/arm/configs/da850_kx_debugconfig  
<kernel>/arch/arm/configs/da850_kx_defconfig  
<kernel>/arch/arm/configs/da850_kx_ubidebugconfig  
<kernel>/arch/arm/mach-davinci/board-da850-kx.c  
<kernel>/drivers/base/class.c  
<kernel>/drivers/usb/gadget/Kconfig  
<kernel>/drivers/usb/gadget/Makefile  
<kernel>/drivers/usb/gadget/X.c  
<kernel>/drivers/usb/gadget/X_fops.c  
<kernel>/drivers/usb/gadget/X_sysfs.c  
<kernel>/drivers/usb/gadget/f_X.c  
<kernel>/drivers/usb/gadget/f_X.h  
<kernel>/drivers/usb/gadget/f_loopback.c  
<kernel>/drivers/usb/gadget/storage_common.c  
<kernel>/include/linux/device.h  
<appskx>/hostgw/hostgw.c  
<appskx>/hostgw/hostgw.h  
<appskx>/hostgw/msqlib.h  
/fs/rootfs-base/overwrite-tree/etc/init.d/X  
/host-drivers/*
```

Configurazione kernel:

Da menuconfig:

- 64) Abilitare DeviceDrivers->USB Support->Inventra Highspeed Dual Role Controller (TI, ADI, ...)
- 65) Abilitare DeviceDrivers->USB Support->DA8xx/OMAP-L1xx->USB Peripheral (gadged stack)
- 66) Abilitare DeviceDrivers->USB Support->USB Gadget Support-->USB Peripheral Controller->Inventra ..
- 67) Abilitare DeviceDrivers->USB Support->USB Gadget Support-->USB Gadget Drivers->Gadget Serial

Per caricare il modulo:

```
target $ modprobe g_X
```

Verifica:

```
target $ cat /sys/devices/platform/musb-da8xx/musb-hdrc/mode  
b_peripheral
```

kernel boot related messages:

```
Dec 9 18:22:59 kx user.info kernel: [ 18.902743] gadget: dual speed X: IN/ep1in, OUT/ep1out  
Dec 9 18:22:59 kx user.info kernel: [ 18.908465] X usb gadget device (250,0)  
Dec 9 18:22:59 kx user.info kernel: [ 18.915912] KX USB Interface, version: 1.0 (bcdDevice 0x126)  
Dec 9 18:22:59 kx user.info kernel: [ 18.921886] gadget: gX ready
```

Debugging:

Un modo più semplice in NFS è ricompilare hostgw in modalità debug

```
host $ cd <appskx>/hostgw  
host $ make clean  
host $ DEBUG=y make install
```

Per sapere se la connessione usb è attiva basta leggere un file

```
target $ cat /sys/class/gX/host_plugged
```

Il file ritorna 1/0 se l'USB client è connesso/disconnesso

54.14 Utilizzare l'utilità *dtest* sul target

I test eseguiti sul KX vengono memorizzati nella cartella <archive>/tests del target fs. I test sono numerati in ordine progressivo

```
target $ ls <archive>/tests/
NO-NAME-TEST.00000
d0261978-cbe4-432f-afc2-fdb5aa034d06.00001
                                         (<- test non associate a nessun soggetto)
                                         (<- test associate a un soggetto con UUID)
```

dtest è un eseguibile standalone e i sorgenti C si trovano in <dev>/apps/appskx/dtest/

I test possono essere visualizzati singolarmente attraverso l'utilità *dtest*

```
target $ cd /usr/share/X/kmain
target $ ./dtest
                                         (<- stampa le opzioni del programma)
Usage: dtest [-l]           prints tests list
          [-t] test_number   prints test content in TXT mode
          [-c] test_number   prints test content in CSV mode
          [-o] test_number   prints test content in CSV mode (OMNIA format)
          [-r] [--print]      repairs corrupted test files
          [-v]                prints version
```

Per vedere la lista dei test presenti

```
target $ ./dtest -l
# Type    Time     Date Duration Items First name Last name             UUID
2 BxB 17:20:58 2017/02/15 00:06:12 357 firstname lastname3 a7ce662f-614f-4023-92f3-7af84941927d
5 BxB 18:11:38 2017/02/15 03:59:45 12815 firstname lastname5 334ceea5-8061-48f2-8455-5504f1f884a8
3 Mix 17:35:15 2017/02/15 00:16:50 156 firstname lastname4 bed89e64-d659-46b9-81c7-5da5720d10c4
4 BxB 17:55:57 2017/02/15 00:11:12 630 firstname lastname4 bed89e64-d659-46b9-81c7-5da5720d10c4
6 BxB 09:43:50 2017/02/16 00:00:00 0 PC_TEST_6 PC_TEST_6 5e4695dd-85df-44b4-bccb-7b875c78d779
1 Mix 17:16:39 2017/02/15 00:03:50 18 firstname lastname2 332773d8-555a-426a-9338-b76a743ef89f
```

Si noti che c'è un test con Items 0 e Duration 0. Significa che è corrotto, oppure è ancora in corso (questi campi vengono scritti solamente a conclusione del test).

Per sanare un test corrotto si può lanciare il comando seguente

```
target $ ./dtest -r --print
```

Per vedere il contenuto di un test

```
target $ ./dtest -t 1
# Type    Time     Date Duration Items First name Last name             UUID
1 Mix 17:16:39 2017/02/15 00:03:50 18 firstname lastname2 332773d8-555a-426a-9338-b76a743ef89f
filename: /mnt/archive/tests/332773d8-555a-426a-9338-b76a743ef89f.00001
name: /mnt/archive/tests/332773d8-555a-426a-9338-b76a743ef89f.00001
UUID 120:
TM version:            3
Test type:              Mix
Test no.:               1
Duration:               00:03:50
Items:                  18
BTPS:                   11006
BTPSE:                  10200
STPD:                   8255
HRMax:                  0
uuid: 332773d8-555a-426a-9338-b76a743ef89f
GUID: 00000000-0000-0000-0000-000000000000
Date: 2017/02/15
Time: 17:16:39
Firstname: firstname
Lastname: lastname2
SSNID:
Birth date: 1999/03/31
Gender: Male
Ethnicity: Caucasian
Height: 150.0
Weight: 50.0
Notes:
TimeOnCal: 0
TaTest: 237
PbTest: 759
```

Engineering Specification - confidential

La prima riga riporta il nome delle grandezze (items)

La seconda riga riporta il numero di cifre decimali dopo la virgola per ciascun item

Le righe seguenti identificano i singoli atti del respiro

“BxB” identifica il tipo di test

23 è il numero di item che compongono il respiro

La modalità CVS consente una facile importazione in Excel

Per spostare i test dal target all'host tramite la rete è sufficiente eseguire sull'host il seguente comando:

```
host $ scp root@<ipaddr>:/mnt/archive/tests/* <host folder path>
```

E' anche possibile popolare l'archivio di un KX da host

```
host $ scp <host_folder_path>/`* root@<ipaddr>:/mnt/archive/tests/
```

Nella GUI del KX esiste la possibilità di esportare i test nella SD Card

54.15 Utilizzare l'utilità *dcals* sul target

Le calibrazioni eseguite sul KX vengono memorizzate nella cartella `/usr/share/X/kmain/settings/gcal` del target fs.

Il numero di calibrazioni e la data di esecuzione possono essere visualizzati attraverso l'utilità dcals

```
target $ cd /usr/share/X/kmain  
target $ ./dcals -l  
dcals version: 1.0 (CC 2015.0)
```

Engineering Specification - confidential

```
TOTAL COUNT: 19
COUNT TYPE DATE TIME RESULT
0 TURBINE 2015/10/15 06:10:59 PASS
1 TURBINE 2015/10/15 06:11:00 PASS
2 TURBINE 2015/10/15 06:11:01 PASS
3 TURBINE 2015/10/15 06:11:02 PASS
4 TURBINE 2015/10/15 06:12:25 PASS
5 TURBINE 2015/10/15 06:12:26 PASS
6 TURBINE 2015/10/15 06:12:27 PASS
7 TURBINE 2015/10/15 06:12:28 PASS
8 TURBINE 2015/10/15 06:19:28 PASS
9 TURBINE 2015/10/15 06:22:59 PASS
10 TURBINE 2015/10/15 06:23:01 PASS
11 TURBINE 2015/10/15 06:23:28 PASS
12 TURBINE 2015/10/15 06:23:29 PASS
13 TURBINE 2015/10/15 06:23:30 PASS
14 TURBINE 2015/10/15 06:23:31 PASS
15 TURBINE 2015/10/15 06:23:32 PASS
16 TURBINE 2015/10/15 06:24:12 PASS
17 TURBINE 2015/10/15 06:40:12 PASS
18 TURBINE 2015/10/15 06:40:12 PASS
```

target \$./dcals -s

dcals version: 1.0 (GC-2015.09)

COUNT	PASSED	FAILED
19	19	0

54.16 Utilizzare l'utilità *dmap* sul target

I risultati di calibrazione e i vari descrittori utilizzati nei test del KX vengono memorizzati in file .map.
Per leggerli

```
target $ cd /usr/share/X/kmain
target $ ./dmap
dmap v1.0 (GC-07.2014)
PARCODE: count=045 len=0105 key=55AAAA56 type=0000 alloc_ptr=04CA

### - LEN : ID : VAL
000 - 0026 : 0000 : 0000 0000 0000 0435 0002 0000 0000 0000 598C 0002 0000 03E8 03E8 F3E0 F3E0 03E8 0435 1000 1000
001 - 0026 : 00C0 : 0000 0003 0000 2710 0002 0000 0001 0003 598C 0002 0000 03E8 03E8 0000 0000 03E8 2710 1000 1000
002 - 0026 : 0400 : 0000 0010 0000 2710 0002 0000 0000 0010 598C 0002 0000 03E8 03E8 0000 0000 03E8 2710 1000 1000
003 - 0026 : 0340 : 0000 0000 0000 1377 0002 0001 0000 000D 057A 0001 032A 03DF 03DF 0064 0000 03E8 1377 082B 0640 (<- BxB: O2 Speed, Delay, Gain, Baseline)
004 - 0026 : 0300 : 0000 000C 0000 2710 0002 0001 0001 000C 1770 0001 029E 03D5 03E8 0ED8 0ED8 0ED8 0ED8 2710 0006 01F4 (<- BxB: CO2 Delay)
005 - 0026 : 0200 : 0000 0008 0000 2710 0002 0000 0001 0008 1770 0002 0000 03E8 03E8 0000 0000 03E8 2710 1000 1000
006 - 0026 : 0780 : 0000 001E 0000 2710 0002 0001 0001 001E 1770 0001 0000 03E8 03E8 0000 0000 03E8 2710 1000 1000
007 - 0026 : 07C0 : 0000 001F 0000 2710 0002 0001 0001 001F 1770 0001 0000 03E8 03E8 0000 0000 03E8 2710 1000 1000
008 - 0026 : 0500 : 0000 0014 0000 2710 0002 0000 0001 0014 1770 0002 0000 03E8 03E8 0000 0000 03E8 2710 1000 1000
009 - 0026 : 0680 : 0000 001A 0000 2710 0002 0000 0001 001A 1770 0002 0000 03E8 03E8 0000 0000 03E8 2710 1000 1000
00A - 0026 : 0600 : 0000 0018 0000 2710 0002 0000 0001 0018 1770 0002 0000 03E8 03E8 0000 0000 03E8 2710 1000 1000
00B - 0026 : 06C0 : 0000 001B 0000 2710 0002 0000 0001 001B 1770 0002 0000 03E8 03E8 0000 0000 03E8 2710 1000 1000
00C - 0026 : 0380 : 0000 000E 0000 2710 0002 0000 0001 000E 1770 0002 0000 03E8 03E8 0000 0000 03E8 2710 1000 1000
00D - 0026 : 03C0 : 0000 000F 0000 1040 0002 0000 0000 000F 1770 0002 0000 03E8 03E8 0090 0090 03E8 1040 1000 1000
00E - 0026 : 0100 : 0000 0004 0000 2710 0002 0000 0000 0004 1770 0002 0000 03E8 03E8 01F4 01F4 03E8 2710 1000 1000
00F - 0026 : 0140 : 0000 0005 0000 2710 0002 0000 0000 0005 1770 0002 0000 03E8 03E8 0000 0000 03E8 2710 1000 1000
010 - 0026 : 0180 : 0000 0006 0000 2710 0002 0000 0000 0006 1770 0002 0000 03E8 03E8 0000 0000 03E8 2710 1000 1000
011 - 0026 : 01C0 : 0000 0007 0000 2710 0002 0000 0001 0007 1770 0002 0000 03E8 03E8 0000 0000 03E8 2710 1000 1000
012 - 0026 : 0040 : 0000 0001 0001 E4CD 0002 0000 0000 0001 1770 0002 0000 044D 03E8 0320 0320 044B E4CD 1000 1000
013 - 0026 : 02C0 : 0000 0008 0000 2710 0002 0000 0001 000B 1770 0002 0000 03E8 03E8 0000 0000 03E8 2710 1000 1000
014 - 0026 : 0080 : 0000 0002 0000 2710 0002 0000 0001 0002 1770 0002 0000 03E8 03E8 0000 0000 03E8 2710 1000 1000
015 - 0026 : 0240 : 0000 0009 0000 2710 0002 0000 0001 0009 1770 0002 0000 03E8 03E8 0000 0000 03E8 2710 1000 1000
016 - 0026 : 05C0 : 0000 0017 0000 2710 0002 0000 0001 0017 1770 0002 0000 03E8 03E8 0000 0000 03E8 2710 1000 1000
017 - 0026 : 04C0 : 0000 0013 0000 2710 0001 0001 0001 0013 1770 0001 0000 03E8 03E8 0000 0000 03E8 2710 1000 1000
018 - 0026 : 0440 : 0000 0011 0000 1377 0002 0001 0000 000D 04E2 0001 032A 03F8 0468 0049 0000 03E8 1377 082B 0640 (<- DMC: O2 Delay)
019 - 0026 : 0480 : 0000 0012 0000 2710 0002 0001 0001 000C 1770 0001 029E 03B5 03E8 0EBD 0ED8 03E8 2710 0006 01F4 (<- DMC: CO2 Delay)
01A - 0026 : 0280 : 0000 000A 0000 2710 0002 0000 0000 000A 1770 0002 0000 03E8 03E8 0000 0000 03E8 2710 1000 1000
01B - 0002 : 0801 : 01F4
01C - 0002 : 080D : 001C
01D - 0002 : 080E : 004E
01E - 0002 : 080F : 03E9
01F - 0002 : 0810 : 03FA
020 - 0002 : 0818 : 0000
021 - 0002 : 0819 : 0BB8
022 - 0002 : 081D : 0000
023 - 0002 : 081F : 0000
024 - 0002 : 0820 : 0000
025 - 0002 : 0828 : 0000
026 - 0002 : 0829 : 02D4 (BxB: PCal)
027 - 0002 : 082A : 02E5 (DMC: PCal)
028 - 0002 : 082B : 00E3
```

```
029 - 0002 : 082C : 02E6
02A - 0002 : 082D : 099B
02B - 0002 : 082E : 01F4
02C - 0002 : 0833 : 0064
02D - 0002 : 0834 : 0001
02E - 0002 : 0836 : 0154
02F - 0002 : 0837 : 024E (BxB: O2 Trimmer)
030 - 0002 : 0838 : 022D (DMC: O2 Trimmer)
031 - 0002 : 0839 : 01DE
032 - 0002 : 083A : 01D7
033 - 0002 : 083B : 0000
034 - 0002 : 083C : 0400
035 - 0002 : 0842 : 0000
036 - 0002 : 0843 : 00C8
037 - 0002 : 0844 : 00C8
038 - 0002 : 085E : 082D
039 - 0002 : 085F : 0003
03A - 0002 : 0868 : 0000
03B - 0002 : 087F : 0000
03C - 0002 : 2002 : 2B1D
03D - 0002 : 200F : 27DB
03E - 0002 : 2013 : 1F74
03F - 0002 : 6000 : 0154
040 - 0020 : 2003 : 0100 0302 0804 0E0B 0C0D 100F 1413 1A17 1E1B 201F 0A1A 1E1B 201F 0000 0000 0000
041 - 000C : 2011 : 3032 3431 3031 3030 3330 0000
042 - 0030 : 087B : BA23 780C 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000
043 - 001C : 200D : 0000 00FC 03E8 03E8 0001 3130 302F 2F31 3430 3100 3A32 3030 303A 0030
044 - 0006 : 2009 : 03E8 07D0 0BB8
```

Le stringhe più lunghe sono i descrittori

54.17 Aggiungere un parametro nello step

Supponiamo di voler aggiungere un parametro allo step BxB

I files che devono essere modificati sono

TODO

54.18 Generare facilmente una password sicura su host

Per generare la password lunga e complicata da una breve e facile da ricordare:

```
host $ echo XKX | md5sum (<- genera un checksum)
```

Quindi prendere i primi n caratteri.

54.19 Installazione driver Intel su host

Avendo cambiato il PC host, alcune periferiche non hanno un driver aggiornato nella distribuzione.
E' quindi necessario aggiornare i driver

```
$ sudo gedit /etc/apt/sources.list
Aggiungere:
deb http://ppa.launchpad.net/xorg-edgers/ppa/ubuntu lucid main #xorg-edgers PPA
deb-src http://ppa.launchpad.net/xorg-edgers/ppa/ubuntu lucid main #xorg-edgers PPA
$ sudo apt-key adv --recv-keys --keyserver keyserver.ubuntu.com 8844C542
$ sudo apt-get update
$ sudo apt-get dist-upgrade
$ da sinaptic installare: linux-image-2.6.35-32-generic e linux-headers-2.6.35-32-generic
```

54.20 Scrivere un log message da shell o da applicazione

Per creare una linea nel system log file si può utilizzare il comando logger da shell.

Esempio:

```
target $ logger System rebooted for hard disk upgrade
```

Il messaggio è presente nel file /var/log/message, infatti

```
target $ tail -f /var/log/message
Jan 26 20:53:31 dell6400 logger: System rebooted for hard disk upgrade
```

Il comando logger si può usare negli script.

Esempio:

```
#!/bin/bash
HDBS="db1 db2 db3 db4"
BAK="/sout/email"
[ ! -d $BAK ] && mkdir -p $BAK ||
/bin/rm $BAK/*
NOW=$(date +"%d-%m-%Y")
ATTCH="/sout/backup.$NOW.tgz"
[ -f $ATTCH ] && /bin/rm $ATTCH ||
MTO="you @yourdomain.com"
for db in $HDBS
do
FILE="$BAK/$db.$NOW-$(date +"%T").gz"
mysqldump -u admin -p'password' $db | gzip -9 > $FILE
done
tar -jcvf $ATTCH $BAK
mutt -s "DB $NOW" -a $ATTCH $MTO <<EOF
$DBS $(date)
EOF
[ "$?" != "0" ] && logger "$0 - MySQL Backup failed" || :
```

L'ultima linea esegue il log del message in /var/log/message se il backup fallisce

Il sistema del logging può anche essere implementato nel codice applicativo per debug.

Esempio:

Il programma demone *hostgw.cpp*, quando compilato con l'opzione *DEBUG=y*

```
host $ cd kx-dev/app/appskx/hostgw
host $ make clean
host $ DEBUG=y make
host $ sudo make install
```

in corrispondenza di certi eventi di errore o debug stampa dei messaggi del tipo

```
fprintf(stderr, "...", ...);
```

Se lo standard error del processo viene ridirezionato su un file di logging */tmp/log*

```
(sleep 7 ; /usr/share/X/bin/hostgw& ) >& /tmp/log
```

dal terminale si potrà usare il seguente comando per vedere in tempo reale il logging dei messaggi

```
target $ tail -f /tmp/log
DEBUG[iwrap_read_from_server:330]: Read 157 bytes from iwrap manager
DEBUG[iwrap_response_handler:240]: Response handler called
Subscribed to iwrap device OMNIA1
...
```

54.21 Utilizzare la shared memory

To do

```
#include "tcshared.h"
#include "custom-io.h"
#include <poll.h>

#define BLE_SH "/usr/share/X/bin/blegw"
#define BLE_SEM "/usr/share/X/bin/kxctrl-test"

typedef int ble_ht;
#define BLE_SH_LEN sizeof(ble_ht)
static TCSHARED *tcs;
static ble_ht *htm_p;
```

```
void atexit_closeshared(void)
{
    tcshared_close(tcs);
}

void tcshared_set_htm(TC_SHARED* p, ble_ht *a, int htm){
    static int last_htm;
    if(htm==last_htm)
        return;
    tcshared_lock(p);
    *a = htm;
    tcshared_unlock(p);
    last_htm = htm;
}

...
tcshared_set_htm(tcs, htm_p, (int)(float)(value*10.0));
...
```

Per elencare le shared memory presenti nel Sistema

```
target $ ipcs
```

```
----- Message Queues -----
key      msqid   owner   perms  used-bytes messages
----- Shared Memory Segments -----
key      shmid   owner   perms   bytes  nattch  status
0x610c2646 0       root    666     24      3
0x610c2649 32769   root    666      4      1
0x00000000 65538   root    600    1024      1      dest
0x610c00d9 98307   root    666     176      2
0x610c00e7 131076   root    666      32      2
----- Semaphore Arrays -----
key      semid   owner   perms  nsems
0x620c2651 0       root    666      1
0x620c2644 32769   root    666      1
0x641005c8 65538   root    600      1
0x620c2639 98307   root    666      1
0x620c00da 131076   root    666      1
0x620c009e 163845   root    666      1
```

Per rimuovere le shared memory presenti nel sistema

```
target $ ipcrm -a
```

54.22 Compilazione host distribuita

E' possibile compilare con gcc utilizzando più PC in rete
Todo

54.23 Utilizzare gnuplot

Gnuplot è un è una utility grafica multiplattaforma, concepito per scienziati e studenti per visualizzare iterativamente funzioni matematiche 2D/3D, web scripting, etc. E' possibile interagire con il mouse e supporta molti formati di output (pdf, jpeg, png, LaTex, etc.)

Installare e lanciare

```
host $ sudo apt-get update
host $ sudo apt-get install gnuplot-x11
host $ gnuplot
GNUPLOT
```

Engineering Specification - confidential

Version 4.6 patchlevel 6 last modified September 2014
Build System: Linux armv7l

Copyright (C) 1986-1993, 1998, 2004, 2007-2014
Thomas Williams, Colin Kelley and many others

gnuplot home: <http://www.gnuplot.info>
faq, bugs, etc: type "help FAQ"
immediate help: type "help" (plot window: hit 'h')
gnuplot >

(<- prompt)

Esempio:

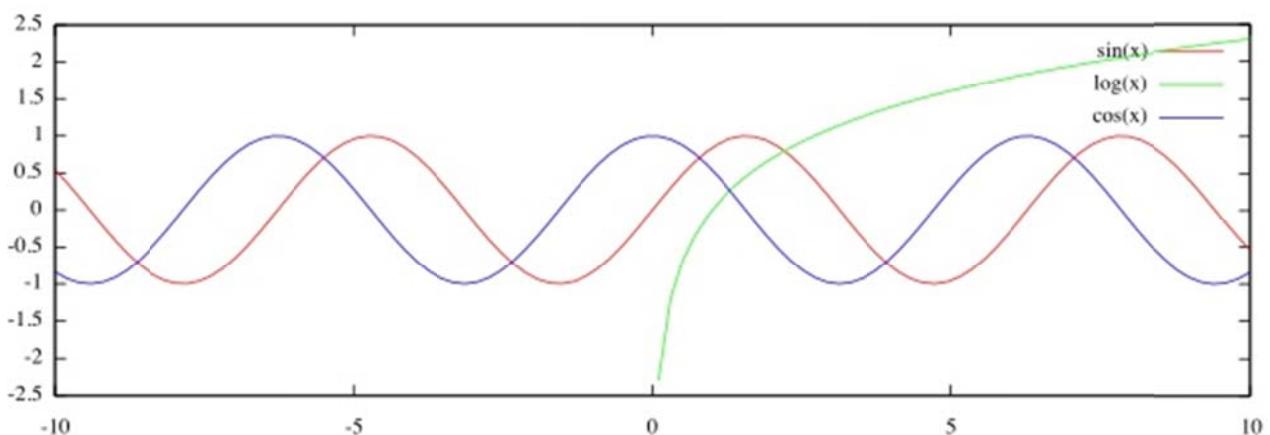
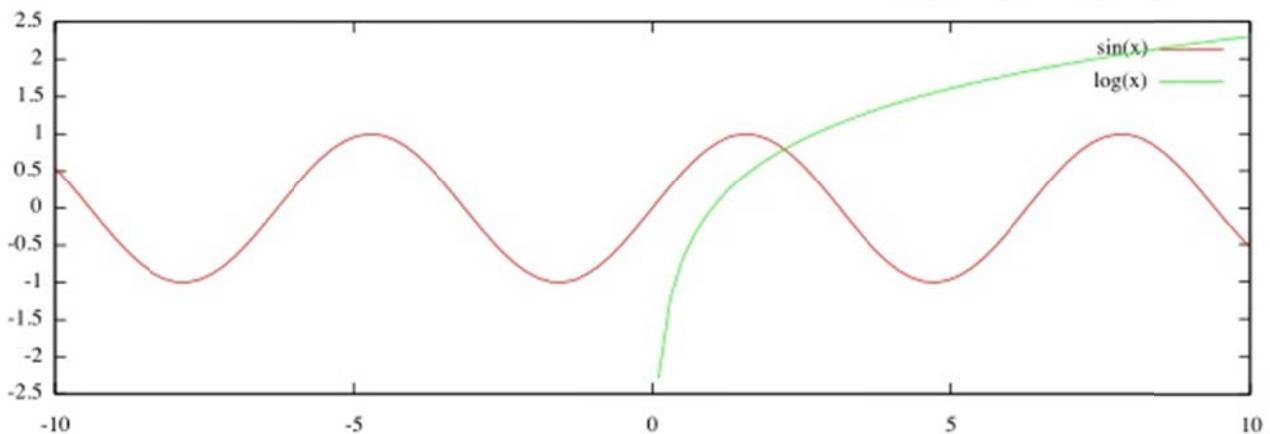
Multiple graphs

```
set multiplot      # multiplot mode (prompt changes to 'multiplot')
set size 1, 0.5

set origin 0.0,0.5
plot sin(x), log(x)

set origin 0.0,0.0
plot sin(x), log(x), cos(x)

unset multiplot    # exit multiplot mode (prompt changes back to 'gnuplot')
```



54.24 Aggiungere supporto runtime a 32-bit su host Ubuntu a 64-bit

Per aggiungere il supporto runtime a 32-bit a un host PC con installato 64-bit Ubuntu

```
host $ sudo apt-get update
host $ sudo dpkg --add-architecture i386
host $ sudo apt-get install libc6:i386 libncurses5:i386 libstdc:i386
```

```
host $ sudo apt-get install zlib1g:i386
```

54.25 Compilazione CPP di una libreria C

Per fare in modo che una libreria C mylib.so possa essere utilizzata da un programma in C++ bisogna inserire la seguente direttiva nel file mylib.h

```
#ifdef __cplusplus
extern "C" {
#endif

...
#ifndef __cplusplus
}
#endif
```

Per forzare un programma eseguibile *myprog*, scritto per utilizzare una certa libreria *mylib.so*, a utilizzare una versione modificata della libreria *newlib* e presente nella stessa directory del programma

```
target $ LD_PRELOAD=./newlib.so ./myprog
```

54.26 Attivare iAP protocol su Bluetooth

Una volta caricato il FW giusto, ricordarsi di eliminare da filesystem il file /etc/iwrap.init. Questo faà in modo che al successivo boot il modulo WT41 venga inizializzato di nuovo e il profilo iAP sarà operativo

55 XNRG

Per conoscere la versione di U-Boot

KX:

```
target # version
U-Boot 2013.10 (Mar 16 2015 - 17:32:55)
arm-linux-gnueabihf-gcc (crosstool-NG linaro-1.13.1-4.7-2013.03-20130313 - Linaro GCC 2013.03) 4.7.3 20130226 (prerelease)
GNU ld (crosstool-NG linaro-1.13.1-4.7-2013.03-20130313 - Linaro GCC 2013.03) 2.23.1
```

VAR-SOM-AM33:

```
target # version
U-Boot 2013.10 (Mar 16 2015 - 17:32:55)
arm-linux-gnueabihf-gcc (crosstool-NG linaro-1.13.1-4.7-2013.03-20130313 - Linaro GCC 2013.03) 4.7.3 20130226 (prerelease)
GNU ld (crosstool-NG linaro-1.13.1-4.7-2013.03-20130313 - Linaro GCC 2013.03) 2.23.1
```

BBB C3:

```
target # version
U-Boot 2015.01-00001-gb2412df (Jan 29 2015 - 15:01:06)
arm-linux-gnueabihf-gcc (Linaro GCC 2014.11) 4.9.3 20141031 (prerelease)
GNU ld (GNU Binutils) Linaro 2014.11-3-git 2.24.0.20141017
```

Per conoscere la versione di linux e della cpu

KX:

```
target $ cat /proc/version
Linux version 3.3.0-kx-006 (gabriele@fub1204) (gcc version 4.5.3 20110311 (prerelease) (GCC) ) #1 PREEMPT Mon Jun 22 09:11:01
CEST 2015
```

```

target $ cat /proc/cpuinfo
Processor      : ARM926EJ-S rev 5 (v5l)
BogoMIPS       : 227.32
Features       : swp half thumb fastmult edsp java
CPU implementer : 0x41
CPU architecture: 5TEJ
CPU variant    : 0x0
CPU part       : 0x926
CPU revision   : 5

Hardware      : X KX
Revision       : 12110003
Serial         : 00000000780cba23

VAR-SOM-AM33:
target $ cat /proc/version
Linux version 3.14.26 (am33@bobo) (gcc version 4.7.3 20130226 (prerelease) (crosstool-NG linaro-1.13.1-4.7-2013.03-20130313 - Linaro
GCC 2013.03) ) #1 Mon Mar 16 16:58:15 IST 2015

target $ cat /proc/cpuinfo
processor      : 0
model name     : ARMv7 Processor rev 2 (v7l)
Features       : swp half thumb fastmult vfp edsp thumbee neon vfpv3 tls vfpd32
CPU implementer : 0x41
CPU architecture: 7
CPU variant    : 0x3
CPU part       : 0xc08
CPU revision   : 2

Hardware      : Generic AM33XX (Flattened Device Tree)
Revision       : 0000
Serial         : 0000000000000000

BBB C3:
target $ cat /proc/version

```

55.1 Creare git repo locale e remota Spartacus

Creare un branch “spartacus” (sia locale che remoto)

host \$ git checkout -b spartacus	(creo il branch locale “spartacus”)
host \$ git push -u origin spartacus	(creo anche il branch remoto “spartacus”, sempre utile come backup del locale)
host \$ git pull	(aggiorno il branch gabriele locale con il spartacus remoto)

Su ogni altra stazione importare il branch “spartacus”

host \$ git pull
host \$ git checkout --track origin/spartacus

A questo punto non dovrò più fare dei merge completi dal master (con i comandi *merge* e *rebase*), ma quando serve fare dei micro-merge dei singoli commit, specificandone l’hash.

Dovendo applicare su *spartacus* un singolo commit <hash> che sta in *master*

host \$ git checkout spartacus	
host \$ git cherry-pick <hash>	(a questo punto il commit è pronto per essere pushato)
host \$ git push	

Nota: Per trovare l’hash del commit si può sfruttare *gitk* magari filtrandolo su una determinate cartella <dirpath> (oppure su un determinato file <filepath>) con

host \$ gitk --all [<dirpath> <filepath>]

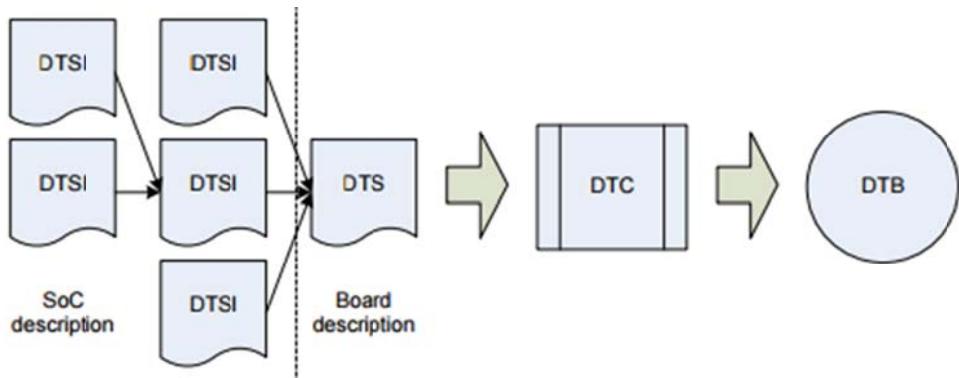
a questo punto si seleziona il commit desiderato e si copia il relativo <hash>

Se invece devo applicare molti commit consecutivi a *spartacus*, supponiamo da <hash1> a <hash2>

host \$ git checkout master	(rientro nel branch master)
host \$ git checkout -b tmp <hash2>	(entro in un nuovo branch temporaneo contenente tutti i commit fino a <hash2>)
host \$ git rebase --onto spartacus <hash1>^	(<- verificare!!!)
host \$ git checkout spartacus && git merge --ff-only tmp	(<- verificare!!!)

55.2 DTS

Il progetto XNRG è basato su DTS, mentre il progetto KX è basato su Board Support Package. Il DTS (Device Tree Source) è un sistema che descrive l'hardware della scheda. Esso include files analoghi che descrivono il SoC e alla fine genera un file binario .dtb che viene utilizzato dal kernel al boot per configurare l'hardware dinamicamente.



Ad esempio, per configurare nel kernel del SOM VAR-SOM-AM33 la porta USB-OTG come host-only:

- 1) Editare il file var-som-am33.dts nei sorgenti del kernel

```

host $
host $ cd <linux>
vim arch/arm/boot/dts/var-som-am33.dts
...
usb@47401800 {
    status = "okay";
    dr_mode = "host"; /* <<- Add this line */
}
...

```

- 2) Ricompilare var-som-am33.dtb e memorizzarlo nella flash (o su NFS)
- 3) Reboot

Per la BeagleBone Black il file DT non si trova nel kernel mainline, ma deve essere cercato nel repository <git://github.com/beagleboard/linux.git>
Tale DT è *arch/arm/boot/dts/am335x-boneblack.dts*

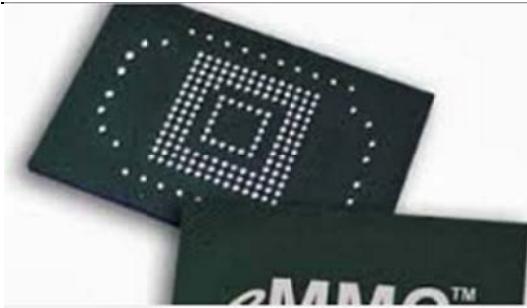
Per maggiori informazioni vedere http://devicetree.org/Device_Tree_Usage

Per il prodotto XNRG il file .dts relativo è

<linux>/arch/arm/boot/dts/am335x-xnrg.dts

55.3 eMMC (BBB)

La flash della BBB è una eMMC da 4GB. Essa è identificata come MMC device 1.



A differenza della NAND, organizzata a apagine, la eMMC è organizzata a settori (o blocchi) di 512B. Il minimo blocco cancellabile è di 512KB.

La eMMC è internamente è strutturata come segue

Boot 1	4MiB
Boot 2	4MiB
RPMB	4MiB (zona lagata all'encryption dei dati)
User	3.6GiB (area di utente, dove si trovano le partizioni)

```
Target# mmcinfo
Device: OMAP SD/MMC
Manufacturer ID: 70
OEM: 100
Name: S1000
Tran Speed: 52000000
Rd Block Len: 512
MMC version 4.5
High Capacity: Yes
Capacity: 3.6 GiB
Bus Width: 4-bit
Erase Group Size: 512 KiB
HC WP Group Size: 4 MiB
User Capacity: 3.6 GiB
Boot Capacity: 4 MiB ENH
RPMB Capacity: 4 MiB ENH
```

Su XNRG abbiamo definito le seguenti partizioni per la eMMC, tutte allineate all'erase block size (512KB)

Address	Type	Size	Part Nr.	Description	Filename
0x00000000 - 0x0001FFFF	raw	0x00020000	p1: MLO	Bootloader SPL	MLO
0x00020000 - 0x0003FFFF	raw	0x00020000	p2: MLO.b0	Bootloader SPL (bkup 0)	MLO
0x00040000 - 0x0005FFFF	raw	0x00020000	p3: MLO.b1	Bootloader SPL (bkup 1)	MLO
0x00060000 - 0x0007FFFF	raw	0x00020000	p4: MLO.b2	Bootloader SPL (bkup 2)	MLO
0x00080000 - 0x004BFFFF	raw	0x00440000	p5: uboot	Bootloader	u-boot.img
0x004C0000 - 0x004DFFFF	raw	0x00020000	p6: uboot.env	Bootloader Env	
0x004E0000 - 0x004FFFFFF	raw	0x00020000	p7: XNRG.env	XNRG Env	
0x00500000 - 0x01B4FFFF	raw	0x01B00000	p8: Kernel	Kernel	uImage
0x01B50000 – 0x0FB4FFFF	ext4 (0x83)	0x0E000000	p9: rootfs	Filesystem	
0x10000000 - 0x1043FFFF	raw	0x00440000	p10: U-boot.b	Bootloader (secondary)	u-boot.img
	raw	0x00020000	p11: U-Boot.env.b	Bootloader Env (Secondary)	
	raw	0x00020000	p12: XNRG.env.b	XNRG Env (secondary)	
	raw	0x01B00000	p13: Kernel.b	Kernel (secondary)	uImage
	ext4 (0x83)	0x0E000000	p14: rootfs.b	Filesystem (secondary)	
	ext4 (0x83)	0xC0530000	p15: archive	Archivio	

OFFSET (KB)	OFFSET (KB hex)	Size (KB)	Partition Name
0	0x00000000	128	MLO
128	0x00020000	128	MLO.b0
256	0x00040000	128	MLO.b1
384	0x00060000	128	MLO.b2
512	0x00080000	4608	uboot
5120 (5M)	0x00500000	256+256	uboot-env + redund
5632	0x00580000	4608	uboot.b
10240 (10M)	0x00A00000	256+256	uboot-env.b + redund
10752	0x00A80000	256+256	XNRG-env + redund
11264 (11M)	0x00B00000	10M	kernel
21504 (21M)	0x01500000	512	dtb
22016	0x01580000	10M	kernel.b
32256	0x01F80000	512	dtb.b
32768 (32M)	0x02000000	512M	rootfs (ext4)
557056 (544M)	0x22000000	3G+	Archive (ext4)

55.3.1 Formato ext4 del rootf

I filesystem di linux possono essere di diversi tipi. Il KX utilizza UBIFS che va bene sulle flash. Al momento dell'aggiornamento FW è necessario copiare il rootfs nella partizione della memoria destinata ad esso. Usando UBI l'immagine può avere dimensione minore della capacità della partizione, e il processo di copia sarà limitato a questa dimensione, infatti ci pensa il kernel a inizializzare a zero la parte restante della partizione.

Sulla eMMC UBI non funziona, per cui si può usare ext2 oppure ext4.

Utilizzando ext4, purtroppo, la dimensione dell'immagine del rootfs deve essere necessariamente uguale alla dimensione della partizione destinata a ospitare il rootfs. Questo perché i dati sulla partizione non occupano una zona contigua, ma sono sparsi.

La conseguenza è che l'aggiornamento FW del XNRG dura sempre lo stesso tempo, il tempo che ci vuole per scrivere l'intera partizione.

Esiste anche un'altra partizione, sia in KX che in XNRG, che è quella dedicata all'archivio.

Esiste ancora un'altra partizione, sia INFRA che in XWRC, che è quella dedicata all'archivio. Fortunatamente, quando si aggiorna il FW, l'archivio deve essere integralmente mantenuto oppure cancellato. La cancellazione di una intera partizione non è un grosso problema, in quanto il kernel, proveniente da un aggiornamento FW, può essere istruito a formattare la partizione, operazione relativamente veloce.

55.4 Inizializzazione eMMC di una BBB nuova connessa in rete

Sul target (connesso in rete) la prima cosa da fare è aggiornare MLO e U-Boot con relativo environment

Engineering Specification - confidential

```
1.0 MB/s
done
Bytes transferred = 5767168 (580000 hex)
=> mmc dev 1
switch to partitions #0, OK
mmc1(part 0) is current device
=> mmc write ${loadaddr} 0 2C00

MMC write: dev # 1, block # 0, count 11264 ... 11264 blocks written: OK
=>
target# reset
resetting ...
```

Nota: notare che il comando di scrittura copre da 0 a 2C00h in multipli di 512 bytes, cioè da 0 a 0x00580000, che è esattamente lo spazio occupato dalle copie multiple di MLO, U-Boot e relativo environment.

55.5 Aggiornamento FW da rete

Dopo avere effettuato l'inizializzazione come da paragrafo precedente, il secondo passaggio è l'aggiornamento completo della eMMC, sempre da rete.

Sull'host bisogna assicurarsi di avere generato l'immagine del FW

```
host $ cd <dev>/fs/rootfs/  
host $ sudo ./gen-fs.sh  
host $ sudo ./gen-img.sh
```

Sul target i comandi sono

```
target# setenv serverip 192.168.0.142
target# setenv ipaddr 192.168.0.149
target# run upd_mmc
....                                         (← richiede circa 7 minuti)
MMC write: dev # 1, block # 851968, count 262144 (filesize 134217728)... 262144 blocks written: OK
=>
target# reset
resetting ...
```

55.6 Aggiornamento FW da USB

L'aggiornamento da USB prevede di collegare al XNRG un USB Mass Storage device formattato FAT o FAT32, con all'interno il file di aggiornamento **X-xnrg-fw.bin**

Per creare il file di aggiornamento, collegare la USB Storage Device a host, quindi

```
host $ cd <dev>/fw-updates  
host $ ./deploy-on-usb.sh <usb_path>           (<- aggiorna tutto, e azzera anche l'archivio)  
dove <usb_path> è la cartella dove linux ha montato la memoria
```

Nel caso in cui si voglia anche formattare la memoria USB si possono usare le seguenti opzioni alternative:

```
host $ cd <dev>/fw-updates  
host $ ./deploy-on-usb.sh <sd dev> 1
```

Engineering Specification - confidential

dove <sd_dev> è **/dev/sdb1** (p.es.) se la memoria USB è già formattata e si vuole solo copiare i file all'interno, oppure è semplicemente **/dev/sdb** se la si vuole anche formattare (FAT32).

Se la eMMC del sistema ha una versione di u-boot e di MLO valide allora, una volta collegata la pennetta al XNRG, è sufficiente alimentare il sistema.

Se invece la eMMC del sistema non ha una versione valida di u-boot e di MLO, come nel caso di una BBB appena acquistata, si può inserire una micro SD nello slot della BBB, anch'essa formattata FAT, contenente le versioni valide di u-boot e MLO

u-boot.img

MLO

Per creare il file di aggiornamento, inserire la SD card sul host, quindi

```
host $ cd <dev>/fw-updates  
host $ ./make-init-sd-card <sd_dev>
```

dove <sd_dev> è **/dev/sdb1** (p.es.) se la card è già formattata e si vuole solo copiare i file all'interno, oppure è semplicemente **/dev/sdb** se la si vuole anche formattare.

Sulla Digital Board del XNRG è (sarà) presente un pulsantino che, premuto al power-up, forza il boot della BBB da USB interface.

La prima cosa che U-Boot fa è cercare un file di aggiornamento FW valido sulla porta USB. Se c'è, lancia l'aggiornatore, che risiede nello stesso file (invece nel KX l'aggiornatore era in U-Boot).

Il boot con aggiornamento da USB appare come segue

```
U-Boot 2016.11-xnrg-001 (Jan 19 2017 - 15:47:20 +0100)

CPU : AM335X-GP rev 2.1
I2C: ready
DRAM: 512 MiB
MMC: OMAP SD/MMC: 0, OMAP SD/MMC: 1
Set fb memory uncachable [9fd00000:9fe2c020]
In: ns16550_serial
Out: ns16550_serial
Err: ns16550_serial
USB0: scanning bus 0 for devices... 3 USB Device(s) found
1 Storage Device(s) found
Magic: 0xc05b3d01
Firmware version: 0 2 1 0
Total size: 159199744 bytes
Flags: 0x00030003
Prescript crc: 0x00000000, size: 0 (bytes)
Kernel crc: 0xbcc53a41f, size: 3229848 (bytes)
DTB crc: 0x8ba5228b, size: 63614 (bytes)
Initrd crc: 0x9427faa3, size: 16037260 (bytes)
Partcfg crc: 0x80cd01e3, size: 175 (bytes)
MLO crc: 0xd9e1895c, size: 66064 (bytes)
Uboot crc: 0x177b05bc, size: 2542476 (bytes)
Rootfs crc: 0x190d59cc, size: 137257683 (bytes)
Archive crc: 0x00000000, size: 0 (bytes)
Header crc: 0x0f423f2e, size: 96 (bytes)
XNRG: update file found.
Starting update...
XNRG fwupdate: block partcfg (size 175; crc 0x80cd01e3; pos 0x0126fc00)
XNRG fwupdate: loading partcfg block
XNRG fwupdate: checking partcfg block
XNRG fwupdate: block prescript (size 0; crc 0x00000000; pos 0x000000200)
XNRG fwupdate: block kernel (size 3229848; crc 0xbcc53a41f; pos 0x000000200)
XNRG fwupdate: loading kernel block
XNRG fwupdate: checking kernel block
XNRG fwupdate: block initrd (size 16037260; crc 0x9427faa3; pos 0x00324600)
XNRG fwupdate: loading initrd block
XNRG fwupdate: checking initrd block
XNRG fwupdate: block dtb (size 63614; crc 0x8ba5228b; pos 0x00314c00)
XNRG fwupdate: loading dtb block
XNRG fwupdate: checking dtb block
Writing to MMC(1)... done
Starting Linux with arguments
blkdevparts=mmcblk1:512k(MLO),4608k(uboot),512k(ubootenv0),4608k(uboot1),512k(ubootenv1),512k(XNRGenv0),10m(kernel0),512k(dt)
```

(<- trova la Pennetta USB)
(<- trova il file di aggiornamento e ne legge l'header)
(<- la nuova versione FW)
(<- la dimensione totale del file)
(<- le flag presenti nell'header)
(<- nessun file di script U-boot da eseguire)
(<- fwupd-initrd.img. Deve essere caricato in ram)
(<- partizionamento della eMMC. Può essere modificato)
(<- rootfs.tar.gz, il nuovo filesystem)
(<- archive.tar.gz, in questo caso non c'è)

(<- aggiornamento partizioni eMMC ?)

Engineering Specification - confidential

```
b0),10m(kernel1),512k(dtb1),512m(rootfs).-(archive) root=/dev/ram0 rootfstype=ramfs console=ttyS0,115200n8 vt.cur_default=1
vt.global_cursor_default=0 consoleblank=0
(<- boot da eMMC con RAM disk filesystem)

## Booting kernel from Legacy Image at 82000000 ...
Image Name: Linux-4.4.39-ti-rt-r76-X
Created: 2017-01-17 18:04:04 UTC
Image Type: ARM Linux Kernel Image (uncompressed)
Data Size: 3229784 Bytes = 3.1 MiB
Load Address: 82000000
Entry Point: 82000000
Verifying Checksum ... OK
## Loading init Ramdisk from Legacy Image at 88080000 ...
Image Name:
Created: 2017-01-19 14:40:40 UTC
Image Type: ARM Linux RAMDisk Image (uncompressed)
Data Size: 16037196 Bytes = 15.3 MiB
Load Address: 00000000
Entry Point: 00000000
Verifying Checksum ... OK
## Flattened Device Tree blob at 88000000
Booting using the fdt blob at 0x88000000
Loading Kernel Image ... OK
Loading Ramdisk to 8f0b4000, end 8fffff54c ... OK
Loading Device Tree to 8f0a1000, end 8f0b387d ... OK

Starting kernel ...

[ 0.00000] Booting Linux on physical CPU 0x0
[ 0.00000] Linux version 4.4.39-ti-rt-r76-X (ingrassia@erdinger) (gcc version 4.9.3 (crosstool-NG crosstool-ng-1.22.0) ) #1 PREEMPT RT
Tue Jan 17 19:03:59 CET 2017
[ 0.00000] CPU: ARMv7 Processor [413fc082] revision 2 (ARMv7), cr=50c5387d
[ 0.00000] CPU: PIPT / VIPT nonaliasing data cache, VIPT aliasing instruction cache
[ 0.00000] Machine model: TI AM335x BeagleBone Black
[ 0.00000] cma: Reserved 24 MiB at 0x9e800000
[ 0.00000] Memory policy: Data cache writeback
[ 0.00000] CPU: All CPU(s) started in SVC mode.
[ 0.00000] AM335X ES2.1 (sgx neon )
[ 0.00000] Built 1 zonelists in Zone order, mobility grouping on. Total pages: 130048
[ 0.00000] Kernel command line:
blkdevparts=mmcblk1:512k(MLO),4608k(uboot),512k(ubootenv0),4608k(uboot1),512k(ubootenv1),512k(XNRGenv0),10m(kernel0),512k(dt
b0),10m(kernel1),512k(dtb1),512m(rootfs).-(archive) root=/dev/ram0 rootfstype=ramfs console=ttyS0,115200n8 vt.cur_default=1
vt.global_cursor_default=0 consoleblank=0
[ 0.00000] PID hash table entries: 2048 (order: 1, 8192 bytes)
[ 0.00000] Dentry cache hash table entries: 65536 (order: 6, 262144 bytes)
[ 0.00000] Inode-cache hash table entries: 32768 (order: 5, 131072 bytes)
[ 0.00000] Memory: 471828K/524288K available (3678K kernel code, 248K rdata, 2204K rodata, 788K init, 401K bss, 27884K
reserved, 24576K cma-reserved, 0K highmem)
[ 0.00000] Virtual kernel memory layout:
[ 0.00000]   vector : 0xffff0000 - 0xffff1000 ( 4 kB)
[ 0.00000]   fixmap : 0xffc00000 - 0xfff00000 (3072 kB)
[ 0.00000]   vmalloc : 0xe0800000 - 0xff800000 ( 496 MB)
[ 0.00000]   lowmem : 0xc0000000 - 0xe0000000 ( 512 MB)
[ 0.00000]   pkmap : 0xbfe00000 - 0xc0000000 ( 2 MB)
[ 0.00000]   modules : 0xbff800000 - 0xbfe00000 ( 6 MB)
[ 0.00000]   .text : 0xc0008000 - 0xc05c6a2c (5883 kB)
[ 0.00000]   .init : 0xc05c7000 - 0xc068c000 ( 788 kB)
[ 0.00000]   .data : 0xc068c000 - 0xc06ca1d8 ( 249 kB)
[ 0.00000]   .bss : 0xc06cd000 - 0xc07316c8 ( 402 kB)
[ 0.00000] SLUB: HWalign=64, Order=0-3, MinObjects=0, CPUs=1, Nodes=1
[ 0.00000] Preemptible hierarchical RCU implementation.
[ 0.00000]   Build-time adjustment of leaf fanout to 32.
[ 0.00000] NR_IRQS:16 nr_irqs:16 16
[ 0.00000] IRQ: Found an INTC at 0xfa200000 (revision 5.0) with 128 interrupts
[ 0.00000] OMAP clockevent source: timer2 at 24000000 Hz
[ 0.00001] sched_clock: 32 bits at 24MHz, resolution 41ns, wraps every 89478484971ns
[ 0.00002] clocksource: timer1: mask: 0xffffffff max_cycles: 0xffffffff, max_idle_ns: 79635851949 ns
[ 0.000027] OMAP clocksource: timer1 at 24000000 Hz
[ 0.000621] clocksource_probe: no matching clocksources found
[ 0.000745] Console: colour dummy device 80x30
[ 0.019227] Calibrating delay loop... 995.32 BogoMIPS (lpj=1990656)
[ 0.019232] pid_max: default: 32768 minimum: 301
[ 0.019324] Mount-cache hash table entries: 1024 (order: 0, 4096 bytes)
[ 0.019329] Mountpoint-cache hash table entries: 1024 (order: 0, 4096 bytes)
[ 0.019831] CPU: Testing write buffer coherency: ok
[ 0.020081] Setting up static identity map for 0x800081c0 - 0x80008214
[ 0.022775] devtmpfs: initialized
[ 0.039060] VFP support v0.3: implementor 41 architecture 3 part 30 variant c rev 3
[ 0.052008] omap_hwmod: debugss: _wait_target_disable failed
[ 0.105450] clocksource: jiffies: mask: 0xffffffff max_cycles: 0xffffffff, max_idle_ns: 7645041785100000 ns
```

Engineering Specification - confidential

```
[ 0.106926] pinctrl core: initialized pinctrl subsystem
[ 0.108055] NET: Registered protocol family 16
[ 0.109869] DMA: preallocated 256 KiB pool for atomic coherent allocations
[ 0.119240] cpuidle: using governor ladder
[ 0.131237] cpuidle: using governor menu
[ 0.134529] GPIO line 19 (hdmidis_gpio0_19) hogged as output/low
[ 0.135026] OMAP GPIO hardware version 0.1
[ 0.135425] GPIO line 59 (hdmidkdis_gpio1_27) hogged as output/low
[ 0.136132] GPIO line 64 (ignored_gpio2_0) hogged as input
[ 0.136815] GPIO line 114 (ignored_gpio3_18) hogged as input
[ 0.136831] GPIO line 116 (ignored_gpio3_20) hogged as input
[ 0.156565] edma 49000000.edma: TI EDMA DMA engine driver
[ 0.159404] SCSI subsystem initialized
[ 0.159663] usbcore: registered new interface driver usbf
[ 0.159720] usbcore: registered new interface driver hub
[ 0.159821] usbcore: registered new device driver usb
[ 0.160230] omap_i2c 44e0b000.i2c: could not find pctldev for node
/ocp/l4_wkup@44c00000/scm@210000/pinmux@800/pinmux_i2c0_pins, deferring probe
[ 0.160270] omap_i2c 4802a000.i2c: could not find pctldev for node
/ocp/l4_wkup@44c00000/scm@210000/pinmux@800/pinmux_i2c1_pins, deferring probe
[ 0.160296] omap_i2c 4819c000.i2c: could not find pctldev for node
/ocp/l4_wkup@44c00000/scm@210000/pinmux@800/pinmux_touch_i2c2_pins, deferring probe
[ 0.160410] pps_core: LinuxPPS API ver. 1 registered
[ 0.160416] pps_core: Software ver. 5.3.6 - Copyright 2005-2007 Rodolfo Giometti <giometti@linux.it>
[ 0.160440] PTP clock support registered
[ 0.161056] omap-mailbox 480c8000.mailbox: omap mailbox rev 0x400
[ 0.162084] clocksource: Switched to clocksource timer1
[ 0.170311] NET: Registered protocol family 2
[ 0.170852] TCP established hash table entries: 4096 (order: 2, 16384 bytes)
[ 0.170907] TCP bind hash table entries: 4096 (order: 4, 98304 bytes)
[ 0.171020] TCP: Hash tables configured (established 4096 bind 4096)
[ 0.171085] UDP hash table entries: 256 (order: 2, 16384 bytes)
[ 0.171110] UDP-Lite hash table entries: 256 (order: 2, 16384 bytes)
[ 0.171267] NET: Registered protocol family 1
[ 0.171578] RPC: Registered named UNIX socket transport module.
[ 0.171589] RPC: Registered udp transport module.
[ 0.171594] RPC: Registered tcp transport module.
[ 0.171600] RPC: Registered tcp NFSv4.1 backchannel transport module.
[ 0.172311] Trying to unpack rootfs image as initramfs...
[ 7.941786] Freeing initrd memory: 15664K (cf0b4000 - d0000000)
[ 7.944648] futex hash table entries: 256 (order: 1, 8192 bytes)
[ 7.951654] NFS: Registering the id_resolver key type
[ 7.951720] Key type id_resolver registered
[ 7.951727] Key type id_legacy registered
[ 7.951983] fuse init (API version 7.23)
[ 7.956398] NET: Registered protocol family 38
[ 7.956630] Block layer SCSI generic (bsg) driver version 0.4 loaded (major 250)
[ 7.956646] io scheduler noop registered (default)
[ 7.957352] pinctrl-single 44e10800.pinmux: 142 pins at pa f9e10800 size 568
[ 7.960365] wkup_m3_ipc 44e11324.wkup_m3_ipc: could not get rproc handle
[ 7.960789] Serial: 8250/16550 driver, 6 ports, IRQ sharing disabled
[ 7.962528] console [ttyS0] disabled
[ 7.962616] 44e09000.serial: ttyS0 at MMIO 0x44e09000 (irq = 158, base_baud = 3000000) is a 8250
[ 8.578770] console [ttyS0] enabled
[ 8.583698] omap_rng 48310000.rng: OMAP Random Number Generator ver. 20
[ 8.590469] [drm] Initialized drm 1.1.0 20060810
[ 8.595459] panel panel: found backlight
[ 8.600483] [drm] Supports vblank timestamp caching Rev 2 (21.10.2013).
[ 8.607139] [drm] No driver support for vblank timestamp query.
[ 8.622580] Console: switching to colour frame buffer device 128x37
[ 8.640983] tilcdc 4830e000.lcdc: fb0: frame buffer device
[ 8.646746] [drm] Initialized tilcdc 1.0.20121205 on minor 0
[ 8.655598] brd: module loaded
[ 8.663860] loop: module loaded
[ 8.669216] ad_dpdt spi2.1: ad5235 1024-Position Digital Potentiometer registered
[ 8.726100] davinci_mdio 4a101000.mdio: davinci mdio revision 1.6
[ 8.732231] davinci_mdio 4a101000.mdio: detected phy mask ffffffe
[ 8.738796] davinci_mdio: dt: updated phy_id[0] from phy_mask[ffffffff]
[ 8.752432] libphy: 4a101000.mdio: probed
[ 8.756489] davinci_mdio 4a101000.mdio: phy[0]: device 4a101000.mdio:00, driver SMSC LAN8710/LAN8720
[ 8.766370] cpsw 4a100000.ethernet: Detected MACID = 04:a3:16:b0:7d:f6
[ 8.774390] ehci_hcd: USB 2.0 'Enhanced' Host Controller (EHCI) Driver
[ 8.780989] ehci-platform: EHCI generic platform driver
[ 8.786409] ehci-omap: OMAP-EHCI Host Controller driver
[ 8.791845] usbcore: registered new interface driver usb-storage
[ 8.797994] usbcore: registered new interface driver usbserial
[ 8.803894] usbcore: registered new interface driver ftdi_sio
[ 8.809709] usbserial: USB Serial support registered for FTDI USB Serial Device
```

Engineering Specification - confidential

```
[ 8.818630] 47401300.usb phy supply vcc not found, using dummy regulator
[ 8.827339] musb-hdrc musb-hdrc.0.auto: Failed to request rx1.
[ 8.833255] musb-hdrc musb-hdrc.0.auto: musb_init_controller failed with status -517
[ 8.841742] 47401b00.usb phy supply vcc not found, using dummy regulator
[ 8.850369] musb-hdrc musb-hdrc.1.auto: Failed to request rx1.
[ 8.856274] musb-hdrc musb-hdrc.1.auto: musb_init_controller failed with status -517
[ 8.870803] omap_rtc 44e3e000 rtc: rtc core: registered 44e3e000 rtc as rtc0
[ 8.878240] i2c /dev entries driver
[ 8.883207] omap_wdt: OMAP Watchdog Timer Rev 0x01: initial timeout 60 sec
[ 8.890410] cpuidle: enable-method property 'ti,am3352' found operations
[ 8.897680] omap_hsmmc 48060000.mmc: Got CD GPIO
[ 8.967782] omap-aes 53500000.aes: OMAP AES hw accel rev: 3.2
[ 8.974956] omap-sham 53100000.sham: hw accel on OMAP rev 4.3
[ 8.981677] X_tiecap 48300100.ecap: Found new device 48300100.ecap[-1]
[ 8.988771] X_tiecap 48300100.ecap: mode pwm
[ 8.993610] X_tiecap 48300100.ecap: clk freq: 100000000 Hz
[ 8.999190] mmc0: host does not support reading read-only switch, assuming write-enable
[ 9.002896] mmc0: new high speed SD card at address e624
[ 9.010138] mmcblk0: mmc0:e624 SU02G 1.84 GiB
[ 9.015641] mmcblk0: p1
[ 9.027185] X_tiecap 48300100.ecap: eCap PWM device [-1] ready (io fa300100, len 128, irq 184)
[ 9.036471] X_tiecap 48304100.ecap: Found new device 48304100.ecap[-1]
[ 9.043556] X_tiecap 48304100.ecap: mode ecap
[ 9.048476] X_tiecap 48304100.ecap: clk freq: 100000000 Hz
[ 9.054615] X_tiecap 48304100.ecap: eCap capture device [-1] ready (io fa304100, len 128, irq 185)
[ 9.056702] mmc1: MAN_BKOPS_EN bit is not set
[ 9.064064] mmc1: new high speed MMC card at address 0001
[ 9.070119] mmcblk1: mmc1:0001 S10004 3.56 GiB
[ 9.074119] mmcblk1boot0: mmc1:0001 S10004 partition 1 4.00 MiB
[ 9.078115] mmcblk1boot1: mmc1:0001 S10004 partition 2 4.00 MiB
[ 9.078503] mmcblk1: p1(MLO) p2(uboot) p3(ubootenv0) p4(uboot1) p5(ubootenv1) p6(XNRGenv0) p7(kernel0) p8(dtbo) p9(kernel1)
p10(dtbo1) p11(rootfs) p12(archive)
[ 9.108453] remoteproc0: wkup_m3 is available
[ 9.112920] remoteproc0: Note: remoteproc is still under development and considered experimental.
[ 9.121915] remoteproc0: THE BINARY FORMAT IS NOT YET FINALIZED, and backward compatibility isn't yet guaranteed.
[ 9.133293] NET: Registered protocol family 17
[ 9.137903] Key type dns_resolver registered
[ 9.142339] omap_voltage_late_init: Voltage driver support not added
[ 9.149419] PM: Cannot get wkup_m3_ipc handle
[ 9.153841] ThumbEE CPU extension supported.
[ 9.160481] input: tps65217_pwr_but as /devices/platform/ocp/44e0b000.i2c/i2c-0/0-0024/input/input0
[ 9.189654] tps65217 0-0024: TPS65217 ID 0xe version 1.2
[ 9.195355] at24 0-0050: 32768 byte 24c256 EEPROM, writable, 1 bytes/write
[ 9.202280] omap_i2c 44e0b000.i2c: bus 0 rev0.11 at 400 kHz
[ 9.223659] pca953x 1-0021: interrupt support not compiled in
[ 9.229564] omap_i2c 4802a000.i2c: bus 1 rev0.11 at 100 kHz
[ 9.558140] edt_ft5x06 2-0038: touchscreen probe failed
[ 9.563427] edt_ft5x06: probe of 2-0038 failed with error -121
[ 9.569302] omap_i2c 4819c000.i2c: bus 2 rev0.11 at 400 kHz
[ 9.577715] remoteproc0: powering up wkup_m3
[ 9.584889] musb-hdrc musb-hdrc.1.auto: MUSB HDRC host driver
[ 9.584918] musb-hdrc musb-hdrc.1.auto: new USB bus registered, assigned bus number 1
[ 9.585102] usb usb1: New USB device found, idVendor=1d6b, idProduct=0002
[ 9.585109] usb usb1: New USB device strings: Mfr=3, Product=2, SerialNumber=1
[ 9.585114] usb usb1: Product: MUSB HDRC host driver
[ 9.585119] usb usb1: Manufacturer: Linux 4.4.39-ti-rt-r76-X musb-hcd
[ 9.585124] usb usb1: SerialNumber: musb-hdrc.1.auto
[ 9.585683] hub 1-0:1.0: USB hub found
[ 9.585716] hub 1-0:1.0: 1 port detected
[ 9.589908] PM: bootloader does not support rtc-only!
[ 9.594441] input: gpio_keys as /devices/platform/gpio_keys/input/input2
[ 9.594781] omap_rtc 44e3e000 rtc: setting system clock to 2000-01-01 00:00:01 UTC (946684801)
[ 9.594794] of_cfs_init
[ 9.594826] of_cfs_init: OK
[ 9.671590] remoteproc0: Booting fw image am335x-pm-firmware.elf, size 217148
[ 9.679702] Freeing unused kernel memory: 788K (c05c7000 - c068c000)
[ 9.689463] remoteproc0: remote processor wkup_m3 is now up
[ 9.689505] wkup_m3_ipc 44e11324.wkup_m3_ipc: CM3 Firmware Version = 0x192
INIT: version 2.88 booting
Starting logging: [ 9.974154] usb 1-1: new high-speed USB device number 2 using musb-hdrc
OK
[ 10.110426] usb 1-1: New USB device found, idVendor=0424, idProduct=2514
[ 10.121548] usb 1-1: New USB device strings: Mfr=0, Product=0, SerialNumber=0
[ 10.154930] hub 1-1:1.0: USB hub found
[ 10.162910] hub 1-1:1.0: 4 ports detected
Initializing random number generator... [ 10.202836] random: dd: uninitialized urandom read (512 bytes read, 6 bits of entropy available)
done.
Update script executed with parameters: start (<- parte l'esecuzione script di aggiornamento)
```

Engineering Specification - confidential

```
[ 10.442279] usb 1-1.1: new high-speed USB device number 3 using musb-hdrc
[ 10.550988] usb 1-1.1: New USB device found, idVendor=0403, idProduct=6011
[ 10.557958] usb 1-1.1: New USB device strings: Mfr=1, Product=2, SerialNumber=0
[ 10.565723] usb 1-1.1: Product: Quad RS232-HS
[ 10.570178] usb 1-1.1: Manufacturer: FTDI
[ 10.575030] ftdi_sio 1-1.1:1.0: FTDI USB Serial Device converter detected
[ 10.582202] usb 1-1.1: Detected FT4232H
[ 10.586585] usb 1-1.1: FTDI USB Serial Device converter now attached to ttyUSB0
[ 10.594642] ftdi_sio 1-1.1:1.1: FTDI USB Serial Device converter detected
[ 10.601573] usb 1-1.1: Detected FT4232H
[ 10.605920] usb 1-1.1: FTDI USB Serial Device converter now attached to ttyUSB1
[ 10.613851] ftdi_sio 1-1.1:1.2: FTDI USB Serial Device converter detected
[ 10.620928] usb 1-1.1: Detected FT4232H
[ 10.625168] usb 1-1.1: FTDI USB Serial Device converter now attached to ttyUSB2
[ 10.633023] ftdi_sio 1-1.1:1.3: FTDI USB Serial Device converter detected
[ 10.640151] usb 1-1.1: Detected FT4232H
[ 10.644385] usb 1-1.1: FTDI USB Serial Device converter now attached to ttyUSB3
[ 10.738279] usb 1-1.2: new high-speed USB device number 4 using musb-hdrc
[ 10.846677] usb 1-1.2: New USB device found, idVendor=abcd, idProduct=1234
[ 10.853602] usb 1-1.2: New USB device strings: Mfr=1, Product=2, SerialNumber=3
[ 10.861375] usb 1-1.2: Product: UDisk
[ 10.866112] usb 1-1.2: Manufacturer: General
[ 10.870518] usb 1-1.2: SerialNumber: ?
[ 10.875090] usb-storage 1-1.2:1.0: USB Mass Storage device detected
[ 10.881887] scsi host0: usb-storage 1-1.2:1.0
Waiting /dev/sda1 to appear... [ 11.886932] scsi 0:0:0:0: Direct-Access General UDisk      5.00 PQ: 0 ANSI: 2
[ 11.887948] sd 0:0:0:0: [sda] 1968128 512-byte logical blocks: (1.01 GB/961 MiB)
[ 11.888081] sd 0:0:0:0: [sda] Write Protect is off
[ 11.888215] sd 0:0:0:0: [sda] No Caching mode page found
[ 11.888223] sd 0:0:0:0: [sda] Assuming drive cache: write through
[ 11.889889] sda: sda1
done.
Trying mounting /dev/sda1...[ 11.925582] sd 0:0:0:0: [sda] Attached SCSI removable disk
[ 12.491075] random: nonblocking pool is initialized
Firmware update: core ...
blkdevparts=mmcblk1:512k(MLO),4608k(uboot),512k(ubootenv0),4608k(uboot1),512k(ubootenv1),512k(XNRGenv0),10m(kernel0),512k(dt
b0),10m(kernel1),512k(dtb1),512m(rootfs).-(archive)
parse_partcfg: found device mmcblk1
found 12 partitions
root device: mmcblk1
part 1: MLO
part 2: uboot
part 3: ubootenv0
part 4: uboot1
part 5: ubootenv1
part 6: XNRGenv0
part 7: kernel0
part 8: dtb0
part 9: kernel1
part 10: dtb1
part 11: rootfs
part 12: archive
do_update_block: writing block 'MLO' on dev '/dev/mmcblk1p1' (offset 0).
do_update_block: writing block 'MLO' on dev '/dev/mmcblk1p1' (offset 131072).
do_update_block: writing block 'MLO' on dev '/dev/mmcblk1p1' (offset 262144).
do_update_block: writing block 'MLO' on dev '/dev/mmcblk1p1' (offset 393216).
do_erase_part: erase 524288 bytes on dev '/dev/mmcblk1p3'.
do_update_core: unable to find blockid for block name ubootenv, skip.
do_update_block: writing block 'uboot' on dev '/dev/mmcblk1p2' (offset 0).
do_update_block: writing block 'kernel' on dev '/dev/mmcblk1p7' (offset 0).
do_update_block: writing block 'dtb' on dev '/dev/mmcblk1p8' (offset 0).
do_erase_part: erase 524288 bytes on dev '/dev/mmcblk1p5'.
do_update_core: unable to find blockid for block name ubootenv, skip.
do_update_block: writing block 'uboot' on dev '/dev/mmcblk1p4' (offset 0).
do_update_block: writing block 'kernel' on dev '/dev/mmcblk1p9' (offset 0).
do_update_block: writing block 'dtb' on dev '/dev/mmcblk1p10' (offset 0).
Formatting archive partition @ /dev/mmcblk1p12... mke2fs 1.42.13 (17-May-2015)          (<- inizia a copiare i vari pezzi nella eMMC)
fs_types for mke2fs.conf resolution: 'ext4'
Discarding device blocks: 4096/794624528384/794624      done
Filesystem label=archive
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
Stride=0 blocks, Stripe width=0 blocks
198800 inodes, 794624 blocks
39731 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=813694976          (<- formatta partizione archive)
```

Engineering Specification - confidential

```
25 block groups
32768 blocks per group, 32768 fragments per group
7952 inodes per group
Filesystem UUID: 0a957632-7393-40d2-994a-e9ec9b5705d6
Superblock backups stored on blocks:
    32768, 98304, 163840, 229376, 294912

Allocating group tables: 0/25  done
Writing inode tables: 0/25  done
Creating journal (16384 blocks): done
Writing superblocks and filesystem accounting information: 0/25 2/25  done

done.[ 32.062423] EXT4-fs (mmcblk1p12): mounted filesystem with ordered data mode. Opts: (null)
Firmware update: check for rootfs ...
do_update_block: writing block 'rootfs' on dev '/mnt/archive/rootfs.tar.gz' (offset 0).      (<- copia rootfs.tar.gz su partizione archive)
Retrieved compressed rootfs from fw file
Formatting root partition @ /dev/mmcblk1p1... mke2fs 1.42.13 (17-May-2015)      (<- formatta partizione rootfs)
fs_types for mke2fs.conf resolution: 'ext4'
Discarding device blocks: 4096/131072      done
Filesystem label=rootfs
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
Stride=0 blocks, Stripe width=0 blocks
32768 inodes, 131072 blocks
6553 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=134217728
4 block groups
32768 blocks per group, 32768 fragments per group
8192 inodes per group
Filesystem UUID: 99d3f98a-f0e2-4d36-ab22-3ba6e0eb5fa9
Superblock backups stored on blocks:
    32768, 98304

Allocating group tables: 0/4  done
Writing inode tables: 0/4  done
Creating journal (4096 blocks): done
Writing superblocks and filesystem accounting information: 0/4  done

done.[ 46.399920] EXT4-fs (mmcblk1p11): mounted filesystem with ordered data mode. Opts: (null)
Firmware update: processing rootfs ...
Firmware update: processing rootfs done.
Firmware update: check for archive ...
do_update_block: no data for block 'archive', skip.
Update complete. System will now reboot in 10 seconds...
Update complete. System will now reboot in 9 seconds...
Update complete. System will now reboot in 8 seconds...
Update complete. System will now reboot in 7 seconds...
Update complete. System will now reboot in 6 seconds...
Update complete. System will now reboot in 5 seconds...
Update complete. System will now reboot in 4 seconds...
Update complete. System will now reboot in 3 seconds...
Update complete. System will now reboot in 2 seconds...
Update complete. System will now reboot in 1 seconds...
Update complete. System will now reboot in 0 seconds...
INIT: INIT: Sending processes the TERM signal
Update script executed with parameters: stop
Update script invoked with stop, exit.
Saving random seed... done.
Stopping logging: OK
mount: dev is already mounted or /dev busy
    dev is already mounted on /dev
umount: /dev/shm: mountpoint not found
umount: /: not mounted

Welcome to X XNRG
xnrgr login: [ 109.234957] musb-hdrc musb-hdrc.1.auto: remove, state 1
[ 109.240252] usb usb1: USB disconnect, device number 1
[ 109.245328] usb 1-1: USB disconnect, device number 2
[ 109.250316] usb 1-1.1: USB disconnect, device number 3
[ 109.255852] ftdi_sio ttyUSB0: FTDI USB Serial Device converter now disconnected from ttyUSB0
[ 109.264371] ftdi_sio 1-1.1:0: device disconnected
[ 109.269571] ftdi_sio ttyUSB1: FTDI USB Serial Device converter now disconnected from ttyUSB1
[ 109.278091] ftdi_sio 1-1.1:1: device disconnected
[ 109.283271] ftdi_sio ttyUSB2: FTDI USB Serial Device converter now disconnected from ttyUSB2
[ 109.291772] ftdi_sio 1-1.1:2: device disconnected
[ 109.296996] ftdi_sio ttyUSB3: FTDI USB Serial Device converter now disconnected from ttyUSB3
```

Engineering Specification - confidential

```
[ 109.305503] ftdi_sio 1-1.1:1.3: device disconnected
[ 109.310668] usb 1-1.2: USB disconnect, device number 4
[ 109.328347] musb-hdrc musb-hdrc.1.auto: USB bus 1 deregistered
[ 109.364949] reboot: Restarting system

U-Boot SPL 2016.11-xnrg-001 (Jan 19 2017 - 15:47:20)
Trying to boot from MMC1                                     (<- è il file rootfs.tar.gz)
reading u-boot.img
reading u-boot.img

U-Boot 2016.11-xnrg-001 (Jan 19 2017 - 15:47:20 +0100)

CPU : AM335X-GP rev 2.1
I2C: ready
DRAM: 512 MiB
MMC: OMAP SD/MMC: 0, OMAP SD/MMC: 1
** First descriptor is NOT a primary desc on 1:1 **
*** Warning - bad CRC, using default environment

Set fb memory uncachable [9fd00000;9fe2c020]
In: ns16550_serial
Out: ns16550_serial
Err: ns16550_serial
<ethaddr> not set. Validating first E-fuse MAC
Writing to redundant MMC(1)... done
XNRG: search for firmware update on next boot
Net: cpsw, usb_ether
Hit any key to stop autoboot: 2 1 0
switch to partitions #0, OK
mmc0 is current device
SD/MMC found on device 0
reading boot.scr
** Unable to read file boot.scr **
reading uEnv.txt
** Unable to read file uEnv.txt **
Booting from eMMC ...
Loading kernel from eMMC...
** First descriptor is NOT a primary desc on 1:1 **
switch to partitions #0, OK
mmc1(part 0) is current device

MMC read: dev # 1, block # 22528, count 20480 ... 20480 blocks read: OK
Loading DTB from eMMC...
** First descriptor is NOT a primary desc on 1:1 **
switch to partitions #0, OK
mmc1(part 0) is current device

MMC read: dev # 1, block # 43008, count 1024 ... 1024 blocks read: OK
## Booting kernel from Legacy Image at 82000000 ...
Image Name: Linux-4.4.39-ti-rt-r76-X
Created: 2017-01-17 18:04:04 UTC
Image Type: ARM Linux Kernel Image (uncompressed)
Data Size: 3229784 Bytes = 3.1 MiB
Load Address: 82000000
Entry Point: 82000000
Verifying Checksum ... OK
## Flattened Device Tree blob at 88000000
Booting using the fdt blob at 0x88000000
Loading Kernel Image ... OK
Loading Device Tree to 8ffed000, end 8ffff87d ... OK

Starting kernel ...

[ 0.000000] Booting Linux on physical CPU 0x0
[ 0.000000] Linux version 4.4.39-ti-rt-r76-X (ingrassia@erdinger) (gcc version 4.9.3 (crosstool-NG crosstool-ng-1.22.0) ) #1 PREEMPT RT
Tue Jan 17 19:03:59 CET 2017
[ 0.000000] CPU: ARMv7 Processor [413fc082] revision 2 (ARMv7), cr=50c5387d
[ 0.000000] CPU: PIPT / VIPT nonaliasing data cache, VIPT aliasing instruction cache
[ 0.000000] Machine model: TI AM335x BeagleBone Black
[ 0.000000] cma: Reserved 24 MiB at 0x9e800000
[ 0.000000] Memory policy: Data cache writeback
[ 0.000000] CPU: All CPU(s) started in SVC mode.
[ 0.000000] AM335X ES2.1 (sgx neon )
[ 0.000000] Built 1 zonelists in Zone order, mobility grouping on. Total pages: 130048
[ 0.000000] Kernel command line: root=/dev/mmcblk1p11 rw ip=off vt.cur_default=1 vt.global_cursor_default=0 consoleblank=0
console=ttyS0,115200n8
```

Engineering Specification - confidential

```
blkdevparts=mmcblk1:512k(MLO),4608k(uboot),512k(ubootenv0),4608k(uboot1),512k(ubootenv1),512k(XNRGenv0),10m(kernel0),512k(dt  
b0),10m(kernel1),512k(dtb1),512m(rootfs).-(archive)  
[ 0.000000] PID hash table entries: 2048 (order: 1, 8192 bytes)  
[ 0.000000] Dentry cache hash table entries: 65536 (order: 6, 262144 bytes)  
[ 0.000000] Inode-cache hash table entries: 32768 (order: 5, 131072 bytes)  
[ 0.000000] Memory: 487496K/524288K available (3678K kernel code, 248K rdata, 2204K rodata, 788K init, 401K bss, 12216K  
reserved, 24576K cma-reserved, 0K highmem)  
[ 0.000000] Virtual kernel memory layout:  
[ 0.000000]   vector : 0xffff0000 - 0xffff1000 ( 4 kB)  
[ 0.000000]   fixmap : 0xffc00000 - 0xfff00000 (3072 kB)  
[ 0.000000]   vmalloc : 0xe0800000 - 0xff800000 ( 496 MB)  
[ 0.000000]   lowmem : 0x00000000 - 0xe0000000 ( 512 MB)  
[ 0.000000]   pkmap : 0xbfe00000 - 0xc0000000 ( 2 MB)  
[ 0.000000]   modules : 0xbff800000 - 0xbfe00000 ( 6 MB)  
[ 0.000000]   .text : 0xc0008000 - 0xc05c6a2c (5883 kB)  
[ 0.000000]   .init : 0xc05c7000 - 0xc068c000 ( 788 kB)  
[ 0.000000]   .data : 0xc068c000 - 0xc06ca1d8 ( 249 kB)  
[ 0.000000]   .bss : 0xc06cd000 - 0xc07316c8 ( 402 kB)  
[ 0.000000] SLUB: HWalign=64, Order=0-3, MinObjects=0, CPUs=1, Nodes=1  
[ 0.000000] Preemptible hierarchical RCU implementation.  
[ 0.000000]   Build-time adjustment of leaf fanout to 32.  
[ 0.000000] NR_IRQS:16 nr_irqs:16 16  
[ 0.000000] IRQ: Found an INTC at 0xfa200000 (revision 5.0) with 128 interrupts  
[ 0.000000] OMAP clockevent source: timer2 at 24000000 Hz  
[ 0.000013] sched_clock: 32 bits at 24MHz, resolution 41ns, wraps every 89478484971ns  
[ 0.000024] clocksource: timer1: mask: 0xffffffff max_cycles: 0xffffffff, max_idle_ns: 79635851949 ns  
[ 0.000029] OMAP clocksource: timer1 at 24000000 Hz  
[ 0.000628] clocksource_probe: no matching clocksources found  
[ 0.000750] Console: colour dummy device 80x30  
[ 0.019231] Calibrating delay loop... 995.32 BogoMIPS (lpj=1990656)  
[ 0.019235] pid_max: default: 32768 minimum: 301  
[ 0.019326] Mount-cache hash table entries: 1024 (order: 0, 4096 bytes)  
[ 0.019331] Mountpoint-cache hash table entries: 1024 (order: 0, 4096 bytes)  
[ 0.019835] CPU: Testing write buffer coherency: ok  
[ 0.020087] Setting up static identity map for 0x800081c0 - 0x80008214  
[ 0.022789] devtmpfs: initialized  
[ 0.039060] VFP support v0.3: implementor 41 architecture 3 part 30 variant c rev 3  
[ 0.051992] omap_hwmod: debugss: _wait_target_disable failed  
[ 0.105449] clocksource: jiffies: mask: 0xffffffff max_cycles: 0xffffffff, max_idle_ns: 7645041785100000 ns  
[ 0.106922] pinctrl core: initialized pinctrl subsystem  
[ 0.108056] NET: Registered protocol family 16  
[ 0.109861] DMA: preallocated 256 KiB pool for atomic coherent allocations  
[ 0.119243] cpuidle: using governor ladder  
[ 0.131239] cpuidle: using governor menu  
[ 0.134528] GPIO line 19 (hdmidis_gpio0_19) hogged as output/low  
[ 0.135026] OMAP GPIO hardware version 0.1  
[ 0.135423] GPIO line 59 (hdmiclkdis_gpio1_27) hogged as output/low  
[ 0.136135] GPIO line 64 (ignored_gpio2_0) hogged as input  
[ 0.136820] GPIO line 114 (ignored_gpio3_18) hogged as input  
[ 0.136835] GPIO line 116 (ignored_gpio3_20) hogged as input  
[ 0.156589] edma 49000000.edma: TI EDMA DMA engine driver  
[ 0.159446] SCSI subsystem initialized  
[ 0.159704] usbcore: registered new interface driver usbfsl  
[ 0.159759] usbcore: registered new interface driver hub  
[ 0.159859] usbcore: registered new device driver usb  
[ 0.160273] omap_i2c 44e0b000.i2c: could not find pctldev for node  
/ocp/l4_wkup@44c00000/scm@210000/pinmux@800/pinmux_i2c0_pins, deferring probe  
[ 0.160310] omap_i2c 4802a000.i2c: could not find pctldev for node  
/ocp/l4_wkup@44c00000/scm@210000/pinmux@800/pinmux_i2c1_pins, deferring probe  
[ 0.160337] omap_i2c 4819c000.i2c: could not find pctldev for node  
/ocp/l4_wkup@44c00000/scm@210000/pinmux@800/pinmux_touch_i2c2_pins, deferring probe  
[ 0.160452] pps_core: LinuxPPS API ver. 1 registered  
[ 0.160459] pps_core: Software ver. 5.3.6 - Copyright 2005-2007 Rodolfo Giometti <giometti@linux.it>  
[ 0.160484] PTP clock support registered  
[ 0.161107] omap-mailbox 480c8000.mailbox: omap mailbox rev 0x400  
[ 0.162133] clocksource: Switched to clocksource timer1  
[ 0.170328] NET: Registered protocol family 2  
[ 0.170874] TCP established hash table entries: 4096 (order: 2, 16384 bytes)  
[ 0.170928] TCP bind hash table entries: 4096 (order: 4, 98304 bytes)  
[ 0.171041] TCP: Hash tables configured (established 4096 bind 4096)  
[ 0.171104] UDP hash table entries: 256 (order: 2, 16384 bytes)  
[ 0.171130] UDP-Lite hash table entries: 256 (order: 2, 16384 bytes)  
[ 0.171281] NET: Registered protocol family 1  
[ 0.171596] RPC: Registered named UNIX socket transport module.  
[ 0.171606] RPC: Registered udp transport module.  
[ 0.171611] RPC: Registered tcp transport module.  
[ 0.171617] RPC: Registered tcp NFSv4.1 backchannel transport module.  
[ 0.175139] futex hash table entries: 256 (order: 1, 8192 bytes)
```

Engineering Specification - confidential

```
[ 0.181959] NFS: Registering the id_resolver key type
[ 0.182014] Key type id_resolver registered
[ 0.182021] Key type id_legacy registered
[ 0.182394] fuse init (API version 7.23)
[ 0.186800] NET: Registered protocol family 38
[ 0.187031] Block layer SCSI generic (bsg) driver version 0.4 loaded (major 250)
[ 0.187048] io scheduler noop registered (default)
[ 0.187729] pinctrl-single 44e10800.pinmux: 142 pins at pa f9e10800 size 568
[ 0.190847] wkup_m3_ipc 44e11324.wkup_m3_ipc: could not get rproc handle
[ 0.191286] Serial: 8250/16550 driver, 6 ports, IRQ sharing disabled
[ 0.192818] console [ttyS0] disabled
[ 0.192902] 44e09000.serial: ttyS0 at MMIO 0x44e09000 (irq = 158, base_baud = 3000000) is a 8250
[ 0.797626] console [ttyS0] enabled
[ 0.802578] omap_rng 48310000.rng: OMAP Random Number Generator ver. 20
[ 0.809353] [drm] Initialized drm 1.1.0 20060810
[ 0.814374] panel panel: found backlight
[ 0.819385] [drm] Supports vblank timestamp caching Rev 2 (21.10.2013).
[ 0.826033] [drm] No driver support for vblank timestamp query.
[ 0.841417] Console: switching to colour frame buffer device 128x37
[ 0.859783] tilcdc 4830e000.lcdc: fb0: frame buffer device
[ 0.865543] [drm] Initialized tilcdc 1.0.0 20121205 on minor 0
[ 0.874384] brd: module loaded
[ 0.882722] loop: module loaded
[ 0.888117] ad_dpot spi2.1: ad5235 1024-Position Digital Potentiometer registered
[ 0.942145] davinci_mdio 4a101000.mdio: davinci mdio revision 1.6
[ 0.948277] davinci_mdio 4a101000.mdio: detected phy mask ffffffe
[ 0.954847] davinci_mdio: dt: updated phy_id[0] from phy_mask[ffffffff]
[ 0.962467] libphy: 4a101000.mdio: probed
[ 0.966510] davinci_mdio 4a101000.mdio: phy[0]: device 4a101000.mdio.00, driver SMSC LAN8710/LAN8720
[ 0.976166] cpsw 4a100000.ethernet: Detected MACID = 04:a3:16:b0:7d:f6
[ 0.983990] ehci_hcd: USB 2.0 'Enhanced' Host Controller (EHCI) Driver
[ 0.990591] ehci-platform: EHCl generic platform driver
[ 0.996021] ehci-omap: OMAP-EHCI Host Controller driver
[ 1.001459] usbcore: registered new interface driver usb-storage
[ 1.007607] usbcore: registered new interface driver usbserial
[ 1.013504] usbcore: registered new interface driver ftdi_sio
[ 1.019312] usbserial: USB Serial support registered for FTDI USB Serial Device
[ 1.028137] 47401300.usb-phy supply vcc not found, using dummy regulator
[ 1.036812] musb-hdrc musb-hdrc.0.auto: Failed to request rx1.
[ 1.042728] musb-hdrc musb-hdrc.0.auto: musb_init_controller failed with status -517
[ 1.051208] 47401b00.usb-phy supply vcc not found, using dummy regulator
[ 1.059862] musb-hdrc musb-hdrc.1.auto: Failed to request rx1.
[ 1.065767] musb-hdrc musb-hdrc.1.auto: musb_init_controller failed with status -517
[ 1.079877] omap_rtc 44e3e000 rtc: already running
[ 1.085008] omap_rtc 44e3e000 rtc: rtc core: registered 44e3e000 rtc as rtc0
[ 1.092410] i2c /dev entries driver
[ 1.097320] omap_wdt: OMAP Watchdog Timer Rev 0x01: initial timeout 60 sec
[ 1.104527] cpuidle: enable-method property 'ti,am3352' found operations
[ 1.111807] omap_hsmmc 48060000.mmc: Got CD GPIO
[ 1.179758] omap-aes 53500000.aes: OMAP AES hw accel rev: 3.2
[ 1.186953] omap-sham 53100000.sham: hw accel on OMAP rev 4.3
[ 1.193659] X_ti_ecap 48300100 ecap: Found new device 48300100.ecap[-1]
[ 1.200753] X_ti_ecap 48300100.ecap: mode pwm
[ 1.205595] X_ti_ecap 48300100.ecap: clk freq: 100000000 Hz
[ 1.211317] mmc0: host does not support reading read-only switch, assuming write-enable
[ 1.215023] mmc0: new high speed SD card at address e624
[ 1.219738] mmcblk0: mmc0:e624 SU02G 1.84 GiB
[ 1.224706] mmcblk0: p1
[ 1.239220] X_ti_ecap 48300100.ecap: eCap PWM device [-1] ready (io fa300100, len 128, irq 184)
[ 1.248505] X_ti_ecap 48304100.ecap: Found new device 48304100.ecap[-1]
[ 1.255594] X_ti_ecap 48304100.ecap: mode ecap
[ 1.260515] X_ti_ecap 48304100.ecap: clk freq: 100000000 Hz
[ 1.264757] mmc1: MAN_BKOPS_EN bit is not set
[ 1.271030] X_ti_ecap 48304100.ecap: eCap capture device [-1] ready (io fa304100, len 128, irq 185)
[ 1.272103] mmc1: new high speed MMC card at address 0001
[ 1.278274] mmcblk1: mmc1:0001 S10004 3.56 GiB
[ 1.282173] mmcblk1boot0: mmc1:0001 S10004 partition 1 4.00 MiB
[ 1.286165] mmcblk1boot1: mmc1:0001 S10004 partition 2 4.00 MiB
[ 1.286595] mmcblk1: p1(MLO) p2(uboot) p3(ubootenv0) p4(uboot1) p5(ubootenv1) p6(XNRGenv0) p7(kernel0) p8(dtbo) p9(kernel1)
p10(dtbo) p11(rootfs) p12(archive)
[ 1.320565] remoteproc0: wkup_m3 is available
[ 1.325034] remoteproc0: Note: remoteproc is still under development and considered experimental.
[ 1.334031] remoteproc0: THE BINARY FORMAT IS NOT YET FINALIZED, and backward compatibility isn't yet guaranteed.
[ 1.345407] NET: Registered protocol family 17
[ 1.350008] Key type dns_resolver registered
[ 1.354448] omap_voltage_late_init: Voltage driver support not added
[ 1.361534] PM: Cannot get wkup_m3_ipc handle
[ 1.365959] ThumbEE CPU extension supported.
```

Engineering Specification - confidential

```
[ 1.372584] input: tps65217_pwr_but as /devices/platform/ocp/44e0b000.i2c/i2c-0/0-0024/input/input0
[ 1.401432] tps65217 0-0024: TPS65217 ID 0xe version 1.2
[ 1.407136] at24 0-0050: 32768 byte 24c256 EEPROM, writable, 1 bytes/write
[ 1.414061] omap_i2c 44e0b000.i2c: bus 0 rev0.11 at 400 kHz
[ 1.435729] pca953x 1-0021: interrupt support not compiled in
[ 1.441638] omap_i2c 4802a000.i2c: bus 1 rev0.11 at 100 kHz
[ 1.774186] edt_ft5x06 2-0038: touchscreen probe failed
[ 1.779475] edt_ft5x06: probe of 2-0038 failed with error -121
[ 1.785350] omap_i2c 4819c000.i2c: bus 2 rev0.11 at 400 kHz
[ 1.793758] remoteproc0: powering up wkup_m3
[ 1.800908] musb-hdrc musb-hdrc.1.auto: MUSB HDRC host driver
[ 1.800937] musb-hdrc musb-hdrc.1.auto: new USB bus registered, assigned bus number 1
[ 1.801117] usb usb1: New USB device found, idVendor=1d6b, idProduct=0002
[ 1.801123] usb usb1: New USB device strings: Mfr=3, Product=2, SerialNumber=1
[ 1.801128] usb usb1: Product: MUSB HDRC host driver
[ 1.801133] usb usb1: Manufacturer: Linux 4.4.39-ti-rt-r76-X musb-hcd
[ 1.801138] usb usb1: SerialNumber: musb-hdrc.1.auto
[ 1.801705] hub 1-0:1.0: USB hub found
[ 1.801744] hub 1-0:1.0: 1 port detected
[ 1.805946] PM: bootloader does not support rtc-only!
[ 1.810498] input: gpio_keys as /devices/platform/gpio_keys/input/input2
[ 1.810836] omap_rtc 44e3e000.rtc: setting system clock to 2000-01-01 00:01:49 UTC (946684909)
[ 1.810849] of_cfs_init
[ 1.810883] of_cfs_init: OK
[ 1.887786] remoteproc0: Booting fw image am335x-pm-firmware.elf, size 217148
[ 1.896001] EXT4-fs (mmcblk1p11): couldn't mount as ext3 due to feature incompatibilities
[ 1.902203] remoteproc0: remote processor wkup_m3 is now up
[ 1.902234] wkup_m3_ipc 44e11324.wkup_m3_ipc: CM3 Firmware Version = 0x192
[ 1.917584] EXT4-fs (mmcblk1p11): couldn't mount as ext2 due to feature incompatibilities
[ 1.934401] EXT4-fs (mmcblk1p11): mounted filesystem with ordered data mode. Opts: (null)
[ 1.942678] VFS: Mounted root (ext4 filesystem) on device 179:43.
[ 1.949297] devtmpfs: mounted
[ 1.953073] Freeing unused kernel memory: 788K (c05c7000 - c068c000)
INIT: version 2.88 booting
[ 2.090417] EXT4-fs (mmcblk1p11): re-mounted. Opts: (null)
[ 2.190312] usb 1-1: new high-speed USB device number 2 using musb-hdrc
Running first boot operations...
Creating /mnt/archive directory...
Installing SGX drivers...
0x10205
release8.x
installing 8.x SGX release user libraries
[ 2.326545] usb 1-1: New USB device found, idVendor=0424, idProduct=2514
[ 2.336226] usb 1-1: New USB device strings: Mfr=0, Product=0, SerialNumber=0
[ 2.350947] hub 1-1:1.0: USB hub found
[ 2.358740] hub 1-1:1.0: 4 ports detected

Installing PowerVR Consumer/Embedded DDK 'sgxddk_1.10@2359475' on target

File system installation root is /

Nothing to un-install.
boot script rc.pvr -> /etc/init.d/rc.pvr
[ 2.638323] usb 1-1.1: new high-speed USB device number 3 using musb-hdrc
kernel module pvrsvrkm.ko -> /lib/modules/4.4.39-ti-rt-r76-X/extra/pvrsvrkm.ko
[ 2.738687] usb 1-1.1: New USB device found, idVendor=0403, idProduct=6011
kernel module omaplfb.ko -> /lib/modules/4.4.39-ti-rt-r76-X[ 2.747825] usb 1-1.1: New USB device strings: Mfr=1, Product=2, SerialNumber=0
/extradata/omaplfb.ko
[ 2.765075] usb 1-1.1: Product: Quad RS232-HS
[ 2.778335] usb 1-1.1: Manufacturer: FTDI
[ 2.786982] ftdi_sio 1-1.1:1.0: FTDI USB Serial Device converter detected
[ 2.798113] usb 1-1.1: Detected FT4232H
[ 2.807940] usb 1-1.1: FTDI USB Serial Device converter now attached to ttyUSB0
[ 2.824132] ftdi_sio 1-1.1:1.1: FTDI USB Serial Device converter detected
[ 2.838488] usb 1-1.1: Detected FT4232H
[ 2.846823] usb 1-1.1: FTDI USB Serial Device converter now attached to ttyUSB1
[ 2.862948] ftdi_sio 1-1.1:1.2: FTDI USB Serial Device converter detected
[ 2.875936] usb 1-1.1: Detected FT4232H
[ 2.884359] usb 1-1.1: FTDI USB Serial Device converter now attached to ttyUSB2
[ 2.893463] ftdi_sio 1-1.1:1.3: FTDI USB Serial Device converter detected
[ 2.908320] usb 1-1.1: Detected FT4232H
[ 2.916679] usb 1-1.1: FTDI USB Serial Device converter now attached to ttyUSB3
shared library libGLES_CM.so -> /usr/lib/libGLES_CM.so.1.10.2359475
[ 3.002319] usb 1-1.2: new high-speed USB device number 4 using musb-hdrc
shared library libusc.so -> /usr/lib/libusc.so.1.10.2359475[ 3.102724] usb 1-1.2: New USB device found, idVendor=abcd, idProduct=1234
[ 3.114222] usb 1-1.2: New USB device strings: Mfr=1, Product=2, SerialNumber=3
```

Engineering Specification - confidential

```
[ 3.125605] usb 1-1.2: Product: UDisk
[ 3.138677] usb 1-1.2: Manufacturer: General
[ 3.147117] usb 1-1.2: SerialNumber: ?
[ 3.156082] usb-storage 1-1.2:1.0: USB Mass Storage device detected
[ 3.174300] scsi host0: usb-storage 1-1.2:1.0
shared library libGLESv2.so -> /usr/lib/libGLESv2.so.1.10.2359475
shared library libglslcompiler.so -> /usr/lib/libglslcompiler.so.1.10.2359475
shared library libIMGegl.so -> /usr/lib/libIMGegl.so.1.10.2359475
shared library libEGL.so -> /usr/lib/libEGL.so.1.10.2359475
shared library libpvr2d.so -> /usr/lib/libpvr2d.so.1.10.2359475
shared library libpvrPVR2D_BLITWSEGL.so -> /usr/lib/libpvrPVR2D_BLITWSEGL.so.1.10.2359475
shared library libpvrPVR2D_FLIPWSEGL.so -> /usr/lib/libpvrPVR2D_FLIPWSEGL.so.1.10.2359475
shared library libpvrPVR2D_FRONTWSEGL.so -> /usr/lib/libpvrPVR2D_FRONTWSEGL.so.1.10.2359475
shared library libpvrPVR2D_LINUXFBWSEGL.so -> /usr/lib/libpvrPVR2D_LINUXFBWSEGL.so.1.10.2359475
[ 4.186824] scsi 0:0:0:0: Direct-Access General UDisk 5.00 PQ: 0 ANSI: 2
[ 4.187975] sd 0:0:0:0: [sda] 1968128 512-byte logical blocks: (1.01 GB/961 MiB)
[ 4.188124] sd 0:0:0:0: [sda] Write Protect is off
[ 4.188263] sd 0:0:0:0: [sda] No Caching mode page found
[ 4.188270] sd 0:0:0:0: [sda] Assuming drive cache: write through
[ 4.189890] sda: sda1
shared library libsrv_um.so -> /usr/lib/libsrv_um.so.1.10.2359475
[ 4.258315] sd 0:0:0:0: [sda] Attached SCSI removable disk
shared library libsrv_init.so -> /usr/lib/libsrv_init.so.1.10.2359475
shared library libPVRScopeServices.so -> /usr/lib/libPVRScopeServices.so.1.10.2359475
binary pvrsvrctl -> /usr/local/bin/pvrsvrctl
binary sgx_init_test -> /usr/local/bin/sgx_init_test
binary services_test -> /usr/local/bin/services_test
binary sgx_blt_test -> /usr/local/bin/sgx_blt_test
binary sgx_clipblt_test -> /usr/local/bin/sgx_clipblt_test
binary sgx_flip_test -> /usr/local/bin/sgx_flip_test
binary sgx_render_flip_test -> /usr/local/bin/sgx_render_flip_test
binary pvr2d_test -> /usr/local/bin/pvr2d_test
shader glsltest1_vertshader.txt -> /usr/local/bin/glsltest1_vertshader.txt
shader glsltest1_fragshaderA.txt -> /usr/local/bin/glsltest1_fragshaderA.txt
shader glsltest1_fragshaderB.txt -> /usr/local/bin/glsltest1_fragshaderB.txt
```

Installation complete!

You may now reboot your target.

Module pvrsvkm failed to load. Retrying.

Running /sbin/depmod

Loaded PowerVR consumer services.

'/etc/init.d/rc.pvr' -> '/etc/init.d/S92pvr'

First boot operations completed!

removed '/cleanboot'

X init script: [5.789980] EXT4-fs (mmcblk1p12): mounted filesystem with ordered data mode. Opts: (null)

Enabling buffers (GPIO 132)

```
[ 5.916655] ad768x spi2.0: Sampling 4 channels
[ 5.921234] ad768x spi2.0: found ad7689 (8 channels, 16 bits per word, max 5000000 HZ)
[ 5.929492] ad768x spi2.0: device configured: f053
[ 5.935616] mpl3115a2 1-0060: I2C adapter functionalities 0x0efe000d
[ 5.959642] mpl3115a2 1-0060: no IRQ line connected
[ 5.966227] mpl3115a2 1-0060: BAR IN 101326 pa
[ 5.971546] mpl3115a2 1-0060: OFF_P 0 Pa, OFF_T 0.00 C, OFF_H 0 m
[ 8.335731] ksampler: Analog offset does not fit in a s16 variable: 72076
[ 8.342804] ksampler: Analog pressure calibration: [ 1403 -> 1403 ] => offset 0
```

OK

Starting logging: OK

Initializing random number generator using /etc/random-seed...[9.019610] random: dd: uninitialized urandom read (512 bytes read, 55 bits of entropy available)

done.

Starting system message bus: [9.089857] random: dbus-uuidgen: uninitialized urandom read (12 bytes read, 56 bits of entropy available)

```
[ 9.099724] random: dbus-uuidgen: uninitialized urandom read (8 bytes read, 56 bits of entropy available)
```

```
[ 9.132104] random: dbus-daemon: uninitialized urandom read (12 bytes read, 57 bits of entropy available)
```

```
[ 9.156312] random: dbus-daemon: uninitialized urandom read (12 bytes read, 57 bits of entropy available)
```

done

Starting rpcbind: OK

Starting network...

```
[ 9.427499] random: ssh-keygen: uninitialized urandom read (32 bytes read, 60 bits of entropy available)
```

```
Starting sshd: [ 9.477828] random: sshd: uninitialized urandom read (32 bytes read, 61 bits of entropy available)
```

OK

Running iWrap daemon start script...

Starting cron ... done.

running hostgw ...

launching kmain ...

Kmain: file X.conf not found, using default

Kmain: file Signals.conf not found, using default

Kmain: file archive-init.sql not found, using default

Engineering Specification - confidential

```
Kmain: file userstrings-init.sql not found, using default
INIT: Entering runlevel: 3
[ 10.152503] random: xnrgfw: uninitialized urandom read (4 bytes read, 71 bits of entropy available)

Welcome to X XNRG
xnrg login: [ 12.332854] random: sql: uninitialized urandom read (256 bytes read, 88 bits of entropy available)
[ 12.668900] random: sql: uninitialized urandom read (256 bytes read, 104 bits of entropy available)
[ 17.133578] random: nonblocking pool is initialized

Welcome to X XNRG
xnrg login: root
Last login: Sat Jan 1 01:02:21 on ttyS0
#
```

55.6.1 boot_mmc

La scheda partirà automaticamente in questa modalità, ma solamente se la variabile bootcmd_XNRG è quella giusta.

```
target# setenv bootcmd_XNRG "run boot_mmc"
target# saveenv
```

Una volta che il kernel è partito possiamo vedere la bootcmd

```
host $ cat /proc/cmdline
root=/dev/mmcblk1p11 rw ip=off vt.cur_default=1 vt.global_cursor_default=0 console=ttyS0,115200n8
blkdevparts=mmcblk1:512k(MLO),4608k(uboot),512k(uboot.env),4608k(uboot.b),512k(uboot.env.b),512k(XNRG.env),10m(kernel),512k(dtbo),10m(kernel.b),512k(dtbo.b),512m(rootfs),-(archive)
```

Nota: come si nota la rete è disabilitata durante il boot. Solo in seguito il kernel la inizializzerà (da fare!)

55.7 Update MLO

Mentre nel KX vi era un bootloader di secondo livello (SPL) esterno a U-Boot (UBL), nella BBB è U-Boot stesso che fornisce il supporto per la compilazione del SPL e questo SPL si chiama MLO.

Quindi il file MLO viene generato dallo stesso processo di compilazione di U-Boot ed è un file che si chiama *MLO*.

In XNRG l'immagine di MLO si chiama *MLO*

Se vogliamo aggiornare MLO da U-Boot

```
target# upd_mmc_mlo
Questo comando espande in
if tftp ${loadaddr} ${target}/MLO; then mmc dev 1 ; mmc writefilesize ${loadaddr} 0x100; mmc writefilesize ${loadaddr} 0x200; mmc
writefilesize ${loadaddr} 0x300; fi
```

WARNING: il comando

```
mmc writefilesize <b> <o>
```

Scrive bytes all'offset <o>*512, in quanto il Block size della eMMC è 512 Bytes.

55.8 Update U-Boot

L'immagine u-boot.bin deve essere memorizzata in flash con un header comprensibile al SPL. Nel KX l'header cambiava (come pure l'UBL) a seconda che l'immagine fosse caricata nella SPI Flash o nella NAND Flash.

Nella BBB l'immagine di U-Boot con l'header MLO-compatibile si chiama *u-boot.img*

Se vogliamo far partire la nostra immagine di U-Boot appena compilata sull'host

```
target# setenv serverip 192.168.0.142
target# setenv ipaddr 192.168.0.149
```

Engineering Specification - confidential

```
target# tftpboot 80800000 XNRG/u-boot.bin
link up on port 0, speed 100, full duplex
Using cpsw device
TFTP from server 192.168.0.142; our IP address is 192.168.0.149
Filename 'XNRG/u-boot.bin'.
Load address: 0x80800000
Loading: #####
# #####
1.6 MiB/s
done
Bytes transferred = 420420 (66a44 hex)
target# go 80800000
## Starting application at 0x80800000 ...
```

WARNING: La versione U-Boot di default presente sulla BBB non consente di salvare la variabili di ambiente!!!

Esploriamo le partizioni flash della BBB

```
target# mmc dev 1
switch to partitions #0, OK
mmc1(part 0) is current device

target# mmc info
Device: OMAP SD/MMC
Manufacturer ID: 70
OEM: 100
Name: S1000
Tran Speed: 52000000
Rd Block Len: 512
MMC version 4.5
High Capacity: Yes
Capacity: 3.6 GiB
Bus Width: 4-bit

target# mmc part
Partition Map for MMC device 1 -- Partition Type: DOS
Part Start Sector Num Sectors UUID Type
1 2048 196608 00000000-01 0e Boot
2 198656 7272448 00000000-02 83
```

```
target# fatls mmc 1
    app/
    docs/
    drivers/
    scripts/
        40 id.txt
    41174 license.txt
    16838 readme.htm
    292 readme.md
    16838 start.htm
    288 autorun.inf
    667 nfs-uenv.txt
    system volume information/
```

7 file(s), 5 dir(s)

```
target# ext2ls mmc 1:2
<DIR> 4096 .
<DIR> 4096 ..
<DIR> 16384 lost+found
<DIR> 4096 bin
<DIR> 4096 boot
<DIR> 4096 dev
<DIR> 4096 etc
<DIR> 4096 home
<DIR> 4096 lib
<DIR> 4096 media
<DIR> 4096 mnt
<DIR> 4096 opt
<DIR> 4096 proc
<DIR> 4096 root
<DIR> 4096 run
<DIR> 4096 sbin
<DIR> 4096 selinux
<DIR> 4096 srv
<DIR> 4096 sys
```

Engineering Specification - confidential

```
<DIR> 4096 tmp
<DIR> 4096 usr
<DIR> 4096 var
    976 uEnv.txt

target# ext2ls mmc 1:2 boot
<DIR> 4096 .
<DIR> 4096 ..
<DIR> 4096 dtbs
<DIR> 4096 uboot
    488 SOC.sh
    976 uEnv.txt
2425213 System.map-3.8.13-bone70
109319 config-3.8.13-bone70
2867606 initrd.img-3.8.13-bone70
5617184 vmlinuz-3.8.13-bone70

target# ext2ls mmc 1:2 boot/uboot
<DIR> 4096 .
<DIR> 4096 ..
U-Boot# ext2ls mmc 1:2 boot/
<DIR> 4096 .
<DIR> 4096 ..
<DIR> 4096 dtbs
<DIR> 4096 uboot
    488 SOC.sh
    976 uEnv.txt
2425213 System.map-3.8.13-bone70
109319 config-3.8.13-bone70
2867606 initrd.img-3.8.13-bone70
5617184 vmlinuz-3.8.13-bone70
```

Oppure

```
target# run upd_mmc_uboot
che espande in
if tftp ${loadaddr} ${target}/u-boot.img; then mmc dev 1 ; mmc writefilesize ${loadaddr} 0x400; fi
```

Per aggiornare l'immagine di backup per U-Boot

```
target# run upd_mmc_uboot_backup
che espande in
if tftp ${loadaddr} ${target}/u-boot.img; then mmc dev 1 ; mmc writefilesize ${loadaddr} 0x80000; fi
```

55.8.1 boot_net

Analogamente a quanto avviene nel caso del KX, l'esecuzione in U-Boot del comando *boot_net*, lancia il boot in modalità NFS, con kernel e relativo DTB reperiti dall'host tramite TFTP.

Quelle che seguono sono le tutte le sole levaribili di ambiente che definiscono per intero il comando.

```
console=ttyS0,115200n8
ipaddr=192.168.0.149
serverip=192.168.0.142
netmask=255.255.255.0
hostname=XNRG
target="XNRG"
nfsserverip=192.168.0.142
rootpath="/var/lib/nfs-Rootfs"
nfsver=3
kernel_addr_r=0x82000000
fdt_addr_r=0x88000000
addipdyn=setenv bootargs ${bootargs} ip=dhcp
addipsta=setenv bootargs ${bootargs} ip=${ipaddr}:${serverip}:${netmask}:off panic=1
load_tftp_kernel=echo Loading kernel from TFTP... tftp ${kernel_addr_r} ${target}/ulimage
load_tftp_dtb=echo Loading DTB from TFTP... tftp ${fdt_addr_r} ${target}/am335x-xnrg.dtb
load_tftp_kernel_dtb;if run load_tftp_kernel; then run load_tftp_dtb ; fi
addcon=setenv bootargs ${bootargs} console=tty0 console=${console}
addip;if test -n ${ipdyn};then run addipdyn; else run addipsta;fi
addnfs=setenv bootargs root=/dev/nfs rw nfsroot=${nfsserverip}:${rootpath},nfsvers=${nfsver} nfsrootdebug rooldelay=4
addblkpart=setenv bootargs ${bootargs} blkdevparts=mmcblk1:128k(MBR),128k(MLO),128k(MLO.b1),128k(MLO.b2),4352k(U-boot),128k(U-boot.env),128k(XNRG.env),27m(Kernel),224m(rootfs),4352k(U-boot.b),128k(U-boot.env.b),128k(XNRG.env.b),27m(Kernel.b),224m(rootfs.b),-(archive)
boot_net=echo Booting from network ...; run load_tftp_kernel_dtb; run addnfs addip addblkpart; bootm ${kernel_addr_r} - ${fdt_addr_r}
```

Una volta che il kernel è partito possiamo vedere la bootcmd

Engineering Specification - confidential

```
host $ cat /proc/cmdline
root=/dev/nfs rw nfsroot=192.168.0.142:/var/lib/nfs-XNRG-rootfs,nfsvers=3 nfsrootdebug roothdelay=4
ip=192.168.0.149:192.168.0.142::255.255.255.0:XNRG:eth0:off panic=1 vt.cur_default=1 vt.global_cursor_default=0
console=ttyS0,115200n8
blkdevparts=mmcblk1:512k(MLO),4608k(uboot),512k(uboot.env),4608k(uboot.b),512k(uboot.env.b),512k(XNRG.env),10m(kernel),512k(dtbo),10m(kernel.b),512k(dtbo.b),512m(rootfs),-(archive)
```

55.9 Micro SD recovery (BBB)

Se vogliamo rifare il boot della BBB da micro SD (perché magari abbiamo corrotto la flash), occorre innanzitutto preparare la micro SD con l'immagine Debian disponibile in

<https://beagleboard.org/latest-images>

Scaricare quindi la Debian image raccomandata. E' un file del tipo

bone-debian-8.4-lxqt-4gb-armhf-2016-05-13-4gb.img.xz

Decomprimere, ottenendo il file .img

Inserire la micro SD nel PC linux, controllare il nodo (sia p.es. sdb) e smontarla

```
host $ sudo umount /dev/sdb
```

Copiare il contenuto

```
host $ sudo dd if=bone-debian-8.4-lxqt-4gb-armhf-2016-05-13-4gb.img of=/dev/sdb bs=1M
```

Inserire la SD card nella BBB, collegare alimentazione e cavo console. Accendere la BBB tenendo premuto il tasto per il boot da SD card. Premere la barra SPACE per arrestare il boot del kernel. Settare le variabili di ambiente ipaddr, serverip e aggiornare u-boot.

55.10 Install Qt Creator (standalone)

Da www.qt.io/download-open-source scaricare e copiare in /home/Download

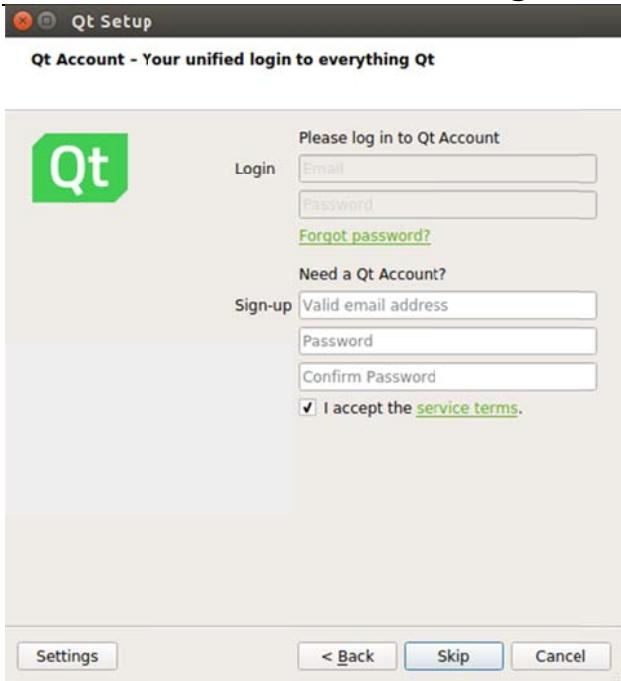
qt-unified-linux-x64-x.x.x-x-online.run

Attualmente la versione corrente è 2.0.3-1

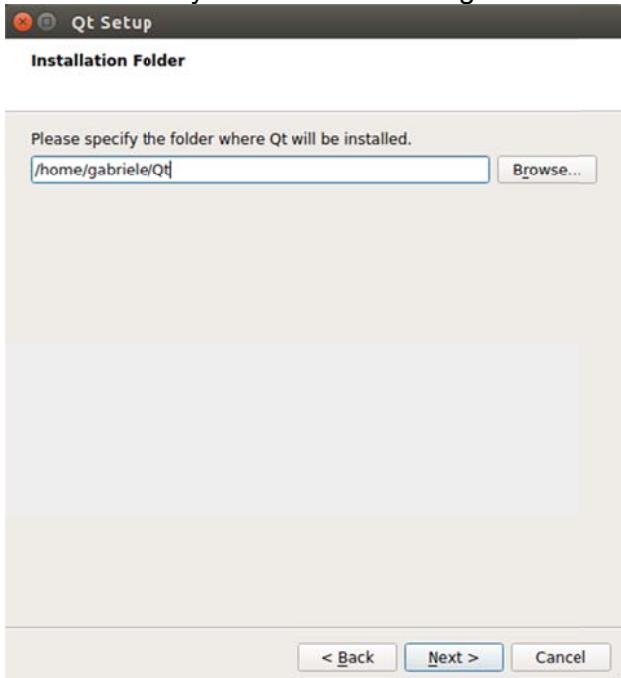
- 6) Rendere eseguibile e lanciare l'installazione

```
host $ chmod u+x qt-unified-linux-x64-2.0.3-1-online.run
host $ ./qt-unified-linux-x64-2.0.3-1-online.run
```

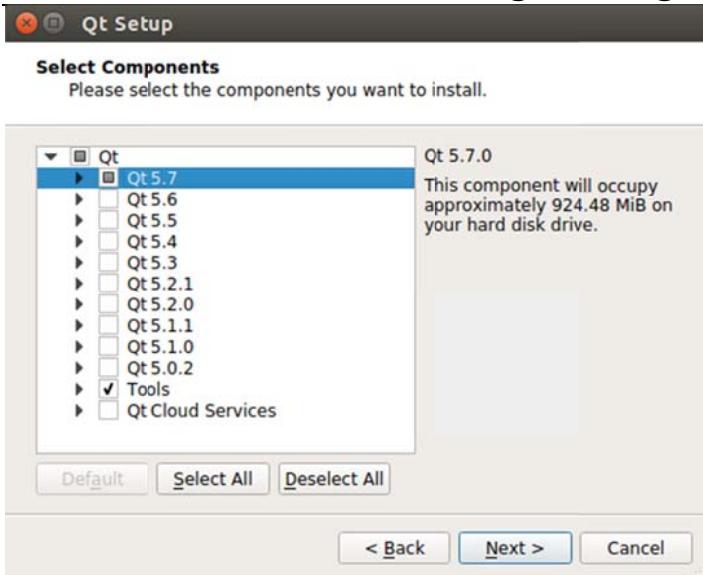
Al momento di definire un account, premere Skip



Come directory di installazione scegliere la cartella di default <qt>=/home/<user>/qt



Selezionare componenti



Procedere fino a installazione completa.

55.11 Compile qmlrun for host using QtCreator

qmlrun è il nome dell'applicazione principale, sia su KX che su XNRG.

Sul KX, che usa libreria Qt4.8, un qws server lancia l'applicazione.

Su XNRG, che usa Qt5, una nuova classe di base svolge le funzioni del qws server.

A differenza del KX, i file della repository del progetto XNRG (xnrg-dev) permettono di compilare sia per host che per ARM.

Compilare per host permette di sviluppare direttamente sul PC, e quindi più velocemente.

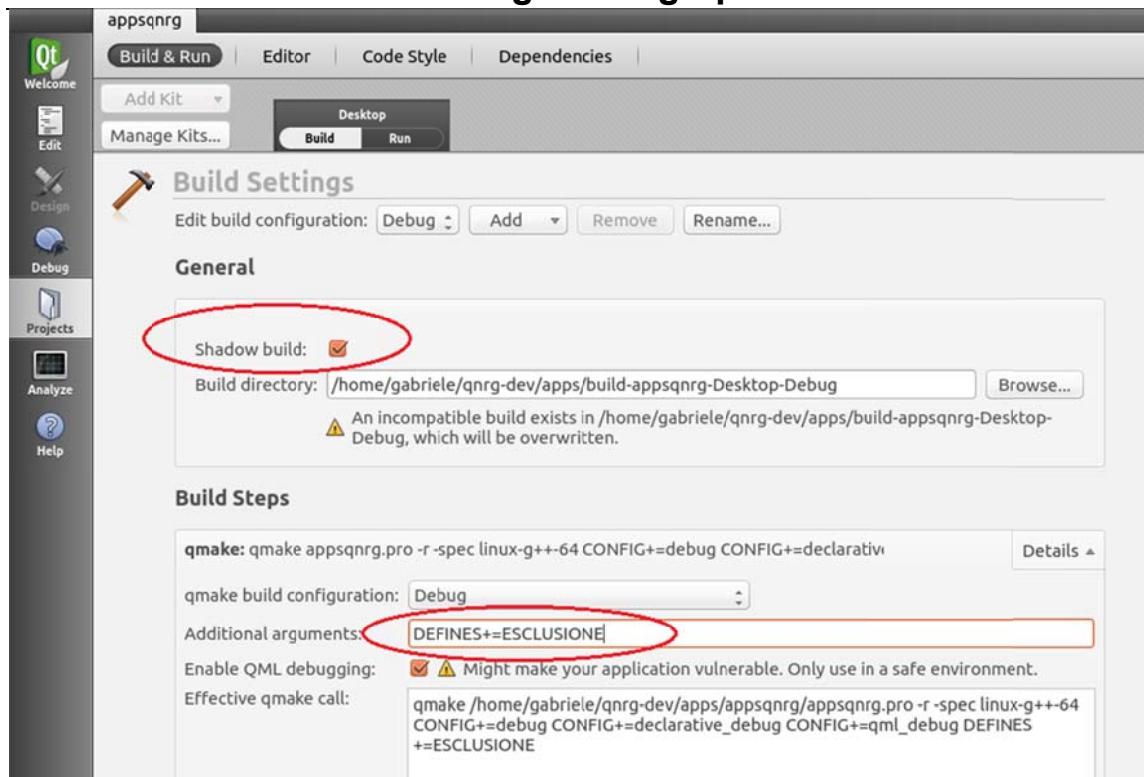
La compilazione e il run su host utilizza QtCreator.

i Makefiles creati da QtCreator per l'host non si mescoleranno con i Makefile per l'ARM, che invece stanno nelle stesse cartelle dei sorgenti.

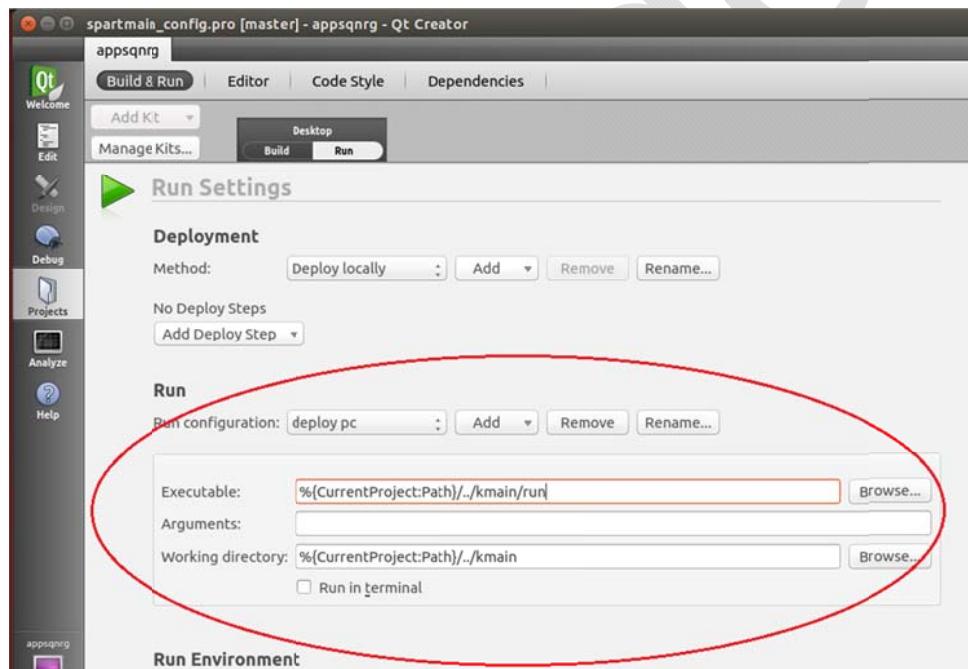
Procedura:

- 1) In *Project>Build>General* abilitare il setting Shadow build del QtCreator (vedi figura)
- 2) In *Project>Build>BuildSteps>qmake* aggiungere in Additional arguments
“**DEFINES+=ESCLUSIONE**” (vedi figura)

Engineering Specification - confidential



- 3) In Project>Run Add>Custom configuration e rinominarla “deploy pc”
- 4) Nel campo Executable usare “%{CurrentProject:Path}/..kmain/run”
- 5) Nel campo Working directory usare “%{CurrentProject:Path}/..kmain”



- 6) Build
- 7) Run

Il Run aggiorna ogni volta il file
/xnrg-dev/app/appsnrg/appsnrg.pro.user

56 Applicazioni Desktop Varie

56.1 Utilizzare GIMP

GIMP (GNU Image Manipulation Program) è un editor di immagini open cross-platform (<https://www.gimp.org/>)

Vediamo come si può facilmente prendere una immagine BMP (*xnrgsplash_1024x600.png*) e dimezzare la componente Green.



Dopo avere installato GIMP

- 1) Aprire file *xnrgsplash_1024x600.png*
- 2) Export as *xnrgsplash_1024x600.bmp*
Selezionare Advanced Options >16-bit R5G6B5
- 3) Selezionare Image>Mode>RGB
- 4) Aprire pannello Colors>Components>ChannelMixer..
Selezionare Green come Output Channel
Impostare valore da 100% a 50%
- 5) Export as *xnrgsplash_1024x600.png*

Il risultato è *xnrgsplash_1024x600_greenAt50.png*



E' possibile lavorare con batch script GIMP su uno o molti file immagine contemporaneamente.

Esempio su un file (simple)

Per applicare una semplice maschera sul file *foo.png*

Entriamo nella cartella */home/<user>/gimp-2.8/scripts*

Copiamo all'interno il file da modificare *foo.png*

Creiamo un file con estensione.scm contenente le seguenti definizioni

```
(define (simple-unsharp-mask filename
    radius
    amount
    threshold)
  (let* ((image (car (gimp-file-load RUN-NONINTERACTIVE filename filename)))
        (drawable (car (gimp-image-get-active-layer image))))
    (plug-in-unsharp-mask RUN-NONINTERACTIVE image drawable radius amount threshold)
    (gimp-file-save RUN-NONINTERACTIVE image drawable filename filename)
    (gimp-image-delete image)))
```

Lanciamo il seguente comando

```
host $ gimp -i -b '(simple-unsharp-mask "foo.png" 5.0 0.5 0) -b '(gimp-quit 0)'
```

In questo caso l'immagine trasformata viene sovrascritta sul file originale.

Esempio su molti (batch)

Per eseguire la stessa cosa su tutti i files .png presenti nella cartella

Creiamo un file con estensione.scm contenente le seguenti definizioni

```
(define (batch-unsharp-mask pattern
    radius
    amount
    threshold)
  (let* ((filelist (cadr (file-glob pattern 1))))
    (while (not (null? filelist))
      (let* ((filename (car filelist))
            (image (car (gimp-file-load RUN-NONINTERACTIVE filename filename))))
```

Engineering Specification - confidential

```
(drawable (car (gimp-image-get-active-layer image))))
(plug-in-unsharp-mask RUN-NONINTERACTIVE image drawable radius amount threshold)
(gimp-file-save RUN-NONINTERACTIVE image drawable filename filename)
(gimp-image-delete image))
(set! filelist (cdr filelist))))
```

Lanciamo il seguente comando

```
host $ gimp -i -b '(batch-unsharp-mask "*.png" 5.0 0.5 0)' -b '(gimp-quit 0)'
```

Le funzioni a disposizione sono molte e presenti in plug-ins.

Per esempio vogliamo installare alcuni plugin molto usati.

```
host $ sudo apt-get install gimp-data-extras gimp-draw gimp-resynthesizer gimp-plugin-registry gimp-texturiz
```

Per cambiare la saturazione del verde dovremo utilizzare la sequente macro

```
plug-in-colors-channel-mixer
```

Che prende 13 parametri

Ma non ho apito bene come si usa!!!

56.2 Utilizzare XFIG

XFIG è un editor di immagini interattivo open cross-platform (<http://mcj.sourceforge.net/>)

Per installarlo sull'host Ubuntu, al solito

```
host $ sudo apt-get install xfig
```

Siccome vi è un bug noto, conviene eseguire anche

```
host $ sudo apt-get update
```

56.3 Utilizzare Dropbox

Dropbox è uno spazio disco remoto (<https://www.dropbox.com/>)

Per installarlo su una macchina a 64-bit eseguire il comando

```
host $ cd ~ && wget -O - "https://www.dropbox.com/download?plat=lnx.x86_64" | tar xzf -
```

Quindi, eseguire il daemon di Dropbox dalla cartella **.dropbox-dist** appena creata.

```
host $ ~/dropbox-dist/dropboxd
```

Se stai eseguendo Dropbox sul server per la prima volta, ti verrà chiesto di copiare un link e incollarlo in un browser per creare un nuovo account o aggiungere il tuo server a un account esistente.

Per fare in modo che il demone Dropbox parta all'avvio dell'host e resti in background, è necessario copiare in **/etc/rc.local**

```
exec "/home/gabriele/.dropbox-dist/dropbox-Lnx.x86_64-21.4.25/dropboxd"
```

Un altro modo, più semplice, è installare **nautilus-dropbox**

```
host $ sudo apt-get install nautilus-dropbox
```

e poi dare il comando (una sola volta)

```
host $ dropbox autostart y
```

Da questo momento in poi la cartella locale di Dropbox è sincronizzata con la cartella su host remoto.

56.4 Utilizzare Python3

Python3 è un linguaggio di scripting evoluto

Installazione

```
host $ sudo apt-get install python3
```

Verifica della versione

```
host $ python3 -V
```

56.5 Utilizzare Tensor Flow

TensorFlow è un software open source multipiattaforma per la manipolazione di array multidimensionali

```
TODo
```

56.6 Utilizzare Ubertooth

Ubertooth (<https://github.com/greatscottgadgets/ubertooth>) è un progetto open source HW/SW che permette di monitorare messaggi di connessioni Bluetooth di tipo

- BLE (Bluetooth Smart)
- BR (Basic Rate) Bluetooth Classic

L'HW si chiama Ubertooth One, mentre il network monitor per l'analisi dei pacchetti è wireshark.

Facciamo un clone del progetto

```
host $ git clone git://github.com/greatscottgadgets/ubertooth.git
```

```
host $ git
```

56.6.1 Installare wireshark

Eseguire

```
host $ sudo apt-get install wireshark
```

Dare le corrette autorizzazioni all'utente corrente per accedere alle interfacce di rete

```
$ sudo dpkg-reconfigure wireshark-common
```

Alla domanda: "Permettere ai non-superuser di catturare i pacchetti?" rispondere <Si>.

Assegnare l'utente corrente al gruppo wireshark:

```
$ sudo usermod -a -G wireshark $USER
```

Riavviare il sistema

Lanciare il software

```
$ wireshark
```

56.6.2 Flashare il firmware in Ubertooth One

Instructions for flashing the firmware can be found [on the corresponding Wiki page](<https://github.com/greatscottgadgets/ubertooth/wiki/Firmware>).

56.6.3 Ricompilare libbrbb

Instructions for building libbrbb can be found [on the corresponding Wiki page](<https://github.com/greatscottgadgets/ubertooth/wiki/Build-Guide>).

57 Debugging tips

Specialmente le prime volte che un sistema esegue il boot possono insorgere problemi. Qui ho elencato alcuni di essi e le soluzioni trovate.

Sintomo

Il sistema parte, ma dopo 10 minuti l'interfaccia grafica si spegne

Soluzione

Disabilitare da U-Boot la funzione di spegnimento display che il kernel implementa di default dopo 600 secondi, aggiungendo consoleblank=0 tra i parametri della console

```
addcon=setenv bootargs ${bootargs} vt.cur_default=1 vt.global_cursor_default=0 consoleblank=0  
console=${console}
```

Se il problema persiste significa che è una libreria che setta di nuovo il parametro a un valore diverso da zero. Per esempio nel nostro caso era QT

In ogni momento si può leggere il valore del parametro con

```
host $ cat /sys/module/kernel/parameters/consoleblank
```

Sintomo

Il sistema parte, ma l'applicazione no

Soluzione

Il processo che lancia l'applicazione viene lanciato prima di un processo essenziale per l'applicazione stessa. In /etc/init.d vi sono degli script numerati. Aumentare il numero dello script che lancia l'applicazione

Sintomo

Il sistema parte, ma il boot del kernel si arresta sul setup della rete

Soluzione

U-Boot inizializza la rete in DHCP mode, quindi il kernel è in attesa che un server DHCP gli assegna un indirizzo. E' sufficiente disabilitare completamente la rete con ip=off. Si farà in modo che sia il kernel a configurare la rete

Sintomo

Il programma di test che usa il driver *ksampler* dà un errore.

```
target$ test_ksampler /dev/ksampler  
open failed: No such device or address
```

Soluzione

Bisogna verificare che il major e il minor number del nodo utilizzato siano uguali a quelli registrati dal driver. In effetti vediamo che sono diversi

```
target$ ls -l /dev/ksampler
```

Engineering Specification - confidential

```
crw-r--r-- 1 root root 245, 0 Jan 1 01:00 /dev/ksampler
target$ cat /proc/devices | grep ksampler
246 ksampler
```

Cancelliamo il nodo e ricreiamolo con il giuto major number

```
target$ rm /dev/ksampler
target$ mknod /dev/ksampler c 246 0
```

Sintomo

Si verifica un kernel oops exception. Il contenuto dei registri fornisce gli indirizzi delle varie istruzioni. Sarebbe necessario individualre la riga di codice dove il problema si è verificato

Ad esempio, supponiamo di vedere in dmasg la seguente exception

```
Feb 13 12:07:17 kx user.warn kernel: [ 29.619554] irq 70, desc: c0454944, depth: 0, count: 0, unhandled: 0
Feb 13 12:07:17 kx user.warn kernel: [ 29.626049] ->handle_irq(): c0050d08, handle_bad_irq+0x0/0x214
Feb 13 12:07:17 kx user.warn kernel: [ 29.632000] ->irq_data(chip()): c044f3d0, 0xc044f3d0
Feb 13 12:07:17 kx user.warn kernel: [ 29.636896] ->action(): c7a9a400
Feb 13 12:07:17 kx user.warn kernel: [ 29.640118] ->action->handler(): c02546e4, ecap_isr+0x0/0x120
Feb 13 12:07:17 kx user.warn kernel: [ 29.645896] IRQ_NOPROBE set
```

Soluzione

Il problema è originato dalla funzione handle_bad_irq, che si trova da c0050d08+0x0 a c0050d08+0x214.

Per avere l'istruzione esatta, si deve ricorrere a qualche altro tool, come **addr2line**.

Esiste anche il tool **objdump**, che permette di avere il listato assembler del kernel e l'indirizzo assoluto delle istruzioni

```
target$ cd <linux>
target$ arm-arago-linux-gnueabi-objdump -dS vmlinux >> /tmp/kernel.s
```

```
c02546d0: e890a878        ldm    sp, {r3, r4, r5, r6, fp, sp, pc}
c02546d4: c0489ae4        .word   0xc0489ae4
c02546d8: c03b3aa6        .word   0xc03b3aa6
c02546dc: c03b3aaa        .word   0xc03b3aaa
c02546e0: c03be0b0        .word   0xc03be0b0

c02546e4 <ecap_isr>:
c02546e4: e1a0c00d        mov    ip, sp
c02546e8: e92dd8f0        push   {r4, r5, r6, r7, fp, ip, lr, pc}
c02546ec: e24cb004        sub    fp, ip, #4
c02546f0: e59f4100        ldr    r4, [pc, #256] ; c02547f8 <ecap_isr+0x114>
c02546f4: e1a06001        mov    r6, r1
c02546f8: e5940004        ldr    r0, [r4, #4]
c02546fc: ebfcee8f        bl    c0190140 <__gpio_get_value>
c0254700: e596301c        ldr    r3, [r6, #28]
c0254704: e1d352be        ldrh   r5, [r3, #46] ; 0x2e
c0254708: e3550000        cmp    r5, #0
c025470c: 0a000037        beq    c02547f0 <ecap_isr+0x10c>
c0254710: e594c124        ldr    ip, [r4, #292] ; 0x124
```

Note: il comando objdump si può usare anche con altri eseguibili, p.es.

```
$ objdump -d /lib32/libc.so.6 | grep system
```

l'opzione -d traduce solamente in assembler, mentre -dS interpone anche il codice sorgente

Sintomo

Durante il funzionamento del programma KX si verifica, saltuariamente, un kernel OOPS che determina il crash del sistema.

```
[ 21.831919] Bad mode in data abort handler detected
[ 21.836834] Internal error: Oops - bad mode: 0 [#1] PREEMPT
[ 21.842399] Modules linked in: mic3291(O) hrecg(O) kxsensors(O) g_X mpl3115a2(O) ad768x(O)
kxpowers(O)
[ 21.852105] CPU: 0 Tainted: G O (3.3.0-kx-006 #2)
```

Engineering Specification - confidential

```
[ 21.858028] PC is at 0xc00000240
[ 21.861190] LR is at __clk_disable+0x78/0x84
[ 21.865467] pc : [<c0000240>] lr : [<c0012494>] psr: 800000d1
[ 21.865485] sp : c7233e78 ip : c7233e48 fp : c7233edc
[ 21.876927] r10: c6a3d118 r9 : c03354d4 r8 : c7a4e008
[ 21.882143] r7 : c6a3d100 r6 : 00343b30 r5 : 00000000 r4 : 00000000
[ 21.888659] r3 : 00000000 r2 : ffffffff r1 : 00000001 r0 : 00000000
[ 21.895177] Flags: Nzcv IRQs off FIQs off Mode FIQ_32 ISA ARM Segment user
[ 21.902476] Control: 0005317f Table: c7060000 DAC: 00000015
[ 21.908213] Process ergotest (pid: 1317, stack limit = 0xc7232270)
[ 21.914384] Stack: (0xc7233e78 to 0xc7234000)
[ 21.918740] 3e60: 00000000 00000001
[ 21.926920] 3e80: ffffffff 00000000 00000000 00343b30 c6a3d100 c7a4e008 c03354d4
[ 21.935101] 3ea0: c6a3d118 c7233edc c7233e48 c7233e78 c0012494 c0000240 800000d1 ffffffff
[ 21.943280] 3ec0: 20000093 00000002 c7233f70 00000002 c7233efc c7233ee0 c0254fb0 c0254e2c
[ 21.951459] 3ee0: c7233efc 00000000 c005eb48 c7a4f140 c7233f0c c7233f00 c01ca784 c0254f6c
[ 21.959639] 3f00: c7233f3c c7233f10 c00d9b08 c01ca774 c7233f70 00000002 c721b280 b6eec024
[ 21.967817] 3f20: c7233f70 00000002 c7232000 00000000 c7233f6c c7233f40 c00884a8 c00d9a08
[ 21.975995] 3f40: c721b280 c69d3000 c7233f94 c721b280 b6eec024 00000000 00000000 00000002
[ 21.984172] 3f60: c7233fa4 c7233f70 c008872c c00883fc 00000000 00000000 00000000 00000000
[ 21.992350] 3f80: c00095c4 00000019 b6efdff8 00000000 00000004 c00095c4 00000000 c7233fa8
[ 22.000528] 3fa0: c0009440 c00886f8 00000019 b6efdff8 00000019 b6eec024 00000002 00000000
[ 22.008705] 3fc0: 00000019 b6efdff8 00000000 00000004 00000000 000001e6 00000ec5 00000001
[ 22.016884] 3fe0: 00012046 be961e90 b6eea540 48507dec 60000010 00000019 00000000 00000000
[ 22.025035] Backtrace:
[ 22.027539] [<c0254e1c>] (set_pwmgas+0x0/0x140) from [<c0254fb0>] (pwmgas_store+0x54/0x68)
[ 22.035791] r6:00000002 r5:c7233f70 r4:00000002 r3:20000093
[ 22.041521] [<c0254f5c>] (pwmgas_store+0x0/0x68) from [<c01ca784>] (dev_attr_store+0x20/0x2c)
[ 22.050028] r4:c7a4f140
[ 22.052603] [<c01ca764>] (dev_attr_store+0x0/0x2c) from [<c00d9b08>] (sysfs_write_file+0x110/0x148)
[ 22.061661] [<c00d99f8>] (sysfs_write_file+0x0/0x148) from [<c00884a8>] (vfs_write+0xbc/0x148)
[ 22.070276] [<c00883ec>] (vfs_write+0x0/0x148) from [<c008872c>] (sys_write+0x44/0x70)
[ 22.078177] r8:00000002 r7:00000000 r6:00000000 r5:b6eec024 r4:c721b280
[ 22.084930] [<c00886e8>] (sys_write+0x0/0x70) from [<c0009440>] (ret_fast_syscall+0x0/0x2c)
[ 22.093264] r8:c00095c4 r7:00000004 r6:00000000 r5:b6efdff8 r4:00000019
[ 22.100003] Code: 00000005 676e6f6c 20706d6a 73756163 (75207365)
```

In generale si deve andare a vedere la sezione Backtrace, dove sono elencate le ultime funzioni chiamate, in ordine inverso a quello di esecuzione

`sysfs_write_file -> dev_attr_store -> pwmgas_store -> set_pwmgas`

Generalmente il registro **pc** contiene l'indirizzo dell'istruzione che ha determinato l'OOPS, mentre **lr** contiene l'istruzione. Ma in questo caso specifico il contenuto di pc e lr non porta a nulla perché qui siamo in presenza di un DATA ABORT, come si evince dalla prima riga

```
[ 21.831919] Bad mode in data abort handler detected
```

Un data abort exception è una eccezione che viene segnalata dalla MMU all'ARM, in quanto la MMU ha abortito un accesso alla memoria di sistema.

Da una ricerca su internet si viene a sapere che questo può capitare quando si gestiscono le interrupt di tipo FIQ. In effetti, dalla seguente riga si evince che il programma era in contesto FIQ

```
[ 21.895177] Flags: Nzcv IRQs off FIQs off Mode FIQ_32 ISA ARM Segment user
```

L'accesso alla memoria in questione è quello che riguarda uno spinlock (*gearlock*). Questo spinlock serviva in quanto la valvola pwm può essere aperta sia dall'applicazione sia dal driver turbina. La funzione incriminata era *kxleevalve_notify_semiperiod* che accedeva alle variabili count e gear, protette dallo spinlock.

Soluzione

La soluzione è consistita nel rimuovere il lock e rendere count e gear delle variabili atomiche.
La modifica ha riguardato i due file

```
<linux>/drivers/platform/X/kx/am18x_ecap_pwm_leevalve_config.c  
<linux>/drivers/platform/X/kx/am18x_pwm_leevalve.c
```

Verifica

Per verificare il funzionamento della modifica apportata è stato sufficiente stressare il sistema effettuando un test DMC in aria e introducendo da console accessi concorrenti in scrittura alla variabile interna gear, in questo modo

```
$ target cd /sys/class/pwm/ecap.2/kx_lee_valve_pwm  
$ target while true ; do echo 8 > gear ; done
```

Se il sistema non va in crash allora la modifica ha avuto successo.

Riferimenti

<http://linux-arm-kernel.infradead.narkive.com/h7NnVRIr/memory-access-exception-in-fiq-an-infrequent-random-occurrence>

<http://www.spinics.net/lists/arm-kernel/msg325250.html>

Sintomo

Eseguo lo script di build della distribuzione XNRG come root (con sudo), ma l'esecuzione viene interrotta col messaggio

```
..  
/home/gabriele/xnrg-dev/u-boot/src/u-boot/tools/mkimage  
This script cannot run as root
```

Soluzione

Il build della distribuzione XNRG prevede come primo step la ricompilazione di U-Boot. Qualche giorno prima avevo ricompilato U-Boot singolarmente come root

```
host $ sudo ./X-build.sh --build <conf>
```

Non avrei dovuto farlo.

La soluzione è fare il distclean di U-Boot come root, in maniera da ripulire tutto.

Sintomo

Su XNRG, dopo un certo numero di ore, l'applicazione va in crash

Soluzione

Si trattava di un *out of memory* dovuto al fatto che ogni 15 sec veniva lanciato un processo idle_sampler che, per qualche motivo, consumava un po' di memoria. La shell di linux stampava la seguente eccezione. Si nota che il processo *qml/run* viene killato dal SO.

```
[34924.443709] QSGRenderThread invoked oom-killer: gfp_mask=0x24000c2, order=0, oom_score_adj=0  
[34924.453810] CPU: 0 PID: 350 Comm: QSGRenderThread Tainted: G          O  4.4.39-ti-r76-X #4  
[34924.463928] Hardware name: Generic AM33XX (Flattened Device Tree)  
[34924.470513] [<c0012611>] (unwind_backtrace) from [<c00108ab>] (show_stack+0xb/0xc)  
[34924.478823] [<c00108ab>] (show_stack) from [<c0095685>] (dump_header+0x39/0x14)  
[34924.486830] [<c0095685>] (dump_header) from [<c0070aa5>] (oom_kill_process+0x251/0x348)  
[34924.495363] [<c0070aa5>] (oom_kill_process) from [<c0070df5>] (out_of_memory+0x209/0x234)  
[34924.508644] [<c0070df5>] (out_of_memory) from [<c0074443>] (_alloc_pages_nodemask+0x627/0x654)  
[34924.518898] [<c0074443>] (_alloc_pages_nodemask) from [<bf839865>] (NewAllocPagesLinuxMemArea+0xac/0x144 [pvrsrvkm])  
[34924.530482] [<bf839865>] (NewAllocPagesLinuxMemArea [pvrsrvkm]) from [<bf83ae21>] (OSAllocPages _Impl+0x44/0x76 [pvrsrvkm])  
[34924.542779] [<bf83ae21>] (OSAllocPages _Impl [pvrsrvkm]) from [<bf83582b>] (BM_ImportMemory+0x192/0x2a0 [pvrsrvkm])  
[34924.554459] [<bf83582b>] (BM_ImportMemory [pvrsrvkm]) from [<bf8383e7>] (RA_Alloc+0x7e/0x178 [pvrsrvkm])  
[34924.564719] [<bf8383e7>] (RA_Alloc [pvrsrvkm]) from [<bf835a3f>] (BM_Alloc+0x106/0x208 [pvrsrvkm])  
[34924.582052] [<bf835a3f>] (BM_Alloc [pvrsrvkm]) from [<bf838803>] (AllocDeviceMem+0x66/0xc4 [pvrsrvkm])  
[34924.592364] [<bf838803>] (AllocDeviceMem [pvrsrvkm]) from [<bf838db9>] (_PVRSRVAllocDeviceMemKM+0x8c/0x134 [pvrsrvkm])  
[34924.604174] [<bf838db9>] (_PVRSRVAllocDeviceMemKM [pvrsrvkm]) from [<bf83fdb5>] (PVRSRVAllocDeviceMemBW+0x110/0x1e4 [pvrsrvkm])  
[34924.616699] [<bf83fdb5>] (PVRSRVAllocDeviceMemBW [pvrsrvkm]) from [<bf840b5d>] (BridgedDispatchKM+0x8c/0xc4 [pvrsrvkm])  
[34924.628624] [<bf840b5d>] (BridgedDispatchKM [pvrsrvkm]) from [<bf83fdb5>] (PVRSRV_BridgeDispatchKM+0x8a/0x12e [pvrsrvkm])  
[34924.641382] [<bf83fdb5>] (PVRSRV_BridgeDispatchKM [pvrsrvkm]) from [<c00a20d9>] (do_vfs_ioctl+0x2e1/0x440)  
[34924.652715] [<c00a20d9>] (do_vfs_ioctl) from [<c00a225b>] (SyS_ioctl+0x23/0x40)  
[34924.660499] [<c00a225b>] (SyS_ioctl) from [<c000e541>] (ret_fast_syscall+0x1/0x4a)  
[34924.668661] Mem-Info:  
[34924.671204] active_anon:74452 inactive_anon:14 isolated_anon:0
```

Engineering Specification - confidential

```
[34924.671204] active_file:80 inactive_file:773 isolated_file:0
[34924.671204] unevictable:0 dirty:1 writeback:0 unstable:0
[34924.671204] slab_reclaimable:302 slab_unreclaimable:4180
[34924.671204] mapped:42013 shmem:17 pagetables:366 bounce:0
[34924.671204] free:5006 free_pcp:4 free_cma:4835
[34924.713018] Normal free:22108kB min:2792kB low:3488kB high:4188kB active_anon:297808kB inactive_anon:56kB active_file:128kB inactive_file:1116kB
unevictable:0kB isolated(anon):0kB isolated(file):0kB present:524288kB managed:512848kB mlocked:0kB dirty:0kB writeback:0kB mapped:166556kB
shmem:68kB slab_reclaimable:1208kB slab_unreclaimable:16720kB kernel_stack:872kB pagetables:1464kB unstable:0kB bounce:0kB free_pcp:88kB
local_pcp:88kB free_cma:20164kB writebackTmp:0kB pages_scanned:1432 all_unreclaimable? no
[34924.761032] lowmem_reserve[], 0 0 0
[34924.764878] Normal: 1*4kB (U) 2*8kB (UC) 10*16kB (C) 13*32kB (UC) 11*64kB (C) 5*128kB (UC) 4*256kB (C) 6*512kB (UC) 2*1024kB (C) 0*2048kB
1*4096kB (C) 1*8192kB (C) = 20372kB
[34924.788581] 75 total pagecache pages
[34924.793769] 131072 pages RAM
[34924.798976] 0 pages HighMem/MovableOnly
[34924.804628] 2860 pages reserved
[34924.808211] 6144 pages cma reserved
[34924.814181] [pid] uid tgid total_vm rss nr_ptes nr_pmds swapents oom_score_adj name
[34924.824841] [ 175] 0 175 2541 22 4 0 0 0 alarm
[34924.837478] [ 235] 0 235 463 23 4 0 0 0 syslogd
[34924.850882] [ 237] 0 237 979 573 4 0 0 0 klogd
[34924.863505] [ 252] 1000 252 725 34 4 0 0 0 dbus-daemon
[34924.876552] [ 258] 0 258 550 34 4 0 0 0 rpcbind
[34924.889030] [ 274] 0 274 1286 69 5 0 0 -1000 sshd
[34924.901319] [ 279] 0 279 500 19 3 0 0 iwrapd
[34924.910620] [ 283] 0 283 429 16 5 0 0 crond
[34924.919937] [ 302] 0 302 533 25 5 0 0 hostgw
[34924.929105] [ 312] 0 312 160736 128499 303 0 0 0 xnrgfw
[34924.938442] [ 315] 0 315 723 47 4 0 0 0 login
[34924.947534] [ 333] 0 333 641 26 4 0 0 0 tasker
[34924.957374] [ 335] 0 335 682 29 5 0 0 0 cuml
[34924.966756] [ 336] 0 336 724 30 4 0 0 0 cuml
[34924.975924] [ 420] 0 420 864 74 4 0 0 0 sh
[34924.984971] Out of memory: Kill process 312 (xnrgfw) score 973 or sacrifice child
[34924.993054] Killed process 333 (tasker) total-vm:2564kB, anon-rss:96kB, file-rss:8kB
[34925.016245] QSGRenderThread invoked oom-killer: gfp_mask=0x24000c2, order=0, oom_score_adj=0
[34925.025414] CPU: 0 PID: 350 Comm: QSGRenderThread Tainted: G O 4.4.39-ti-r76-X #4
[34925.035716] Hardware name: Generic AM33XX (Flattened Device Tree)
[34925.047389] [<c0012611>] (unwind_backtrace) from [<c00108ab>] (show_stack+0xb/0xc)
[34925.055721] [<c00108ab>] (show_stack) from [<c0095685>] (dump_header+0x39/0x14)
[34925.063756] [<c0095685>] (dump_header) from [<c0070aa5>] (oom_kill_process+0x251/0x348)
[34925.072345] [<c0070aa5>] (oom_kill_process) from [<c0070df5>] (out_of_memory+0x209/0x234)
[34925.081194] [<c0070df5>] (out_of_memory) from [<c0074443>] (_alloc_pages_nodemask+0x627/0x654)
[34925.090887] [<c0074443>] (_alloc_pages_nodemask) from [<bf839865>] (NewAllocPagesLinuxMemArea+0xac/0x144 [pvrsrvkm])
[34925.09952] [<bf839865>] (NewAllocPagesLinuxMemArea [pvrsrvkm]) from [<bf83ae21>] (OSAllocPages_Impl+0x44/0x76 [pvrsrvkm])
[34925.122228] [<bf83ae21>] (OSAllocPages_Impl [pvrsrvkm]) from [<bf83582b>] (BM_ImportMemory+0x192/0x2a0 [pvrsrvkm])
[34925.133558] [<bf83582b>] (BM_ImportMemory [pvrsrvkm]) from [<bf8383e7>] (RA_Alloc+0x7e/0x178 [pvrsrvkm])
[34925.144077] [<bf8383e7>] (RA_Alloc [pvrsrvkm]) from [<bf835a3f>] (BM_Alloc+0x106/0x208 [pvrsrvkm])
[34925.161135] [<bf835a3f>] (BM_Alloc [pvrsrvkm]) from [<bf838803>] (AllocDeviceMem+0x66/0x4 [pvrsrvkm])
[34925.174729] [<bf838803>] (AllocDeviceMem [pvrsrvkm]) from [<bf838db9>] (PVRSRVAllocDeviceMemKM+0x8c/0x134 [pvrsrvkm])
[34925.189944] [<bf838db9>] (PVRSRVAllocDeviceMemKM [pvrsrvkm]) from [<bf83fdb5>] (PVRSRVAllocDeviceMemBW+0x110/0x1e4 [pvrsrvkm])
[34925.209026] [<bf83fdb5>] (PVRSRVAllocDeviceMemBW [pvrsrvkm]) from [<bf840b5d>] (BridgedDispatchKM+0x8c/0xc4 [pvrsrvkm])
[34925.225610] [<bf840b5d>] (BridgedDispatchKM [pvrsrvkm]) from [<bf83bfd5>] (PVRSRV_BridgeDispatchKM+0x8a/0x12e [pvrsrvkm])
[34925.239005] [<bf83bfd5>] (PVRSRV_BridgeDispatchKM [pvrsrvkm]) from [<c00a20d9>] (do_vfs_ioctl+0x2e1/0x440)
[34925.249587] [<c00a20d9>] (do_vfs_ioctl) from [<c00a225b>] (SyS_ioctl+0x23/0x40)
[34925.257612] [<c00a225b>] (SyS_ioctl) from [<c000e541>] (ret_fast_syscall+0x1/0x4a)
[34925.265648] Mem-Info:
[34925.268138] active_anon:74428 inactive_anon:14 isolated_anon:0
[34925.268138] active_file:78 inactive_file:377 isolated_file:0
[34925.268138] unevictable:0 dirty:0 writeback:0 unstable:0
[34925.268138] slab_reclaimable:302 slab_unreclaimable:4181
[34925.268138] mapped:41700 shmem:17 pagetables:362 bounce:0
[34925.268138] free:5422 free_pcp:19 free_cma:5153
[34925.304716] Normal free:20572kB min:2792kB low:3488kB high:4188kB active_anon:297712kB inactive_anon:56kB active_file:312kB inactive_file:2688kB
unevictable:0kB isolated(anon):0kB isolated(file):0kB present:524288kB managed:512848kB mlocked:0kB dirty:0kB writeback:4kB mapped:167576kB
shmem:68kB slab_reclaimable:1208kB slab_unreclaimable:16724kB kernel_stack:872kB pagetables:1448kB unstable:0kB bounce:0kB free_pcp:40kB
local_pcp:40kB free_cma:19860kB writebackTmp:0kB pages_scanned:1904 all_unreclaimable? no
[34925.362926] lowmem_reserve[], 0 0 0
[34925.366675] Normal: 57*4kB (U) 111*8kB (UMC) 34*16kB (UMC) 16*32kB (UMC) 11*64kB (C) 5*128kB (UC) 4*256kB (C) 6*512kB (UC) 2*1024kB (C)
0*2048kB 1*4096kB (C) 1*8192kB (C) = 21948kB
[34925.384238] 539 total pagecache pages
[34925.388293] 131072 pages RAM
[34925.391459] 0 pages HighMem/MovableOnly
[34925.395777] 2860 pages reserved
[34925.399215] 6144 pages cma reserved
[34925.403513] [pid] uid tgid total_vm rss nr_ptes nr_pmds swapents oom_score_adj name
[34925.413112] [ 175] 0 175 2541 22 4 0 0 0 alarm
[34925.423644] [ 235] 0 235 463 214 4 0 0 0 syslogd
[34925.440669] [ 237] 0 237 979 573 4 0 0 0 klogd
[34925.454448] [ 252] 1000 252 725 34 4 0 0 0 dbus-daemon
[34925.467661] [ 258] 0 258 550 34 4 0 0 0 rpcbind
[34925.481475] [ 274] 0 274 1286 69 5 0 0 -1000 sshd
[34925.497566] [ 279] 0 279 500 19 3 0 0 0 iwrapd
[34925.509479] [ 283] 0 283 429 16 5 0 0 0 crond
[34925.523038] [ 302] 0 302 533 25 5 0 0 0 hostgw
[34925.535953] [ 312] 0 312 160736 128375 303 0 0 0 xnrgfw
[34925.553046] [ 315] 0 315 723 47 4 0 0 0 login
[34925.564399] [ 335] 0 335 682 29 5 0 0 0 cuml
[34925.576892] [ 336] 0 336 724 30 4 0 0 0 cuml
```

```
[34925.588700] [ 420]   0 420 864 74 4 0 0 0 sh
[34925.601592] Out of memory: Kill process 312 (xnrgfw) score 973 or sacrifice child
[34925.613074] Killed process 312 (xnrgfw) total-vm:642944kB, anon-rss:293740kB, file-rss:219700kB
```

Ogni processo ha un *oom_score_adj* visibile con
host \$ cat /proc/<pid>/oom_score_adj

Questo numero serve al SO per decidere quali processi sacrificare nel momento in cui la memoria sta finendo. Per esempio, nel messaggio di cui sopra si nota che il demone *sshd* ha indice -1000, probabilmente perché questo processo è talmente importante da non dovere essere mai sacrificato.

Zombie

58 Links utili

- Mainline kernel sources git repository
<git://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git>
- Mainline u-boot sources git repository
<git://git.denx.de/u-boot.git>
- TI E2E Community on Sitara AM18x
http://e2e.ti.com/support/dsp/sitara_arm174_micropocessors/default.aspx
- Wiki pages on Sitara AM18x
<http://processors.wiki.ti.com/index.php/Category:AM1x>
http://processors.wiki.ti.com/index.php?title=Sitara_Linux_Software_Developer's_Guide&oldid=43627
http://processors.wiki.ti.com/index.php/GSG:_DA8x/OMAP-L1/AM1x_DVEVM_Additional_Procedures
- AM1808 product page
<http://focus.ti.com/docs/prod/folders/print/am1808.html>
- U-Boot documentation
<http://www.denx.de/wiki/U-Boot/WebHome>
- Sourcery G++ LITE tools are available at:
<http://www.codesourcery.com/sgpp/lite/arm/portal/release858>
- TI Code Composer Studio documentation
http://processors.wiki.ti.com/index.php/CCSv5_Getting_Started_Guide
- Qt
http://processors.wiki.ti.com/index.php/Qt_Framework_Demo_Setup
- TFTP
http://processors.wiki.ti.com/index.php/Setting_Up_a_TFTP_Server
- NFS
http://processors.wiki.ti.com/index.php/GSG:_Setting_up_OMAP-L1_Target_File_System
- TI Linux Community for DaVinci Processors (mailing list):
<http://linux.davincidsp.com>
Mailing list archives: <http://www.mail-archive.com/davinci-linux-open-source@linux.davincidsp.com/>
Mailing list archives (text): <http://linux.omap.com/pipermail/davinci-linux-open-source/>
- Linux changes reports:
http://kernelnewbies.org/Linux_2.6
- UBIFS
http://plugcomputer.org/plugwiki/index.php/Enabling_UBIFS
http://processors.wiki.ti.com/index.php/UBIFS_Support
- Eclipse
<http://wiki.eclipse.org/Category:FAQ>

Engineering Specification - confidential

- TI SDK
<http://www.ti.com/lscds/ti/arm/training/OLT312000.page>
- AM35xx related topics
http://processors.wiki.ti.com/index.php/How_to_Flash_Linux_System_from_U-boot
http://processors.wiki.ti.com/index.php/AMSDK_u-boot_User%27s_Guide
http://processors.wiki.ti.com/index.php/AMSDK_Linux_User%27s_Guide
- bzimage structure
<http://lkml.iu.edu/hypermail/linux/kernel/9909.3/0625.html>
- Device tree data structure
http://devicetree.org/Device_Tree_Usage
- Giving linux the boot
http://training.ti.com/system/files/docs/sitara_boot_camp_03_giving_linux_the_boot.pptx
- Raspberry boot process
http://elinux.org/RPi_Software#Overview
- Two stage u-boot design
http://processors.wiki.ti.com/index.php/AM335x_U-Boot_User's_Guide#Two_stage_U-Boot_design
- U-boot image formats
<http://www.denx.de/wiki/view/DULG/UBootImages>
- vmlinuz_le format
https://www.ibm.com/developerworks/community/blogs/mhaque/entry/anatomy_of_the_initrd_and_vmlinuz?lang=en