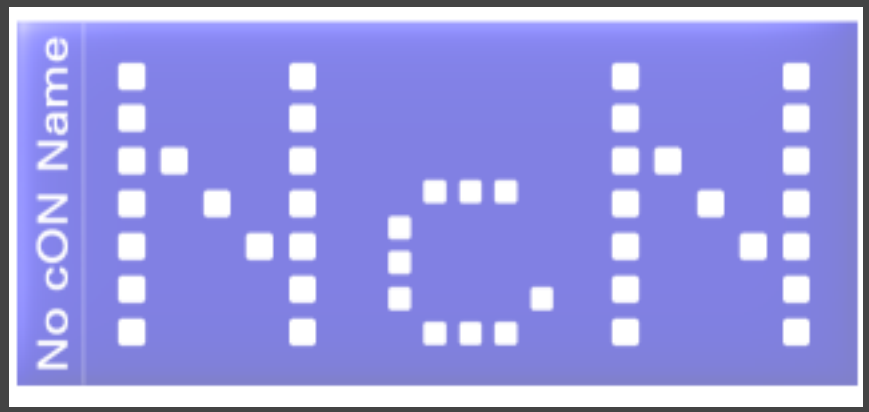
HTC Nand Dumping
for forensics purposes



NCN 2010 - Pau Oliva Fora
<pof@eslack.org>

Table of Contents

Introducción

Quien es HTC:

Fabricante taiwanés de móviles de gama media / alta.
ODM desde 1997 hasta 2007 (i-mate, Qtek, etc...)
Sistemas operativos Windows Mobile y Android (entre otros)

Procesadores (ARM):

- Texas Instruments OMAP 850
- Intel PXA + Qualcomm MSM6250
- Samsung SC2442 + Qualcomm MSM6275
- Qualcomm MSM7200, MSM7201A, MSM7225,...
- Qualcomm MSM8260 (SnapDragon)

Memorias "flash"

NOR:

- Celdas conectadas en paralelo
- permite leer/programar los datos de cada celda individualmente.

Ejemplos: M-Systems DiskOnChip G3/G4/H3

NAND:

- Celdas conectadas en serie (menor tamaño y menor coste)
- se debe leer/programar por página

Ejemplos:

Samsung OneNand

Samsung, Hynix, Micron, Toshiba...

Formato de la NAND

Cuando dumpeamos NAND tenemos que tener en cuenta esta información:

- Lectura / escritura por página
- Borrado por bloque
- Cada bloque se compone de varias páginas
- 1 página = 512 bytes lógicos / 528 bytes físicos
  (16 bytes usados para ECC y bad-block management)
  - byte 517 != 0xFF --> Bad Block

Según como dumpeamos, tendremos que "reconstruir" el dumpeo para eliminar la información de ECC y los bad blocks.

¿Qué información queremos recuperar? I

MSISDN, SIM S/N, IMSI, IMEI

PIM Data: Agenda de contactos, calendario, lista TO-DO, reminders, tareas, voice memos...

Listado de últimas llamadas (entrantes, salientes, perdidas) y fecha/hora de éstas.

Mensajes SMS y MMS (guardados en telefono y/o en SIM)

¿Qué información queremos recuperar? II

Aplication data: emails, ficheros adjuntos, browser history, twitts, docs, PDFs, etc...

Fotos y videos (tomados con la camara del terminal)

Ultima posición GPS conocida, coarse data

Listado de WLANs (SSID, BSSID, encryption key), donde se ha asociado el movil...

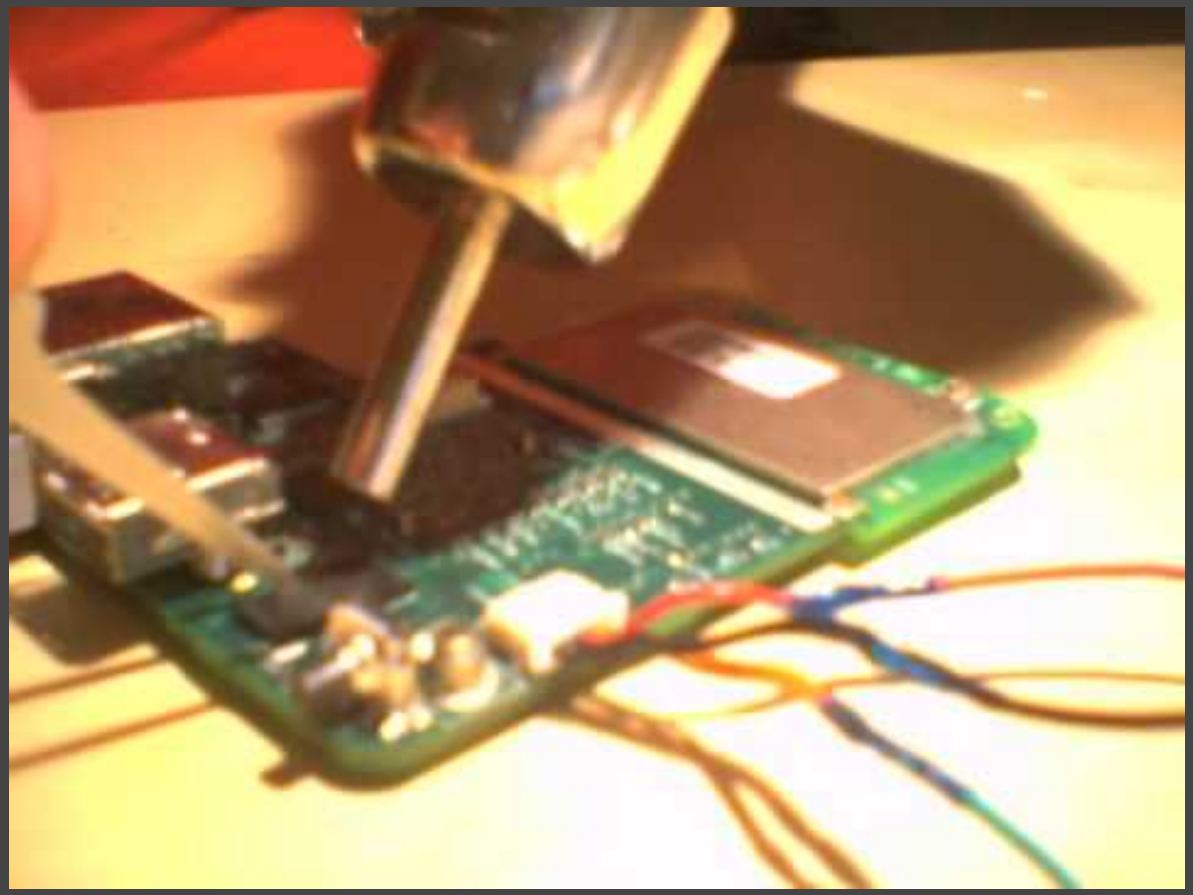
Métodos de dumpeo por hardware

Desoldar BGA

Desoldar BGA (I)

Desoldar BGA (II)

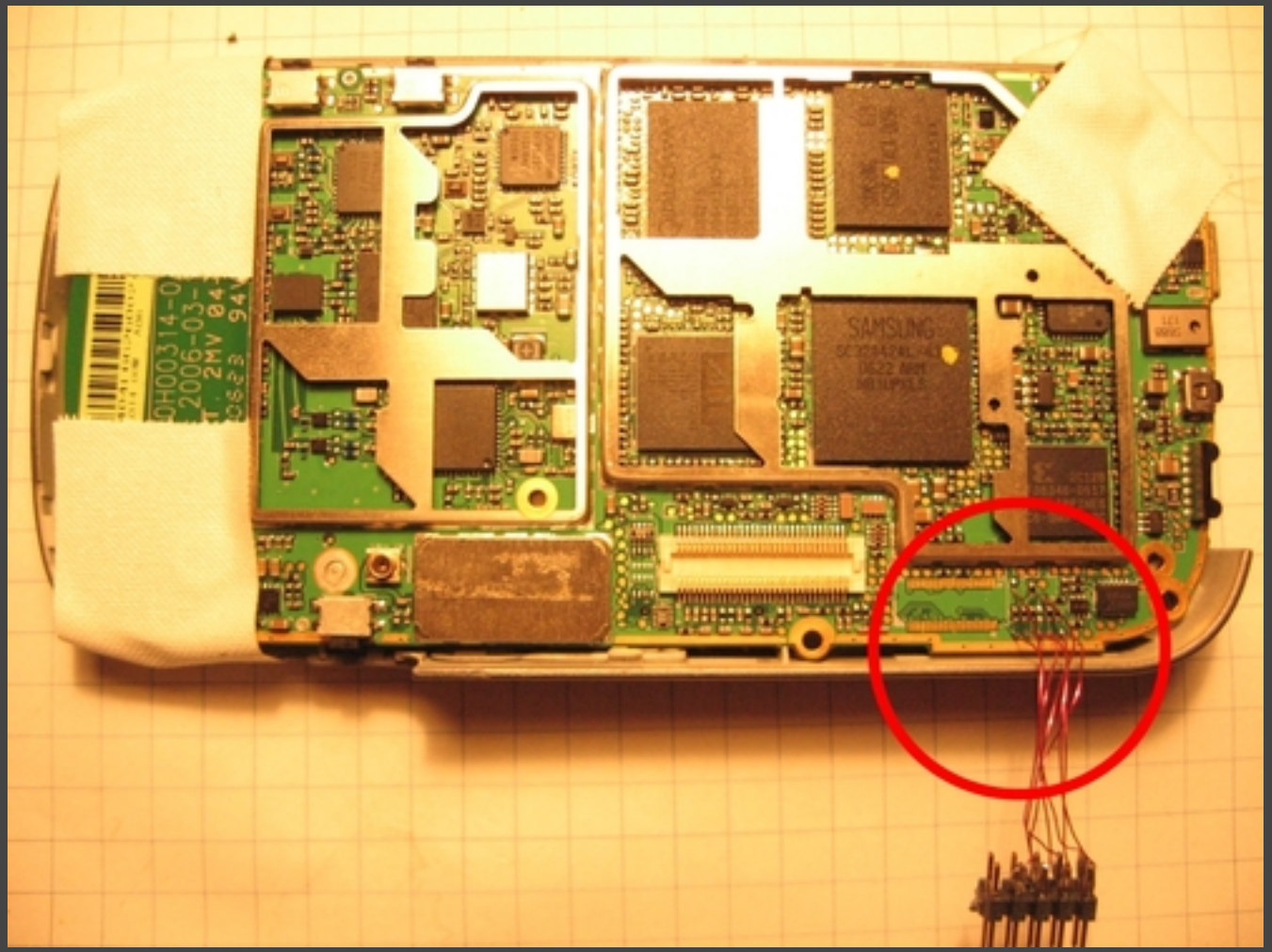
Método "casero"



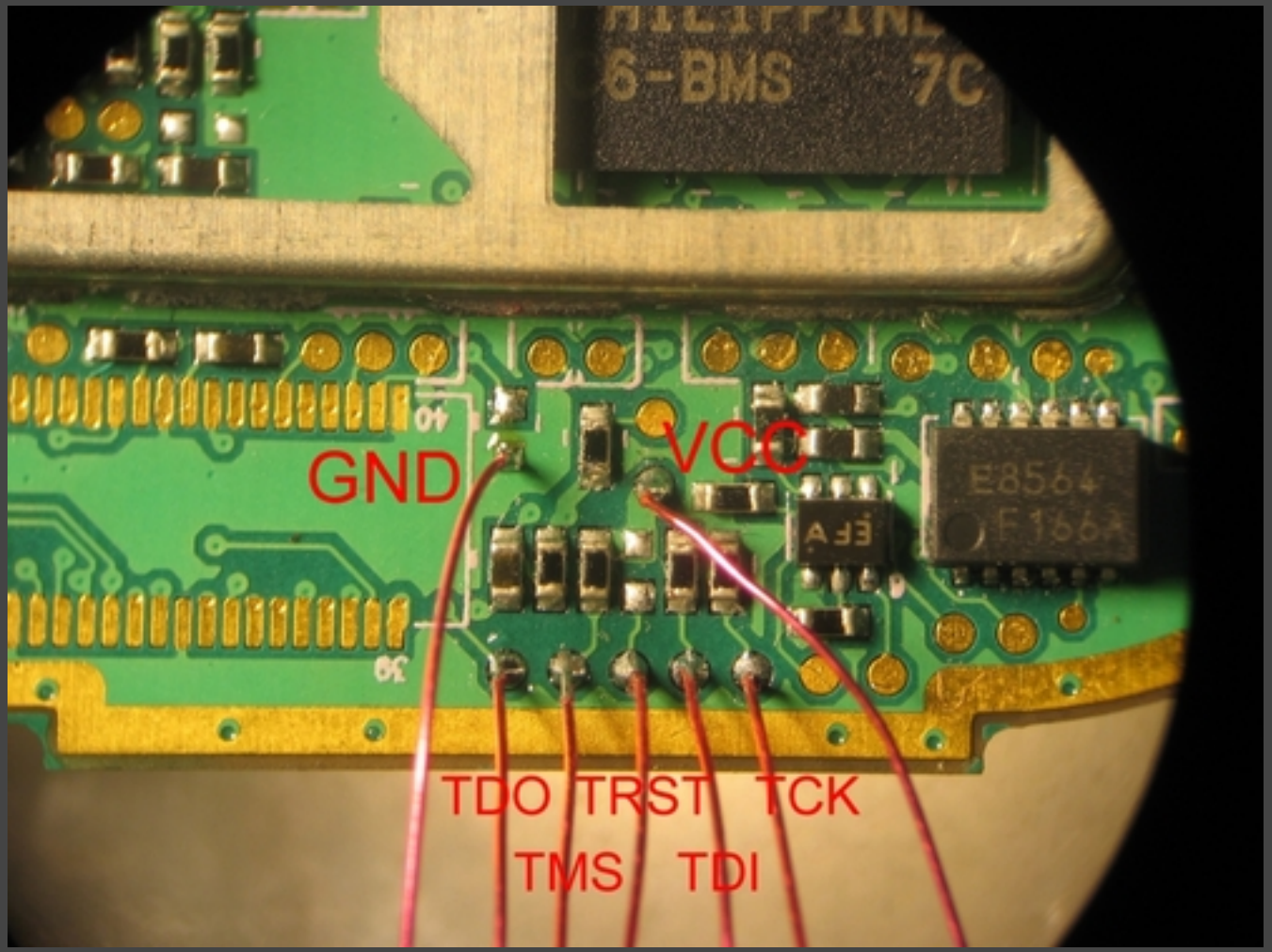
Métodos de dumpeo por hardware

JTAG

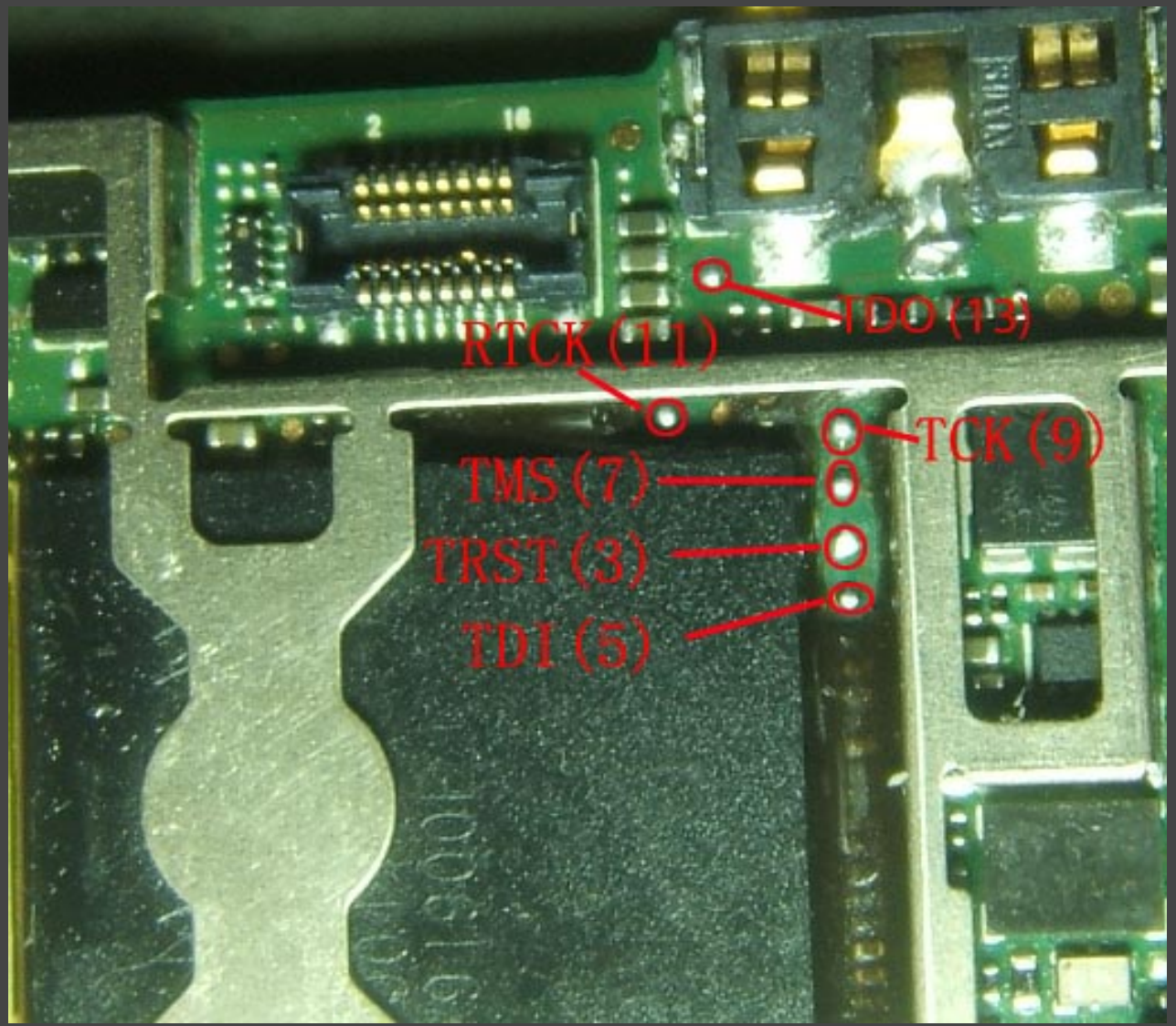
JTAG (I)



JTAG (II)



JTAG (III)



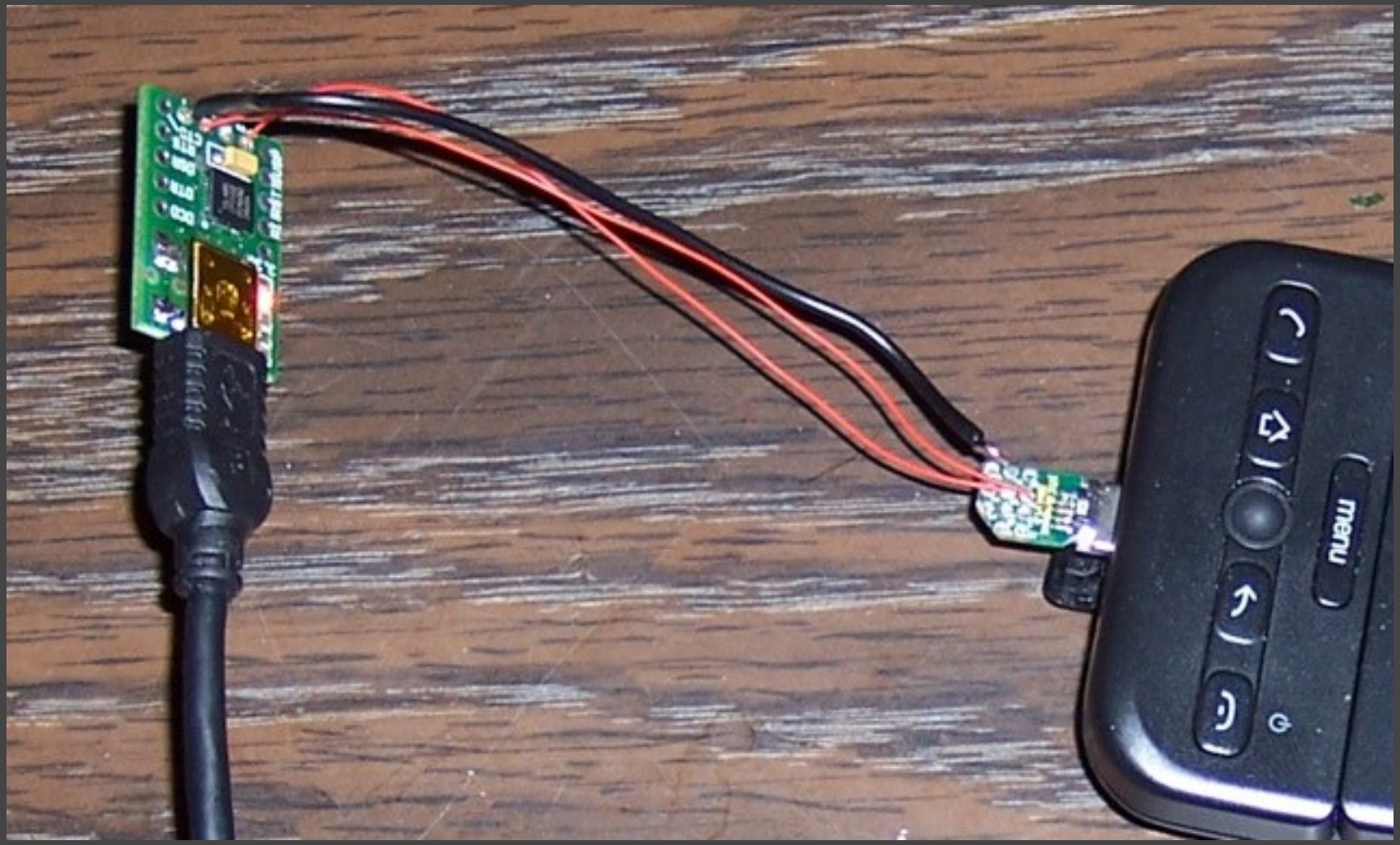
Dumpeo de la NAND sin acceder al sistema operativo

Dumpeo desde OEMSBL

Acceder desde SPL (rtask a), o a través de cable serie.

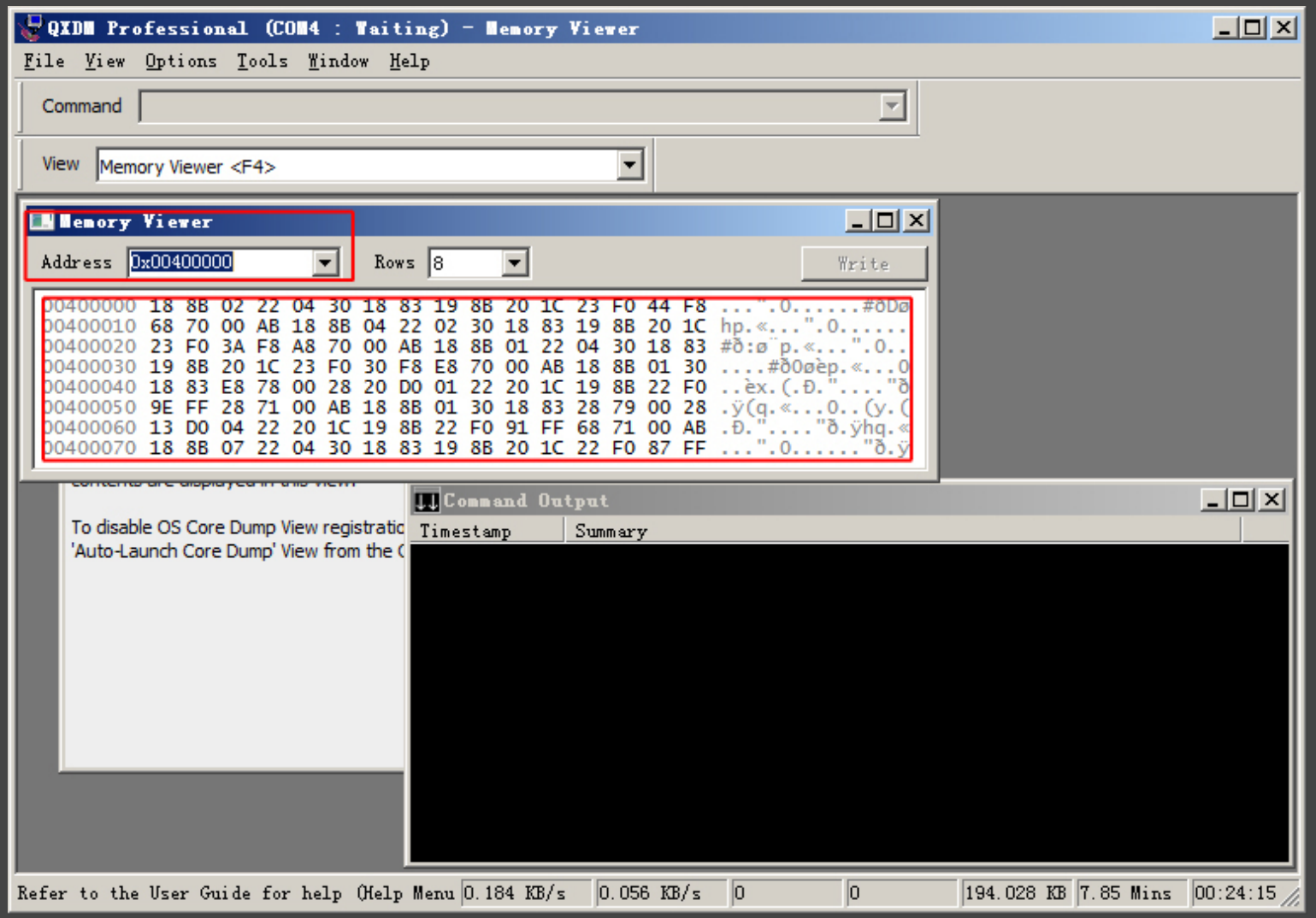
```
Command: inpw
Usage: inpw <address_in_hex>
Purpose: Read word in assigned address.

Command: inpdw
Usage: inpdw <address_in_hex>
Purpose: Read dword in assigned address.
```

Cable USB-Serie



Dumpeo sin acceder al Sistema Operativo

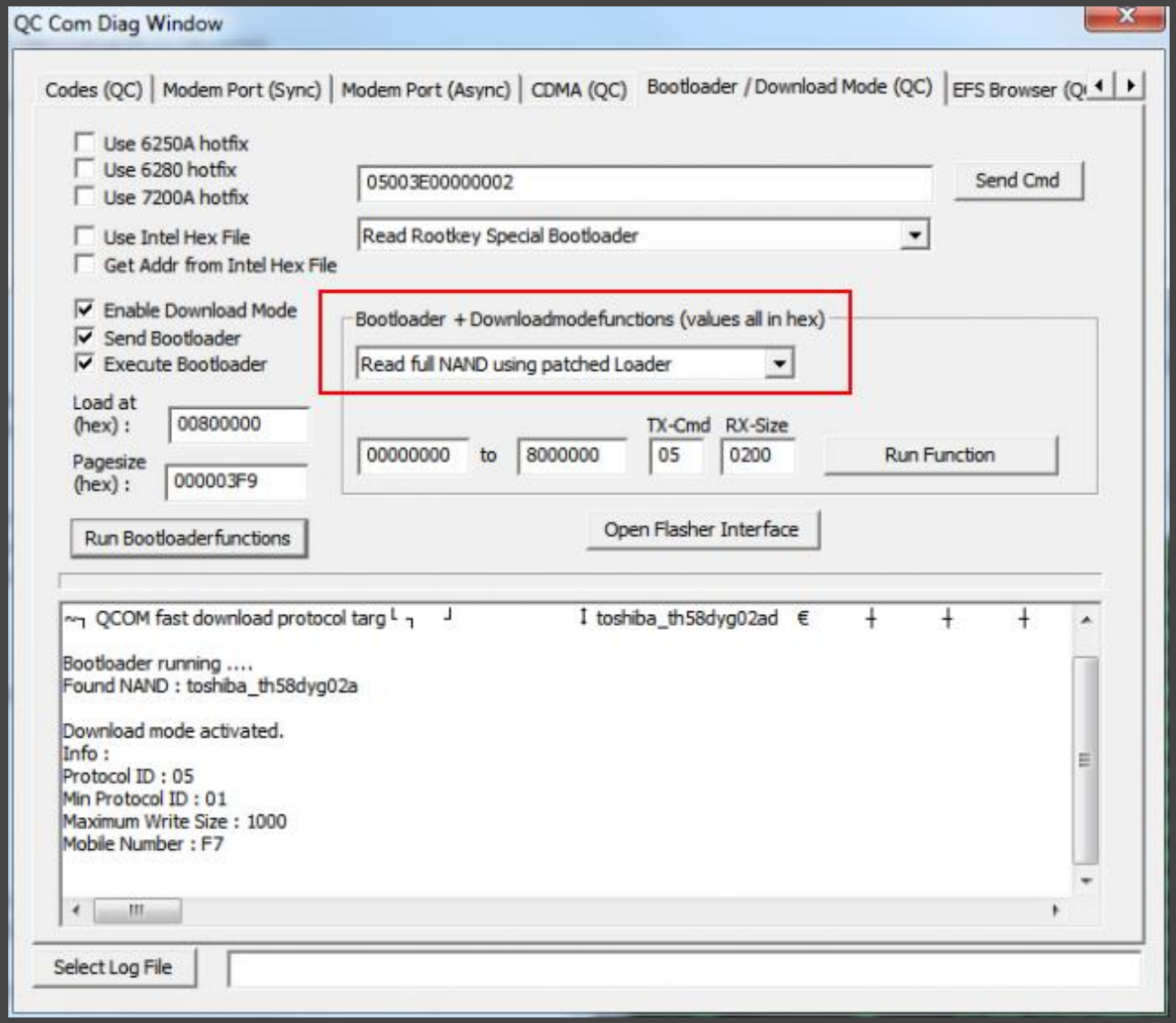
Qualcomm Tools: QPST, QXDM y CDMA Workshop



Dumpeo sin acceder al Sistema Operativo

PSAS: Phone System Analysis Software (antiguo QMAT)

http://psas.revskills.de/



Dumpeo sin acceder al Sistema Operativo

Desde BootLoader (SPL):

rbmc [FileName [StartAddr [Len]]]

Read back the memory content from the specified address to the host and save the data to specified file name.

FileName : Full file path for save data of memory(default=c:\temp\Mem.nb).

StartAddr : Start address of memory(default(hex)=A0000000).

Len : How many bytes will be read. And if not given value, it will be total ROM size on board - ((StartAddress & 0x0FFFFFFF) - (ROM base address(0) & 0x0FFFFFFF)).

Dumpeo sin acceder al Sistema Operativo

Desde BootLoader (SPL):

Abusar del comando 'checksum':

checksum [[StartAddr [Len]]]

Return CRC checksum of **RAM memory**.

StartAddr : Start address of ROM(default(hex)=A0000000).

Len : How many bytes will be calculated.
default(hex)<pre> ROM total size - ((dwStartAddress & 0x0FFFFFFF) - (ROM_BASE & 0x0FFFFFFF))

¿Que pasa si pedimos el CRC de un sólo byte? ;-)

DIY NAND Dumper: leer la NAND con 2 llamdas a 'checksum'.

SPLxploit

En un bootloader con restricciones (comandos checksum y rbmc no existen), debemos 'reemplazar' el bootloader en caliente aprovechando una vulnerabilidad del mismo:

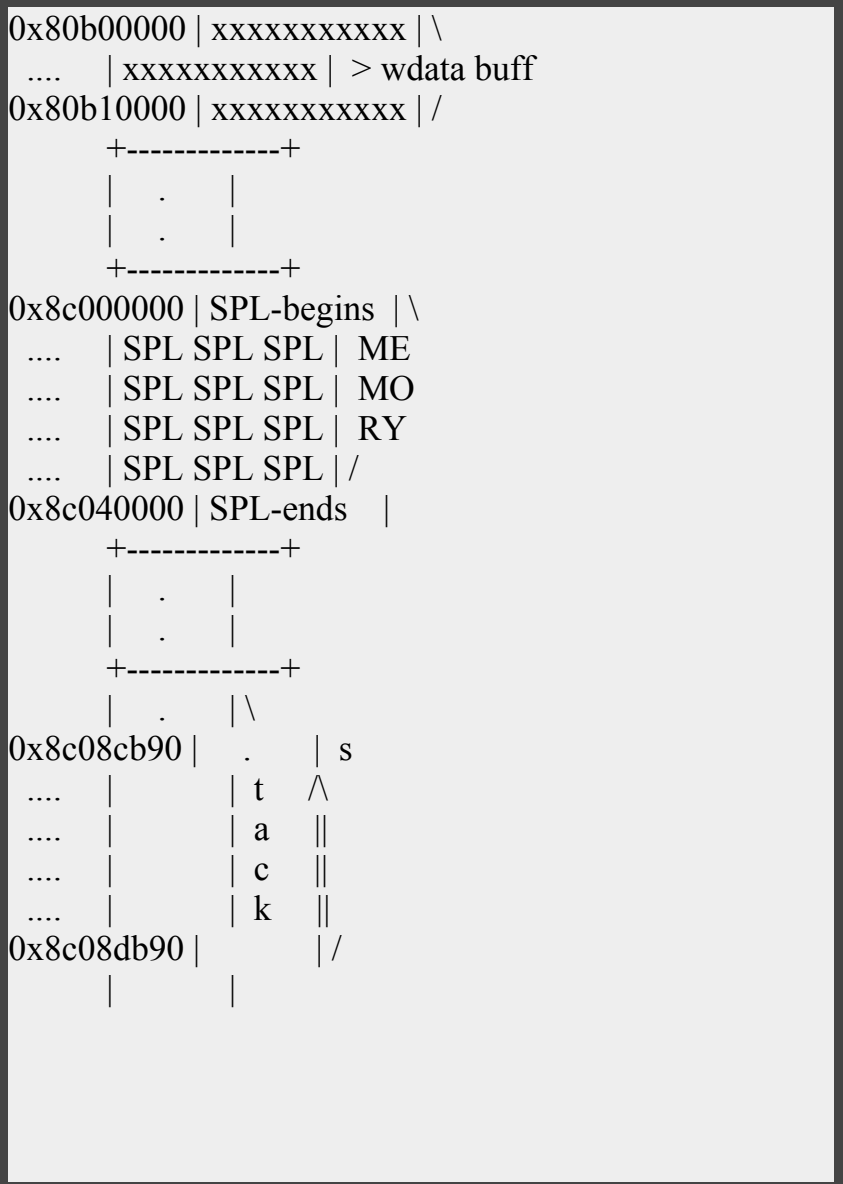
- Existe un stack overflow en todos los bootloaders de HTC. Reportado a HTC el año 2007, todavía no lo han parcheado.

- Haciendo llamadas recursivas al comando 'ruustart' podemos "llenar" la pila, y sobreescrivir código del bootloader.

- Usamos "stack spraying" con jumps relativos a nuestro shellcode para 'saltar' a un bootloader modificado.

+info: http://pof.eslack.org/HTC/splxploit/

SPLxploit - memory layout



1) poner un patron de bytes conocido en la stack
2) checksum, para encontrar el offset del top de la stack y el tamaño de stack frame
3) cargar codigo no firmado usando wdata -> invalid CERT error
4) meteremos una IPL modificada para que no cargue la SPL de la nand i una SPL parcheada
5) iterar comando 'ruustart' hasta llegar al final de la SPL: - primero padding con 0's.
- despues shellcode: handler que ejecuta el loader q reside en RAM y salta al offset 0 para iniciar la IPL. - finalmente: spraying con branches relativos al shellcode
7) llamar a una funcion q tenga el entry point alineado correctamente

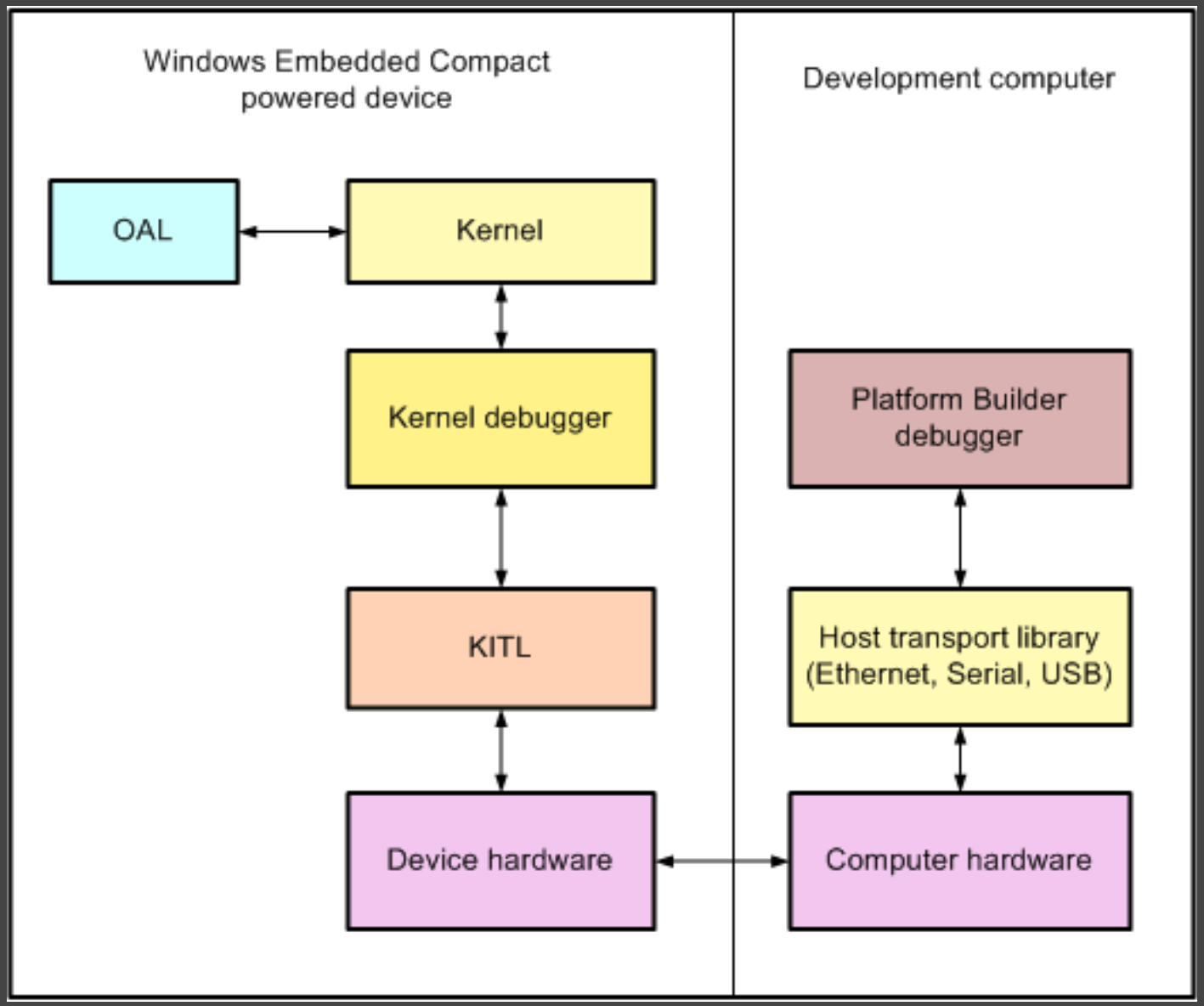
KITL: (Windows CE Only)

Kernel Independent Transport Layer (KITL): is a communication link between your development computer & wince enabled device for debugging purpose.

The Platform Builder debugger and remote tools such as Remote File Viewer, Remote Registry Editor, and Remote Kernel Tracker use KITL. KITL exposes the device hardware to the kernel debugger and works independently of the board and transport (such as Ethernet, serial, or USB) to send and receive data between the computer and the device.

source: MSDN

KITL

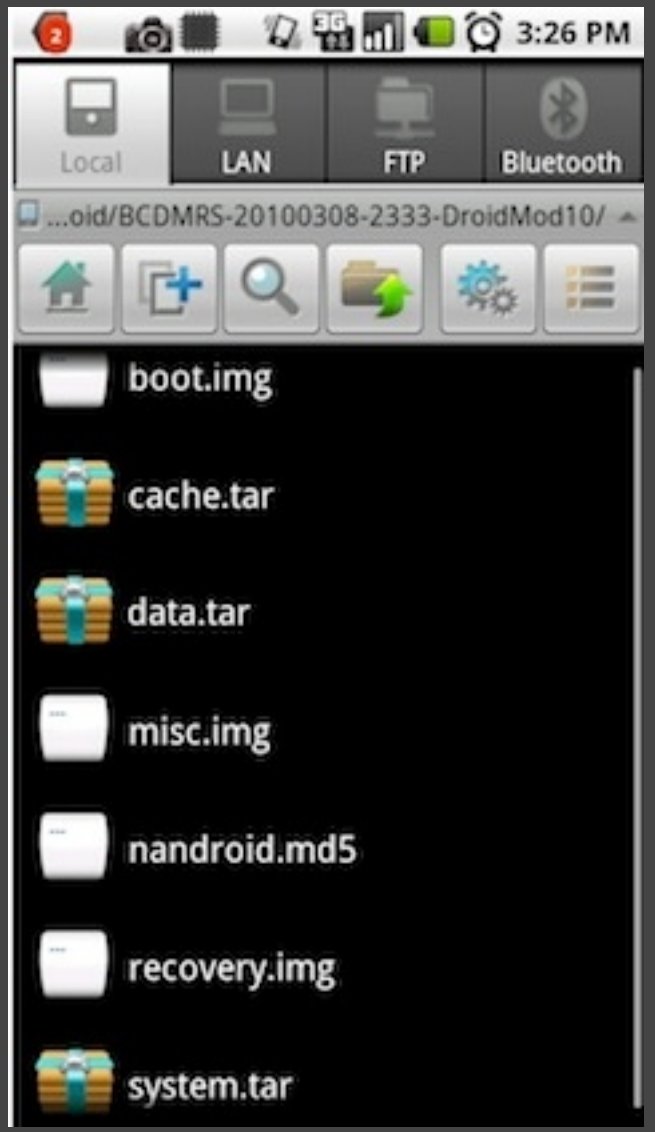

Windows Embedded Compact
powered device
Development computer
OAL
Kernel
Kernel debugger
KITL
Device hardware
Platform Builder
debugger
Host transport library
(Ethernet, Serial, USB)
Computer hardware

SSCOM3.2 (作者:聂小猛(丁丁), 主頁http://www.mcu51.com, Email: mcu52@163.com)2003.6.24

```
160h:01 f0 a0 e1 00 00 a0 e1 00 00 a0 e1 fe ff ff ea
170h:f8 03 9f e5 00 10 90 e5 01 15 81 e3 00 10 80 e5
180h:10 10 a0 e3 01 10 51 e2 fd ff ff 1a d0 03 9f e5
190h:00 10 90 e5 0e 18 81 e3 00 10 80 e5 bc 03 9f e5
1a0h:00 10 90 e5 08 10 81 e3 00 10 80 e5 fe ff ff ea
1b0h:12 13 a0 e3 b9 2f 8f e2 00 00 a0 e1 34 30 81 e2
1c0h:04 40 92 e4 04 40 81 e4 03 00 51 e1 fb ff ff 1a
1d0h:0e f0 a0 e1 0e a0 a0 e1 4e 14 a0 e3 9c 23 9f e5
1e0h:00 20 81 e5 00 20 91 e5 02 2b c2 e3 00 20 81 e5
1f0h:ff 20 a0 e3 04 20 c1 e5 00 30 a0 e3 01 30 83 e2


SMDK2410 Board (MCU S3C2410) Test Program Ver 1.1(20020801) FCLK = 202800000 Hz

 0:ADC                  1:ADC with DMA         2:ADC TSP Seperate     3:ADC TSP Auto
 4:DMA M2M              5:DMA Worst Test       6:External DMA         7:External Interrupt
 8:IIC(KS24C080)INT     9:IIC(KS24C080)POL    10:Reco IIS UDA1341    11:Play IIS UDA1341
12:FIQ Interrupt       13:Change INT Priority 14:UART2 IrDA Rx       15:UART2 IrDA Tx
16:STN 1Bit            17:STN 2Bit            18:STN 4Bit            19:CSTN  8Bit
20:CSTN  8Bit On       21:CSTN 12Bit          22:TFT240320  8Bit     23:TFT240320  8Bit On
24:TFT240320 16Bit     25:TFT640480  1Bit     26:TFT640480  8Bit     27:TFT640480 16Bit
28:TFT640480 BSWP      29:TFT640480 Palette   30:TFT640480 HWSWP     31:MPLL Change
32:MPLL MPS Change     33:MPLL On/Off         34:PMS Slow            35:PMS Hold
36:PMS Idle            37:PMS Idle(MMU)       38:PMS Idle Hard       39:PMS SDRAM Init
40:PMS STOP            41:PMS Power-Off STOP  42:PMS Power-Off 100Hz 43:PMS Measure Power
44:RTC Alarm           45:RTC Display         46:RTC Round Reset     47:RTC Tick
48:SDI Write/Read      49:SPIO RxTx Int       50:SPIO RxTx POLL      51:SPIO Master Tx DMA1
52:SPIO Slave Rx DMA1  53:SPIO Master Rx DMA1 54:SPIO Slave Tx DMA1  55:SPIO Master RxTx INT
56:SPIO Slave RxTx INT 57:Timer Interrupt     58:Timer Tout          59:UART0 Rx/Tx Int
60:UART0 Rx/Tx DMA     61:UART0 Rx/Tx FIFO    62:UART0 AFC Tx        63:UART0 AFC Rx
64:UART1 Rx/Tx Int     65:UART1 Rx/Tx DMA     66:UART1 Rx/Tx FIFO    67:UART1 AFC Tx
68:UART1 AFC Rx        69:UART2 Rx/Tx Int     70:UART2 Rx/Tx DMA     71:UART2 Rx/Tx FIFO
72:USB FIFO Test       73:WDT INT Request     74:External Bus Reqest 75:NonAlgined Access
76:PC Card (PD6710)    77:Read Page Mode      78:SWI                 79:External Wait
80:Stone Test          81:ETC NEC Int         82:nBATT_FAULT int     83:NAND View Bad Block
84:NAND View Page      85:NAND Write          86:NAND ECC            87:NOR Flash Program

Select the function to test :
```

Nandroid Backup desde recovery image

Nandroid Backup is a set of tools and a script that will enable anyone who has **root** on their Android phone plus a recovery image with busybox and adbd running as root to make full system backups.



Source: infernix@xda-devs

Nandroid backup desde recovery image

data.tar (data.img) contiene los datos del usuario.

- extraer montando la imagen yaffs2 en raw como dispositivo loop (requiere mtd-utils y el módulo de kernel de yaffs2)
- unyaffs: extrae directamente el contenido del .img en el directorio local

http://code.google.com/p/unyaffs/

Dumpeo de la NAND desde sistema operativo

Windows Mobile

WinCE TFFS API

True flash file system or TrueFFS is a low level file system designed to run on a raw Solid-state drive.

TrueFFS implements error correction, bad block re-mapping and wear leveling. Externally, TrueFFS presents a normal hard disk interface.

TrueFFS was created by M-Systems, on well-known "DiskOnChip 2000" product line, who were acquired by Sandisk in 2006.

A flash translation layer is used to adapt a fully functional file system to the constraints and restrictions imposed by flash memory devices.

Source: Wikipedia

itsutils pdocread (I)

```
Usage: pdocread [options] start [ length [ filename ] ]
    when no length is specified, 512 bytes are assumed
    when no filename is specified, a hexdump is printed
    -t     : find exact disk size
    -l     : list all diskdevices
    -v     : be verbose
    -s OFS : seek into source file ( for writing only )
    -b SIZE: specify sectorsize to use when accessing disk
    -B SIZE: specify blocksize to use when accessing disk
    -G SIZE: specify blocksize to use when transfering over activesync
    -u PASSWD : unlock DOC device
    -S BK1x : specify alternate disksignature ( e.g. BIPO, BK1A .. BK1G )
Source:
    -d NAME : devicename or storename
    -p NAME : partitionname
    -h HANDLE : directly specify handle
  either specify -d and optionally -p, or specify -h
Method:
    -n NUM : binarypartition number ( normal p if omitted )
    -w     : read via windows disk api
    -o     : read OTP area
if the filename is omitted, the data is hexdumped to stdout
if no length is specified, 512 bytes are printed
```

itsutils pdocread (II)

List NAND Partitions:

```
$ pdocread.exe -l
 85.88M (0x55e0000) FLASHDR
|          3.12M (0x31f000) Part00 - image update kernel part.
|          3.50M (0x380000) Part01 - regular kernel part. (XIP)
|         41.38M (0x2960000) Part02 - IMGFS
|         37.88M (0x25e0000) Part03 - User filesystem
STRG handles:
handle c34713fe 37.88M (0x25e0000)
handle e348c912 41.38M (0x2960000)
handle c348c8ee  3.50M (0x380000)
handle 2348c71e  3.12M (0x31f000)
```

Dump NAND Partitions:

```
$ pdocread.exe -w -d FLASHDR -b 0x800 -p Part00 0 0x31f000 Part00.raw
$ pdocread.exe -w -d FLASHDR -b 0x800 -p Part01 0 0x380000 Part01.raw
$ pdocread.exe -w -d FLASHDR -b 0x800 -p Part02 0 0x2960000 Part02.raw
$ pdocread.exe -w -d FLASHDR -b 0x800 -p Part03 0 0x25e0000 Part03.raw
```

itsutils bkondisk

Permite acceder a Samsung OnDisk Flash:

copy bkondisk.exe to \windows on your device, then:

prun bkondisk [targetdir]

will save all partitions on all volumes in files on [targetdir]

Particiones que podemos dumpear:

```
        partnr    attr    start     size   ( in blocks )
pi0.2:  00000002 00000002 00000005 00000020  gsm etc
pi0.3:  00000003 00000002 00000025 00000160  os
pi0.6:  00000006 00000022 00000001 00000004  spl
pi0.8:  00000008 00000001 00000185 00000264  userfs
```

JumpSPL

Aplicacion para WinCE que permite poner un bootloader en RAM y saltar a éste, sin necesidad de que esté flasheado en el dispositivo.

- deshabilita la cache de instrucciones de ARM y el direccionamiento virtual

(para entendernos: "mata" el kernel de windows CE, y deja el dispositivo como si acabara de arrancar la IPL, tras el inicio del hardware)

+info: http://forum.xda-developers.com/showthread.php?t=334667

Dumpeo de la NAND desde sistema operativo

Android

Dumpeo de NAND en Android

Security OFF (S-OFF):
@secuflag - radio NVRAM flag que controla la "seguridad" de la NAND.

¿Como pasar de S-ON a S-OFF?

- Engineering device (with engineering HBOOT)

- unrevoked3 (para conseguir root) + AlphaRev 1.5 (parchea el HBOOT para que no consulte @secuflag)

http://www.unrevoked.com/
http://alpharev.nl

- unrevokedforever (escribe en NVRAM - sólo CDMA)

http://unrevoked.com/rootwiki/doku.php/public/forever

Dumpeo de NAND en Android

- Char driver interface to dump physical memory

http://gitorious.org/androidfan-flash/kernel/commit/2ae612479f5e2ae891fb9b22a2ea3415c4fec017

Módulo de kernel que crea el dispositivo */dev/memdump*, desde donde podemos dumpear el contenido de la NAND.

That's all folks!

Any questions?