J U L Y 1 9 9 8

WRL
Technical Note TN-54

The Itsy Pocket Computer Version 1.5 User's Manual

Marc A. Viredaz

Western Research Laboratory 250 University Avenue Palo Alto, California 94301 USA

The Itsy Pocket Computer Version 1.5
User's Manual

Marc A. Viredaz

July 1998

Abstract

The *Itsy pocket computer* is a flexible research platform developed at Compaq Computer Corporation's *Western Research Laboratory (WRL)*. Its aim is to enable hardware and software research in pocket computing, including low-power hardware, power management, operating systems, wire-less networking, user interfaces, and applications. This document describes the architecture and the low-level programming model of the Itsy computer.

Revision 1.0

Contents

List of Figures

List of Tables

Acknowledgements

The Itsy pocket-computer project was started and led by Bill Hamburgen. The hardware described in this document was designed jointly by Bill Hamburgen and the author.

The author would like to acknowledge the support of Marco Annaratone, Roy Levin, and Bob Supnik.

The author is grateful to the design team of the *StrongARM SA-1100 processor*, in particular to Richard Witek, the chief architect, and to Gregg Mack, who designed the processor's evaluation platform (a.k.a. *Brutus*) which served as a starting point to the design of the Itsy computer. Other members of the StrongARM team who provided invaluable services are Lynn Comp, Tim Litch, Jeanne Meyer, Jim Montanaro, Tom Schild, and Jeff Slaton.

The author is also indebted to the Itsy team at the *Western Research Laboratory (WRL)* and *System Research Center (SRC)*: Joel Bartlett, Lawrence Brakmo, David Chaiken, Jeff Dean, Puneet Kumar, Bob Mayo, Sharon Perl, Barton Sano, Carl Waldspurger, and Deborah Wallach, for their comments and feed-back. Special thanks go to Wayne Mack, who built several of the early prototypes, and to Annie Warren, Steve Jeske, and Fran McGroary-Dehn for taking care of the project's logistics.

1 Introduction

This document describes the architecture and the programmer's model of the *Itsy pocket computer*. It should be considered as a guide for low-level software developers and daughter-card designers.

1.1 History

The first *printed-circuit board (PCB)*, referred to as *Itsy mother-board version 1.0*, was completed in November 1997. This first prototype had several flaws, all of which could, fortunately, be corrected. The logic design corresponding to a modified (i.e., patched) version 1.0 board is known as *Itsy mother-board version 1.1*. This design corresponds only to a set of schematics, no physical PCB having been manufactured. A total of six version 1.0/1.1 systems were built.

A second prototype, named *Itsy mother-board version 1.5* was first built in March 1998. It corresponds to a version 1.1 system with a few additional features. From the programmer's point-of-view, there are almost no differences between the versions 1.1 and 1.5.

This document describes the Itsy mother-board version 1.5. All relevant differences between the versions 1.1 and 1.5 are outlined in foot-notes.

1.2 Notations

In this report, electrical signals are represented as upper-case names in a sans-serif font (e.g., PWR_EN). Active-low signals are denoted by over-lines (e.g., $\overline{\mathsf{RESET}}$), while buses and element of buses are specified by subscripts (e.g., $\mathsf{D}_{31..0}$, A_0). In the schematics and PLD listing, the same signals are represented using the syntax and conventions of the corresponding CAD tools. For example, the signal $\overline{\mathsf{CS}}_0$ appears as `~cs[0]` in the schematics, according to the WindowSIL [Tha97] conventions, and is represented as `!CS0` in the PLD listing, following the ABEL syntax of Synario [Syn96].

On the schematics, each component is uniquely identified by a short reference (2 – 4 letters). In this document, these component references are represented in a fixed-size font (e.g., `s01`).

2 Architecture

Figure 1 presents the architecture of the Itsy computer. The left part of the figure shows the implementation of the Itsy mother-board. The right part of the figure represents all the resources available through the daughter-card interface (see Section 2.7). Figure 2 shows the front panel of the Itsy computer and the placement of input/output units. The remainder of this section describes each individual unit.

2.1 Processor

The *central processing unit (CPU)* of the Itsy computer is the StrongARM SA-1100 processor, developed by Digital Equipment Corporation's Digital Semiconductor division, which became part of Intel in May 1998. A good knowledge of the manufacturer's documentation [DEC98a, DEC98b] is assumed throughout this report.

The main crystal frequency is 3.6864 MHz. Using the processor's *phase-locked loop (PLL)*, this makes it possible to vary the CPU core frequency from 59.0 MHz to 206.4 MHz. The real-time crystal frequency is 32.768 kHz.

2.2 Power supply

The Itsy computer is powered by a pair of standard AAA alkaline batteries, suppling a voltage V_{batt}. A measurement device enables external monitoring of the power consumption. The voltage after this device is referred to as V_{pwrin}. For most practical purposes, these two voltage can be considered equal (i.e., $V_{\mathrm{batt}} \approx V_{\mathrm{pwrin}}$). With new batteries, the maximum battery voltage is $V_{\mathrm{batt}} \approx 3.2\,\mathrm{V}$. On the Itsy mother-board version 1.5, an external power supply (i.e., $2.0\,\mathrm{V} < V_{\mathrm{batt}} \leq 3.3\,\mathrm{V}$) can be

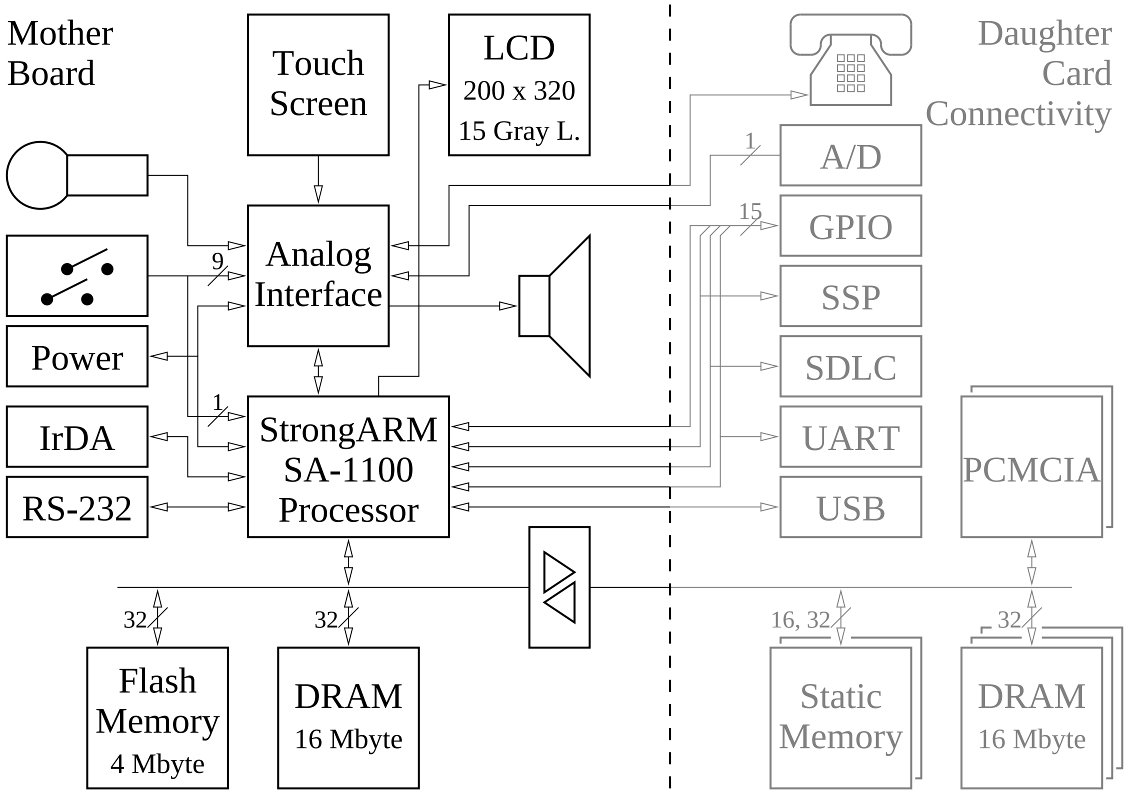

Figure 1: Architecture of the Itsy pocket computer.

connected using the serial-interface connector (see Section 2.6.3).[1] Since, there is no mechanism to automatically disconnect the batteries when an external power supply is used, the batteries must be removed before connecting the external power supply. Failure to do this carries a risk of leakage and possibly fire.

Two voltages are regulated from the batteries or external power supply: the main power-supply voltage V_{dd} (3.3 V nominal) and the CPU core power-supply voltage V_{cc} (1.5 V nominal).

2.2.1 Power monitoring

The Itsy computer features several mechanisms to monitor the power-supply voltages. At the hardware level, the signals BATT_FAULT and VDD_FAULT of the StrongARM SA-1100 processor [DEC98b] force a transition to sleep mode if the voltage V_{pwrin} falls below the threshold $V_{\mathrm{pwrin,\,fault}}$ (i.e., $V_{\mathrm{pwrin}} \leq V_{\mathrm{pwrin,\,fault}} \approx 2.0\,\mathrm{V}$) or if the voltage V_{dd} falls below the threshold $V_{\mathrm{dd,\,fault}}$ (i.e., $V_{\mathrm{dd}} \leq V_{\mathrm{dd,\,fault}} \approx 2.7\,\mathrm{V}$), respectively. Table 1 gives the voltage tolerances for the thresholds $V_{\mathrm{pwrin,\,fault}}$ and $V_{\mathrm{dd,\,fault}}$. Although there is no mechanism to monitor the voltage V_{cc}, the corresponding power supply has been designed such that this voltage should always remain stable as

[1]This feature is not available on the Itsy mother-board version 1.1.

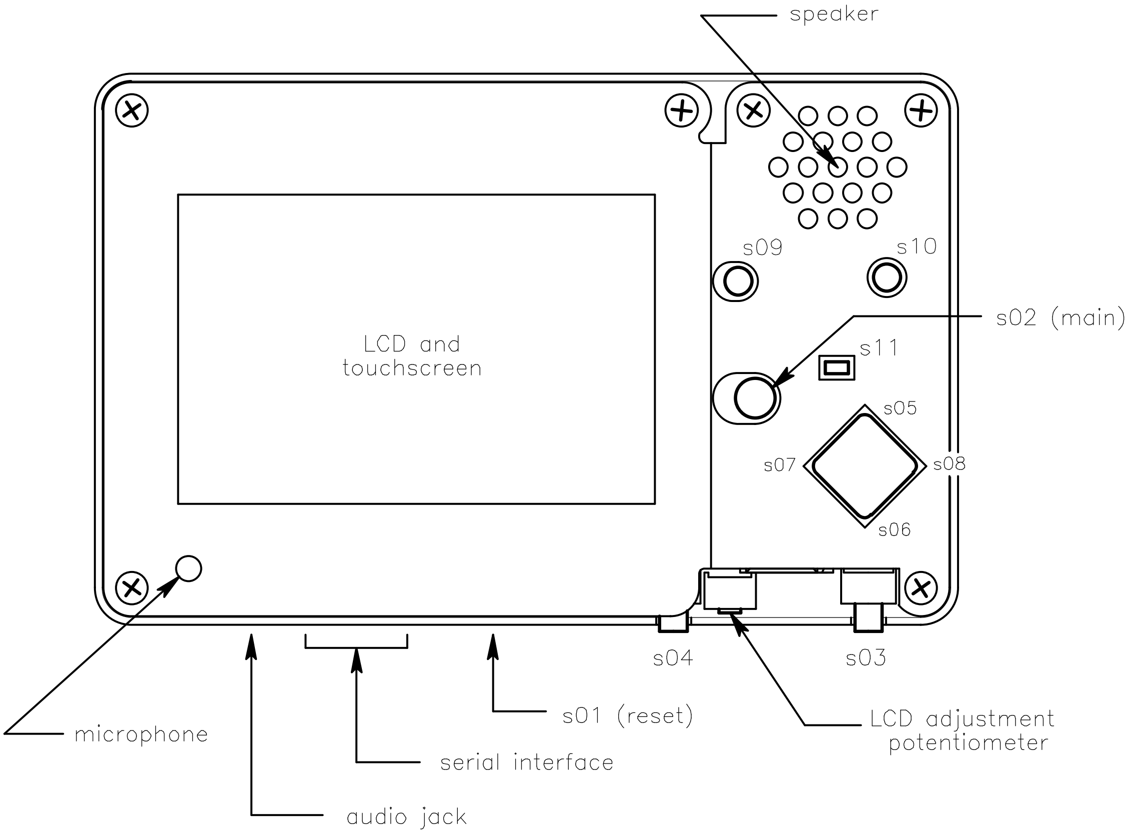

Figure 2: Front panel of the Itsy computer.

long as the battery voltage V_{pwrin} is above the threshold $V_{\mathrm{pwrin,\,fault}}$.

The input signal GPIO_2 (PWROK) is set to 1 during normal operation and toggles to 0 if the main power-supply voltage V_{dd} drops below the threshold $V_{\mathrm{dd,\,low}}$ (i.e., $V_{\mathrm{dd}} \leq V_{\mathrm{dd,\,low}} \approx 3.0\,\mathrm{V}$), given in Table 1 (see Section 3.2). This signal can be polled by software or used to generate an interrupt. A software transition to sleep mode can then be forced, after the current state has been saved as appropriate. As shown in Table 1, the signal GPIO_2 (PWROK) will always toggle to 0 before the signal VDD_FAULT is asserted. Since most components (including the processor) are not specified to operate correctly at a voltage V_{dd} lower than 3.0 V, the signal GPIO_2 (PWROK) must be monitored. The signal VDD_FAULT should only be considered as a "safety" mechanism.

Voltage threshold	Min.	Typ.	Max.
$V_{\mathrm{pwrin,\,fault}}$	1.88 V	1.93 V	1.98 V
$V_{\mathrm{dd,\,fault}}$	2.58 V	2.67 V	2.76 V
$V_{\mathrm{dd,\,low}}$	2.82 V	2.91 V	3.01 V

Table 1: Voltage thresholds for power monitoring.

Finally, the three voltages V_{pwrin}, V_{dd}, and V_{cc} are connected to the general-purpose analog input pins AD_0, AD_1, and AD_2 of the UCB1200 analog interface [Phi97b] and can be measured using the corresponding analog-to-digital converters. Since there is no early warning that the battery voltage V_{pwrin} is about to reach the threshold $V_{\mathrm{pwrin,fault}}$, the software must use this interface to infer the state of the batteries.

2.3 Reset scheme

A reset pulse is applied to the signal $\overline{\mathsf{RESET}}$ upon power-up or when the reset push-button (`s01`) is pressed (see Figure 2). This signal is also asserted when the power-supply voltage V_{dd} falls below a level that is at least 30 mV lower than the threshold $V_{\mathrm{dd,fault}}$. This should only happen at the very end of the batteries' life, when the voltage V_{batt} (or V_{pwrin}) is so low that the power supply is unable to keep the voltage V_{dd} regulated even at low sleep-mode current.

2.4 Control logic

An external controller, implemented using a Philips PZ3032-8BC *programmable logic device (PLD)* [Phi97a] is used to select the boot memory and to implement the *auxiliary LCD controller*.

Static-memory bank 0 mirrors either bank 1 or bank 2, depending whether the StrongARM SA-1100 processor [DEC98b] should boot [2] from the mother-board or from the daughter-card (see Sections 2.7 and 3.2). After a hardware reset (i.e., power-up or push-button reset) or while exiting sleep mode, the daughter-card signal $\overline{\mathsf{DCBOOT}}$ and the processor signal GPIO_{19} (DCEN) are sampled.[3] If the former signal is asserted (0) and the latter signal is set to 1, static-memory bank 0 mirrors bank 2 and the processor boots from the daughter-card. Otherwise, bank 0 mirrors bank 1 and the processor boots from the mother-board. After a hardware reset, the boot memory is solely selected by the signal $\overline{\mathsf{DCBOOT}}$, since the value of the signal GPIO_{19} (DCEN) is always 1 (see Section 3.2).

The auxiliary LCD controller makes it possible to capture a black-and-white image (no grey levels) on the LCD and to maintain it even when the processor is in sleep mode (see Section 2.6.1). With the signal GPIO_{20} (LCDEN) set to 1 and the signal GPIO_{21} (AUXLCDEN) set to 0 (see Section 3.2), the software should display a black-and-white image on the LCD using the processor's LCD controller. The signal GPIO_{21} (AUXLCDEN) should then be toggled to 1. After the complete image has been transmitted, it is captured and remains displayed as long as the signals GPIO_{20} (LCDEN) and GPIO_{21} (AUXLCDEN) both stay at 1. The exact capture time occurs at the first end-of-frame that the value of the signal GPIO_{21} (AUXLCDEN) is 1.

2.5 Memory system

On the mother-board, the memory system consists of a *flash memory* decoded as static-memory bank 1 and of *dynamic random-access memory (DRAM)* bank 0. Static-memory banks 2 and 3

[2] The initialization sequence that follows a reset (i.e., hardware, software, or watch-dog reset) is very similar to the sequence that is executed while exiting sleep mode. Thus, for the sake of simplicity, the verb "boot" is used in this document as a generic term to refer to any of these actions.

[3] On the Itsy mother-board version 1.1, these signals are also sampled after a software or watch-dog reset.

as well as DRAM banks 1, 2, and 3 are available to the daughter-card interface (see Section 2.7). Static-memory bank 0 — from which the StrongARM SA-1100 processor [DEC98b] boots — mirrors either bank 1 or bank 2 (see Sections 2.4, 2.7, and 3.2).

2.5.1 Flash memory

A pair of 16-bit flash-memory circuits implement the 32-bit mother-board flash memory. Many different devices can be accommodated:

AMD Am29LV160B$OssPT$ [AMD97e]
AMD Am29LV800O-$ssPT$/Am29LV800B$OssPT$ [AMD97c, AMD97d]
AMD Am29LV400O-$ssPT$ [AMD97a]
AMD Am29LV200O-$ssPT$ [AMD97b]
Hitachi HN29VO800P-ss/HN29WO800P-ss [Hit97c, Hit97d]
Motorola M29F800A2$OPss$/M29F800A3$OPss$ [Mot97]
Sharp LH28F800SGP-Lss [Sha97]

where "O" specifies the internal sector organization, "ss" specifies the speed, "P" specifies the package, and "T" specifies the temperature range. Any other compatible parts can also be used.

Following the convention described in Sections 2.7.1 and 3.3, a non-volatile memory identification structure describes the characteristics of the specific parts used on a given system and hence allows the software to configure the memory interface correctly.

The *reset/power-down pin* of the flash-memory circuits is asserted (0) during a reset (i.e., hardware, software, or watch-dog reset) and during sleep mode. When the flash memory does not need to be accessed, this signal can also be asserted (0) by setting the signal GPIO_3 ($\overline{\mathsf{FLFOFF}}$) to 0 (see Section 3.2). When the Itsy computer must boot from the mother-board, this signal should never be set to 0 during sleep mode, since the processor would be unable to read the boot memory upon wake up. This can easily be achieved by setting bit 3 of the *power manager GPIO sleep state register* PGSR of the StrongARM SA-1100 processor [DEC98b] to 1.

The *ready/not-busy pins* of the flash-memory circuits can be monitored using the signals GPIO_4 ($\mathsf{FL0RY}/\overline{\mathsf{BY}}$), for the least significant 16 data bits $\mathsf{D}_{15\,..\,0}$, and GPIO_5 ($\mathsf{FL1RY}/\overline{\mathsf{BY}}$), for the most significant 16 data bits $\mathsf{D}_{31\,..\,16}$ (see Section 3.2).

The hardware write-protection mechanism, featured by some of the supported parts, is never used, and the corresponding pin is always de-asserted (1).

2.5.2 Dynamic RAM

A pair of 64 Mbit (i.e., 2^{12} rows $\times$ 2^{10} columns $\times$ 16 bits) self-refresh DRAM circuits implement the 32-bit mother-board DRAM. Many different *fast-page mode* or *enhanced data out (EDO)* devices can be accommodated:

Hitachi HM5165160ALTT-ss [Hit97a]
Hitachi HM5165165ALTT-ss [Hit97b]
Samsung KM416V4100AS-Lss/KM416V4100BS-Lss [Sam97a, Sam98a]
Samsung KM416V4104AS-Lss/KM416V4104BS-Lss [Sam97b, Sam98b]

Toshiba TC5165165AFTS-*ss* [Tos96]

where "*ss*" specifies the speed. Any other compatible parts can also be used.

Since — unlike with the flash memory — the software can not recognize which are the specific parts used on a given system, all Itsy computers version 1.5 have been assembled using the fastest available parts, that is, 50 ns EDO DRAM circuits (KM416V4104AS-L5, KM416V4104BS-L5, or TC5165165AFTS-50).

The choice of the mother-board DRAM affects which types of DRAM can be used on a daughter-card. Since all banks must have the same number of rows, only parts with 2^{12} rows can be considered. Moreover, since all banks must be accessed at the speed of the slowest one, it is best to use the same type of circuits everywhere.

2.6 Input/output devices

This section describes the different input/output units of the Itsy mother-board. All analog devices are handled by the Philips UCB1200 analog interface [Phi97b]. This circuit is connected to StrongARM SA-1100 processor [DEC98b] using the *Multi-media Communications Port (MCP)* engine of serial port 4.

2.6.1 Display and touch-screen

The display of the Itsy computer is an Epson TCM-A0822-*xx liquid crystal display (LCD)* [Eps97], where "*xx*" specifies the revision. It features 200 lines of 320 pixels each. The interface is 8 bits wide. The LCD controller of the StrongARM SA-1100 processor [DEC98b] is used as the main controller. It can display 15 levels of grey. A trim potentiometer is used to control the brightness of the LCD (see Figure 2).

The signal GPIO_{20} (LCDEN) is connected to the enable pins $\overline{\mathsf{DOFF1}}$ and $\overline{\mathsf{DOFF2}}$ of the LCD (see Section 3.2). The LCD is enabled when this signal is set to 1 and disabled when it is set to 0. Similarly, the signal GPIO_{21} (AUXLCDEN) is used to control an auxiliary LCD controller, which makes it possible to maintain a black-and-white image (no grey levels) on the LCD, even when the processor is in sleep mode (see Sections 2.4 and 3.2). The auxiliary LCD controller is enabled when this signal is set to 1 and disabled when it is set to 0. To capture a black-and-white image, the image should be displayed using the processor's LCD controller with the auxiliary LCD controller disabled. The signal GPIO_{21} (AUXLCDEN) should then be toggled to 1. After the complete image has been transmitted, it is captured and remains displayed as long as the signals GPIO_{20} (LCDEN) and GPIO_{21} (AUXLCDEN) both stay at 1.

On the Itsy mother-board version 1.5, the four signals TSXP, TSXN, TSYP, and TSYN of the resistive touch-screen are directly connected to the TSPX, TSMX, TSPY, and TSMY pins of the UCB1200 analog interface [Phi97b], respectively.[4]

[4] On the Itsy mother-board version 1.1, the signals TSXP, TSXN, TSYP, and TSYN of the touch-screen are connected to the TSMX, TSPX, TSPY, and TSMY pins of the UCB1200 analog interface, respectively (i.e., the polarity of the X-axis is inverted).

Pin	Signal	Function	Dir.
1	GND	**Gro**und (common terminal)	
2	MIC	**Mic**rophone (audio input)	I
3	SPKR	**Sp**ea**k**er (audio output)	O

Table 2: Audio jack connector pin-out. Pin 1 is the sleeve, pin 2 is the tip, and pin 3 is the ring (i.e., middle contact).

In order to minimize the noise generated by the LCD, it is possible to synchronize the sampling of the touch-screen voltages with the LCD line pulse. The synchronization signal TSCRSMPL — connected to the ADCSYNC pin of the UCB1200 analog interface — is identical to the LCD line clock LCDLCLK.

2.6.2 Audio interface

The audio interface consists of a microphone, a speaker, and a 3-contact 2.5 mm jack connector (see Figure 2). The mother-board microphone and speaker are disconnected when the jack connector is used. Table 2 provides the pin-out of this connector.

The audio input is connected to the MICP pin of the UCB1200 analog interface [Phi97b], through a 1 μF decoupling capacitor. The negative terminal of the mother-board microphone is connected to the MICGND pin of the UCB1200 analog interface. When an external microphone is used, its negative terminal is connected to the system ground (GND).

Similarly, the audio output is connected to the SPKRP pin of the UCB1200 analog interface, through a 47 μF decoupling capacitor. The negative terminal of the mother-board speaker is connected to the SPKRN pin of the UCB1200 analog interface, allowing differential drive of the speaker. When an external speaker is used, its negative terminal is connected to the system ground (GND). In this case, the speaker is no longer differentially driven.

2.6.3 Serial interface

Serial port 3 of the StrongARM SA-1100 processor [DEC98b] provides the *universal asynchronous receiver/transmitter (UART)* engine that is used as the RS-232 interface of the Itsy mother-board (see Figure 2). It is connected to a Hirose 3260-8S1 connector through a Maxim MAX3223CAP RS-232 driver [Max96].

The choice of a non-standard serial-interface connector for the serial interface has been made in order to decrease the size of the Itsy computer. Table 3 provides the pin-out of this connector. On the Itsy mother-board version 1.5, it can also be used to connect to an external power supply (see Section 2.2).[5]

A consequence of having a non-standard connector is that special cables must be used. Figure 3 shows the connection from the serial-interface connector to a 6-pin male MMJ connector. The same cable can be used to connect the Itsy computer either to a *data communication equipment (DCE)*,

[5] This feature is not available on the Itsy mother-board version 1.1.

Pin	Signal	Function	Dir.
1	RSDTR	**RS**-232 interface **d**ata **t**erminal **r**eady (DTR)	O
2	BATT *	**Batt**ery voltage V_{batt}	
3	RSTXD	**RS**-232 interface **t**ransmit **d**ata (TxD)	O
4	GND	**G**rou**nd**	
5	GND	**G**rou**nd**	
6	RSRXD	**RS**-232 interface **r**eceive **d**ata (RxD)	I
7	BATT *	**Batt**ery voltage V_{batt}	
8	RSDSR	**RS**-232 interface **d**ata **s**et **r**eady (DSR)	I

* On the Itsy mother-board version 1.1, pins 2 and 7 are connected to ground (GND).

Table 3: Serial-interface connector pin-out.

i.e., a modem, or to a *data terminal equipment (DTE)*, i.e., a computer or a terminal. Figure 4 presents two examples of connection to 9-pin DIN connectors. The cable represented by Figure 4 (a) is used to connect the Itsy computer to a DCE, while the cable represented by Figure 4 (b) is used to connect the Itsy computer to a DTE. It can be noted that an MMJ cable (see Figure 3) used in conjunction with the DEC H8571-J MMJ-to-DIN adapter corresponds to the connection described by Figure 4 (b). Some applications might require different connections and, hence, use different cables.

The signals GPIO_{23} ($\overline{\mathsf{UARTFOFF}}$) and GPIO_{24} (UARTFON) control the MAX3223CAP RS-232 driver (see Section 3.2). When the former signal is set to 0, the driver is disabled. When it is set to 1, the driver is either enabled or in *AutoShutdown mode*, depending whether the signal GPIO_{24} (UARTFON) is set to 1 or to 0, respectively. In AutoShutdown mode, the driver enables and disables itself depending whether a valid RS-232 input signal is detected (i.e., the serial interface is connected to a powered system) or not (i.e., the serial interface is unconnected or is connected to a unpowered system). When the driver is disabled in AutoShutdown mode, transmitting data

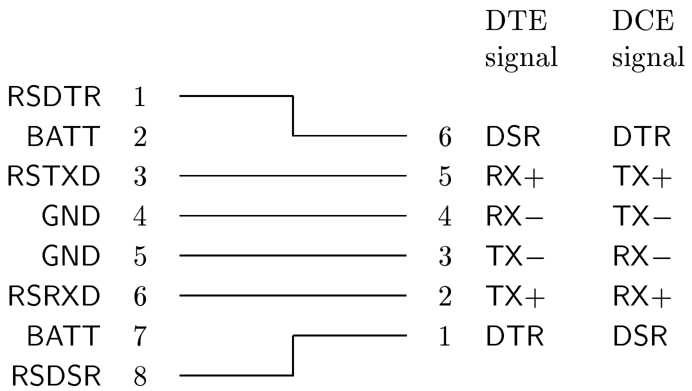

Figure 3: Serial-interface cable to 6-pin male MMJ connector.

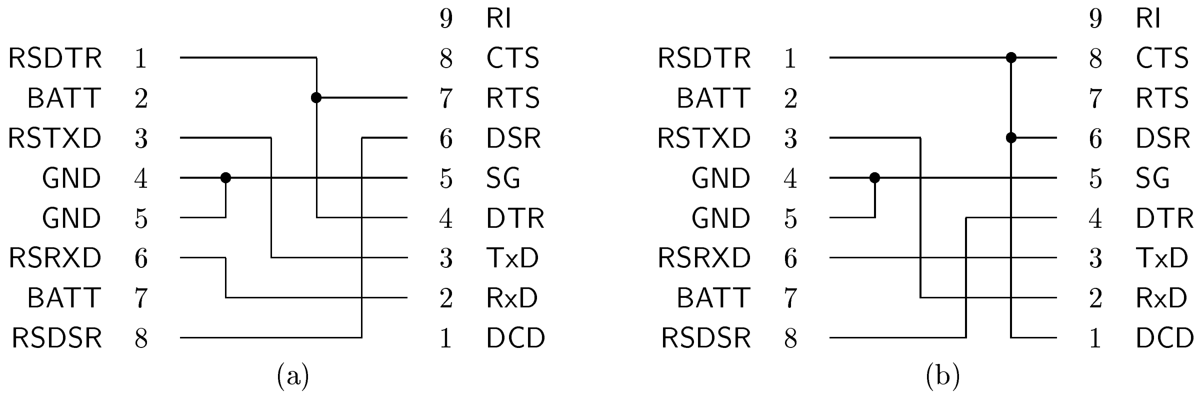

Figure 4: Serial-interface cables to 9-pin male and female DIN connectors. (a) Connection from the Itsy computer to a DCE (9-pin male DIN connector). (b) Connection from the Itsy computer to a DTE (9-pin female DIN connector, null-modem cable).

will not enable it. As a consequence, a potential dead-lock might happen when the serial interface is connected to another similar interface (e.g., when two Itsy computers are connected together), since both drivers might remain disabled. To avoid this, the software should briefly set the signal GPIO_{24} (UARTFON) to 1 when establishing a connection. The driver can then be switched to AutoShutdown mode by toggling this signal to 0.

The signal GPIO_{7} (UARTVALID) is set to 1 when a valid RS-232 input signal is detected and to 0 when no valid signals are detected (see Section 3.2).

The signals GPIO_{8} (UARTDSR) and GPIO_{9} (UARTDTR) are connected to the second receive and transmit buffers of the MAX3223CAP RS-232 driver (see Section 3.2). Used in conjunction with an appropriate special cable, they can emulate one additional input signal and one additional output signal of the RS-232 standard. In the examples presented in Figures 3 and 4, they are shown to emulate the *data set ready* signal DSR and the *data terminal ready* signal DTR, respectively. However, since they are general-purpose input/output signals and since the serial-interface connector is non-standard, there are no restrictions on which signals they can emulate. For example, some applications might use these signals to emulate the the *clear to send* signal CTS and the *request to send* signal RTS.

2.6.4 Infrared interface

The infrared interface of the Itsy computer is implemented by a Novalog MiniSIR2 IrDA transceiver [Nov98], used in conjunction with the UART engine of serial port 2 of the StrongARM SA-1100 processor [DEC98b]. This interface is compliant with the *Infrared Data Association (IrDA)* standard version 1.0. Only the UART engine of serial port 2 should be used, the *high-speed serial to parallel (HSSP)* engine should never be used. Moreover, the bit rate should be limited to the range: [2.4 kbit/s .. 115.2 kbit/s]. Since the receive and transmit pins of the MiniSIR2 IrDA transceiver are both active high, the *HSSP control register 2* HSCR2 of the StrongARM SA-1100 processor should be initialized to $000C0000_{16}$.

The signal GPIO_{22} ($\overline{\mathsf{IRDAEN}}$) is used to control the MiniSIR2 IrDA transceiver (see Section 3.2). The transceiver is enabled when this signal is set to 0 and disabled when it is set to 1.

2.6.5 Push-buttons

The Itsy computer features 10 general-purpose push-buttons (see Figure 2). The *main push-button* (`s02`) is connected to the signal GPIO_0 ($\overline{\mathsf{MAINPB}}$) of the StrongARM SA-1100 processor [DEC98b] (see Section 3.2). This signal is set to 0 when the push-button is pressed and to 1 when the push-button is released. This push-button is always enabled. It should be debounced by software. Although this is, otherwise, a general-purpose push-button, it can be used to wake up the processor after a hardware transition to sleep mode due to the assertion of the signal BATT_FAULT or the signal VDD_FAULT (see Section 2.2.1).

The 9 remaining push-buttons (`s03` to `s11`) are connected to the $\mathsf{IO}_{0..8}$ pins of the UCB1200 analog interface [Phi97b]. They are enabled by setting the IO_9 pin to 0 and are disabled when this pin is configured as input or set to 1. The $\mathsf{IO}_{0..8}$ pins are set to 0 when the corresponding push-buttons are pressed and to 1 when they are released. These push-buttons should be debounced by software.

2.7 Daughter-card interface

The aim of the daughter-card interface is to provide hardware designers with all the unused resources of the StrongARM SA-1100 processor [DEC98b] and UCB1200 analog interface [Phi97b]. Daughter-cards are connected to the Itsy mother-board through a 160-pin connector. The functionality available through the daughter-card interface includes:

- 2 static-memory banks (banks 2 and 3). It is possible to boot from the daughter-card. In this case, bank 0 mirrors bank 2.
- 3 DRAM banks (banks 1, 2 and 3).
- 2-socket *Personal Computer Memory Card International Association (PCMCIA)* interface.
- 4 serial interfaces: *universal serial bus (USB)*, *universal asynchronous receiver/transmitter (UART)*, *synchronous data link controller (SDLC)*, and *synchronous serial port (SSP)*.
- 15 general-purpose input/output signals, 13 of which can be used for interrupts.
- 1 telecommunication codec (e.g., for software modem).
- 1 general-purpose analog input (10-bit analog-to-digital converter, nominal voltage range: [0 V .. 7.5 V]).

Not all these features are available simultaneously, since three of the serial interfaces (i.e., UART, SDLC, SSP) are allocated by reconfiguring some of the general-purpose input/output pins.

Table 4 shows the pin-out of the daughter-card connector. Some signals (mostly the memory bus) are buffered by SN74LVCH16244ADGG drivers [TI97a] or SN74LVCH16245ADGG transceivers [TI97b]. Buffered signals are explicitly specified as such in the description below. All other signals are unbuffered. All buffers are enabled when the signal GPIO_{19} (DCEN) is set to 1 and disabled when it is set to 0 (see Section 3.2). The following signals are available:

Pin	Signal	Dir.	Pin	Signal	Dir.	Pin	Signal	Dir.	Pin	Signal	Dir.
1	DCA_0	O	41	$\overline{\mathsf{DCPWAIT}}$	I	81	P3300		121	GND	
2	DCA_1	O	42	$\overline{\mathsf{DCIOIS16}}$	I	82	GND		122	DCD_{14}	I/O
3	P3300		43	GND		83	TINN	I	123	DCD_6	I/O
4	DCA_2	O	44	$\overline{\mathsf{DCPREG}}$	O	84	TINP	I	124	DCD_{29}	I/O
5	DCA_3	O	45	DCPSKTSEL	O	85	GND		125	DCD_{21}	I/O
6	GND		46	P3300		86	GND		126	GND	
7	DCA_4	O	47	GPIO_1	I/O	87	UDCN	I/O	127	DCD_{13}	I/O
8	DCA_5	O	48	$\overline{\mathsf{DCCS}_2}$	O	88	UDCP	I/O	128	DCD_5	I/O
9	DCA_6	O	49	P3300		89	GND		129	P3300	
10	DCA_7	O	50	$\overline{\mathsf{DCPIOW}}$	O	90	RXD_1	I/O	130	DCD_{28}	I/O
11	GND		51	$\overline{\mathsf{DCPIOR}}$	O	91	PWRIN		131	DCD_{20}	I/O
12	DCA_8	O	52	GND		92	TXD_1	I/O	132	GND	
13	DCA_9	O	53	$\overline{\mathsf{DCPWE}}$	O	93	PWRIN		133	DCD_{12}	I/O
14	P3300		54	$\overline{\mathsf{DCPOE}}$	O	94	P3300		134	DCD_4	I/O
15	DCA_{10}	O	55	P3300		95	GPIO_{27}	I/O	135	P3300	
16	DCA_{11}	O	56	$\overline{\mathsf{DCPCE}_2}$	O	96	GPIO_{26}	I/O	136	DCROMSEL	I
17	GND		57	$\overline{\mathsf{DCPCE}_1}$	O	97	P3300		137	$\overline{\mathsf{DCBOOT}}$	I
18	DCA_{12}	O	58	GND		98	GPIO_{25}	I/O	138	P3300	
19	DCA_{13}	O	59	$\overline{\mathsf{DCWE}}$	O	99	GPIO_{18}	I/O	139	DCD_{27}	I/O
20	P3300		60	$\overline{\mathsf{DCOE}}$	O	100	P3300		140	DCD_{19}	I/O
21	$\overline{\mathsf{RESET\_OUT}}$	O	61	$\overline{\mathsf{DCCAS}_3}$	O	101	GPIO_{17}	I/O	141	GND	
22	PWR_EN	O	62	$\overline{\mathsf{DCCAS}_2}$	O	102	GPIO_{16}	I/O	142	DCD_{11}	I/O
23	P3300		63	GND		103	P3300		143	DCD_3	I/O
24	DCA_{14}	O	64	$\overline{\mathsf{DCCAS}_1}$	O	104	GPIO_{15}	I/O	144	P3300	
25	DCA_{15}	O	65	$\overline{\mathsf{DCCAS}_0}$	O	105	GPIO_{14}	I/O	145	DCD_{26}	I/O
26	GND		66	P3300		106	P3300		146	DCD_{18}	I/O
27	DCA_{16}	O	67	$\overline{\mathsf{DCRAS}_3}$	O	107	GPIO_{13}	I/O	147	GND	
28	DCA_{17}	O	68	$\overline{\mathsf{DCRAS}_2}$	O	108	GPIO_{12}	I/O	148	DCD_{10}	I/O
29	P3300		69	GND		109	P3300		149	DCD_2	I/O
30	DCA_{18}	O	70	$\overline{\mathsf{DCRAS}_1}$	O	110	GPIO_{11}	I/O	150	DCD_{25}	I/O
31	DCA_{19}	O	71	$\overline{\mathsf{DCCS}_3}$	O	111	GPIO_{10}	I/O	151	DCD_{17}	I/O
32	GND		72	PWRIN		112	P3300		152	GND	
33	DCA_{20}	O	73	GND		113	DCD_{31}	I/O	153	DCD_9	I/O
34	DCA_{21}	O	74	GND		114	DCD_{23}	I/O	154	DCD_1	I/O
35	DCA_{22}	O	75	TOUTP	O	115	GND		155	P3300	
36	DCA_{23}	O	76	TOUTN	O	116	DCD_{15}	I/O	156	DCD_{24}	I/O
37	GND		77	GND		117	DCD_7	I/O	157	DCD_{16}	I/O
38	DCA_{24}	O	78	GND		118	P3300		158	$\overline{\mathsf{DCRST}}$ *	O
39	DCA_{25}	O	79	DCAD	I	119	DCD_{30}	I/O	159	DCD_8	I/O
40	P3300		80	GND		120	DCD_{22}	I/O	160	DCD_0	I/O

* On the Itsy mother-board version 1.1, pin 158 is connected to ground (GND).

Table 4: Daughter-card connector pin-out.

GND: **gro**u**nd**
27 pins are used to carry the system's ground.

P3300: **p**ower **3.300** V
24 pins are used to carry the 3.3 V-nominal power-supply voltage V_{dd} (see Section 2.2). The amount of current available to daughter-cards varies depending on the processing state of the Itsy computer (i.e., CPU core frequency, enabled/disabled state of all units, etc.). Daughter-cards that do not draw more than 80 mA while the StrongARM SA-1100 processor [DEC98b] is in sleep mode, and do not draw more than 200 mA while the processor is in idle or run modes, can be accommodated in almost any processing states. A further power analysis, beyond the scope of this document, is required to accommodate daughter-cards that require more current.

PWRIN: **pow**e**r in**put
3 pins are used to carry the unregulated battery voltage V_{pwrin} (see Section 2.2). The amount of current available to daughter-cards varies depending on the processing state of the Itsy computer (i.e., CPU core frequency, enabled/disabled state of all units, etc.).

$\overline{\text{DCRST}}$: **d**aughter-**c**ard **r**e**s**e**t**
On the Itsy mother-board version 1.5, this output signal is asserted (0) during a hardware reset (i.e., power-up or push-button reset). It corresponds to the input signal $\overline{\text{RESET}}$ of the StrongARM SA-1100 processor [DEC98b] buffered by an SN74LVCH16244ADGG driver [TI97a] (see Section 2.3).[6]

$\overline{\text{RESET\_OUT}}$: **reset out**put
This output signal is asserted (0) during a reset (i.e., hardware, software, or watch-dog reset) and during sleep mode. It is directly connected to the corresponding pin of the StrongARM SA-1100 processor [DEC98b].

PWR_EN: **pow**e**r en**able
This output signal is asserted (1) during run mode or idle mode and de-asserted (0) during sleep mode. It is connected directly to the corresponding pin of the StrongARM SA-1100 processor [DEC98b].

$\overline{\text{DCBOOT}}$: **d**aughter-**c**ard **boot** select
This input signal defines whether a daughter-card is bootable or not. It is sampled after a hardware reset (i.e., power-up or push-button reset) or while exiting sleep mode. If, at sampling time, this signal is asserted (0) and the signal GPIO_{19} (DCEN) is set to 1, static-memory bank 0 mirrors bank 2 and the StrongARM SA-1100 processor [DEC98b] boots from the daughter-card. Otherwise, static-memory bank 0 mirrors bank 1 and the processor boots from the mother-board. A pull-up resistor keeps this signal de-asserted (1) by default. After a hardware reset, the boot memory is solely selected by the signal $\overline{\text{DCBOOT}}$, since the value of the signal GPIO_{19} (DCEN) is always 1 (see Section 3.2). The signal $\overline{\text{DCBOOT}}$ is not

[6] On the Itsy mother-board version 1.1, the signal $\overline{\text{DCRST}}$ does not exist and pin 158 is connected to ground (GND).

sampled after a software or watch-dog reset, because the processor does not sample the signal ROM_SEL after these types of reset.[7]

DCROMSEL: **d**aughter-**c**ard boot **ROM** width **sel**ect

This input signal defines the width of static-memory bank 2 on a bootable daughter-card. A value of 0 corresponds to a 16-bit bank, while a value of 1 corresponds to a 32-bit bank. A pull-up resistor sets the default value of this signal to 1. This signal is used to set the signal ROM_SEL of the StrongARM SA-1100 processor [DEC98b] when static-memory bank 0 mirrors bank 2 (see signal $\overline{\mathsf{DCBOOT}}$ above).

$\overline{\mathsf{DCCS}}_{3..2}$: **d**aughter-**c**ard static-memory **c**hip **s**elect

These output signals control the accesses to the static-memory banks 2 and 3. The signal $\overline{\mathsf{DCCS}}_2$ is generated directly by the PZ3032-8BC PLD [Phi97a] and is asserted when the signal $\overline{\mathsf{CS}}_2$ of the StrongARM SA-1100 processor [DEC98b] is asserted or when the signal $\overline{\mathsf{CS}}_0$ is asserted and static-memory bank 0 mirrors bank 2 (see signal $\overline{\mathsf{DCBOOT}}$ above). This signal is enabled when the signal GPIO_{19} (DCEN) is set to 1 and disabled when it is set to 0 (see Section 3.2). The signal $\overline{\mathsf{DCCS}}_3$ corresponds to the signal $\overline{\mathsf{CS}}_3$ buffered by an SN74LVCH16244ADGG driver [TI97a].

$\overline{\mathsf{DCRAS}}_{3..1}$: **d**aughter-**c**ard DRAM **r**ow-**a**ddress **s**trobe
$\overline{\mathsf{DCCAS}}_{3..0}$: **d**aughter-**c**ard DRAM **c**olumn-**a**ddress **s**trobe

These output signals control the accesses to the DRAM banks 1, 2, and 3. They correspond to the signals $\overline{\mathsf{RAS}}_{3..1}$ and $\overline{\mathsf{CAS}}_{3..0}$ of the StrongARM SA-1100 processor [DEC98b] buffered by an SN74LVCH16244ADGG driver [TI97a].

$\overline{\mathsf{DCOE}}$: **d**aughter-**c**ard static-memory and DRAM **o**utput **e**nable
$\overline{\mathsf{DCWE}}$: **d**aughter-**c**ard static-memory and DRAM **w**rite **e**nable

These output signals control the accesses to the static-memory banks 2 and 3 (in conjunction with the signals $\overline{\mathsf{DCCS}}_{3..2}$) and to the DRAM banks 1, 2, and 3 (in conjunction with the signals $\overline{\mathsf{DCRAS}}_{3..1}$ and $\overline{\mathsf{DCCAS}}_{3..0}$). They correspond to the signals $\overline{\mathsf{OE}}$ and $\overline{\mathsf{WE}}$ of the StrongARM SA-1100 processor [DEC98b] buffered by an SN74LVCH16244ADGG driver [TI97a].

DCPSKTSEL: **d**aughter-**c**ard **P**CMCIA so**c**ke**t** **sel**ect
$\overline{\mathsf{DCPREG}}$: **d**aughter-**c**ard **P**CMCIA **reg**ister select
$\overline{\mathsf{DCPCE}}_{2..1}$: **d**aughter-**c**ard **P**CMCIA **c**hip **e**nable
$\overline{\mathsf{DCPOE}}$: **d**aughter-**c**ard **P**CMCIA **o**utput **e**nable
$\overline{\mathsf{DCPWE}}$: **d**aughter-**c**ard **P**CMCIA **w**rite **e**nable
$\overline{\mathsf{DCPIOR}}$: **d**aughter-**c**ard **P**CMCIA **i**nput/**o**utput **r**ead strobe
$\overline{\mathsf{DCPIOW}}$: **d**aughter-**c**ard **P**CMCIA **i**nput/**o**utput **w**rite strobe

These output signals control the accesses to the PCMCIA interface. They correspond to the signals PSKTSEL, $\overline{\mathsf{PREG}}$, $\overline{\mathsf{PCE}}_{2..1}$, $\overline{\mathsf{POE}}$, $\overline{\mathsf{PWE}}$, $\overline{\mathsf{PIOR}}$, and $\overline{\mathsf{PIOW}}$ of the StrongARM SA-1100 processor [DEC98b] buffered by SN74LVCH16244ADGG drivers [TI97a].

[7]On the Itsy mother-board version 1.1, the signal $\overline{\mathsf{DCBOOT}}$ is also sampled after a software or watch-dog reset, hence, leading to potential problems when using a 16-bit daughter-card and allowing the value of the signal $\overline{\mathsf{DCBOOT}}$ to change.

$\overline{\mathsf{DCIOIS16}}$: **d**aughter-**c**ard PCMCIA **i**nput/**o**utput **is** **16**-bit wide
$\overline{\mathsf{DCPWAIT}}$: **d**aughter-**c**ard **P**CMCIA **wait**

These input signals provide feed-back from the PCMCIA interface. They correspond to the signals $\overline{\mathsf{IOIS16}}$ and $\overline{\mathsf{PWAIT}}$ of the StrongARM SA-1100 processor [DEC98b] buffered by an SN74LVCH16244ADGG driver [TI97a]. Pull-up resistors keep the signals $\overline{\mathsf{IOIS16}}$ and $\overline{\mathsf{PWAIT}}$ de-asserted (1) when the driver is disabled.

$\mathsf{DCA}_{25\,..\,0}$: **d**aughter-**c**ard **a**ddress

These output signals implement the daughter-card address bus. They correspond to the address bus $\mathsf{A}_{25\,..\,0}$ of the StrongARM SA-1100 processor [DEC98b] buffered by two SN74LVCH16244ADGG drivers [TI97a].

$\mathsf{DCD}_{31\,..\,0}$: **d**aughter-**c**ard **d**ata

These bi-directional signals implement the daughter-card data bus. They correspond to the data bus $\mathsf{D}_{31\,..\,0}$ of the StrongARM SA-1100 processor [DEC98b] buffered by two SN74LVCH16245ADGG transceivers [TI97b].

UDCP: **U**SB **d**evice **c**ontroller **p**ositive data
UDCN: **U**SB **d**evice **c**ontroller **n**egative data

These bi-directional signals implement the USB interface. They are directly connected to the UDC+ and UDC− pins of the StrongARM SA-1100 processor [DEC98b]. No protection circuitry is provided.

RXD_1: **r**eceive **d**ata, serial port **1**
TXD_1: **t**ransmit **d**ata, serial port **1**

These bi-directional signals can be used to implement the SDLC or UART engine of serial port 1 or can be configured as general-purpose input/output signals, as shown in Table 5. They are directly connected to the corresponding pins of the StrongARM SA-1100 processor [DEC98b].

$\mathsf{GPIO}_{27\,..\,25,18\,..\,10,1}$: **g**eneral-**p**urpose **i**nput/**o**utput

These bi-directional signals can be used as general-purpose input/output signals, as interrupts, or to implement their *alternate function*, as shown in Table 5. They are directly connected to the corresponding pins of the StrongARM SA-1100 processor [DEC98b]. It should be noted that the signal GPIO_1 can be used to wake up the processor after a hardware transition to sleep mode due to the assertion of the signal BATT_FAULT or the signal VDD_FAULT (see Section 2.2.1).

TINP: **t**elecommunication **in**put **p**ositive data
TINN: **t**elecommunication **in**put **n**egative data
TOUTP: **t**elecommunication **out**put **p**ositive data
TOUTN: **t**elecommunication **out**put **n**egative data

These input and output signals implement the telecommunication codec (i.e., for software modem). They are directly connected to the corresponding pins of the UCB1200 analog interface [Phi97b].

	Main/alternate function		Gen.-purp. signal		
Signal	Function	Dir.	Dir.	Int.	Sleep
RXD_1	Serial port 1 SDLC/UART receive data	I	I/O	N	Z, 0
TXD_1	Serial port 1 SDLC/UART transmit data	O	I/O	N	Z, 0
$GPIO_1$			I/O	Y	Z, 0, 1
$GPIO_{10}$	Serial port 4 SSP transmit data	O	I/O	Y	Z, 0, 1
$GPIO_{11}$	Serial port 4 SSP receive data	I	I/O	Y	Z, 0, 1
$GPIO_{12}$	Serial port 4 SSP sample clock	O	I/O	Y	Z, 0, 1
$GPIO_{13}$	Serial port 4 SSP sample frame	O	I/O	Y	Z, 0, 1
$GPIO_{14}$	Serial port 1 UART transmit data	O	I/O	Y	Z, 0, 1
$GPIO_{15}$	Serial port 1 UART receive data	I	I/O	Y	Z, 0, 1
$GPIO_{16}$	Serial port 1 SDLC sample clock	I/O	I/O	Y	Z, 0, 1
$GPIO_{17}$	Serial port 1 SDLC abort after frame	O	I/O	Y	Z, 0, 1
$GPIO_{18}$	Serial port 1 UART sample clock	I	I/O	Y	Z, 0, 1
$GPIO_{25}$	1 Hz clock	O	I/O	Y	Z, 0, 1
$GPIO_{26}$	Internal clock ($\frac{1}{2}$ CPU core frequency)	O	I/O	Y	Z, 0, 1
$GPIO_{27}$	32.768 kHz clock	O	I/O	Y	Z, 0, 1

Table 5: Daughter-card general-purpose input/output signals. The "sleep" column describes the possible sleep-mode states, the symbol "Z" means that the corresponding pin is configured as input.

DCAD: **d**aughter-**c**ard general-purpose **a**nalog-to-**d**igital input
: This signal is a general-purpose analog input (10-bit analog-to-digital converter, nominal voltage range: [0 V .. 7.5 V]). It is directly connected to the AD_3 pin of the UCB1200 analog interface [Phi97b].

Daughter-cards are intended to form or replace the back of an Itsy computer. Systems with only a small number of thin components can be entirely packaged on the top side of the daughter-card PCB and, hence, fit in entirely in the case. On the other hand, systems that require more space can grow arbitrarily large on the bottom side of the daughter-card PCB. By sanding off the thin lip on the rear of the case, it is also possible to design daughter-card PCBs that are larger than the case. The mechanical specifications of daughter-cards is provided in Appendix A.

2.7.1 Static-memory identification scheme

To allow self-configuration, the software should be able to determine if a daughter-card is used or not and, if present, it should be able to recognize the daughter-card. A few resources available on the daughter-card (e.g., the DRAM banks) can be detected without any additional hardware, but this is not the case for most devices. Moreover, it is sometimes useful to only partially assemble a daughter-card (e.g., on a memory-extension daughter-card, only some of the available static-memory or DRAM banks may be present). In order to merge these requirements with the need to recognize which flash-memory circuits are used on the mother-board (see Section 2.5.1), a static-memory identification scheme — identical for all four static-memory banks of the StrongARM SA-1100 processor [DEC98b] — has been defined. Any daughter-card, using either static-memory bank 2 or 3 or using any otherwise undetectable resources (e.g., GPIO signals), should conform to this convention.

Several criteria were considered in the design of this mechanism:

- This scheme should be compatible with both 16-bit and 32-bit static-memory banks. That is, the configuration software should recognize a given device without prior knowledge of its width.
- Since the processor's endianess is programmable, this scheme should be defined independently of the current endianess.
- The amount of additional hardware should be minimized.

To meet these goals, each possible static-memory device should implement an 8-bit *class identification register* CID. Since there are only 256 possible values for this register, they should be assigned parsimoniously. Therefore, each value represents a class of devices (e.g., a single value is used for any non-volatile memory). Additional information, defined on a class basis, may be implemented to discriminate between different members of a class.

The address of the CID register should be fixed and easily decodable. The first word of a bank is not a good candidate, since this is the location of the reset vector in the boot static-memory bank. The first address following the exception vectors is not a good candidate either, since this is typically the position of the *fast interrupt request (FIQ)* handler routine. Hence, the last word of a bank has been adopted as the address for the CID register.

This leads to the following convention: each static-memory device should decode a read access at address $\mathsf{A}_{25\,..\,0} = \mathtt{3FFFFFE}_{16}$, with address bit A_0 ignored on a 16-bit device and address bits $\mathsf{A}_{1\,..\,0}$ ignored on a 32-bit device. The device should then provide the 8-bit class identification value CID on the data bits $\mathsf{D}_{7\,..\,0}$. Partial address decoding (or even no decoding) may be used as appropriate.[8]

The class identification value $\mathsf{CID} = 255 = \mathtt{FF}_{16}$ is reserved and will not be assigned to any static-memory device. With this knowledge, the configuration software can detect the absence of any device on a given bank, by driving the value $255 = \mathtt{FF}_{16}$ on the data bits $\mathsf{D}_{7\,..\,0}$ before reading the bank's CID register (the hardware is designed such that the last value driven on the data bus is preserved in the absence of any device driving the bus).

The class identification value $\mathsf{CID} = 0$ is assigned to any non-volatile memory, that is, *read-only memory (ROM)* or flash memory.[9] Additional information for this class is defined in Section 3.3. The CID register as well as all additional information can be simply programmed in the non-volatile memory. On many daughter-cards, it may be useful to implement a non-volatile memory for general

[8] Although only the 26 least significant address bits $\mathsf{A}_{25\,..\,0}$ are available on the pins of the StrongARM SA-1100 processor, a 128 Mbyte address space is internally allocated to each static-memory bank. Therefore, the offset of the CID register within the static-memory bank is a 27-bit value with address bit A_{26} being ignored.

[9] For the purpose of the configuration software, it is only useful to distinguish whether a non-volatile memory can be programmed in-circuit or not. Therefore, *programmable ROM (PROM)* and *erasable programmable ROM (EPROM)* are lumped in the ROM category. Likewise, *electrically-erasable programmable ROM (EEPROM)* are lumped in the flash-memory category.

use in addition to some other hardware (e.g., serial interface, sensors). In order to generalize the configuration software, any such static-memory device should be assigned a class identification value $\mathsf{CID} \leq 127 = 7\mathrm{F}_{16}$ (i.e., $\mathsf{CID}_7 = 0$) while any other device should have a class identification value $\mathsf{CID} > 128 = 80_{16}$ (i.e., $\mathsf{CID}_7 = 1$). Devices of the first category should implement the same additional information as non-volatile memories with the class identification value $\mathsf{CID} = 0$ (see Section 3.3).

The mother-board flash memory, implemented as static-memory bank 1, conforms to the same identification scheme, with the class identification value $\mathsf{CID} = 0$. Daughter-cards using static-memory banks 2 and 3 should implement both CID registers. Daughter-cards that do not use any static-memory bank but use other resources should dedicate one or both banks to the identification scheme. In this case, the CID register could easily be implemented using an 8-bit buffer.

3 Programmer's model

This section presents additional information on the model that the low-level software has of the Itsy hardware.

3.1 Memory map

The internal decoding of the StrongARM SA-1100 processor [DEC98b] provides a general template as how the address space is used. Table 6 shows the memory map implemented by the Itsy computer. The first column give the address range at which a particular device is decoded. The "location" column specifies whether this device is internal to the processor, is implemented on the mother-board, or whether its interface is available to the daughter-card interface. Finally, the last two columns give the device's width and size (or possible widths and sizes, when several different devices can be implemented).

As mentioned in Section 2.5, static-memory bank 0 — from which the processor boots — mirrors either bank 1 or bank 2 (see Sections 2.4, 2.7, and 3.2).

All DRAM banks must have 2^{12} rows, as imposed by DRAM bank 0 (see Section 2.5.2). This limits the size of banks 1, 2, and 3 to 16 Mbyte maximum (i.e., the largest size supported by the processor without external hardware). Using different DRAM circuits, 4 Mbyte and 8 Mbyte banks can also be implemented. Smaller banks can be considered as well. However, they are mapped in a non-contiguous address space.

3.2 Mother-board general-purpose input/output signals

Table 7 shows the general-purpose input/output signals used on the Itsy mother-board. All input signals can be used as interrupts. After a hardware reset (i.e., power-up or push-button reset), all input/output signals are configured as input. Therefore, pull-up and pull-down resistors are used to provide a default value for all signals meant to be configured as output, as shown in the last column of Table 7. The function of all signals are:

Address range	Location	Device	Width [bit]	Size
00000000_{16} $07FFFFFF_{16}$		Static-memory bank 0 mirror of bank 1 or 2 (boot memory)		
08000000_{16} $0FFFFFFF_{16}$	Mother-board	Static-memory bank 1 flash memory	32	4 Mbyte (max.)
10000000_{16} $17FFFFFF_{16}$	Daughter-card	Static-memory bank 2	16/32	64 Mbyte (max.)
18000000_{16} $1FFFFFFF_{16}$	Daughter-card	Static-memory bank 3	16/32	64 Mbyte (max.)
20000000_{16} $3FFFFFFF_{16}$	Daughter-card	PCMCIA interface [DEC98b]	8/16/32	
40000000_{16} $7FFFFFFF_{16}$	Reserved			
80000000_{16} $BFFFFFFF_{16}$	Processor (internal)	StrongARM SA-1100 registers [DEC98b]	32	
$C0000000_{16}$ $C7FFFFFF_{16}$	Mother-board	DRAM bank 0	32	16 Mbyte
$C8000000_{16}$ $CFFFFFFF_{16}$	Daughter-card	DRAM bank 1	32	16 Mbyte (max.)
$D0000000_{16}$ $D7FFFFFF_{16}$	Daughter-card	DRAM bank 2	32	16 Mbyte (max.)
$D8000000_{16}$ $DFFFFFFF_{16}$	Daughter-card	DRAM bank 3	32	16 Mbyte (max.)
$E0000000_{16}$ $E7FFFFFF_{16}$	Processor (internal)	Zero bank (read only) [DEC98b]	32	128 Mbyte
$E8000000_{16}$ $FFFFFFFF_{16}$	Reserved			

Table 6: Memory map.

GPIO$_0$: $\overline{\text{MAINPB}}$: **Main push-button**

This input signal is connected to the main push-button (s02) of the Itsy computer (see Section 2.6.5). It is set to 0 when the push-button is pressed and to 1 when the push-button is released. Although this is, otherwise, a general-purpose push-button, it can be used to wake up the processor after a hardware transition to sleep mode due to the assertion of the signal BATT_FAULT or the signal VDD_FAULT (see Section 2.2.1).

GPIO$_2$: PWROK: **Power OK**

This input signal is set to 1 during normal operation and toggles to 0 if the main power-supply voltage V_{dd} drops below the threshold $V_{\mathrm{dd,low}}$ (i.e., $V_{\mathrm{dd}} \le V_{\mathrm{dd,low}} \approx 3.0\,\mathrm{V}$), given in Table 1 (see Section 2.2.1).

Signal	Name	Function	Dir.	Def.
$GPIO_0$	$\overline{\text{MAINPB}}$	**Main p**ush-**b**utton	I	
$GPIO_2$	PWROK	**Pow**er **OK**	I	
$GPIO_3$	$\overline{\text{FLFOFF}}$	**Fl**ash memory **f**orce **off**	O	1
$GPIO_4$	FL0RY/$\overline{\text{BY}}$	**Fl**ash memory **0 r**ead**y**/**b**us**y**	I	
$GPIO_5$	FL1RY/$\overline{\text{BY}}$	**Fl**ash memory **1 r**ead**y**/**b**us**y**	I	
$GPIO_6$	CODECINT	**Codec int**errupt	I	
$GPIO_7$	UARTVALID	**UART** receive signal **valid**	I	
$GPIO_8$	UARTDSR	**UART d**ata **s**et **r**eady	I	
$GPIO_9$	UARTDTR	**UART d**ata **t**erminal **r**eady	O	0
$GPIO_{19}$	DCEN	**D**aughter-**c**ard **en**able	O	1
$GPIO_{20}$	LCDEN	**LCD en**able	O	0
$GPIO_{21}$	AUXLCDEN	**Aux**iliary **LCD** controller **en**able	O	0
$GPIO_{22}$	$\overline{\text{IRDAEN}}$	**IrDA** transceiver **en**able	O	1
$GPIO_{23}$	$\overline{\text{UARTFOFF}}$	**UART f**orce **off**	O	0
$GPIO_{24}$	UARTFON	**UART f**orce **on**	O	0

Table 7: Mother-board general-purpose input/output signals.

$GPIO_3$: $\overline{\text{FLFOFF}}$: **Fl**ash memory **f**orce **off**

This output signal is used to control the reset/power-down pin of the flash-memory circuits (see Section 2.5.1). When this signal is set to 0, the reset/power-down pins are asserted (0). When it is set to 1, the reset/power-down pins are only asserted (0) during a reset (i.e., hardware, software, or watch-dog reset) and during sleep mode. A pull-up resistor sets the default value of this signal to 1 when the $GPIO_3$ pin is configured as input (e.g., after a hardware reset). When the Itsy computer must boot from the mother-board, this signal should never be set to 0 during sleep mode, since the processor would be unable to read the boot memory upon wake up. This can be easily achieved by setting bit 3 of the *power manager GPIO sleep state register* PGSR of the StrongARM SA-1100 processor [DEC98b] to 1.

$GPIO_4$: FL0RY/$\overline{\text{BY}}$: **Fl**ash memory **0 r**ead**y**/**b**us**y**

This input signal is connected to the ready/not-busy pin of the flash-memory circuit used for the least significant 16 data bits $D_{15..0}$ (see Section 2.5.1). It is set to 0 when the flash-memory circuit is executing an erase or program operation and to 1 when it is ready for use.

$GPIO_5$: FL1RY/$\overline{\text{BY}}$: **Fl**ash memory **1 r**ead**y**/**b**us**y**

This input signal is connected to the ready/not-busy pin of the flash-memory circuit used for the most significant 16 data bits $D_{31..16}$ (see Section 2.5.1). It is set to 0 when the flash-memory circuit is executing an erase or program operation and to 1 when it is ready for use.

GPIO_6: CODECINT: **Codec int**errupt

This input signal is connected to the interrupt pin IRQOUT of the UCB1200 analog interface [Phi97b]. It is set to 1 when an interrupt is pending and to 0 otherwise.

GPIO_7: UARTVALID: **UART** receive signal **valid**

This input signal is connected to the $\overline{\mathsf{INVALID}}$ pin of the MAX3223CAP RS-232 driver [Max96], used with the UART engine of serial port 3 (see Section 2.6.3). It is set to 1 when a valid RS-232 input signal is detected (i.e., the serial interface is connected to a powered system) and to 0 when no valid signals are detected (i.e., the serial interface is unconnected or is connected to a unpowered system).

GPIO_8: UARTDSR: **UART d**ata **s**et **r**eady

This input signal is connected to the second receive buffer pin R2OUT of the MAX3223CAP RS-232 driver [Max96], used with the UART engine of serial port 3 (see Section 2.6.3). It can be used by software to monitor one of the RS-232 incoming signals, for example, the *data set ready* signal DSR. The exact signal that is monitored is determined by the serial-interface cable used.

GPIO_9: UARTDTR: **UART d**ata **t**erminal **r**eady

This output signal is connected to the second transmit buffer of the MAX3223CAP RS-232 driver [Max96], used with the UART engine of serial port 3 (see Section 2.6.3). A pull-down resistor sets the default value of this signal to 0 when the GPIO_9 pin is configured as input (e.g., after a hardware reset). It can be used by software to emulate one of the RS-232 outgoing signals, for example, the *data terminal ready* signal DTR. The exact signal that is emulated is determined by the serial-interface cable used.

GPIO_{19}: DCEN: **D**aughter-**c**ard **en**able

This output signal is used to control the SN74LVCH16244ADGG drivers [TI97a] and the SN74LVCH16245ADGG transceivers [TI97b] used to buffer some daughter-card signals (see Section 2.7). These buffers are enabled when this signal is set to 1 and disabled when it is set to 0. A pull-up resistor sets the default value of this signal to 1 when the GPIO_{19} pin is configured as input (e.g., after a hardware reset). When the Itsy computer should boot from the daughter-card, this signal should not be set to 0 during sleep mode, since the processor would be unable to access the daughter-card upon wake up. This can be easily achieved by setting bit 19 of the *power manager GPIO sleep state register* PGSR of the StrongARM SA-1100 processor [DEC98b] to 1. On the other hand, this signal can be de-asserted (0) during sleep mode, in order to force the processor to boot from the mother-board even if a bootable daughter-card is present (see Sections 2.4 and 2.7).

GPIO_{20}: LCDEN: **LCD en**able

This output signal is connected to the enable pins $\overline{\mathsf{DOFF1}}$ and $\overline{\mathsf{DOFF2}}$ of the TCM-A0822-xx LCD [Eps97] (see Section 2.6.1). The LCD is enabled when this signal is set to 1 and disabled when it is set to 0. A pull-down resistor sets the default value of this signal to 0 when the GPIO_{20} pin is configured as input (e.g., after a hardware reset).

GPIO_{21}**:** AUXLCDEN**: Aux**iliary **LCD** controller **en**able

This output signal is used to control the auxiliary LCD controller (see Sections 2.4 and 2.6.1). The auxiliary LCD controller is enabled when this signal is set to 1 and disabled when it is set to 0. A pull-down resistor sets the default value of this signal to 0 when the GPIO_{21} pin is configured as input (e.g., after a hardware reset). A black-and-white image (no grey levels) is captured at the first end-of-frame that the value of this signal is 1 and remains displayed as long as this signal and the signal GPIO_{20} (LCDEN) both stay at 1, even when the processor is in sleep mode.

GPIO_{22}**:** $\overline{\mathsf{IRDAEN}}$**: IrDA** transceiver **en**able

This output signal is connected to the $\overline{\mathsf{SD}}$ pin of the MiniSIR2 IrDA transceiver [Nov98] (see Section 2.6.4). The transceiver is enabled when this signal is set to 0 and disabled when it is set to 1. A pull-up resistor sets the default value of this signal to 1 when the GPIO_{22} pin is configured as input (e.g., after a hardware reset).

GPIO_{23}**:** $\overline{\mathsf{UARTFOFF}}$**: UART f**orce **off**

This output signal is connected to the control pin $\overline{\mathsf{FORCEOFF}}$ of the MAX3223CAP RS-232 driver [Max96], used with the UART engine of serial port 3 (see Section 2.6.3). When this signal is set to 0, the driver is disabled. When it is set to 1, the driver is either enabled or in AutoShutdown mode, depending on the value of the signal GPIO_{24} (UARTFON). A pull-down resistor sets the default value of this signal to 0 when the GPIO_{23} pin is configured as input (e.g., after a hardware reset).

GPIO_{24}**:** UARTFON**: UART f**orce **on**

This output signal is connected to the control pin FORCEON of the MAX3223CAP RS-232 driver [Max96], used with the UART engine of serial port 3 (see Section 2.6.3). When the signal GPIO_{23} ($\overline{\mathsf{UARTFOFF}}$) is set to 0, this signal has no effect. Otherwise, the driver is enabled when this signal is set to 1 and is in AutoShutdown mode when it is set to 0. A pull-down resistor sets the default value of this signal to 0 when the GPIO_{24} pin is configured as input (e.g., after a hardware reset). In AutoShutdown mode, the driver is only enabled when a valid RS-232 input signal is detected (i.e., the serial interface is connected to a powered system) and is disabled when no valid signals are detected (i.e., the serial interface is unconnected or is connected to a unpowered system). When the driver is disabled in AutoShutdown mode, transmitting data will not enable it. As a consequence, a potential dead-lock might happen when the serial interface is connected to another similar interface (e.g., when two Itsy computers are connected together), since both drivers might remain disabled. To avoid this, the software should briefly set this signal to 1 when establishing a connection. The driver can then be switched to AutoShutdown mode by toggling this signal to 0.

3.3 Non-volatile memory identification structure

As mentioned in Section 2.7.1, the class identification value $\mathsf{CID} = 0$ is assigned to any non-volatile memory (i.e., ROM or flash memory). Moreover, any daughter-card that include a non-volatile memory in addition to some other hardware should be assigned a class identification value $\mathsf{CID} \leq 127 = \mathbf{7F}_{16}$ (i.e., $\mathsf{CID}_7 = 0$). All these devices must implement the *non-volatile memory identification structure* in the last 32 bytes of the non-volatile memory.

In addition to the class identification value CID, these devices must provide the following information on the data bits $\mathsf{D}_{15..0}$, when a read access is decoded at address $\mathsf{A}_{25..0} = \mathbf{3FFFFFE}_{16}$ (with address bit A_0 ignored on a 16-bit devices and address bits $\mathsf{A}_{1..0}$ ignored on a 32-bit devices):

$\mathsf{A}_{25..0}$	15 … 11	10	9	8	7 … 0
$\mathbf{3FFFFFE}_{16}$	SIZE	W	D	R	$\mathsf{CID} \leq 127 = \mathbf{7F}_{16}$

Where the additional fields have the following meaning:

R: **R**ead-only static memory
A value of 0 indicates a flash memory, while a value of 1 indicates a ROM.

D: **D**aughter-card static memory
A value of 0 indicates a non-volatile memory on the mother-board, while a value of 1 indicates a non-volatile memory on a daughter-card. This bit should be set to 0 for static-memory bank 1 and to 1 for banks 2 and 3. It can be used to determine whether bank 0 mirrors bank 1 or bank 2 (see Sections 2.4, 2.7, and 3.2).

W: Static-memory **w**idth
A value of 0 indicates a 32 bit non-volatile memory, while a value of 1 indicates a 16 bit non-volatile memory.

SIZE: Base-2 logarithm of static-memory **size**
This field provides the logarithm in base 2 of the non-volatile memory's size, expressed in bytes.

The rest of the non-volatile memory identification structure contains the initialization values of the *static memory control register* MSC0 or MSC1 of the StrongARM SA-1100 processor [DEC98b], for all possible clock frequencies, as well as the information about which general-purpose input/output signals are associated with the non-volatile memory (if any). Figures 5 and 6 show this structure for 16-bit and 32-bit devices respectively. The initialization software should copy the MSC value for the current CPU frequency f_{cpu} into the appropriate field of the MSC0 or MSC1 register. The fields defining the general-purpose input/output signals have the following meaning:

EN0: Static-memory **en**able **0**
This field provides the number of the output GPIO pin (if any) used to enable or disable the least significant 16 bits ($\mathsf{D}_{15..0}$) of the non-volatile memory. When no such pin is used, this field should be set to the special value $\mathsf{EN0} = 255 = \mathbf{FF}_{16}$. The non-volatile memory should be enabled when this pin is set to 1 and disabled when it is set to 0.

$\mathsf{RY}/\overline{\mathsf{BY}}\mathsf{0}$: Static-memory **r**ead**y**/**b**us**y** **0**
This field provides the number of the input GPIO pin (if any) used to monitor the ready or busy state of the least significant 16 bits ($\mathsf{D}_{15..0}$) of the non-volatile memory (i.e., flash memory).

$A_{25..0}$	15 … 8	7 … 0
$3FFFFE0_{16}$	MSC ($f_{cpu} = 16 \cdot f_{crystal} = 59.0$ MHz)	
$3FFFFE2_{16}$	MSC ($f_{cpu} = 20 \cdot f_{crystal} = 73.7$ MHz)	
$3FFFFE4_{16}$	MSC ($f_{cpu} = 24 \cdot f_{crystal} = 88.5$ MHz)	
$3FFFFE6_{16}$	MSC ($f_{cpu} = 28 \cdot f_{crystal} = 103.2$ MHz)	
$3FFFFE8_{16}$	MSC ($f_{cpu} = 32 \cdot f_{crystal} = 118.0$ MHz)	
$3FFFFEA_{16}$	MSC ($f_{cpu} = 36 \cdot f_{crystal} = 132.7$ MHz)	
$3FFFFEC_{16}$	MSC ($f_{cpu} = 40 \cdot f_{crystal} = 147.5$ MHz)	
$3FFFFEE_{16}$	MSC ($f_{cpu} = 44 \cdot f_{crystal} = 162.2$ MHz)	
$3FFFFF0_{16}$	MSC ($f_{cpu} = 48 \cdot f_{crystal} = 176.9$ MHz)	
$3FFFFF2_{16}$	MSC ($f_{cpu} = 52 \cdot f_{crystal} = 191.7$ MHz)	
$3FFFFF4_{16}$	MSC ($f_{cpu} = 56 \cdot f_{crystal} = 206.4$ MHz)	
$3FFFFF6_{16}$	Reserved	
$3FFFFF8_{16}$	Reserved	
$3FFFFFA_{16}$	$RY/\overline{BY}0$	EN0
$3FFFFFC_{16}$	Reserved	
$3FFFFFE_{16}$	SIZE | 1 | D | R	CID $\leq 127 = 7F_{16}$

Figure 5: Non-volatile memory identification structure for 16-bit devices.

When no such pin is used, this field should be set to the special value $RY/\overline{BY}0 = 255 = FF_{16}$. The ready state should be indicated by a value of 1 and the busy state by a value of 0.

EN1**:** Static-memory **en**able **1**

This field provides the number of the output GPIO pin (if any) used to enable or disable the most significant 16 bits ($D_{31..16}$) of the non-volatile memory. When no such pin is used, this field should be set to the special value $EN1 = 255 = FF_{16}$. The non-volatile memory should be enabled when this pin is set to 1 and disabled when it is set to 0.

$RY/\overline{BY}1$**:** Static-memory **r**ead**y**/**b**us**y** **1**

This field provides the number of the input GPIO pin (if any) used to monitor the ready or busy state of the most significant 16 bits ($D_{31..16}$) of the non-volatile memory (i.e., flash memory).

$A_{25..0}$	31 … 24	23 … 16	15 … 8	7 … 0
$\mathtt{3FFFFE0}_{16}$	MSC ($f_{\text{cpu}} = 16 \cdot f_{\text{crystal}} = 59.0\,\text{MHz}$)		MSC ($f_{\text{cpu}} = 20 \cdot f_{\text{crystal}} = 73.7\,\text{MHz}$)	
$\mathtt{3FFFFE4}_{16}$	MSC ($f_{\text{cpu}} = 24 \cdot f_{\text{crystal}} = 88.5\,\text{MHz}$)		MSC ($f_{\text{cpu}} = 28 \cdot f_{\text{crystal}} = 103.2\,\text{MHz}$)	
$\mathtt{3FFFFE8}_{16}$	MSC ($f_{\text{cpu}} = 32 \cdot f_{\text{crystal}} = 118.0\,\text{MHz}$)		MSC ($f_{\text{cpu}} = 36 \cdot f_{\text{crystal}} = 132.7\,\text{MHz}$)	
$\mathtt{3FFFFEC}_{16}$	MSC ($f_{\text{cpu}} = 40 \cdot f_{\text{crystal}} = 147.5\,\text{MHz}$)		MSC ($f_{\text{cpu}} = 44 \cdot f_{\text{crystal}} = 162.2\,\text{MHz}$)	
$\mathtt{3FFFFF0}_{16}$	MSC ($f_{\text{cpu}} = 48 \cdot f_{\text{crystal}} = 176.9\,\text{MHz}$)		MSC ($f_{\text{cpu}} = 52 \cdot f_{\text{crystal}} = 191.7\,\text{MHz}$)	
$\mathtt{3FFFFF4}_{16}$	MSC ($f_{\text{cpu}} = 56 \cdot f_{\text{crystal}} = 206.4\,\text{MHz}$)		Reserved	
$\mathtt{3FFFFF8}_{16}$	RY/$\overline{\text{BY}}$1	EN1	RY/$\overline{\text{BY}}$0	EN0
$\mathtt{3FFFFFC}_{16}$	Reserved		SIZE | 0 | D | R	CID $\leq 127 = \mathtt{7F}_{16}$

Figure 6: Non-volatile memory identification structure for 32-bit devices.

When no such pin is used, this field should be set to the special value RY/$\overline{\text{BY}}$1 $= 255 = \mathtt{FF}_{16}$. The ready state should be indicated by a value of 1 and the busy state by a value of 0.

When a single GPIO pin is used to enable a 32 bit non-volatile memory, the EN0 and EN1 fields should both provide the same value. The same applies to the RY/$\overline{\text{BY}}$0 and RY/$\overline{\text{BY}}$1 fields.

For example, the address $A_{25..0} = \mathtt{3FFFFF8}_{16}$ of the mother-board flash memory should be set to the value $D_{31..0} = 05030403_{16}$ (see Section 3.2).

References

[AMD97a] Advanced Micro Devices. *Am29LV400T/Am29LV400B: 4 Megabit (524,288 × 8-Bit/ 262,144 × 16-Bit) CMOS 3.0 Volt-Only Sector Architecture Flash Memory*, May 1997. Publication no. 20514, rev. B, amendment +1, preliminary.

[AMD97b] Advanced Micro Devices. *Am29LV200T/Am29LV200B: 2 Megabit (262,144 × 8-Bit/ 131,072 × 16-Bit) CMOS 3.0 Volt-Only Sector Architecture Flash Memory*, August 1997. Publication no. 20513, rev. C, amendment +1, preliminary.

[AMD97c] Advanced Micro Devices. *Am29LV800T/Am29LV800B: 8 Megabit (1,048,576 × 8-Bit/ 524,288 × 16-Bit) CMOS 3.0 Volt-Only, Sectored Flash Memory*, November 1997. Publication no. 20478, rev. D, amendment 0, preliminary.

[AMD97d] Advanced Micro Devices. *Am29LV800B: 8 Megabit (1 M × 8-Bit/512 K × 16-Bit) CMOS 3.0 Volt-Only Boot Sector Flash Memory*, December 1997. Publication no. 21490, rev. E, amendment 0, preliminary.

[AMD97e] Advanced Micro Devices. *Am29LV160B: 16 Megabit (2 M × 8-Bit/1 M × 16-Bit) CMOS 3.0 Volt-Only Boot Sector Flash Memory*, December 1997. Publication no. 21358, rev. F, amendment 0, preliminary.

[DEC98a] Digital Equipment Corporation, Maynard, MA (USA). *DIGITAL Semiconductor SA-1100 Microprocessor: Data Sheet*, January 1998. Document no. EC-R8XUA-TE.

[DEC98b] Digital Equipment Corporation, Maynard, MA (USA). *DIGITAL Semiconductor SA-1100 Microprocessor: Technical Reference Manual*, March 1998. Document no. EC-R5MTC-TE.

[Eps97] Seiko Epson. *Technical Literature TCM-A0822-6*, March 1997. Rev. 0.

[Hit97a] Hitachi. *HM5164160A Series, HM5165160A Series: 4194304-Word × 16-Bit Dynamic RAM*, January 1997. Document no. ADE-203-596 (Z), rev. 0.1, preliminary.

[Hit97b] Hitachi. *HM5164165A Series, HM5165165A Series: 4194304-Word × 16-Bit Dynamic RAM*, April 1997. Document no. ADE-203-453 (Z), rev. 0.4, preliminary.

[Hit97c] Hitachi. *HN29VT800 Series, HN29VB800 Series: 1048576-Word × 8-Bit / 524288-Word × 16-Bit CMOS Flash Memory*, April 1997. Document no. ADE-203-781A (Z), rev. 1.0.

[Hit97d] Hitachi. *HN29WT800 Series, HN29WB800 Series: 1048576-Word × 8-Bit / 524288-Word × 16-Bit CMOS Flash Memory*, May 1997. Document no. ADE-203-537A (Z), rev. 1.0.

[Max96] Maxim. *MAX3221/MAX3223/MAX3243: 1 μA Supply Current, True +3 V to +5.5 V RS-232 Transceivers with AutoShutdown™*, October 1996. Document no. 19-0306, rev. 4.

[Mot97] Motorola. *M29F800A3, M29F800A2: 8 M CMOS Flash Memory*, October 1997. Document no. M29F800A3/D, rev. 4.

[Nov98] Novalog, Costa Mesa, CA (USA). *SIRComm™ MiniSIR2™ 115.2 kbps IrDA® 1.0 Transceiver Module*, April 1998.

[Phi97a] Philips Semiconductors. *PZ3032 32-Macrocell CPLD: Data Sheet*, February 1997. Ic27 data handbook.

[Phi97b] Philips Semiconductors. *UCB1200 Advanced Modem/Audio Analog Front-End: Data Sheet*, August 1997. Preliminary specification.

[Sam97a] Samsung Electronics. *KM416V4000A, KM416V4100A CMOS DRAM: 4 M × 16 Bit CMOS Dynamic RAM with Fast Page Mode*, 1997.

[Sam97b] Samsung Electronics. *KM416V4004A, KM416V4104A CMOS DRAM: 4 M × 16 Bit CMOS Dynamic RAM with Extended Data Out*, 1997.

[Sam98a] Samsung Electronics. *KM416V4000B, KM416V4100B CMOS DRAM: 4 M × 16 Bit CMOS Dynamic RAM with Fast Page Mode*, 1998. Preliminary spec.

[Sam98b] Samsung Electronics. *KM416V4004B, KM416V4104B CMOS DRAM: 4 M × 16 Bit CMOS Dynamic RAM with Extended Data Out*, 1998. Preliminary spec.

[Sha97] Sharp. *LH28F800SG: 8 Mbit (512 Kbit × 16) SmartVoltage Flash Memory*, February 1997. Preliminary.

[Syn96] Synario Design Automation, Redmond, WA (USA). *ABEL-HDL Entry User Manual*, January 1996. Document no. 090-0649-001.

[Tha97] C. Thacker. *WindowSIL/WC User's Guide*. SRC, Digital Equipment Corporation, Palo Alto, CA (USA), February 1997.

[TI97a] Texas Instruments. *SN74LVCH16244A: 16-Bit Buffer/Driver with 3-State Outputs*, January 1997. Document no. SCAS313D.

[TI97b] Texas Instruments. *SN74LVCH16245A: 16-Bit Bus Transceiver with 3-State Outputs*, March 1997. Document no. SCES063E.

[Tos96] Toshiba. *Toshiba MOS Digital Integrated Circuit TC5165165AJS/AFTS-40 TC5165165AJS/AFTS-50 TC5165165AJS/AFTS-60 Silicon Gate CMOS: 4,194,304 Words × 16 Bit EDO (Hyper Page) Dynamic RAM*, October 1996. Tentative data.

A Daughter-card mechanical specifications

Figure 7 presents the mechanical specifications for daughter-cards. On the top side (i.e., connector side), no components should be placed outside the specified component keep-in area. Components whose height does not exceed 1.2 mm can be placed anywhere within the keep-in area. This rule is conservative, since higher components can be positioned in several sub-areas. However, it is beyond the scope of this document to specify these exceptions. Since the daughter-card forms the back of the Itsy computer, there are no restrictions as to where components can be placed on the bottom side.

The 3.4 mm hole is optional. Its aim is to provide an air channel to the back of the noise-canceling microphone. Since this feature improves the quality of the audio input, it is highly recommended to include it on any daughter-card (unless it jeopardizes the placement and/or routing of the daughter-card).

The daughter-card connector to be used with the Itsy mother-board version 1.5 is a Nais AXK5SA6075P model.[10]

Figure 8 specifies the pad layout for the connector.[11] The specified pad dimensions should correspond to the area available for soldering. That is, with PCB technologies using gasketed pads (i.e., the size of the solder mask openings are smaller than the size of the copper pads), the specified dimensions are for the solder mask openings, while with PCB technologies using non-gasketed pads (i.e., the size of the solder mask openings are larger than or equal to the size of the copper pads), these dimensions are those of the copper pads.

[10] The Itsy mother-board version 1.1 uses a different manufacturer and the mating connector to be used on its daughter-cards is a JAE WR-160PB-VF-1 model.

[11] This pad layout accommodates both the JAE WR-160PB-VF-1 and Nais AXK5SA6075P connectors to be used with the Itsy mother-board versions 1.1 and 1.5 respectively.

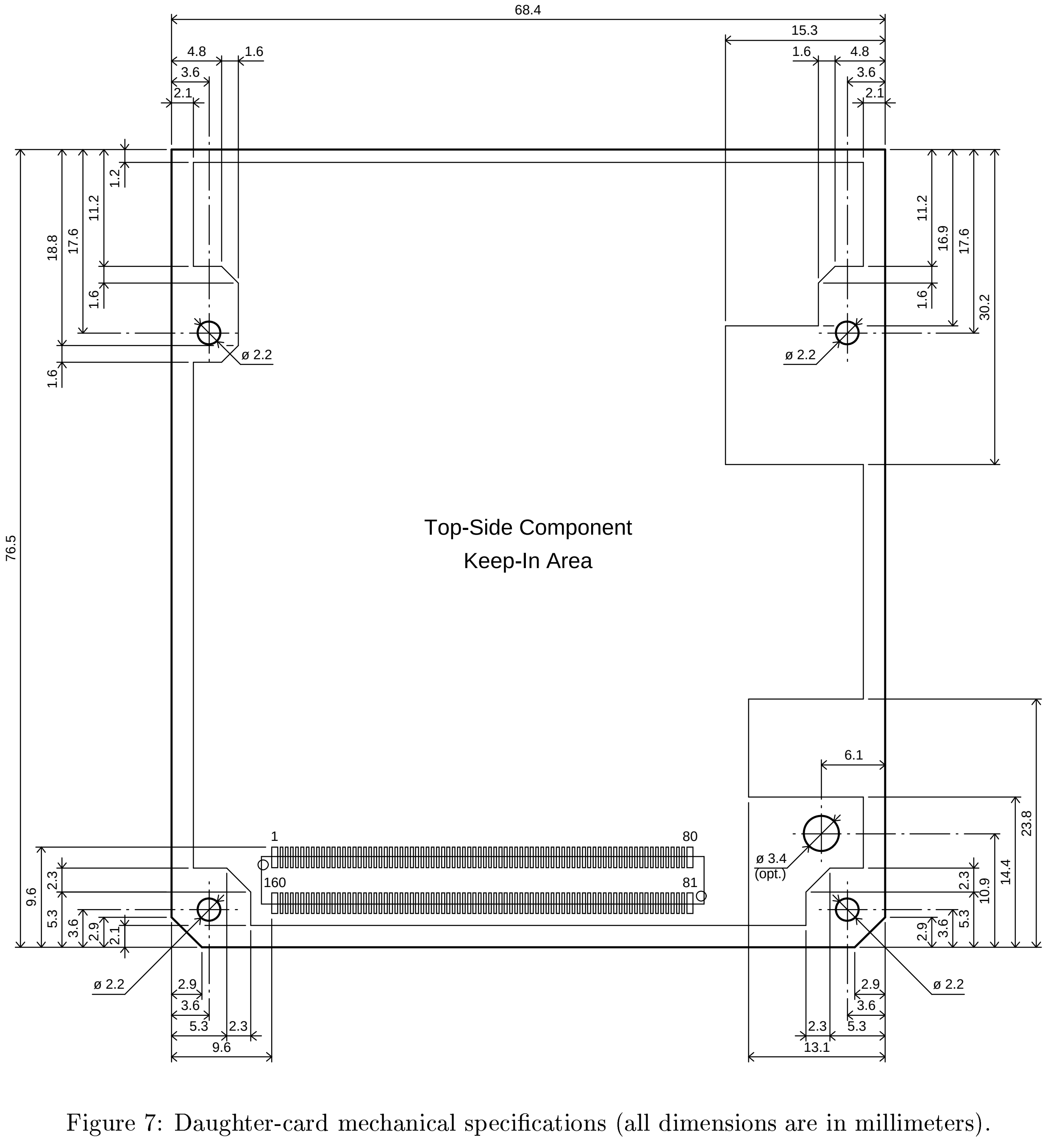

Figure 7: Daughter-card mechanical specifications (all dimensions are in millimeters).

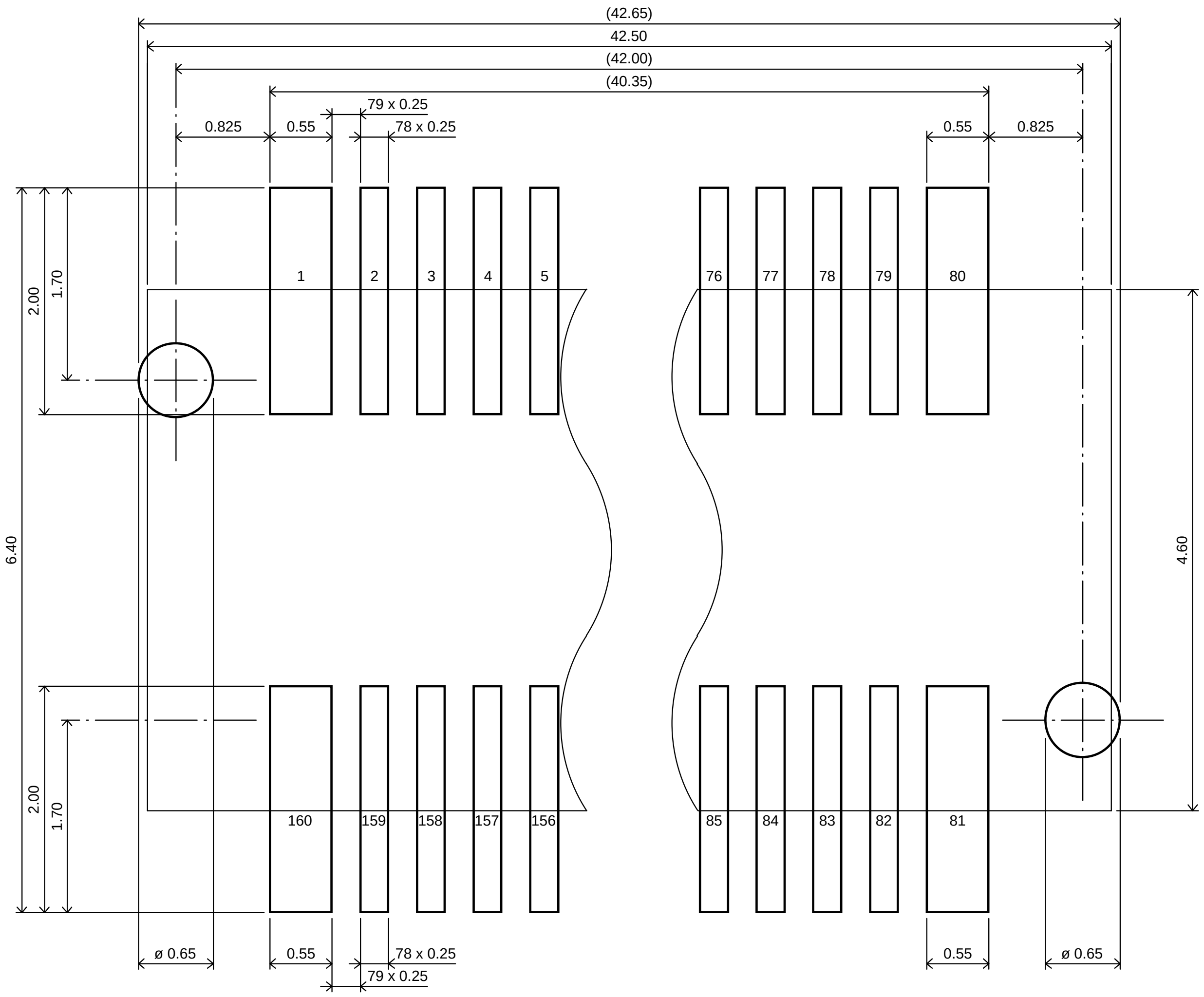

Figure 8: Pad layout specifications for the daughter-card connector (all dimensions are in millimeters).

WRL Research Reports

‘‘Titan System Manual.’’ **Michael J. K. Nielsen.** WRL Research Report 86/1, September 1986.

‘‘Global Register Allocation at Link Time.’’ **David W. Wall.** WRL Research Report 86/3, October 1986.

‘‘Optimal Finned Heat Sinks.’’ **William R. Hamburgen.** WRL Research Report 86/4, October 1986.

‘‘The Mahler Experience: Using an Intermediate Language as the Machine Description.’’ **David W. Wall and Michael L. Powell.** WRL Research Report 87/1, August 1987.

‘‘The Packet Filter: An Efficient Mechanism for User-level Network Code.’’ **Jeffrey C. Mogul, Richard F. Rashid, Michael J. Accetta.** WRL Research Report 87/2, November 1987.

‘‘Fragmentation Considered Harmful.’’ **Christopher A. Kent, Jeffrey C. Mogul.** WRL Research Report 87/3, December 1987.

‘‘Cache Coherence in Distributed Systems.’’ **Christopher A. Kent.** WRL Research Report 87/4, December 1987.

‘‘Register Windows vs. Register Allocation.’’ **David W. Wall.** WRL Research Report 87/5, December 1987.

‘‘Editing Graphical Objects Using Procedural Representations.’’ **Paul J. Asente.** WRL Research Report 87/6, November 1987.

‘‘The USENET Cookbook: an Experiment in Electronic Publication.’’ **Brian K. Reid.** WRL Research Report 87/7, December 1987.

‘‘MultiTitan: Four Architecture Papers.’’ **Norman P. Jouppi, Jeremy Dion, David Boggs, Michael J. K. Nielsen.** WRL Research Report 87/8, April 1988.

‘‘Fast Printed Circuit Board Routing.’’ **Jeremy Dion.** WRL Research Report 88/1, March 1988.

‘‘Compacting Garbage Collection with Ambiguous Roots.’’ **Joel F. Bartlett.** WRL Research Report 88/2, February 1988.

‘‘The Experimental Literature of The Internet: An Annotated Bibliography.’’ **Jeffrey C. Mogul.** WRL Research Report 88/3, August 1988.

‘‘Measured Capacity of an Ethernet: Myths and Reality.’’ **David R. Boggs, Jeffrey C. Mogul, Christopher A. Kent.** WRL Research Report 88/4, September 1988.

‘‘Visa Protocols for Controlling Inter-Organizational Datagram Flow: Extended Description.’’ **Deborah Estrin, Jeffrey C. Mogul, Gene Tsudik, Kamaljit Anand.** WRL Research Report 88/5, December 1988.

‘‘SCHEME->C A Portable Scheme-to-C Compiler.’’ **Joel F. Bartlett.** WRL Research Report 89/1, January 1989.

‘‘Optimal Group Distribution in Carry-Skip Adders.’’ **Silvio Turrini.** WRL Research Report 89/2, February 1989.

‘‘Precise Robotic Paste Dot Dispensing.’’ **William R. Hamburgen.** WRL Research Report 89/3, February 1989.

‘‘Simple and Flexible Datagram Access Controls for Unix-based Gateways.’’ **Jeffrey C. Mogul.** WRL Research Report 89/4, March 1989.

‘‘Spritely NFS: Implementation and Performance of Cache-Consistency Protocols.’’ **V. Srinivasan and Jeffrey C. Mogul.** WRL Research Report 89/5, May 1989.

‘‘Available Instruction-Level Parallelism for Superscalar and Superpipelined Machines.’’ **Norman P. Jouppi and David W. Wall.** WRL Research Report 89/7, July 1989.

‘‘A Unified Vector/Scalar Floating-Point Architecture.’’ **Norman P. Jouppi, Jonathan Bertoni, and David W. Wall.** WRL Research Report 89/8, July 1989.

‘‘Architectural and Organizational Tradeoffs in the Design of the MultiTitan CPU.’’ **Norman P. Jouppi.** WRL Research Report 89/9, July 1989.

‘‘Integration and Packaging Plateaus of Processor Performance.’’ **Norman P. Jouppi.** WRL Research Report 89/10, July 1989.

‘‘A 20-MIPS Sustained 32-bit CMOS Microprocessor with High Ratio of Sustained to Peak Performance.’’ **Norman P. Jouppi and Jeffrey Y. F. Tang.** WRL Research Report 89/11, July 1989.

‘‘The Distribution of Instruction-Level and Machine Parallelism and Its Effect on Performance.’’ **Norman P. Jouppi.** WRL Research Report 89/13, July 1989.

‘‘Long Address Traces from RISC Machines: Generation and Analysis.’’ **Anita Borg, R.E.Kessler, Georgia Lazana, and David W. Wall.** WRL Research Report 89/14, September 1989.

‘‘Link-Time Code Modification.’’ **David W. Wall.** WRL Research Report 89/17, September 1989.

‘‘Noise Issues in the ECL Circuit Family.’’ **Jeffrey Y.F. Tang and J. Leon Yang.** WRL Research Report 90/1, January 1990.

‘‘Efficient Generation of Test Patterns Using Boolean Satisfiablilty.’’ **Tracy Larrabee.** WRL Research Report 90/2, February 1990.

‘‘Two Papers on Test Pattern Generation.’’ **Tracy Larrabee.** WRL Research Report 90/3, March 1990.

‘‘Virtual Memory vs. The File System.’’ **Michael N. Nelson.** WRL Research Report 90/4, March 1990.

‘‘Efficient Use of Workstations for Passive Monitoring of Local Area Networks.’’ **Jeffrey C. Mogul.** WRL Research Report 90/5, July 1990.

‘‘A One-Dimensional Thermal Model for the VAX 9000 Multi Chip Units.’’ **John S. Fitch.** WRL Research Report 90/6, July 1990.

‘‘1990 DECWRL/Livermore Magic Release.’’ **Robert N. Mayo, Michael H. Arnold, Walter S. Scott, Don Stark, Gordon T. Hamachi.** WRL Research Report 90/7, September 1990.

‘‘Pool Boiling Enhancement Techniques for Water at Low Pressure.’’ **Wade R. McGillis, John S. Fitch, William R. Hamburgen, Van P. Carey.** WRL Research Report 90/9, December 1990.

‘‘Writing Fast X Servers for Dumb Color Frame Buffers.’’ **Joel McCormack.** WRL Research Report 91/1, February 1991.

‘‘A Simulation Based Study of TLB Performance.’’ **J. Bradley Chen, Anita Borg, Norman P. Jouppi.** WRL Research Report 91/2, November 1991.

‘‘Analysis of Power Supply Networks in VLSI Circuits.’’ **Don Stark.** WRL Research Report 91/3, April 1991.

‘‘TurboChannel T1 Adapter.’’ **David Boggs.** WRL Research Report 91/4, April 1991.

‘‘Procedure Merging with Instruction Caches.’’ **Scott McFarling.** WRL Research Report 91/5, March 1991.

‘‘Don’t Fidget with Widgets, Draw!.’’ **Joel Bartlett.** WRL Research Report 91/6, May 1991.

‘‘Pool Boiling on Small Heat Dissipating Elements in Water at Subatmospheric Pressure.’’ **Wade R. McGillis, John S. Fitch, William R. Hamburgen, Van P. Carey.** WRL Research Report 91/7, June 1991.

‘‘Incremental, Generational Mostly-Copying Garbage Collection in Uncooperative Environments.’’ **G. May Yip.** WRL Research Report 91/8, June 1991.

‘‘Interleaved Fin Thermal Connectors for Multichip Modules.’’ **William R. Hamburgen.** WRL Research Report 91/9, August 1991.

‘‘Experience with a Software-defined Machine Architecture.’’ **David W. Wall.** WRL Research Report 91/10, August 1991.

‘‘Network Locality at the Scale of Processes.’’ **Jeffrey C. Mogul.** WRL Research Report 91/11, November 1991.

‘‘Cache Write Policies and Performance.’’ **Norman P. Jouppi.** WRL Research Report 91/12, December 1991.

‘‘Packaging a 150 W Bipolar ECL Microprocessor.’’ **William R. Hamburgen, John S. Fitch.** WRL Research Report 92/1, March 1992.

‘‘Observing TCP Dynamics in Real Networks.’’ **Jeffrey C. Mogul.** WRL Research Report 92/2, April 1992.

‘‘Systems for Late Code Modification.’’ **David W. Wall.** WRL Research Report 92/3, May 1992.

‘‘Piecewise Linear Models for Switch-Level Simulation.’’ **Russell Kao.** WRL Research Report 92/5, September 1992.

‘‘A Practical System for Intermodule Code Optimization at Link-Time.’’ **Amitabh Srivastava and David W. Wall.** WRL Research Report 92/6, December 1992.

‘‘A Smart Frame Buffer.’’ **Joel McCormack & Bob McNamara.** WRL Research Report 93/1, January 1993.

‘‘Recovery in Spritely NFS.’’ **Jeffrey C. Mogul.** WRL Research Report 93/2, June 1993.

‘‘Tradeoffs in Two-Level On-Chip Caching.’’ **Norman P. Jouppi & Steven J.E. Wilton.** WRL Research Report 93/3, October 1993.

‘‘Unreachable Procedures in Object-oriented Programing.’’ **Amitabh Srivastava.** WRL Research Report 93/4, August 1993.

‘‘An Enhanced Access and Cycle Time Model for On-Chip Caches.’’ **Steven J.E. Wilton and Norman P. Jouppi.** WRL Research Report 93/5, July 1994.

‘‘Limits of Instruction-Level Parallelism.’’ **David W. Wall.** WRL Research Report 93/6, November 1993.

‘‘Fluoroelastomer Pressure Pad Design for Microelectronic Applications.’’ **Alberto Makino, William R. Hamburgen, John S. Fitch.** WRL Research Report 93/7, November 1993.

‘‘A 300MHz 115W 32b Bipolar ECL Microprocessor.’’ **Norman P. Jouppi, Patrick Boyle, Jeremy Dion, Mary Jo Doherty, Alan Eustace, Ramsey Haddad, Robert Mayo, Suresh Menon, Louis Monier, Don Stark, Silvio Turrini, Leon Yang, John Fitch, William Hamburgen, Russell Kao, and Richard Swan.** WRL Research Report 93/8, December 1993.

‘‘Link-Time Optimization of Address Calculation on a 64-bit Architecture.’’ **Amitabh Srivastava, David W. Wall.** WRL Research Report 94/1, February 1994.

‘‘ATOM: A System for Building Customized Program Analysis Tools.’’ **Amitabh Srivastava, Alan Eustace.** WRL Research Report 94/2, March 1994.

‘‘Complexity/Performance Tradeoffs with Non-Blocking Loads.’’ **Keith I. Farkas, Norman P. Jouppi.** WRL Research Report 94/3, March 1994.

‘‘A Better Update Policy.’’ **Jeffrey C. Mogul.** WRL Research Report 94/4, April 1994.

‘‘Boolean Matching for Full-Custom ECL Gates.’’ **Robert N. Mayo, Herve Touati.** WRL Research Report 94/5, April 1994.

‘‘Software Methods for System Address Tracing: Implementation and Validation.’’ **J. Bradley Chen, David W. Wall, and Anita Borg.** WRL Research Report 94/6, September 1994.

‘‘Performance Implications of Multiple Pointer Sizes.’’ **Jeffrey C. Mogul, Joel F. Bartlett, Robert N. Mayo, and Amitabh Srivastava.** WRL Research Report 94/7, December 1994.

‘‘How Useful Are Non-blocking Loads, Stream Buffers, and Speculative Execution in Multiple Issue Processors?.’’ **Keith I. Farkas, Norman P. Jouppi, and Paul Chow.** WRL Research Report 94/8, December 1994.

‘‘Drip: A Schematic Drawing Interpreter.’’ **Ramsey W. Haddad.** WRL Research Report 95/1, March 1995.

‘‘Recursive Layout Generation.’’ **Louis M. Monier, Jeremy Dion.** WRL Research Report 95/2, March 1995.

‘‘Contour: A Tile-based Gridless Router.’’ **Jeremy Dion, Louis M. Monier.** WRL Research Report 95/3, March 1995.

‘‘The Case for Persistent-Connection HTTP.’’ **Jeffrey C. Mogul.** WRL Research Report 95/4, May 1995.

‘‘Network Behavior of a Busy Web Server and its Clients.’’ **Jeffrey C. Mogul.** WRL Research Report 95/5, October 1995.

‘‘The Predictability of Branches in Libraries.’’ **Brad Calder, Dirk Grunwald, and Amitabh** Srivastava. WRL Research Report 95/6, October 1995.

‘‘Shared Memory Consistency Models: A Tutorial.’’ **Sarita V. Adve, Kourosh Gharachorloo.** WRL Research Report 95/7, September 1995.

‘‘Eliminating Receive Livelock in an Interrupt-driven Kernel.’’ **Jeffrey C. Mogul and K. K. Ramakrishnan.** WRL Research Report 95/8, December 1995.

‘‘Memory Consistency Models for Shared-Memory Multiprocessors.’’ **Kourosh Gharachorloo.** WRL Research Report 95/9, December 1995.

‘‘Register File Design Considerations in Dynamically Scheduled Processors.’’ **Keith I. Farkas, Norman P. Jouppi, Paul Chow.** WRL Research Report 95/10, November 1995.

‘‘Optimization in Permutation Spaces.’’ **Silvio** Turrini. WRL Research Report 96/1, November 1996.

‘‘Shasta: A Low Overhead, Software-Only Approach for Supporting Fine-Grain Shared Memory.’’ **Daniel J. Scales, Kourosh Gharachorloo, and Chandramohan A. Thekkath.** WRL Research Report 96/2, November 1996.

‘‘Efficient Procedure Mapping using Cache Line Coloring.’’ **Amir H. Hashemi, David R. Kaeli, and Brad Calder.** WRL Research Report 96/3, October 1996.

‘‘Optimizations and Placement with the Genetic Workbench.’’ **Silvio Turrini.** WRL Research Report 96/4, November 1996.

‘‘Memory-system Design Considerations for Dynamically-scheduled Processors.’’ **Keith I. Farkas, Paul Chow, Norman P. Jouppi, and Zvonko Vranesic.** WRL Research Report 97/1, February 1997.

‘‘Performance of the Shasta Distributed Shared Memory Protocol.’’ **Daniel J. Scales and Kourosh Gharachorloo.** WRL Research Report 97/2, February 1997.

‘‘Fine-Grain Software Distributed Shared Memory on SMP Clusters.’’ **Daniel J. Scales, Kourosh Gharachorloo, and Anshu Aggarwal.** WRL Research Report 97/3, February 1997.

‘‘Potential benefits of delta encoding and data compression for HTTP.’’ **Jeffrey C. Mogul, Fred Douglis, Anja Feldmann, and Balachander Krishnamurthy.** WRL Research Report 97/4, July 1997.

WRL Technical Notes

‘‘TCP/IP PrintServer: Print Server Protocol.’’ **Brian K. Reid and Christopher A. Kent.** WRL Technical Note TN-4, September 1988.

‘‘TCP/IP PrintServer: Server Architecture and Implementation.’’ **Christopher A. Kent.** WRL Technical Note TN-7, November 1988.

‘‘Smart Code, Stupid Memory: A Fast X Server for a Dumb Color Frame Buffer.’’ **Joel McCormack.** WRL Technical Note TN-9, September 1989.

‘‘Why Aren’t Operating Systems Getting Faster As Fast As Hardware?.’’ **John Ousterhout.** WRL Technical Note TN-11, October 1989.

‘‘Mostly-Copying Garbage Collection Picks Up Generations and C++.’’ **Joel F. Bartlett.** WRL Technical Note TN-12, October 1989.

‘‘Characterization of Organic Illumination Systems.’’ **Bill Hamburgen, Jeff Mogul, Brian Reid, Alan Eustace, Richard Swan, Mary Jo Doherty, and Joel Bartlett.** WRL Technical Note TN-13, April 1989.

‘‘Improving Direct-Mapped Cache Performance by the Addition of a Small Fully-Associative Cache and Prefetch Buffers.’’ **Norman P. Jouppi.** WRL Technical Note TN-14, March 1990.

‘‘Limits of Instruction-Level Parallelism.’’ **David W. Wall.** WRL Technical Note TN-15, December 1990.

‘‘The Effect of Context Switches on Cache Performance.’’ **Jeffrey C. Mogul and Anita Borg.** WRL Technical Note TN-16, December 1990.

‘‘MTOOL: A Method For Detecting Memory Bottlenecks.’’ **Aaron Goldberg and John** Hennessy. WRL Technical Note TN-17, December 1990.

‘‘Predicting Program Behavior Using Real or Estimated Profiles.’’ **David W. Wall.** WRL Technical Note TN-18, December 1990.

‘‘Cache Replacement with Dynamic Exclusion.’’ **Scott McFarling.** WRL Technical Note TN-22, November 1991.

‘‘Boiling Binary Mixtures at Subatmospheric Pressures.’’ **Wade R. McGillis, John S. Fitch, William R. Hamburgen, Van P. Carey.** WRL Technical Note TN-23, January 1992.

‘‘A Comparison of Acoustic and Infrared Inspection Techniques for Die Attach.’’ **John S. Fitch.** WRL Technical Note TN-24, January 1992.

‘‘TurboChannel Versatec Adapter.’’ **David Boggs.** WRL Technical Note TN-26, January 1992.

‘‘A Recovery Protocol For Spritely NFS.’’ **Jeffrey C. Mogul.** WRL Technical Note TN-27, April 1992.

‘‘Electrical Evaluation Of The BIPS-0 Package.’’ **Patrick D. Boyle.** WRL Technical Note TN-29, July 1992.

‘‘Transparent Controls for Interactive Graphics.’’ **Joel F. Bartlett.** WRL Technical Note TN-30, July 1992.

‘‘Design Tools for BIPS-0.’’ **Jeremy Dion & Louis Monier.** WRL Technical Note TN-32, December 1992.

‘‘Link-Time Optimization of Address Calculation on a 64-Bit Architecture.’’ **Amitabh Srivastava and David W. Wall.** WRL Technical Note TN-35, June 1993.

‘‘Combining Branch Predictors.’’ **Scott McFarling.** WRL Technical Note TN-36, June 1993.

‘‘Boolean Matching for Full-Custom ECL Gates.’’ **Robert N. Mayo and Herve Touati.** WRL Technical Note TN-37, June 1993.

‘‘Piecewise Linear Models for Rsim.’’ **Russell Kao, Mark Horowitz.** WRL Technical Note TN-40, December 1993.

‘‘Speculative Execution and Instruction-Level Parallelism.’’ **David W. Wall.** WRL Technical Note TN-42, March 1994.

‘‘Ramonamap - An Example of Graphical Groupware.’’ **Joel F. Bartlett.** WRL Technical Note TN-43, December 1994.

‘‘ATOM: A Flexible Interface for Building High Performance Program Analysis Tools.’’ **Alan Eustace and Amitabh Srivastava.** WRL Technical Note TN-44, July 1994.

‘‘Circuit and Process Directions for Low-Voltage Swing Submicron BiCMOS.’’ **Norman P. Jouppi, Suresh Menon, and Stefanos Sidiropoulos.** WRL Technical Note TN-45, March 1994.

‘‘Experience with a Wireless World Wide Web Client.’’ **Joel F. Bartlett.** WRL Technical Note TN-46, March 1995.

‘‘I/O Component Characterization for I/O Cache Designs.’’ **Kathy J. Richardson.** WRL Technical Note TN-47, April 1995.

‘‘Attribute caches.’’ **Kathy J. Richardson, Michael J. Flynn.** WRL Technical Note TN-48, April 1995.

‘‘Operating Systems Support for Busy Internet Servers.’’ **Jeffrey C. Mogul.** WRL Technical Note TN-49, May 1995.

‘‘The Predictability of Libraries.’’ **Brad Calder, Dirk Grunwald, Amitabh Srivastava.** WRL Technical Note TN-50, July 1995.

‘‘Simultaneous Multithreading: A Platform for Next-generation Processors.’’ **Susan J. Eggers, Joel Emer, Henry M. Levy, Jack L. Lo, Rebecca Stamm and Dean M. Tullsen.** WRL Technical Note TN-52, March 1997.

‘‘Reducing Compulsory and Capacity Misses.’’ **Norman P. Jouppi.** WRL Technical Note TN-53, August 1990.

‘‘The Itsy Pocket Computer Version 1.5: User’s Manual.’’ **Marc A. Viredaz.** WRL Technical Note TN-54, July 1998.

‘‘The Memory Daughter-Card Version 1.5: User’s Manual.’’ **Marc A. Viredaz.** WRL Technical Note TN-55, July 1998.

WRL Research Reports and Technical Notes are available on the World Wide Web, from `http://www.research.digital.com/wrl/techreports/index.html`.