Palm IIIx Internals
(or what 3Com don't tell you)

by Jesús Arias

25th June 2002

Contents

Preface.

This document compiles almost all the information that I have obtained about the Palm IIIx hardware. This story begun when a friend of mine gave me a Palm IIIx with a broken touch-screen. That PDA was useless for a normal use because its screen, but makes a good candidate for reverse engineering because if something broke none would be lost.

The first thing I did to that PDA was a hardware hack which allowed me to boot the Dragonball processor using its internal (well documented) bootloader ROM. Then I tried a lot of DRAM controller configurations until I was able to download some code to the DRAM and to run it.

Then I downloaded all the Flash code to my PC and I disassembled it. I was looking for the general purpose pins configuration, and also for the correct DRAM configuration.

And finally, using some test programs, I tried to test all my assumptions about the Palm. I hope that the following pages were error free, but *"errare humanum est"*...

Chapter 1

The Palm IIIx Hardware.

1.1 The Chips.

When a Palm IIIx is opened, two PCB's are found. The main board includes CPU, memory, power regulation, speaker and buttons. The other board is bonded to a metal case and includes the LCD screen, its associate electronics, and the touch-screen.

The inspection of the main board shows the following known chips:

- MC68EZ328 (Motorola): CPU
- 29LV160B-90 (Fujitsu): Flash memory
- two KM416V1204CT-L6 (SEC, Samsung): DRAM memory.
- ADS7843 (Burr Brown): Touch Screen controller & A/D converter.
- LT1304 (Linear technology) : Switched power regulator.
- SP385E (SIPEX): CMOS to RS232 level converter.
- SP4422A (SIPEX): High voltage generator for back-light.

There are also other chips with less clear functions. These chips seems to be of small scale integration, and are located both in the main board and in the LCD board.

A remarkable passive component is an 0.1 Farad electrolytic capacitor. This capacitor can hold enough charge to maintain the DRAM data and refresh during the battery change. 3Com guarantees that the data will be preserved for one minute without batteries.

1.2 CPU.

The Palm IIIx CPU is the Motorola Dragonball-EZ processor (MC68EZ328). This processor includes in a single chip a 68000 CPU, a real time clock , a PLL clock generator, an interrupt controller, several general purpose I/O ports, up to 8 programmable Chip Select signals, a DRAM controller, a UART, a SPI controller, a timer, a PWM audio output, and a LCD controller.

Unfortunately the Dragonball-EZ processor lacks an MMU, so, it's impossible to implement a good protected operating system like Linux/BSD on it. You can expect as much hangs as in an MS-Windows computer because the system integrity relies on the good behavior of the applications. This also opens the door (or may be the Windows) to the virus developers.

On the other hand, the 68000 CPU is a CISC core designed 24 years ago. Although this CPU had a clever design for 1978, its performance is quite low. A typical instruction takes a minimum 4 clock cycles for execution and complex instructions, like multiplication, are very slow. So don't expect MP3 decompression with this chip.

1.3 DRAM Memory.

The main Palm IIIx memory is of the Dynamic RAM type. This PDA model comes with 4 Mb of DRAM implemented with two SEC KM416V1204CT-L6 chips. Each chip contains 2Mb of memory organized as 1024 rows x 1024 columns x 16 bits. These memories are EDO type and the "L6" suffix suggest a 60 ns read cycle that allows for a proper operation without wait states.

The Dragonball-EZ processor includes a highly configurable DRAM controller that allows a direct connection of the DRAM chips to the processor. But "highly configurable" also means that it is difficult to configure right all the things. The DRAM controller is selected when a memory access matches the Chip Select D (CSD) base address. This address is programmed to 0 by the Palm OS start-up. The DRAM controller also includes two registers that must be correctly programmed before the DRAM use. The first is the DRAM Control Register, that is programmed to 0x8607. This means:

- DRAM Controller Enabled.
- CAS before RAS Refresh.
- Fast Page mode, 2 clocks / transfer.
- Main Refresh Clock: 32 KHz.
- EDO access for LCD DMA.
- Page Size: 1024.
- 0 Wait States.
- Fast Address Multiplexing.
- Normal RAS precharge.
- Low Power Refresh mode.
- Refresh during RESET.
- DWE enabled.

The other register is the DRAM Memory Configuration Register, and it controls how the row and column address are multiplexed, and also the refresh rate. The refresh rate is set to 32KHz and the address multiplexing is set to:

DRAM ADDRESS MAPPING.

col0	col1	col2	col3	col4	col5	col6	col7	col8	col9			
A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13

row0	row1	row2	row3	row4	row5	row6	row7	row8	row9			
A11	A12	A13	A14	A15	A16	A17	A18	A20	A19	A21	A22	A23

DRAM Memory Configuration Register (DRAMMC) = 0x8F01

Note that, while the Dragonball can provide up to 13 bits for the row and column address, only 10 bits are used in the DRAM chips. Also, the low address bits are assigned to the columns in order to allow page reads. Page reads are only used by the LCD DMA.

1.4 FLASH Memory.

The ROM memory of the Palm IIIx is of the Flash Type. That means that the PDA firmware can be easily updated "in-circuit". The Palm IIIx Flash contains the start-up code and the Palm OS.

The Palm IIIx includes only one Fujitsu 29LV160B-90 Flash chip. This chip contains 2Mb of memory organized as 1M x 16 bits. The Flash is divided in 35 sectors that can be individually erased. The first 4 sectors are 16Kb, 8Kb, 8Kb and 32Kb wide respectively, while the rest of the sectors are 64Kb wide. The "-90" suffix suggest a 90 ns read cycle, and therefore the Flash can be read without wait states.

The Flash memory is selected via the CSA chip-select output and it is, therefore, the default boot memory. During start-up this memory is always selected, but after the start-up the CSA chip select logic is programmed, and then the Flash memory it is located only at address 0x10c00000.

1.4.1 FLASH Protection.

The Dragonball processor starts its execution from the Flash memory. More precisely, the first 8 bytes of the Flash must contain the initial values for the supervisor SP and PC registers. If the first sector is erased, the CPU is unable to boot.

In order to prevent the erasure of the crucial memory areas, each Flash sector can be individually protected "off-circuit". The sectors which have been protected cannot be erased without the issue of an "unprotect command". This command needs the application of a high voltage (12 V) not available inside the PDA. This mechanism guarantees that the minimum firmware needed for an OS update is always available in the Flash.

Unfortunately, the Palm IIIx designers did not used this protection for the first Flash sector. This means that any application can erase this sector (accidentally or not) and then the PDA will be unable to boot. If this situation happens nothing can be done in order to recover the OS code without hardware hacks, and most PDA users will throw his Palm away thinking it has a hardware failure. Clever enough ???

But don't think that Palm designers did not known the protection mechanisms. The third Flash sector is protected. This sector only contains an ASCII string with the PDA serial number. For cry !!!

And also the Dragonball-EZ internal bootloader cannot be enabled because the PG5/EMUBRK pin is used for other purposes (see section 1.6).

In summary: The Palm IIIx is a computer that can be broken by its software. A very bad design.

1.5 Dragonball's Peripherals.

The Dragonball-EZ processor includes a lot of on-chip peripherals. Almost all Dragonball's peripherals are used in the Palm IIIx PDA. The following list summarizes these peripherals:

- LCD controller: The LCD screen has an interface with four data bits. The LCD screen is 160x160 pixels wide. The default RESET polarity is used for all the signals. A logic "1" in the video memory is displayed as "black" in the screen. When the back-light is turned on, the LCD image is inverted: a logic "1" is displayed as "burning green".
- UART: The Dragonball's UART is used for the serial cradle connector and for the IRDA transceiver. The serial or IRDA function is selected via the PD5 and PG4 pins (see section 1.6). Note that IRDA data format is not NRZ. When IRDA communications are used, the IRDA data format must be selected in the UART control register.
- SPI: The Serial Peripheral Interface is a synchronous port that communicates the processor and the touch screen A/D converter (see section 1.8).
- PWM: The PWM output drives a single transistor audio amplifier and, in turn, a piezoelectric speaker. The PWM output is able to play 8 bit audio streams, but the frequency response of the speaker is so bad, that this output is only good for beeps. We must also remark that the PWM moduation of sampled audio signals leads to severe harmonic distortion. This problem can be reduced with the predistortion of the samples (interpolation), which in the other hand requires some computing power.

pin	Type	Function
PB6	out	Powers the LCD when high
PC7	out	Enables the LCD when high
PD4/IRQ1	in	Cradle HotSync button. Active low.
PD5	out	Enables the RS232 level converter when high
PF1/IRQ5	in	Pen Interrupt input. Active low.
PF7	out	Enables the LCD backlight when high
PG4	out	Enables the IRDA transceiver when low.
PG5	out	Chip Select for the touch panel converter. Active low.

Table 1.1: General purpose pins and its functions.

- RTC: The real time clock is always on. It runs from the main 32.768KHz oscillator, and keeps the date and the time. It can generate interrupts on a periodic basis or on alarm matchings.
- TIMER: The timer pin cannot be used because it is already used for the LCD. But the timer can be still used internally for periodic interrupt generation and software profiling.

1.6 General purpose pins.

The Dragonball-EZ pins can be used for several functions. When one Dragonball feature is not used its associated pins can be then released for general purpose use. In the Palm IIIx PDA these pins are used for buttons and also for external subsystem enable signals and inputs. These signals were harder to find, and therefore the list depicted in table 1.1 can be incomplete.

The most remarkable pins are:

- PB6, that is also the timer input / output. The Palm IIIx uses this pin to power the LCD display on and off, precluding any timer related use.
- PC7. This pin can be used as LACD (LCD alternate crystal direction). In this case this output is toggled to alternate the crystal polarization on the panel. But, in the Palm IIIx, this pin is used as a general purpose pin. This means that LCD electronics already includes the LCD phase alternation function.
- PG5. This pin can be used as EMUBRK (hardware emulator breakpoint), but it is rarely used for this purpose. However, This pin has another great function: If this pin is hold low during RESET, the Dragonball-EZ CPU boots from an internal ROM. This ROM contains a bootloader program that configures the serial port for 9600 baud, 8 bit, transmissions and reads hexadecimal records from the serial port, allowing for OS updates, hardware tests and other useful things. Look into the MC68EZ328 users manual for the description of the hexadecimal B-Records. Unfortunately, this pin is used to select the A/D touch screen converter, preventing its bootloader use, but it can still be used if a hardware hack is done (as I did: You only need to solder a button switch between PG5 and GND. If this switch is pressed during RESET the CPU enters its internal bootloader code).

1.7 Buttons.

The seven buttons of the Palm IIIx are distributed over a 3x4 array that allows space up to 12 buttons. Figure 1.1 shows how the rows and columns are connected to the Dragonball-EZ processor. The four port D pins are programmed as inputs, with their pull-ups active. In order to detect a button press its row must be selected by making the corresponding PF pin low. The PD inputs can trigger an interrupt in the Dragonball, and this technique is surely used to awake the CPU when the ON button is pressed. Under these circumstances the PF6 pin is held low and the interrupt corresponding to the PD0 pin is enabled. When the ON button is pressed the generated interrupt awake the Dragonball PLL oscillator providing new clock cycles to the CPU.

Figure 1.2 shows where the above mentioned buttons are located in the PDA.

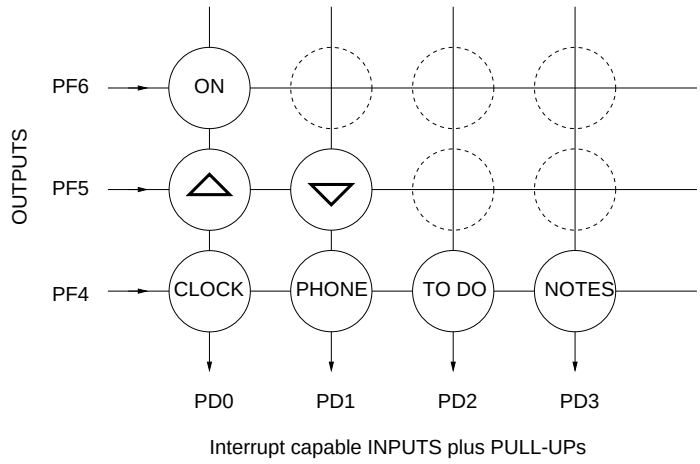


Figure 1.1: Array structure of the Palm IIIx buttons.

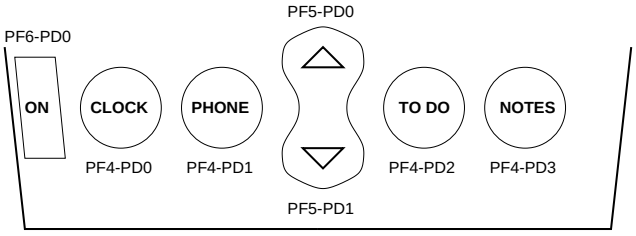


Figure 1.2: Button location in the Palm IIIx PDA.

1.8 Touch-screen

The reading of the pen position in a resistive touch screen requires A/D conversion. The Dragonball-EZ processor lacks A/D converters but it includes a Serial Peripheral port (only Master function) that allows a glueless connection to an external serial A/D converter. In this case the converter is an ADS7843, a touch screen controller made by Texas Instruments (formerly Burr Brown). Figure 1.3 shows how the ADS7843 is connected to the Dragonball processor. Together with the touch screen function, the converter also provides two general purpose analog inputs IN3 and IN4. The IN3 input is connected to the Palm batteries, and can therefore measure its voltage. The input IN4 is unused and it's just tied to ground to avoid noise coupling. The BUSY output of the ADS7843 seems to be not connected.

The following code illustrates how to read one of the four available channels of the ADS7843. This function takes as argument the control word that is sent to the touch-screen controller in the first data exchange (Read the ADS7843 datasheet for a description of control words).

```
unsigned int ads7843(unsigned int control)
{
        unsigned int ad;

        /* Enable SPIM, len=9 bits in order to rip-off the BUSY bit */
        /* Phase=0, Polarity=0, data rate=16.5MHz/128=129.536KHz */

        *((volatile unsigned short *)0xfffff802)=0xa208;
        *((volatile unsigned char *)0xfffff431)&=~0x20;  /* CS=PG5=0 */

        /* wait until CS reaches a proper low level.
           Remember the 10K series resistor */
        delay(50);
```

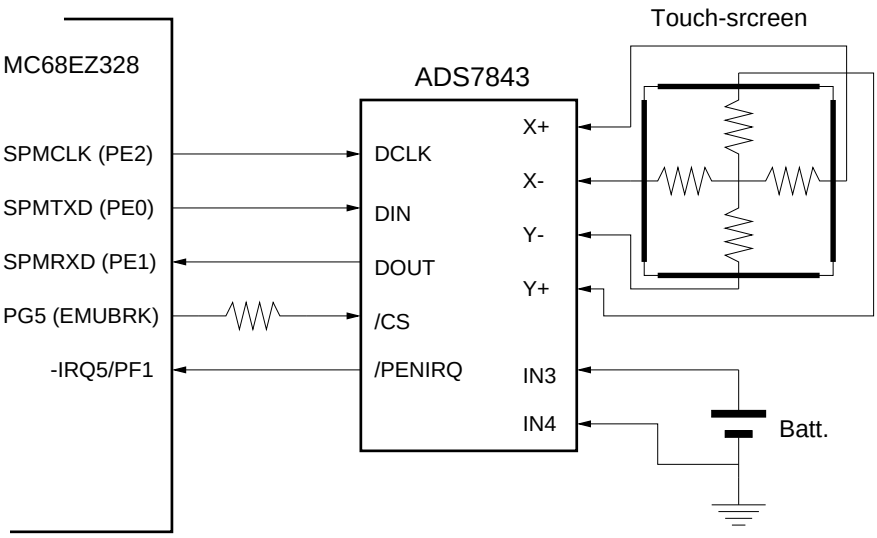


Figure 1.3: Circuit schematic of the Touch-Screen controller section.

```
        *((volatile unsigned short *)0xfffff800)=control<<1; /*Control word */
        *((volatile unsigned char  *)0xfffff802)|=1;    /* Trigger xchg. */
        while(!(*((volatile unsigned char *)0xfffff803)&0x80));
        *((volatile unsigned char *)0xfffff803)&=0x7f;  /* Clear IRQ */
        ad=*((volatile unsigned short *)0xfffff800);

        /* Now len=16 bits for AD data receiving */
        *((volatile unsigned short *)0xfffff802)=0xa20f;

        *((volatile unsigned short *)0xfffff800)=0;
        *((volatile unsigned char  *)0xfffff802)|=1;    /* Trigger xchg. */
        while(!(*((volatile unsigned char *)0xfffff803)&0x80));
        *((volatile unsigned char *)0xfffff803)&=0x7f;  /* Clear IRQ */
        ad=*((volatile unsigned short *)0xfffff800);

        *((volatile unsigned char *)0xfffff431)|=0x20;  /* CS=PG5=1 */
        delay(50);
        *((volatile unsigned char *)0xfffff802)&=0xfd;  /* disable SPIM */

        return ad>>4;   /* Only 12 bits are valid */
}
```

1.9 Cradle connector.

Figure 1.4 shows the cradle connector and the HotSync button circuit. The pin description was obtained via Google, and is good for the old non-EZ Palms. I was unable to find what are the Dragonball pins used for the GPI and GPO pins. GPI (pin 3) is probably not connected while GPO (pin 10) seems to be conected to a high, constant, RS232 voltage (+6V) coming from the SP385E pin 3 (V+) through a 330 Ω resistor.

Note that the cradle HotSync button is active high. This button just connects pin 9 (3.3 V) and pin 3 (HotSync) when pressed. The associated NPN transistor will invert this logic level while protecting the PD4 Dragonball pin from external static discharges.

The rest of the connector pins are the typical RS232 signals. These signals already have RS232 voltage levels thanks to the SP385E chip, so no external converters are needed. As a result, the cradle does not contain any active electronics.

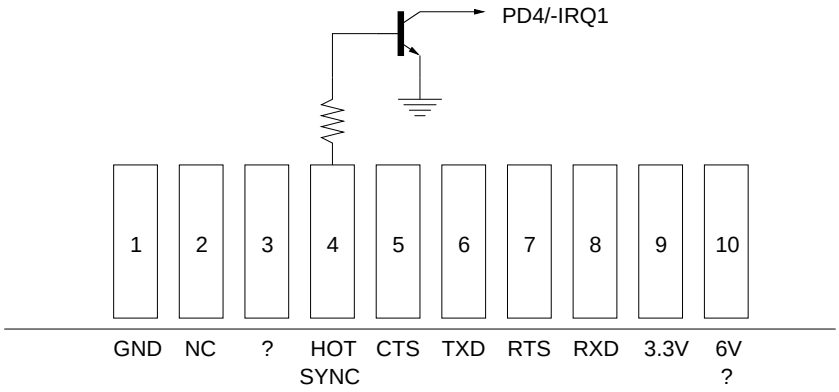
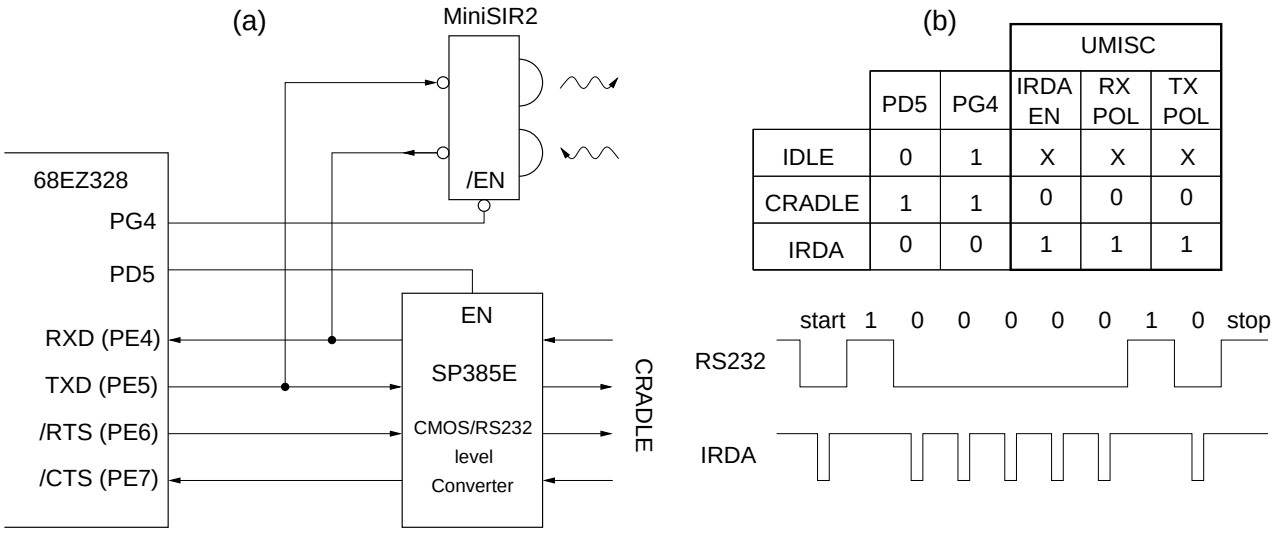


Figure 1.4: Detail of the Cradle connector and the HotSync circuit.



			UMISC		
	PD5	PG4	IRDA EN	RX POL	TX POL
IDLE	0	1	X	X	X
CRADLE	1	1	0	0	0
IRDA	0	0	1	1	1

Figure 1.5: (a) Serial port schematic. (b) Configuration for Cradle and IRDA interfaces and data formats.

Only the RXD, TXD and GND signals are needed to establish a serial connection with a PC, but CTS and RTS can help in the serial stream control. If CTS and RTS are not used, a software flow control is needed (usually XON/XOFF).

1.10 Infrared Port (IRDA)

The Palm iiix PDA comes with an infrared port that is IRDA SIR compliant. Data rates up to 115200 baud are supported. The physical interface requires an asynchronous serial port with IRDA capabilities, as the data format for IRDA transmissions is RZ instead of NRZ (see figure 1.5b).

The 68EZ328 includes one asynchronous serial port that can be configured for both RS323 and IRDA communications. This port is shared by cradle and IRDA interfaces, as shown in figure 1.5a. Figure 1.5b shows a table with the correct configuration for Cradle and IRDA communications and also the timing of the two data formats. The PD5 and PG4 pins are general purpose pins and can be programmed to 0 or 1 in their respective port register. IRDA_EN, RX_POL and TX_POL are control bits of the UART miscellaneous register (UMISC). The two later bits allows the inversion of the RXD and TXD data respectively. Note that, in the Palm iiix, the data polarity must be low for IRDA while high for RS232 communications.

IRDA communications are packet oriented. The data encapsulation and error checking must be done by program because the UART lacks all these functionalities. Simple tests with dumb asynchronous data shows some interesting

things:

1. The first transmission bits, and maybe bytes, are corrupted. This might be due to automatic gain control in IRDA receivers. To avoid this problem the actual data packet must be preceded by a preamble of dummy data. 0xFF is a good character for preamble because it only has a pulse per character, allowing for a good UART synchronization.
2. Reception errors are usual. Each packet must include a reliable error detection mechanism. Parity checking is not enough. Erroneous packets must be detected and resent.
3. Transmitted data is also listened by the IRDA receiver in the transmitter. Thus, in order to avoid interference, IRDA communications must be half-duplex.