StrongARM** SA-1100 Multimedia Development Board with Companion SA-1101 Development Board

User's Guide

October 1998

Order Number: 278114-001

Figures

Tables

Overview 1

1.1 Purpose

This document describes the use and features for the Intel StrongARM** SA-1100 multimedia development and the SA-1101 development boards.

1.2 How to Use This Document

Chapter 2, “Introduction”defines the purpose of the SA-1100 multimedia development board and the optional SA-1101 development board.

All readers should read Chapter 3, “Getting Started” for information about how to connect and turn on power to the board(s), how to select various card modes, and how to connect it to a terminal or host system.

All readers are advised to read Chapter 4, “Hardware Overview” to get an understanding of the overall functionality of the SA-1100 multimedia development and the optional SA-1101 development boards. Subsequent chapters assume a familiarity with the material in that chapter.

Thereafter, software engineers will probably want to refer to the following chapters:

- Chapter 5, “Software Video-Processing Engine Driver” describes the software video-processing engine for the SA-1100 microprocessor.
- Chapter 6, “SCB Library”, describes a set of utilities for using the SA-1100 as an serial control bus (SCB) master.
- Chapter 7, “Firmware Modification” describes the firmware and applications for the SA-1100 multimedia development platform.

A number of appendixes provide general reference material:

- Appendix A, “SA-1100 Board Configuration Guide” describes all of the switches, LEDS, and the header pins on the SA-1100 multimedia development StrongARM** SA-1100 Multimedia Development Board with Companion SA-1101 Development Boardboard.
- Appendix B, “SA-1101 Board Configuration Guide” describes all of the header pins on the SA-1101 development board.
- Appendix C, “CPLD Source Code Process”, describes the resources and the process to program the CPLDs on the SA-1100 multimedia development and the SA-1101 development boards.

1.3 Related Documentation

Other documentation that may be helpful while reading this document are described in the following table:

Title	Location
SA-1100 Microprocessor Technical Reference Manual, order number: 278088	Intel's website for developers is at: http://developer.intel.com
*StrongARM** SA-1101 Microprocessor Technical Reference Manual*, order number: 278170[a]	
*StrongARM** SA-1100 Microprocessor Specification Update*, order number: 278105	
The ARM Debug Monitor: Angel	ARM's website is at: http://www.arm.com
ARM Architecture Reference Manual	

a. This document is helpful only if you are using the SA-1101 development board.

1.4 Conventions

This section defines product-specific terminology, abbreviations, and other conventions used throughout this manual.

1.4.1 Abbreviations

Term	Definition
ADC	Analog-to-digital converter
API	Application programming interface
Bt	Brooktree*
CC	Closed caption
CCIR	Comite Consultatif Internationale de Radio
CIF	Common interchange format (352 × 288)
CPLD	Complex programmable logic device
DSP	Digital signal processor
EDS	Extended data services
fps	Frames per second
IrDA	Infrared data acquisition
MB/s	Megabytes per second
NTSC	National Television Standards Committee (U.S. TV standard)
OS	Operating system
OSP	On-screen programming
PAL	Phased acquisition loop (European television standard)

Term	Definition
PCMCIA	Personal computer memory card interface architecture
PIO	Programmed I/O
PIP	Picture in picture
PIWP	Picture in windows picture
POTS	Plain old telephone system (video conferencing)
QCIF	Quarter CIF (176 × 144)
SCB	Serial Control Bus
SDK	Software development kit
SDT	Software developer's toolkit
SIF	Source input format (360x288)
VBI	Vertical blanking interval
VoIP	Voice over IP
YUV	Color format for PAL and NTSC

1.4.2 Numbering

All numbers are decimal or hexadecimal unless otherwise indicated. The prefix 0x indicates a hexadecimal number. For example, 19 is decimal, but 0x19 and 0x19A are hexadecimal (also see Addresses). Otherwise, the base is indicated by a subscript; for example, 100_2 is a binary number.

Unless otherwise noted, all addresses and offsets are hexadecimal.Binary Multiples

The abbreviations K, M, and G (kilo, mega, and giga) represent binary multiples and have the following values.

K 2^{10} (1024)
M 2^{20} (1,048,576)
G 2^{30} (1,073,741,824)

For example:

2 KB	2 kilobytes	2×2^{10} bytes
4 MB	4 megabytes	4×2^{20} bytes
8 GB	8 gigabytes	8×2^{30} bytes
2 K pixels	2 kilopixels	2×2^{10} pixels
4 M pixels	4 megapixels	4×2^{20} pixels

1.4.3 Bit Notation

Multiple-bit fields can include contiguous and noncontiguous bits contained in brackets ([]). Multiple contiguous bits are indicated by a pair of numbers separated by a colon (:). For example, [9:7,5,2:0] specifies bits 9,8,7,5,2,1, and 0. Similarly, single bits are frequently indicated with brackets. For example, [27] specifies bit 27.

1.4.4 Caution

Cautions indicate potential damage to equipment or loss of data.

1.4.5 Data Units

The following data unit terminology is used throughout this manual.

Term	Words	Bytes	Bits
Byte	½	1	8
Word[a]	1	2	16
Longword	2	4	32

a. A 32-bit quantity in some SA-1100 documentation.

Note: Notes emphasize particularly important information.

1.4.6 Signal Names

All signal names are printed in plain type with mixed case, as follows:

Example	Signal Type
NAME	SA-1100, high-active
nNAME	SA-1100, low-active (**n** prefix indicates low-active)
name	PCI and nonSA-1100, high-active
name#	PCI and nonSA-1100, low-active (**#** suffix indicates low-active)
NAME	Board or third-party device, high-active
NAME_L	Board or third-party device, low-active

Introduction 2

The SA-1100 microprocessor provides a true convergence machine for the appliance, telephony, and embedded computer market segments. The SA-1100 is a low-cost general-purpose microcomputer with a wide variety of I/O, performance optimizations, caching, and a flexible memory controller. All of these features enable it to input, manipulate, and output media streams. The SA-1100 is a 32-bit RISC microprocessor with a 16KB instruction cache, an 8KB write-back data cache, a minicache, a write buffer, a read buffer, and a memory-management unit (MMU) combined in a single device. For more information about the SA-1100 device, see the *SA-1100 Microprocessor Technical Reference Manual*.

The SA-1101 companion device provides additional I/O connections for devices and peripherals, including VGA graphics. These I/O functions enable complete systems to be built with very little "glue" logic. In addition to a complete video subsystem, the SA-1101 includes a USB host controller, extensive support for PCMCIA, two PS/2 ports and keypad support, a full-function parallel port, and other I/O capabilities. For more information about the SA-1101 device, see the *StrongARM** SA-1101 Microprocessor Technical Reference Manual*.

This document describes a multimedia development platform based on these devices, which consists of two circuit boards—the SA-1100 multimedia development board and the optional SA-1101 daughterboard. This platform has two purposes:

- General-purpose hardware and software development platform for the SA-1100 family.
- Multimedia convergence reference design.

2.1 General Development Platform

The SA-1100 multimedia development board provides a breadboard environment for hardware engineers with:

- Many probe points for access to key signals.
- Circuit routing that is extremely flexible through the use of programmable CPLDs. Sample routing for common multimedia functions are provided.
- Two connectors for mounting daughter cards, including one for the SA-1101 daughterboard.
- Memory that is tightly coupled to the SA-1100 to demonstrate maximum memory bandwidth of over 100 Mbytes/sec.
- Programmable CPU clock speeds up to 220 MHz.
- An example integration of video, audio, POTS communication, flash, and DSP co-processor on board.
- A platform that easily accommodates different I/O devices. For example, to not use the example I/O integration, you can:
  - — Keep the current FIFO structure, reprogram the CPLD, and use a different video decoder.
  - — Connect a video input directly to the memory bus, which changes the SA-1100 loading and incurs a small delay.

— Connect a video input to the X-bus, which does not changes the SA-1100 loading but incurs a larger delay.

For software engineers, the SA-1100 multimedia development board provides:

- Angel* Debug Kernel in boot flash compatible with version 2.11A of the ARM SDT.
- PCI form factor allows use in a PC chassis. This allows software development programming in the office as well as lab environments.
- Bulkhead serial interface connector on the board provides debug communications to the SA-1100. Ethernet interface optionally available through the use of PCMCIA Ethernet card on the SA-1101 daughterboard PCMCIA slot.
- Several board support packages available through OS suppliers.
- Sample source code including I/O drivers such as the software video-processing engine drivers and the SCB library available through Intel's developer's web site.

The SA-1101 development board is specifically designed to provide a breadboard environment for hardware engineers with:

- Many probe points that are provided for access to key signals.
- Expanded I/O connections for video, PCMCIA card connector, USB, DSUB and 1284/Parallel ports, PS/2 connections.
- Circuit routing that is extremely flexible through the use of programmable CPLDs. Sample routing for common functions are provided.
- Provides a hardware and software development environment for PCMCIA interfaces, IEEE 1284, and matrix keyboards.

For software engineers, the SA-1101 development board contains all of the system components necessary for a Windows CE* sub-notebook system development platform with two independent video heads.

2.2 Convergence Reference Design

The work instantiated by the SA-1100 multimedia development board is the basis for a full video reference design available from Intel and others that can include the configurable boards, drivers, OS, and application software for video I/O, audio I/O, telephony, web access, email, voice-over IP (VoIP), video conferencing, and similar applications. This reference design demonstrates the low-cost design desirable in the consumer appliance market.

Possible applications enabled by the SA-1100 multimedia reference design include, but is not limited to:

- Video phone with optional web and email access
- Standard digital television (SDTV) engine with capabilities such as PIP, OSD, video phone, and web access
- Voice-over-IP
- Security monitoring/remote security
- Electronic camera
- Wireless gateway

- Speech recognition
- Tapeless recorder

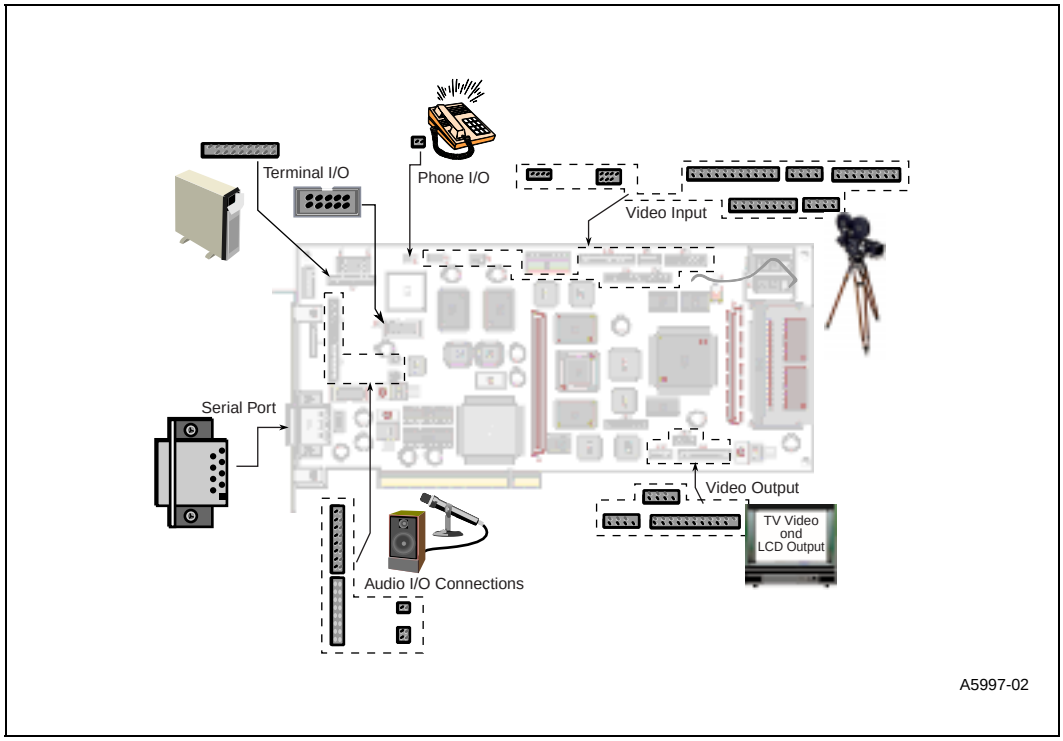
The capabilities demonstrated with the SA-1100 multimedia development reference design platform shows a true convergence of the computer, appliance and telephony market segments, yet the design is flexible enough to accommodate most design requirements. With the expanded I/O capabilities of the SA-1101 development platform, additional memory can be incorporated to optimize throughput.

2.3 Hardware Overview

The reference designs for the SA-1100 and the SA-1101 accept a multimedia input, store the information in memory, process the multimedia information and output the information for display. Both designs benefit from CPLDs that allows the logic to be reconfigured at run time.

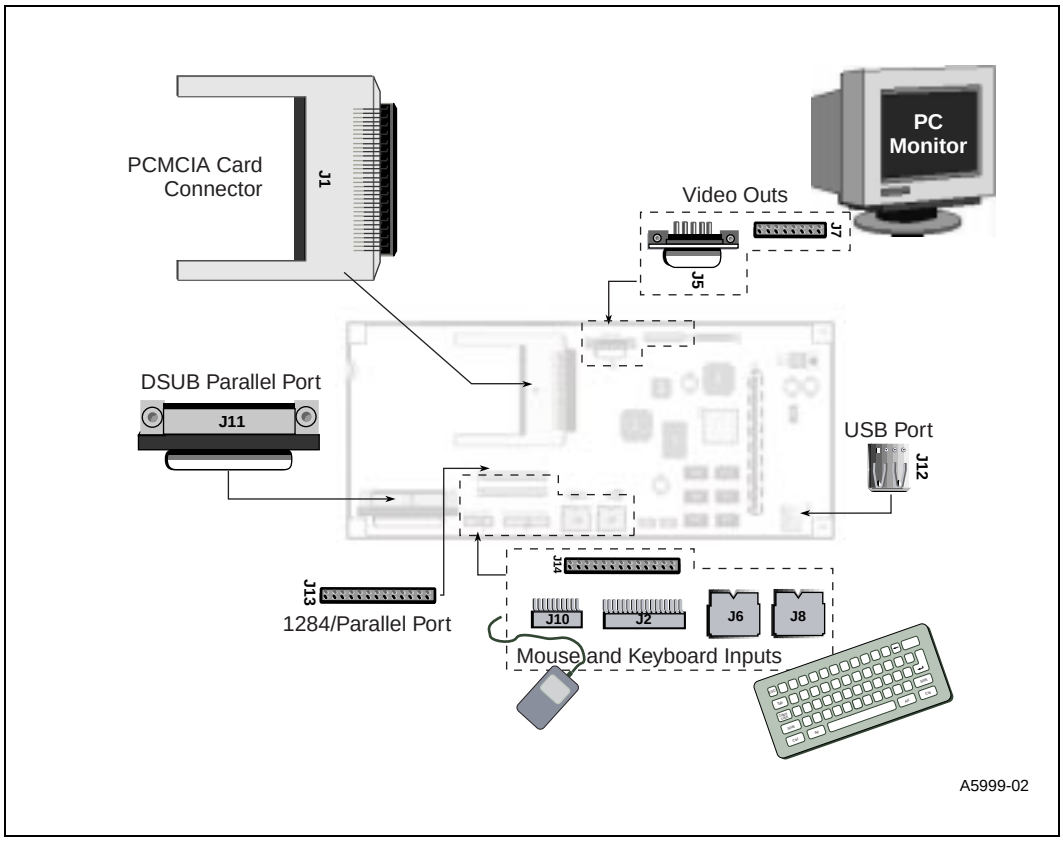
As shown in Figure 2-1, the SA-1100 has audio, video, phone, IrDA, and terminal connections, a serial port, and connectors for daughter cards, such as the SA-1101 development board.

Figure 2-1. SA-1100 Multimedia Development Board I/O Connections



As shown in Figure 2-2, the SA-1101 provides connections for a USB port, keyboard, mouse, touchpad, parallel ports, RGB and PCMCIA.

Figure 2-2. SA-1101 Development Board I/O Connections



2.4 Software Overview

As listed in Section 2.2, this reference design provides many opportunities for software engineers to develop applications. An example is the video pass-through application, which is provided with this reference design. Other software associated with this reference design is available from Intel and third parties.

Several multimedia software architectures are described in this section:

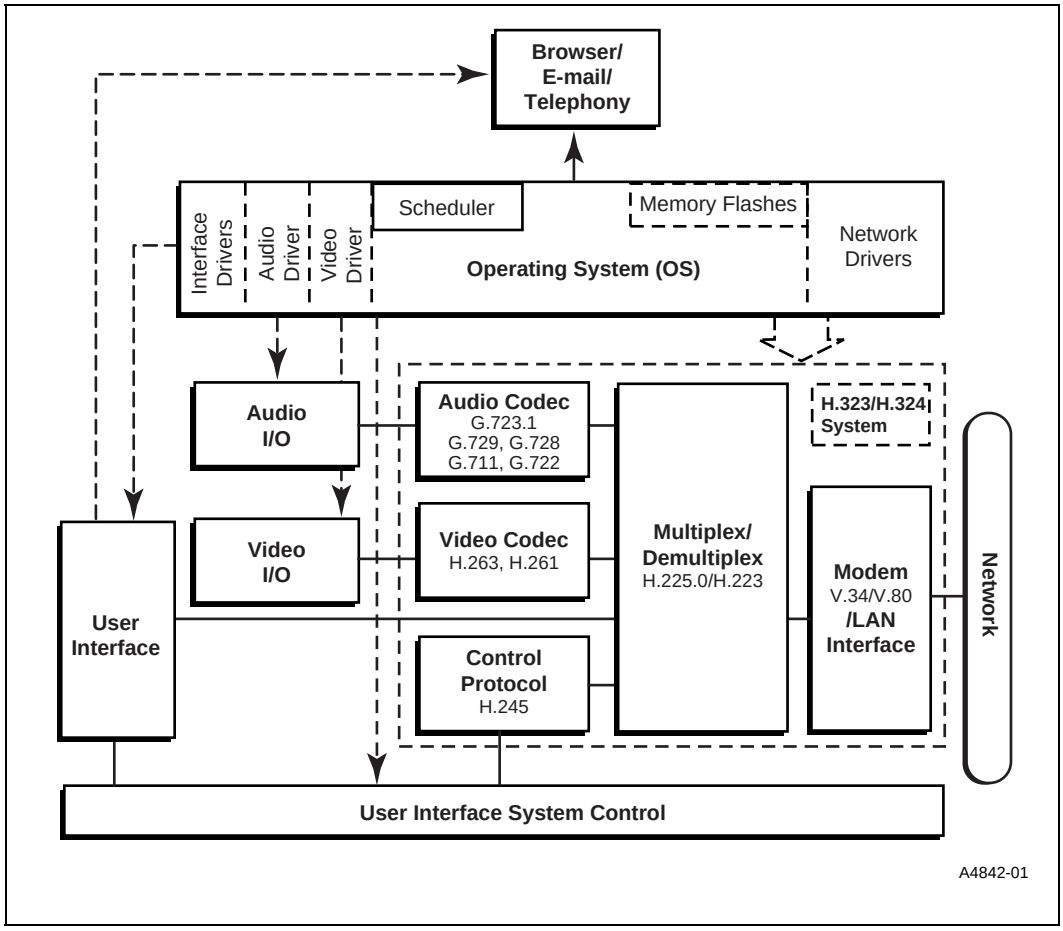
- Web-video phone and VoIP
- Remote security monitoring

2.4.1 Web-Video Phone and VoIP

A video phone with a web access option, as shown in Figure 2-3, can be built based on the reference board design (for an overview of the reference board design, see Chapter 4, “Hardware Overview”). The software package for the web-video phone application includes the OS, network and videoconferencing, and user interface.

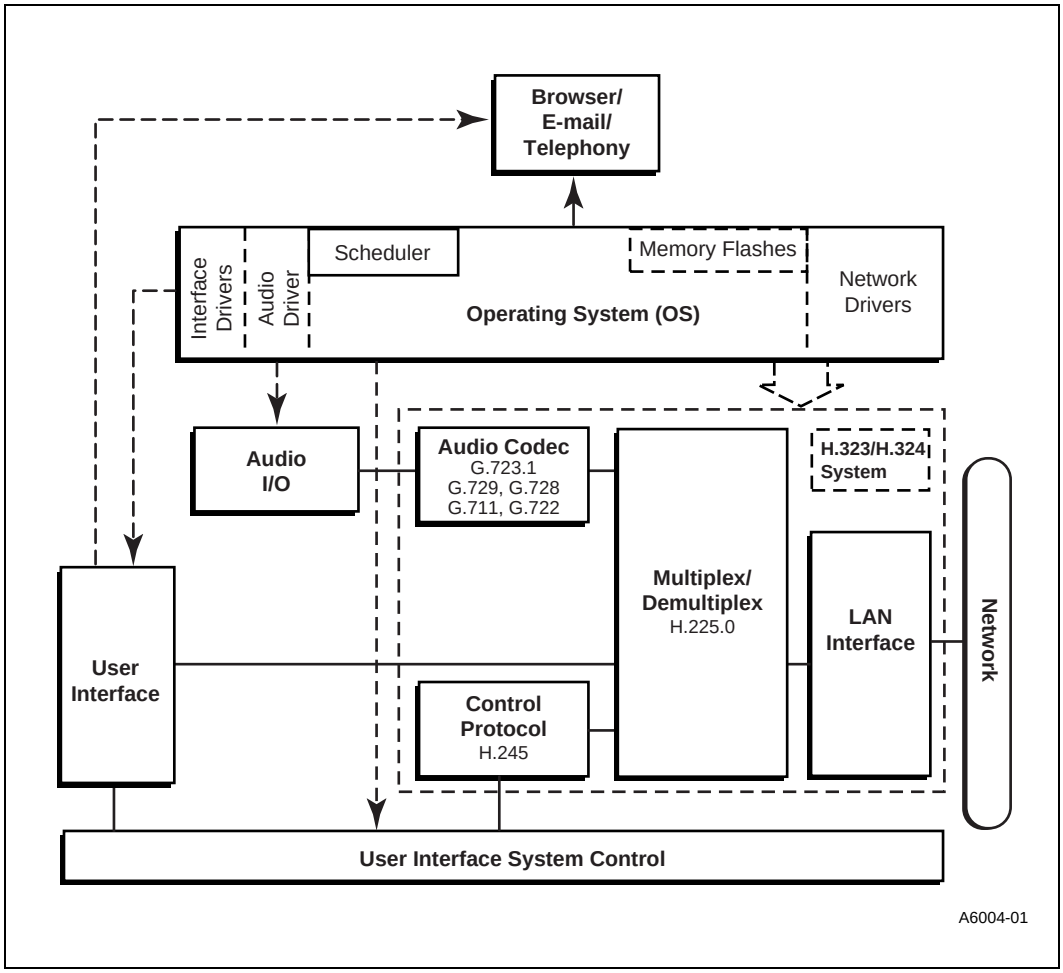
Depending on which type of networks will be used, the video codec, audio codec and multiplexer should be properly selected. For example, POTS line should use H.263 for video codec, G.723.1 for audio codec, V.34/V.80 for modem, and H.324 for system.

Figure 2-3. Web-Video Software



Using the same software package, a digital TV with web-video phone capability can also be built by adding a TV tuner, a digital video/audio decoder and a monitor. Alternately, removing the video I/O and video processing (compression) from the software stack provides a VoIP architecture, as shown in Figure 2-4.

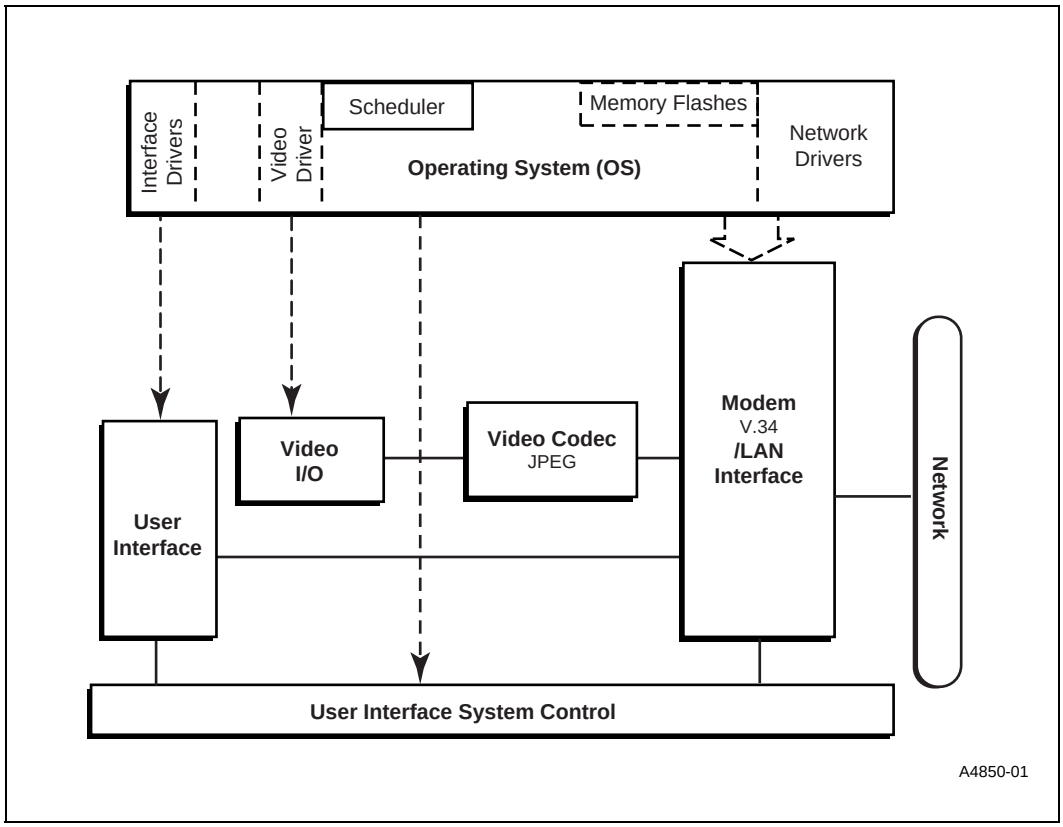
Figure 2-4. VoIP System



2.5 Remote Security Monitoring

As shown in Figure 2-5, the SA-1100 multimedia reference design can be used to build a remote security monitoring system. The software package for this application can be very limited, as it only requires an OS, network and JPEG, and a graphical user interface (GUI) as shown in Figure 2-5.

Figure 2-5. Security Camera System



Getting Started 3

The SA-1100 multimedia development board is supplied as a plug-in card, while the optional SA-1101 development board is supplied as a daughter card. This chapter provides a physical description of the cards and describes how to:

- Unpack the cards and give them a visual inspection
- Select the various card modes
- Configure the cards to suit your application
- Power up the cards for the first time

3.1 Physical Description

Figure 3-1 shows the physical layout of the SA-1100 multimedia development board. It is a single-board computer with the form factor of a PCI add-in card. Figure 3-2 show the physical layout of the optional SA-1101 development board. It is a single-board I/O expander for the SA-1100 with the form factor of a daughter card.

This board contains processor, system controller, memory and input/output devices. Component S1 is used for configuring the card for ROM selection and disabling interrupts.

The bulkhead mounting bracket of the board holds a female 9-way D-type connector and an IrDA header block. The D-type connector provides an RS232 terminal connection to a host system.

The CPLDs, which can be programmed to provide a variety of hardware functions, are pre-programmed to support sample designs. The CPLDs are also capable of being programmed at run-time to support a variety of hardware needs. The in circuit programmability feature of these devices allows SA-1100 system developers to tailor the SA-1100 multimedia design to their specific application. The ability to quickly port a new hardware application to an existing and supported platform allows rapid deployment of software development platforms for the customer application. (Typically software schedules are more critical to time to market than hardware schedules, so the early deployment of functional software development platforms can reduce product delivery schedules.)

The discrete LEDs and the HexLEDs provide status information during power up and self-test. See the README.TXT for interpretation of the LEDs and hexleds.

Figure 3-1. SA-1100 Multimedia Development Board

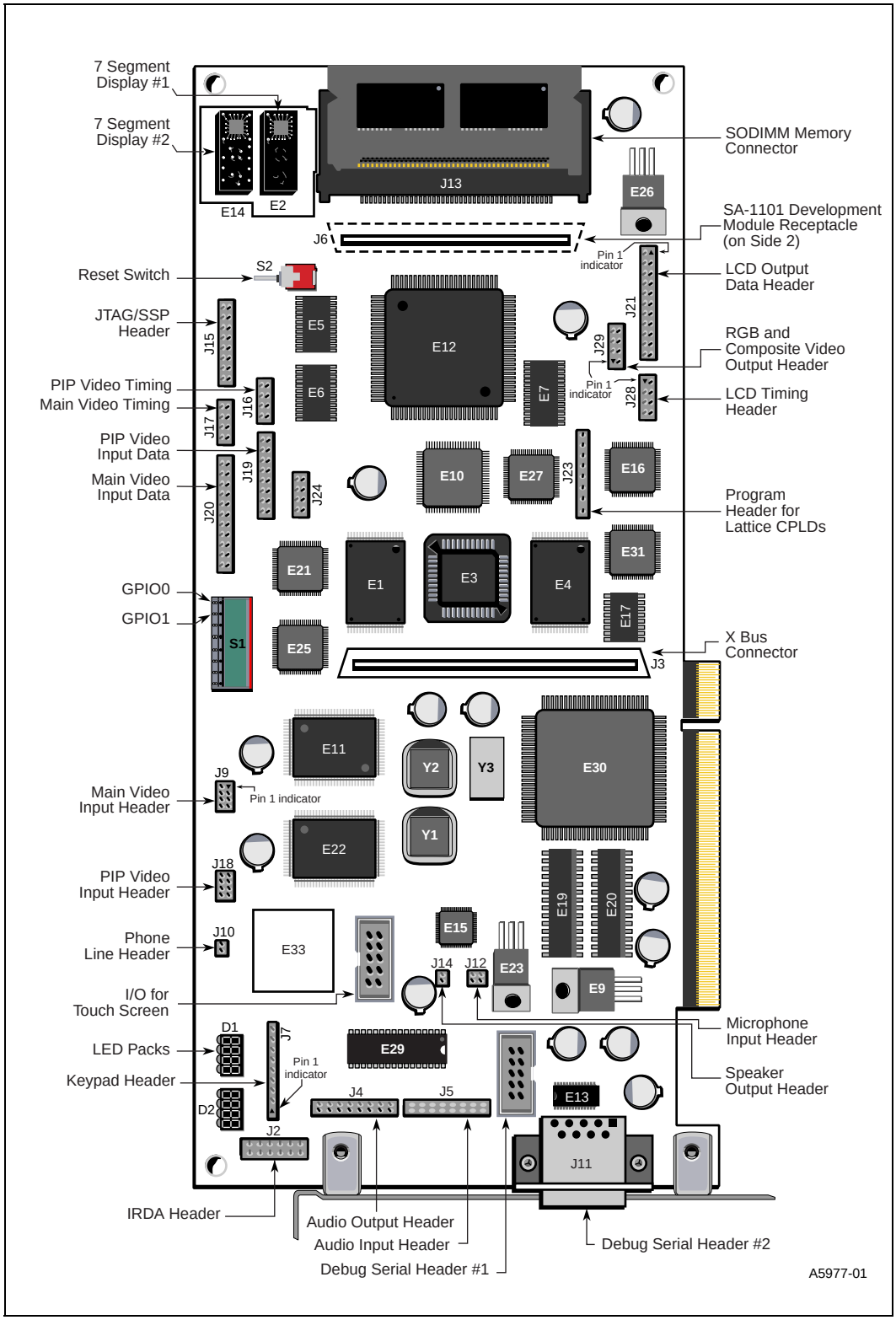
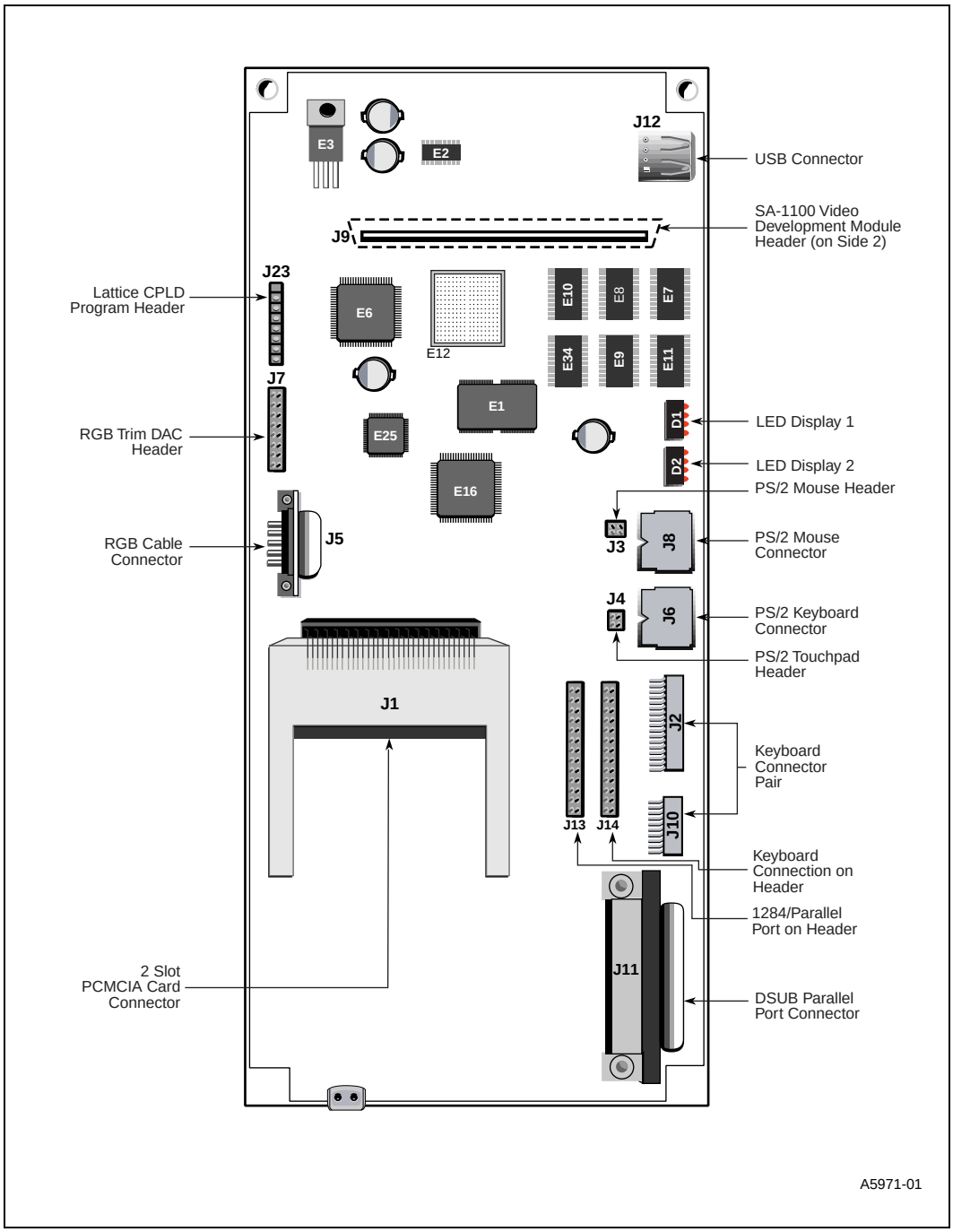


Figure 3-2. Optional SA-1101 Development Board



3.2 Unpacking the Boards

Caution: These boards contain electronic components that are susceptible to permanent damage from electrostatic discharge (static electricity). To prevent electrostatic damage, they are supplied in an antistatic bag. When handling the boards, risk of damage can be alleviated by following a few simple precautions:

- Remove the board from the bag only when you are working on an antistatic, earthed surface and wearing an earthed antistatic wrist strap.
- Keep the antistatic bag that the card was supplied in; if you remove the board from a system, store it back in the bag.
- Do not touch the gold contacts.

3.2.1 SA-1100 Multimedia Development Board Kit Contents

The SA-1100 multimedia development kit is shipped with the following items:

- Two identical adapter cables fitted with a male RCA jack and a two-pin header receiver
- A 16-pad keypad
- A fully-populated SA-1100 multimedia development board with pre-programmed CPLDs
- A SODIMM double-sided module containing 16 MB or 32 MB of SDRAM memory, already pre-installed
- Null-modem cable (9 pin D-Sub female to 9 pin D-Sub female)
- *StrongARM** SA-1100 Multimedia Development Board with Companion SA-1101 Development Board User's Guide (*this document)
- *StrongARM** SA-1100 Microprocessor Specification Update*
- *SA-1100 Microprocessor Technical Reference Manual*
- *README.TXT*

Note: The schematics, parts list, software (including CPLD source code), and copies of the documentation are available from the developer's area on the Intel website.

3.2.2 SA-1100 Multimedia Development Board Software

To purchase an ARM Software Development Toolkit (SDT), see your Intel sales representative.

The following source and executable files are available from the StrongARM section in the developer's area on the Intel website.

- SCB driver application—Industry-standard protocols for configuring multimedia components
- Video pass through application—Captures video input and moves it to the video output buffer
- Keyboard driver—Reads data from keyboard to make available to programs
- Video out application—Program that displays application-generated data
- Diagnostics—Test program that analyzes the functions of the SA-1100 multimedia development board and the SA-1101 development board

- Angel boot loader—Software component of ARM that loads an application from a remote host computer or from the application flash
- Set of microHAL libraries (to be used with Angel)—Set of drivers for communicating with the SA-1100 multimedia development board

Note: All software is available from the StrongARM section in the developer's area on the Intel website.

3.2.2.1 Small Outline Dual-Inline Memory Module (SODIMM)

The SA-1100 multimedia development board is normally supplied with a single, 72-pin plug-in SODIMM double-sided board containing 16 MB or 32 MB of SDRAM memory. If the SODIMM module has not been inserted when you receive your SA-1100 board, install it by following these steps:

1. Locate J13, the SODIMM socket, which is located on the outside edge of the board.
2. Slide the SODIMM into the socket at a 35 degree angle (relative to the SA-1100 board) taking account of the polarity slots. Two polarization slots are cut in the SODIMM; this ensures that the SODIMM is oriented correctly.
3. Pull the tabs out on the socket and gently fold the SODIMM module down towards the SA-1100 board. The notches on the SODIMM module should line up with the tabs with a clicking noise.

Warning: The tabs on the socket are fragile and should be handled with care.

3.2.2.2 Before Installing the SA-1100 Multimedia Development Board

Before installing and power up the board, perform a quick visual inspection:

1. Inspect the card for physical damage.
2. Ensure that all switches for component S1 are down except for Switch Positions 7 and 8 for the video pass through application.
3. Ensure that one SODIMM is fitted in J13 (socket closest to the edge of the card), and that the main portion of the gold contacts on the SODIMM has inserted into the socket along the whole length of the SODIMM.

3.2.3 SA-1101 Development Board Kit Contents

The optional SA-1100 multimedia development board is shipped with the following items:

- A 64-key keyboard
- A fully-populated SA-1101 development board with pre-programmed CPLDs
- *StrongARM** SA-1101 Microprocessor Technical Reference Manual*

Note: The schematics, parts list, software (including CPLD source code), and copies of the documentation are available from the developer's area on the Intel website.

3.2.3.1 Optional SA-1101 Development Board

The SA-1101 development board is an optional daughter card that expands the I/O capabilities of the SA-1100 device. To assemble the SA-1101 development daughter card to the SA-1100 multimedia development board, align J9 of the daughter card with J6 of the mother board and gently press together.

3.2.4 Board Installation

The SA-1100 multimedia development board uses a PCI form factor board outline that can draw power from a standard PCI slot available on a motherboard or a backplane. The board installs in a PC with a PCI expansion bus, which provides a simple way to power up and use the SA-1100 multimedia development board. In a lab environment, using a backplane may be better suited for hardware engineering efforts.

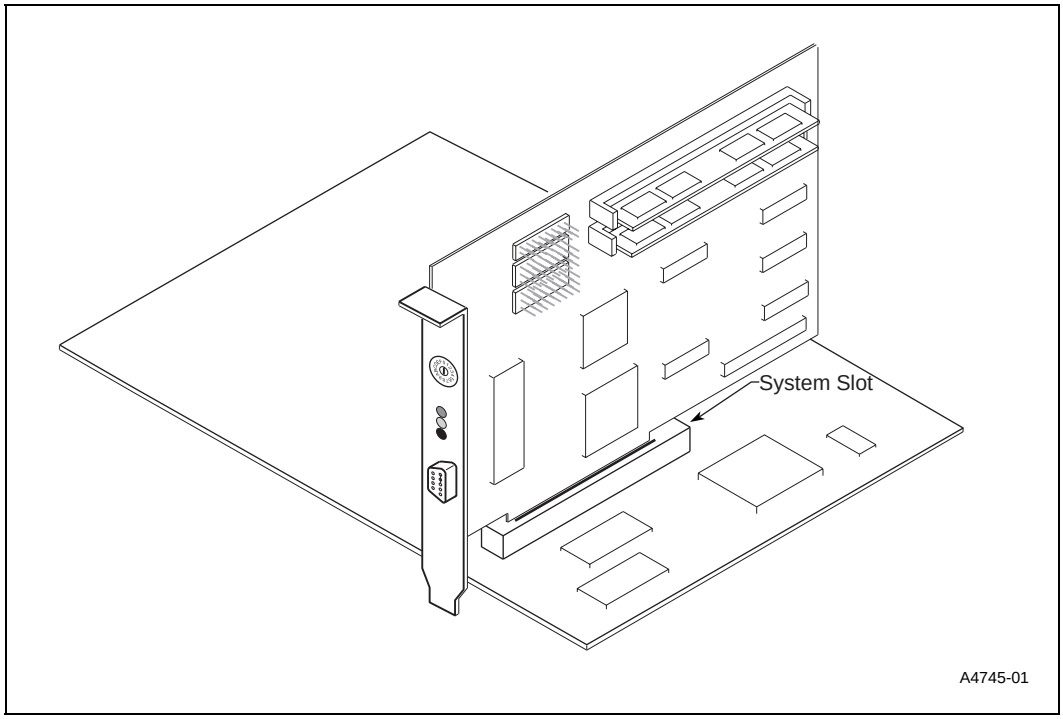
Note: The PCI slot only provides power to the SA-1100 multimedia development board and does not provide communications.

3.2.4.1 Backplane Installation

For hardware desktop applications, a backplane such as the Intel order number 21A85-02, provides power connections for the SA-1100 multimedia development board. A standard motherboard power supply is also required to supply power for the backplane.

Figure 3-3 shows an backplane installation using the SA-1100 multimedia development board and one of the Intel PCI development backplanes.

Figure 3-3. Example Configuration of Multimedia Development Board in a Backplane



3.3 Video Connections

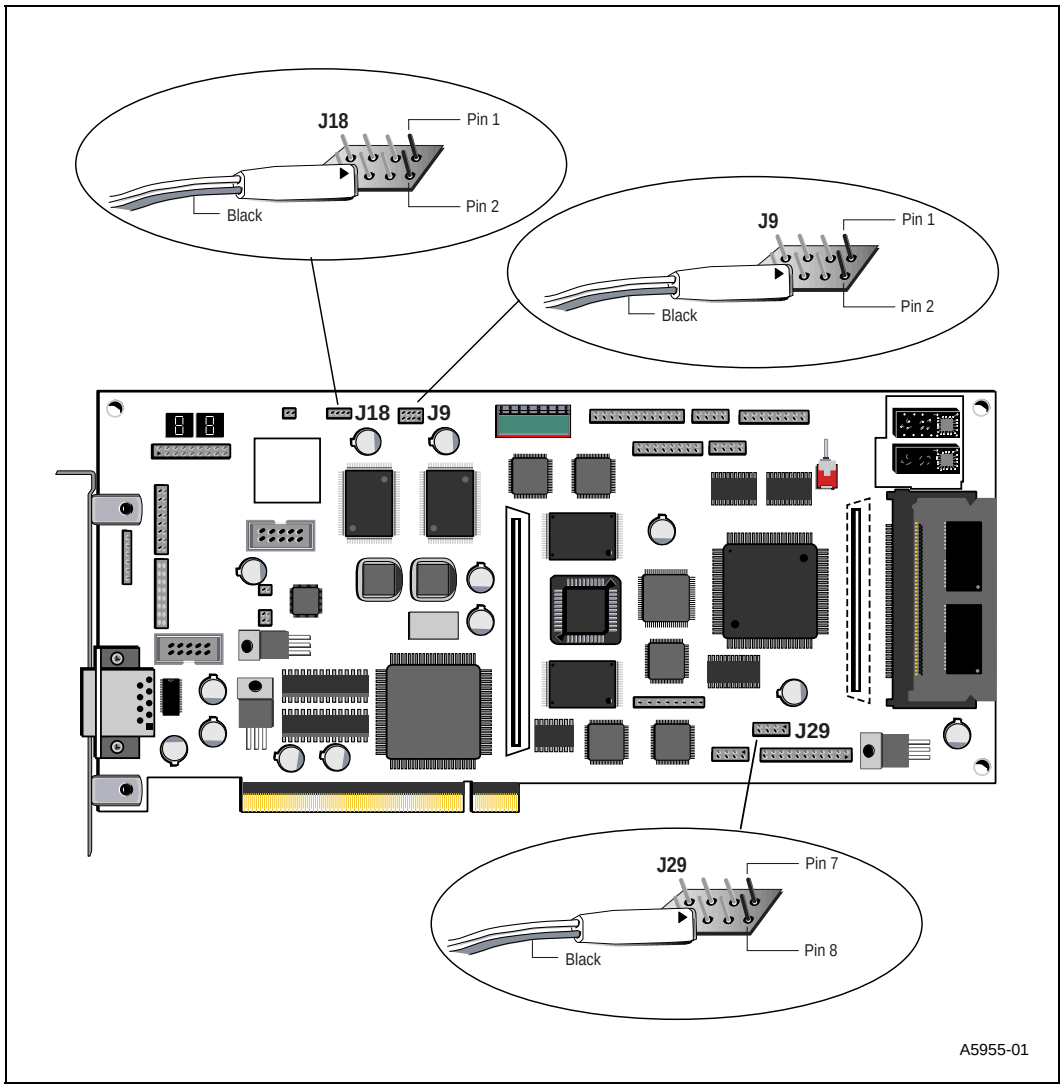
The SA-1100 multimedia development board is shipped with two identical adapter video cables, which provide video connections, and a 16-pad keypad for data entry in applications. The adapter cables have a female RCA jack wired to a two-pin header receiver, with the black wire designated as ground.

Use the following procedure and see Figure 3-4 for information on how to properly make the SA-1100 multimedia development board connections.

1. Using the pin 1 indicator (triangle) as a reference, attach the two-pin header receiver of the adapter cable to J9, Main Video Input Header, using ground (black) on pin 1 and signal (white) on pin 2. Attach the RCA lead to a video input source.
2. Using the pin 1 indicator (triangle) as a reference, attach the two-pin header receiver of the adapter cable to J29, RGB and Composite Video Output Header, using ground (black) on pin 8 and signal (white) on pin 7.
3. Insert the ribbon connector of the 16-pad keyboard into J7, aligning pin 1 of the 9-pin ribbon cable with pin 1 of the 10-pin header.
4. Insert the RS232 null-modem cable to J11, Debug Serial Header #2, aligning the connectors. Attach the other end to an RS232 port on a terminal or terminal emulator.
5. For a secondary video source, use the pin 1 indicator (triangle) as a reference to connect a two-pin header receiver of an adapter cable to J18, PIP video input header, having ground (black) on pin 1 and signal (white) on pin 2

Note: Only two adapter cables are shipped with each kit.

Figure 3-4. Connecting J9, J18, and J29



3.4 Terminal Settings

To configure a terminal for proper communication protocols, use the following settings:

- 9600 baud
- 8-bit data
- 1 stop bit
- no parity
- no flow control

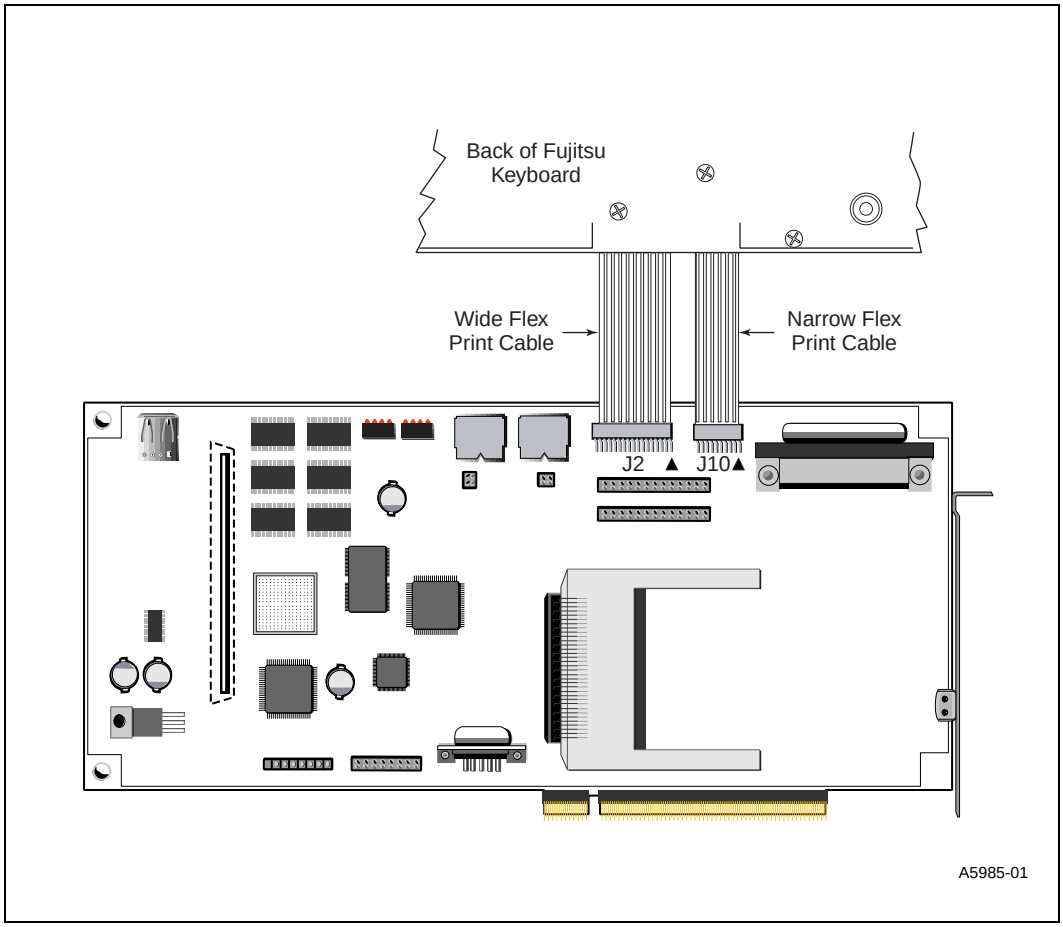
3.5 Cables for Optional SA-1101 Development Board

The optional SA-1101 development board is shipped with a 64-key keyboard and requires a 15-pin RGB connector. Use the following procedure to make these connections:

1. Insert the narrow ribbon cable of the keyboard into J10, Keyboard Connector Pair, aligning the highest order pins of each cable. Be careful not to slide the cable under the connector instead of into the connector.
2. Insert the wide ribbon cable of the keyboard into J2, Keyboard Connector Pair, aligning the pin 1's of each connector. Be careful not to slide the cable under the connector instead of into the connector.
3. Attach the 9-pin RGB connector (not provided) into J5, RGB Cable Connector, aligning the socket connections. Attach the other end of the connector to a color monitor or LCD display.

Note: The flex print cables for the keyboard are center justified with connectors J2 and J10, resulting in unused pins on the left side of J2 and on the right side of J10. Please see Figure 3-5 for proper insertion of the keyboards two flex print cables.

Figure 3-5. Keyboard Connectors



3.6 Bootloader Control

Table 3-1 describes which ROM image is loaded by the bootloader.

Table 3-1. Component S1 Switch Combinations for ROM Image Selections

Switch Position1[a]	Switch Position 0[a]	ROM Image Loaded by Bootloader
0	0	Angel
0	1	ROM image 1
1	0	ROM image 2
1	1	ROM image 3

a. 0 = switch down, 1 = switch up

3.7 Applying Power Using the PCI Connection

Turn on the PC or apply power to the backplane for the development boards. Identify the group of eight discrete LEDs near the bulkhead mounting bracket of the SA-1100 multimedia development board. Power-cycle the system and watch the LEDs.

The LEDs should all illuminate initially and then extinguish after about half a second. At the same time, the terminal screen should display a message similar to this:

```
Angel Debug Monitor for SideARM (FIQ), MMU on, caches enabled, Clock Switching on
(serial)
1.00 (Advanced RISC Machines unreleased) rebuilt on Jul 14 1998 at 14:25:59
```

If you fail to see the behavior described then use this checklist to identify the problem:

- If the SA-1100 multimedia development board has been configured as an add-in card and it stops the host PC from booting correctly, verify the component S1 switch settings on the card.
- If the LEDs behave correctly but the terminal doesn't produce any output, check the terminal cable and terminal settings. The cable can be tested by connecting it between two PC COM ports and running terminal emulation on each port; if the cable is correctly wired you will be able to type characters on either terminal emulator and display them on the other.
- If neither the LEDs nor the terminal behave correctly, then verify that Switch Position 1 and Switch Position 0 of component S1 are in the closed (down) position.
- Attempt to run the onboard diagnostics by following the instructions in the next section.
- If the discrete LEDs are all lit and the HexLEDs are 00, either Angel is missing or a bad Angel is present.

3.8 Running the Onboard Diagnostics

You can get an additional level of confidence that the development board is working correctly by running the onboard diagnostics that are programmed into the flash ROM.

Before starting the diagnostics:

1. Attach the card to a terminal as described in Section 3.4.
2. Power-cycle the system. The diagnostics should start up automatically and report progress on the terminal. Diagnostics first change the LEDs to a predetermined value. See the README.TXT for the predetermined value for this release.
3. The README.TXT describe the output that the diagnostics should produce and describes what to do if the diagnostics fail.

3.9 Video-Pass-Through Test Mode

The video-pass-through test mode is useful while debugging video applications. Table 3-2 describes how to disable the video FIFO interrupts for the main video FIFO interrupt and the PIP video FIFO interrupt.

Table 3-2. Video FIFO Test Mode Selection

To Disable the Video FIFO Interrupts For the:	Make sure that...
Main Video FIFO Interrupt	Switch Position 7 of component S1 is open (up)
PIP Video FIFO Interrupt	Switch Position 8 of component S1 is open (up)

3.10 Using the ARM SDT with the Multimedia Development Board

The ARM Software Development Toolkit (SDT) includes a remote debugger. When running the remote debugger, one part runs on the host (this part includes the user interface) and the other part runs on the target (the SA-1100 multimedia development board). The host and target communicate across a communications channel. By default, the SA-1100 multimedia development board uses its COM0 RS232 port to communicate with the host.

The software that runs on the target is called the *remote debug agent* or *remote debug stub*. By default, the remote debug agent used with the SA-1100 multimedia development board is a program called Angel.

Use an RS232 null-modem cable between the COM0 port on the SA-1100 multimedia development board and the RS232 port on the machine on which the SDT has been installed.

If you are using the SA-1100 multimedia development board as an add-in card, the SDT can run on the same host. This requires COM0 to be connected to one of the COM ports on the host.

For more details on using the SDT, refer to the *ARM Software Development Toolkit Reference Manual*. Chapter 7, “Firmware Modification” describes how to use the SDT to build images that can be executed and debugged on the SA-1100 multimedia development board.

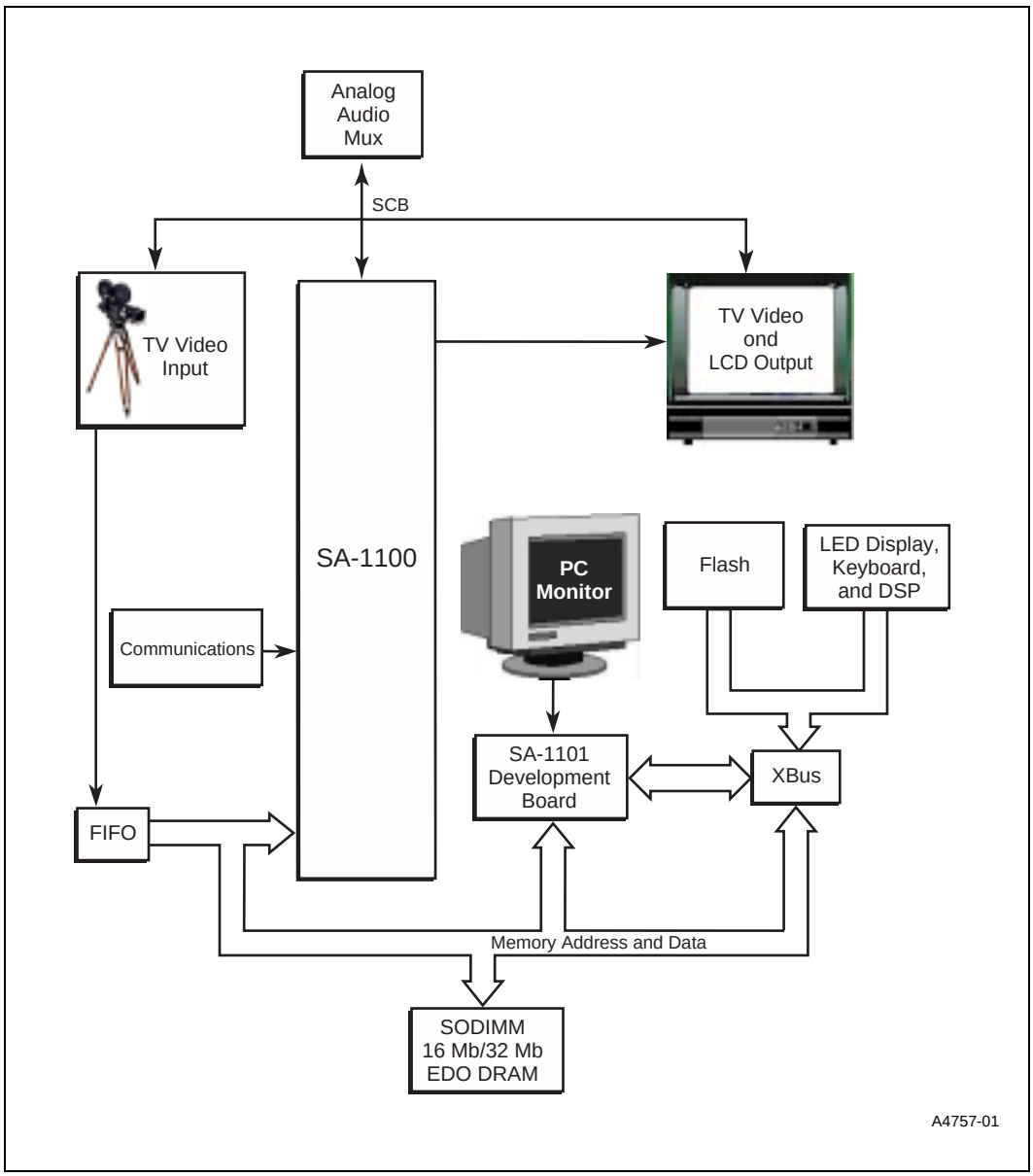
Hardware Overview 4

The SA-1100 multimedia development board is a highly flexible platform for hardware and software development of videophones, Internet TVs, Internet cameras, as well as a general-purpose operating system and application porting platform. With the addition of the SA-1101 development daughter board, the SA-1100 and SA-1101 reference design combination contains all the system components necessary for a Windows CE sub-notebook system development platform with two independent video heads.

The SA-1100 multimedia development board block diagram illustrates the flexibility and extendibility of this design. All device interfacing has been implemented with in-system programmable CPLDs and all system interface points are available on connectors suitable for daughter boards or cables. Although not intended as a ready to manufacture product design, the SA-1100 multimedia development board provides the basis for low-cost derivative designs such as the Internet TV videophone.

Figure 4-1 shows the SA-1100 multimedia development board block diagram.

Figure 4-1. SA-1100 Multimedia Development Board Block Diagram



This example design represents a complete TV system with Internet and videophone capabilities as well as picture in picture (PIP), multiple PIP (used to display captured semi-real time frames of all selected channels), picture in web page (PIWP), on screen programming (OSP), closed captioning (CC) and VBI data capture.

4.1 TV Video Input

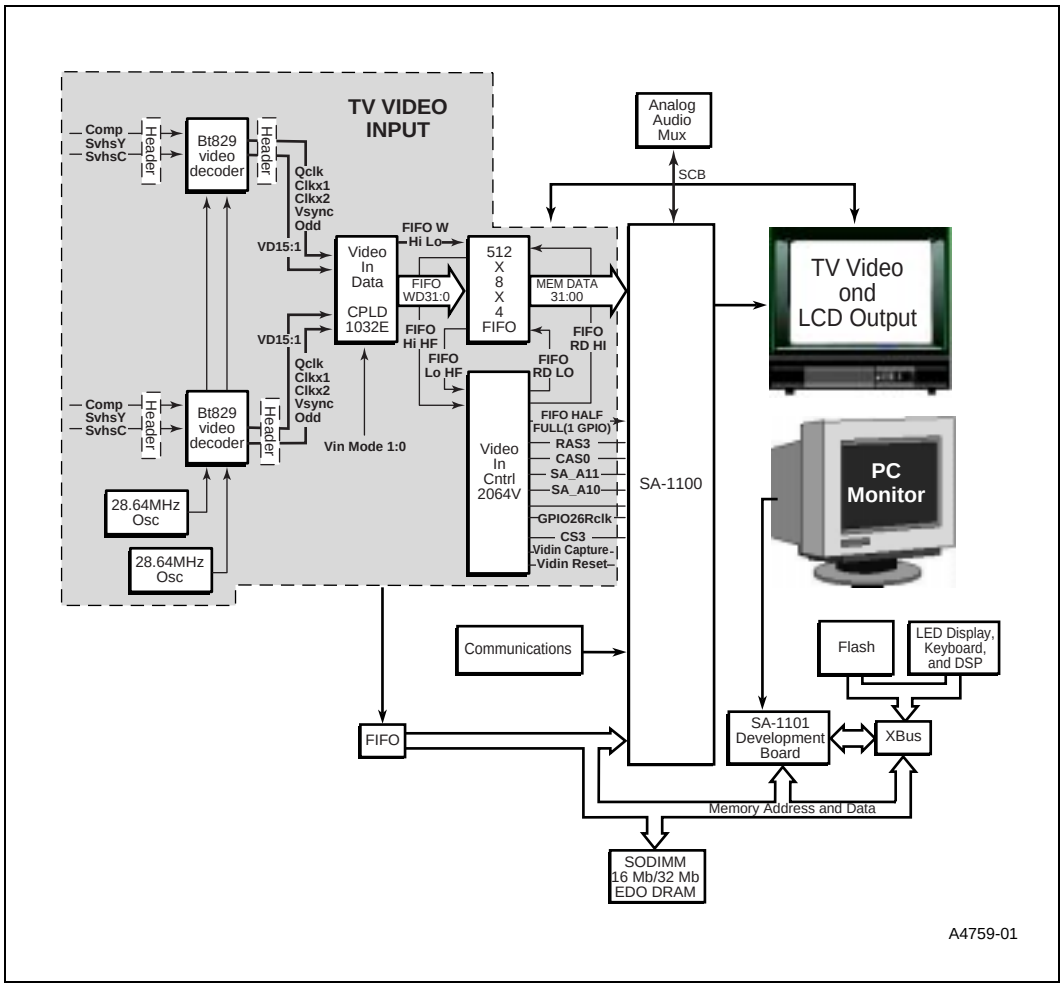
The TV video input section of the SA-1100 multimedia development board allows the SA-1100 processor to capture, store and display in real time, standard analog NTSC or PAL TV signals from cameras, VCRs and TV tuners. It does this by using standard low cost, video decoder devices and FIFO memory devices to provide a high speed programmed I/O (PI/O) video capture subsystem. The SA-1100 multimedia development board provides two TV decoder devices with programmable scaling, which support Picture in Picture (PIP) or multipicture applications similar to PIP features on high end TVs.

The TV video input subsystem may be programmed to capture video from either TV decoder or from both decoders at the same time. The two TV decoders may be programmed with different scaling factors and even different broadcast systems. There are no constraints on video synchronization between the two video sources.

The FIFOs are software configurable to allow them to act as a single 32 bit wide FIFO dedicated to either decoder, or as two separate 16 bit FIFOs, one for each decoder. The FIFOs isolate the SA-1100 processor from the real time timing details of the video signals. The SA-1100 only sees FIFO half-full interrupts with video sync information coded in the FIFO data stream; it never sees video sync interrupts.

The FIFOs are interfaced to the SA-1100 using the same burst timing as the main memory EDO DRAM. Providing a FIFO interface that supports high-speed burst reads is the key to real time video capture using the SA-1100 programmed I/O.

Figure 4-2. TV Video Input Block Diagram

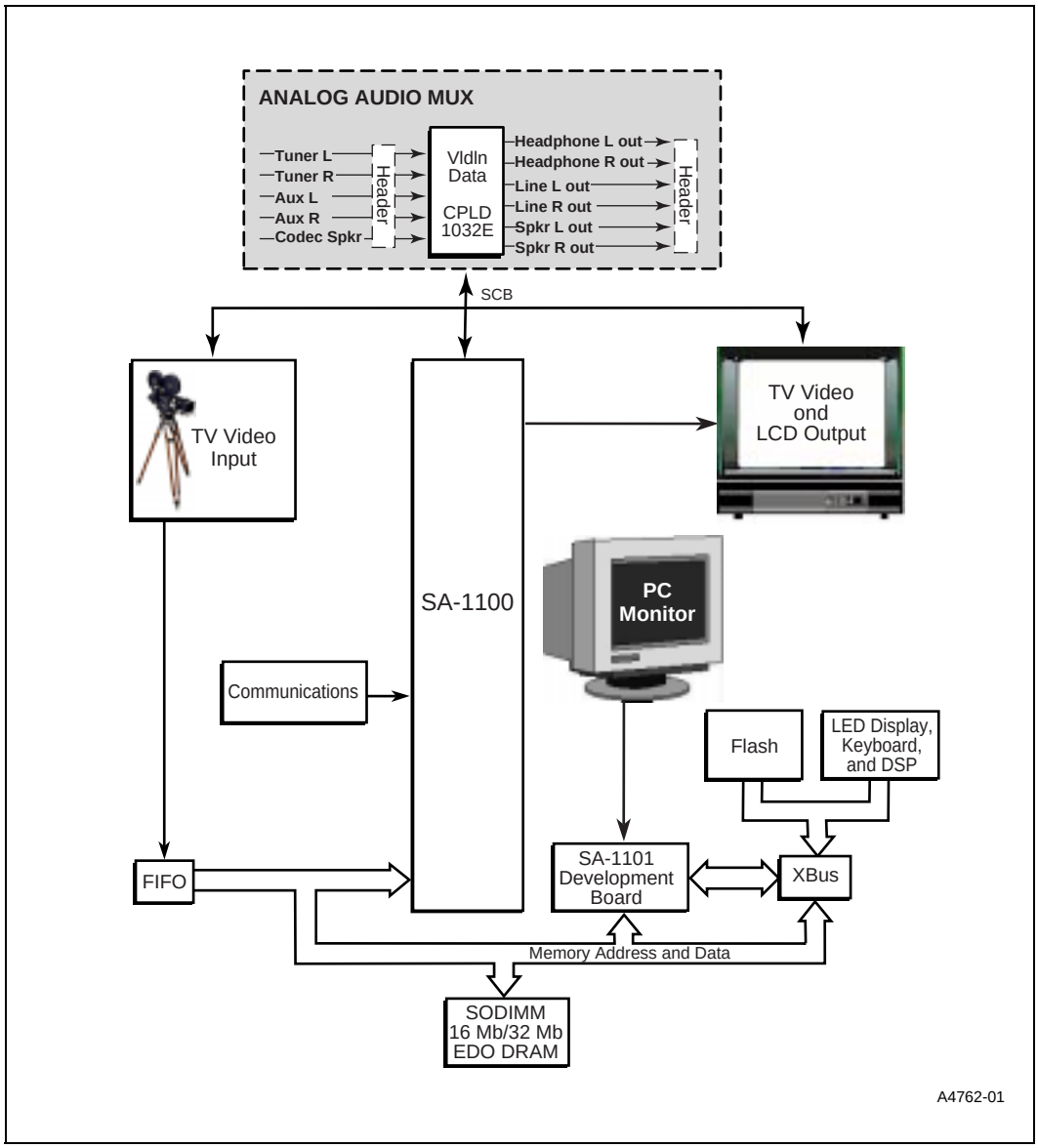


4.2 Analog Audio

The analog audio section of the SA-1100 multimedia development board is a general-purpose analog audio processing unit that manages up to six mono or three stereo analog audio channels. The analog audio section allows easy integration of the SA-1100 multimedia development board in a variety of multimedia applications including videophones and Internet TVs. The TDA9860

analog audio processor device is controlled via the serial control bus (SCB) bus on the SA-1100 multimedia development board. It also provides crossbar switching of any input to any output as well as tone and stereo balance controls.

Figure 4-3. Analog Audio Block

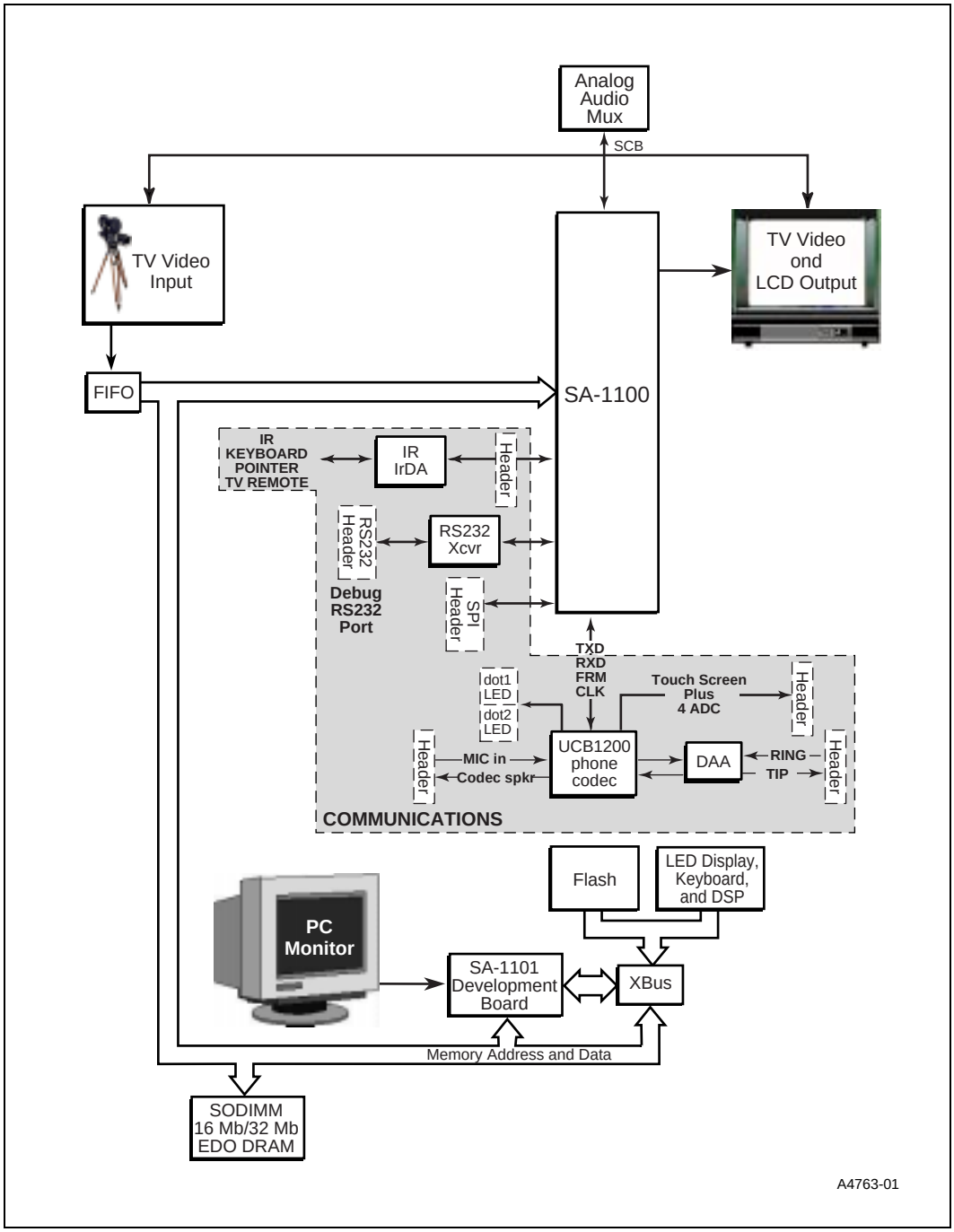


4.3 Communications

The SA-1100 multimedia development board provides two serial ports, a teleconferencing CODEC that supports a telephone line interface, a microphone, and a speaker. One serial port is used by the Angel micro kernel while the second serial port is available for user applications.

An IrDA infrared communications transceiver on the SA-1100 multimedia development board allows direct wireless communication between the SA-1100 multimedia development board and any IrDA compliant device, such as most PC notebooks and hand held PCs. The SA-1100 multimedia development board IrDA is capable of speeds ranging from 300 bps to 4M bps.

Figure 4-4. Communications



4.4 LED Display, Keyboard, DSP, Xbus and Flash

This section describes the LED display, keyboard, DSP, Xbus and flash.

4.4.1 LED Display

The SA-1100 multimedia development board has a two-digit hexadecimal seven-segment LED display as well as a block of eight discrete LEDs. Both of these display groups are accessible as programmed I/O registers and may be written at any time by either the Angel micro kernel or by user applications. The LED displays are typically used for program debug and system status.

4.4.2 Keypad

A 4-by-4 matrix keypad, suitable for developing telephone as well as general purpose applications, is included with the SA-1100 multimedia development board kit. The keypad is the primary user input device for the SA-1100 multimedia development board demonstration applications and is also available for user application development. The keypad is interfaced to the SA-1100 via a CPLD that allows interrupt driven programmed I/O scanning and de-bouncing of the keypad.

4.4.3 Digital Signal Processor

An application-specific (audio) DSP is provided as an optional processor on the SA-1100 multimedia development board to allow development of G7XX and other standard digital audio protocols. The use of the audio DSP is optional as the audio DSP protocols may also be implemented in software on the SA-1100.

The DSP is provided in SA-1100 multimedia development board to allow offloading of MIPs from the SA-1100 in the event that the target application requires more MIPs than the SA-1100 can support.

4.4.4 System Design External Bus (Xbus)

The Xbus[1] is a SA-1100 multimedia development board specific, general-purpose bus that connects most of the programmed I/O devices in the SA-1100 multimedia development board. The Xbus is a fully-buffered version of the SA-1100 pin bus including 32 data lines and 26 address lines with the addition of several general-purpose signals such as resets and address decode enables. Most of the Xbus control signals originate in the SA-1100 multimedia development board CPLDs. The CPLDs may be modified or adapted to customer specific needs. Two Xbus connectors are provided.

The Xbus 96 pin connector provides a convenient fully-buffered 32-bit interface point for customer designed daughter boards or logic analyzer adapters.

The 128 pin connector on the SA-110 multimedia development board is normally used by the SA-1101 development daughter board. This connector includes both a buffered 16-bit Xbus and unbuffered 32-bit SA-1100 main DRAM memory busses suitable for Direct Memory Access (DMA) devices and PCMCIA type devices.

1. Not to be confused with X-Bus, which is an interface used with the 21285 core logic for the SA-110 microprocessor.

4.4.5 Flash Storage

The flash memory banks in the SA-1100 multimedia development board system provide non-volatile program storage for the bootable micro kernel and for user applications. Two banks of flash memory are provided: boot flash and application flash. The SA-1100 multimedia development board is shipped from the factory with the Angel debugger flashed into the boot flash bank.

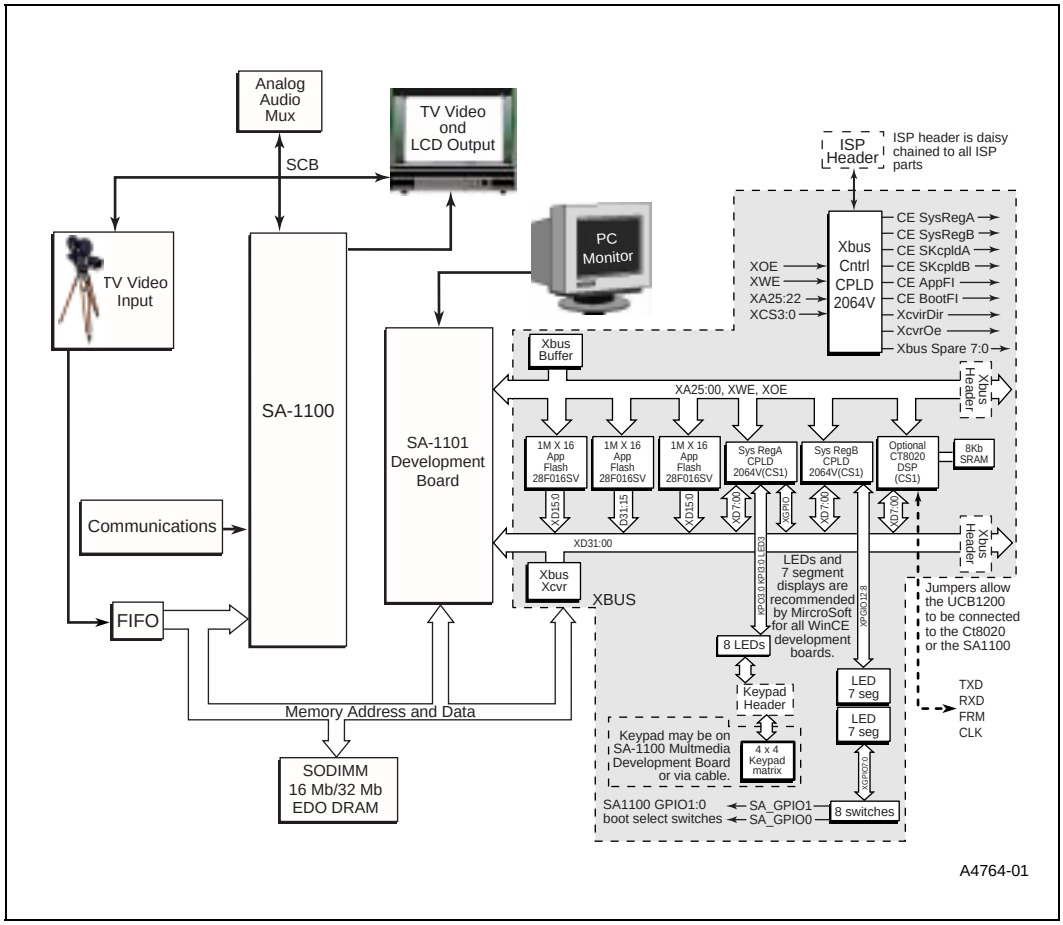
The boot flash bank is a 64 Kx16 (128KB) socketed flash device that is intended for the initial power up boot code as well as micro kernel code. The device is socketed to assist in the initial debugging of hardware and firmware. The device may be flashed while in the SA-1100 multimedia development board or on a dedicated ROM blaster and then plugged into the socket.

The application flash bank is a 4MB block of Intel flash memory that is intended to store user applications.

Both the boot flash and the application flash banks may be written using the Flash Management Utility (FMU). See Chapter 7, “Firmware Modification”for more information about the FMU.

A switch on the SA-1100 multimedia development board allows the addressing and functions of the two flash banks to be swapped. This feature allows the boot flash device to be flashed while the application flash bank functions as the boot flash bank. This scheme allows the SA-1100 multimedia development board to be used to flash and debug boot code without requiring a dedicated ROM emulator or flash ROM programmer.

Figure 4-5. Xbus and Storage Block Diagrams



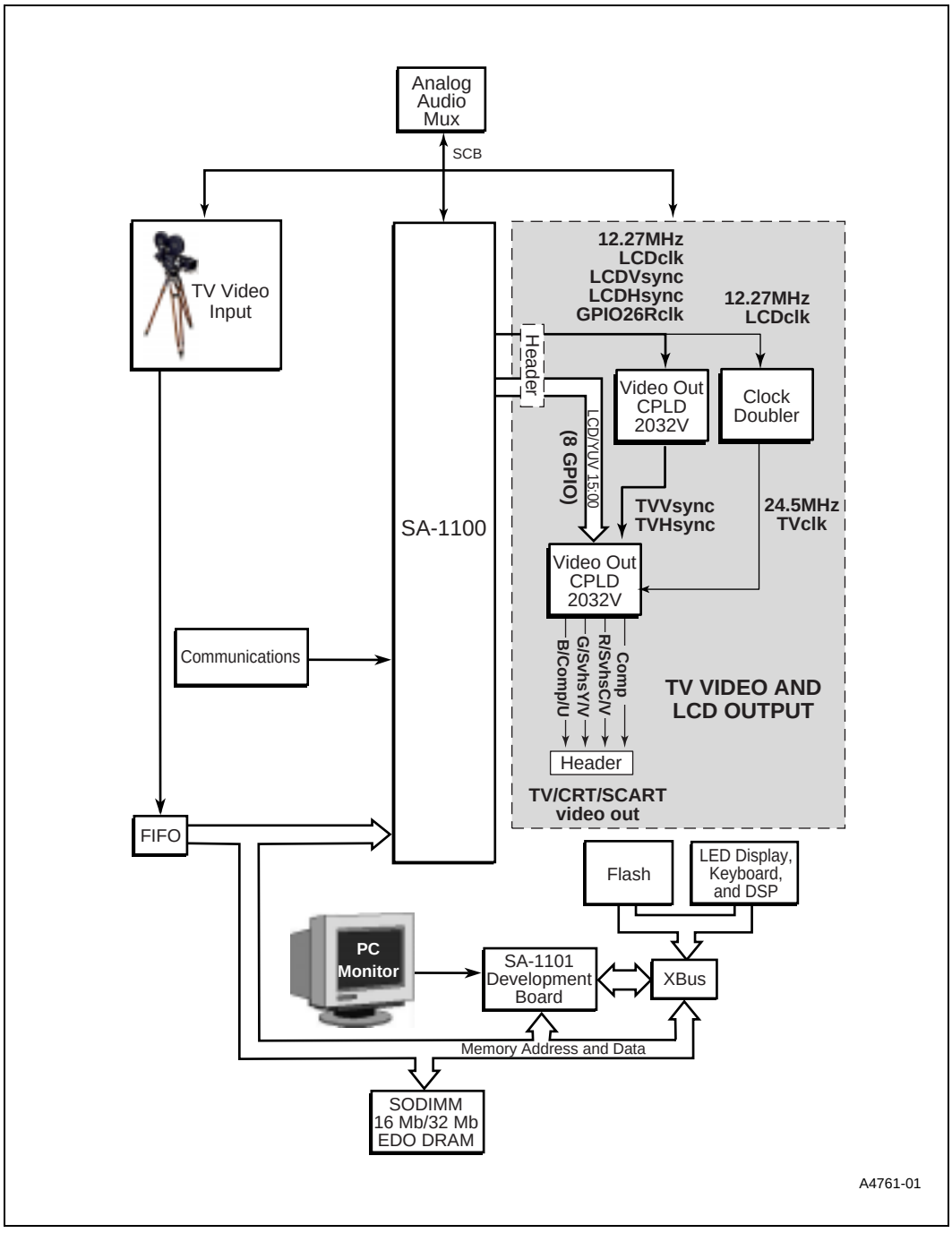
4.5 TV Video Output

The TV video output section of the SA-1100 multimedia development board contains a low-cost digital to NTSC video encoder device that is interfaced to the SA-1100 LCD interface port. The small amount of support logic required to adapt the LCD interface to the NTSC encoder is implemented in the CPLD.

This section also describes a clock doubling PLL device that is no longer required for the NTSC video output. The function of the clock doubling PLL is now included in the CPLD.

The LCD interface in the SA-1100 multimedia development board video out application is programmed to output data.

Figure 4-6. TV Video Output



4.6 SA-1101 Development Board and SODIMM Board

This section describes the SA-1101 development board and the SODIMM board.

4.6.1 SA-1101 Development Board

The SA-1101 development board attaches to the SA-1100 multimedia development board via a 128 pin connector and provides the following functions.

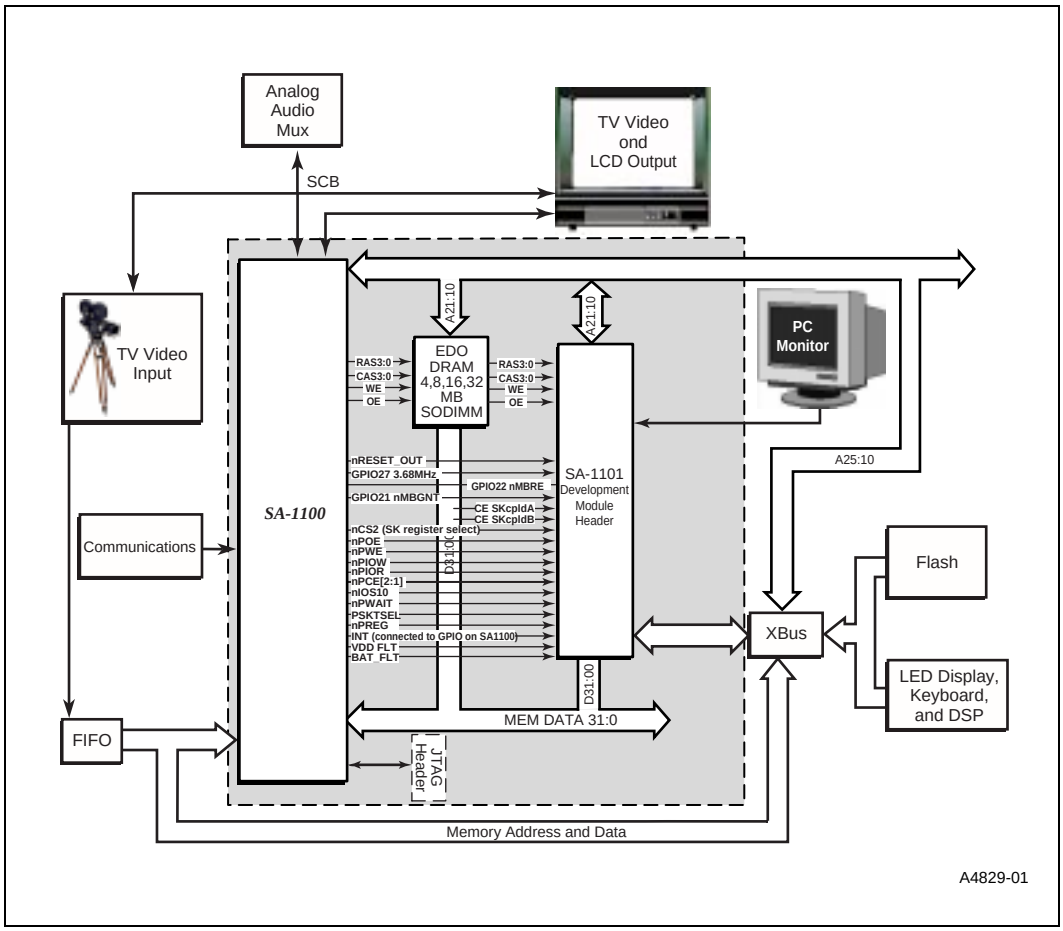
- VGA video up to 1024 by 768 at 8 bits per color
- Two trim DACs used for brightness and contrast controls
- USB host controller
- Glue logic for two PCMCIA slots
- Two PS2 ports
- 16 by 8 matrix keyboard interface
- 1284 parallel printer port

4.6.2 SODIMM EDO DRAM Board

The SODIMM EDO DRAM is a small memory module that provides the main memory for the SA-1100. SODIMMs were chosen for the SA-1100 multimedia development board design to allow different memory configurations and vendors to be evaluated. The SODIMM is a 32-bit wide

memory module typically used in laptop computers. The SODIMMs are available in a variety of sizes and configurations ranging from 4MB to 32MB. For a list of qualified SODIMM vendors, see the README.TXT file.

Figure 4-7. SA-1101 Development Board and SODIMM Board



4.7 High Bandwidth General Purpose Data Acquisition Applications

Although the main purpose of the SA-1100 multimedia development board design is to provide video in and video out on an SA-1100 platform, the flexible nature of the design also allows the video in CPLD and high speed 32 bit wide FIFOs to be used for general purpose, very high speed data acquisition. Acquisition data rates up to 40MBS (66% of SA-1100 system bandwidth) are possible. A general purpose interface may be implemented using the headers on the input side of the video-in-data CPLD. The video-in-data CPLD may be programmed to format or decode digital data streams from 1 bit to 24 bits wide while supporting several sideband signals and clocks.

Medical applications such as data acquisition and processing for various medical scanners and UltraSound scanners may be feasible. In addition, industrial, seismic and scientific data acquisition and processing applications may also benefit from the SA-1100 multimedia development design.

4.8 Video Design Applicable for Video Front End Designs

Two Bt829a video decoders feed data and timing into an IspLSI* 1032E CPLD called the video-in-data CPLD. The main Bt829a has a full 16 bit data path connection to the video-in-data CPLD while the secondary Bt829a has an 8 bit data path. Cost focused customer designs requiring only 8 or 16 bit wide FIFOs may be implemented in the flexible video data path.

The two IDT72V81 512x16 FIFOs configured as a 512 x32 FIFO provide a Half Full (HF) pin that can be connected to a GPIO pin is programmed to interrupt the SA-1100. Each HF interrupt can drain 256 32-bit words which equals 512 pixels from the FIFO using 32 burst cycles of 8 words each. This requires a minimum of 32 x 30ns x 8 = 7680ns of SA-1100 bus time every 512 x 870ns = 445440ns (average QSIF 15fps pixel time is 8*(1/(12.27 * 0.75)) = 870ns). Therefore 15FPS QCIF video acquisition will use about 2% of the total SA-1100 bandwidth. Each HF interrupt drains at least 256 locations of the 512 deep FIFO and has an approximate interrupt frequency of 2KHz for QCIF 15FPS and 20KHz for CCIR601 30FPS.

The 32-bit wide FIFO design makes it possible to process full resolution CCIR601 video through the SA-1100 while using only 50% of the system bandwidth and MIPs. Full 601 square mode acquisition requires about 20MBS of the memory bus bandwidth while full 601 out requires another 20MBS; the write of acquired video data to the output display buffer requires yet another 20MBS. With main EDO DRAM CAS cycle timing of 30ns yielding peak main memory bandwidth of 120MBS, this leaves about 50% of the main memory bandwidth and a greater percentage of the MIPS still available. This allows full resolution 601 pass through mode with about 50% of a 220MHz CPU still available, which is very useful in WebTV applications.

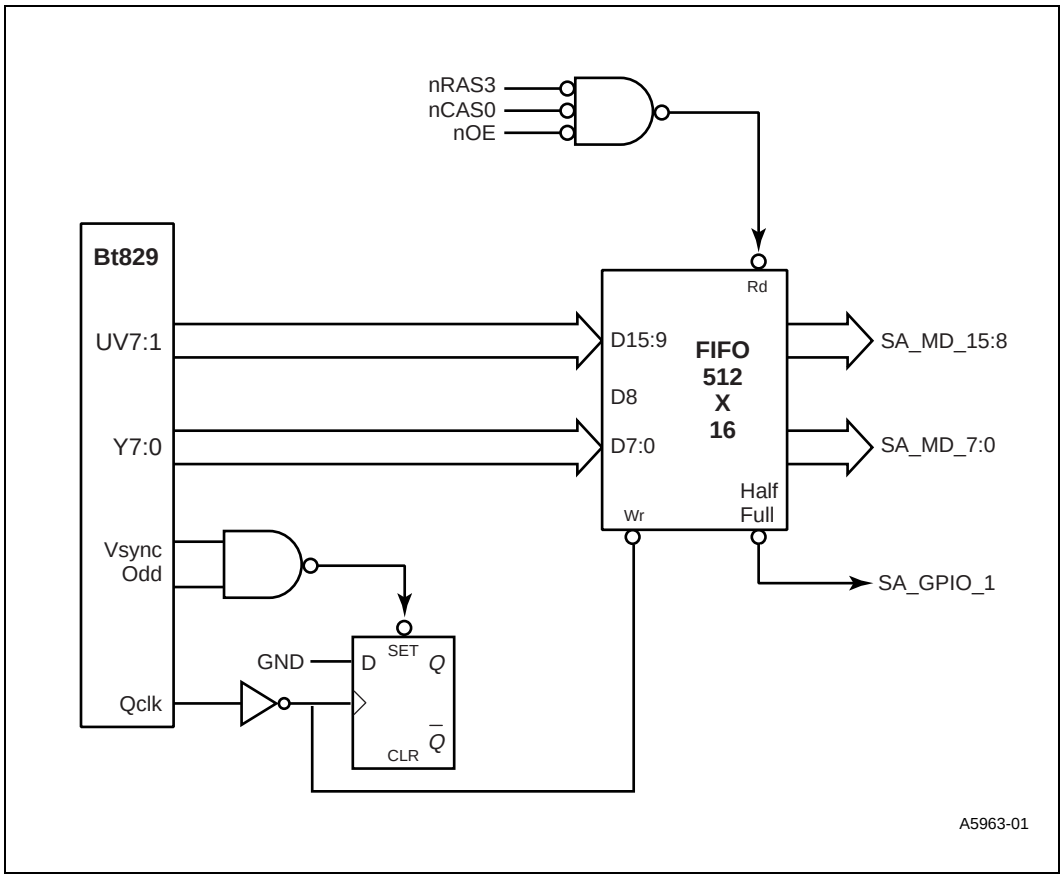
A lower cost implementation that uses only a 16-bit wide FIFO will require an effective input bandwidth of 40MBs, because twice as many reads are required to capture the 20MBs real data rate for full CCIR601. In this design the total bandwidth used during CCIR601 pass through will be 80MBs or about 66% of the available system bandwidth. The SA-1100 multimedia development board CPLDs may be configured to implement this design. This design will not support dual decoders and the two simultaneous video input streams that are required for real time picture in picture (PIP) features. However, semi-real time PIP may be implemented by channel scanning the TV tuner or video sources.

The physical interface between the Bt829a, 72V81 FIFOs, PLS* I2064V CPLD and the SA-1100 will not require 5V to 3.3V level converters. The 72V81 and 2064V are 3.3V parts that are also 5V tolerant. However, the SA-1100 is **not** 5v tolerant.

4.9 Low Cost Implementation

Figure 4-8 shows a simple, low-cost implementation that supports CCIR601 video input on the SA-1100. This implementation only requires a single Bt829A, a 256x16 FIFO, two nand gates, an inverter, and a flip-flop.

Figure 4-8. Low-Cost Video In Design



4.9.1 Video Data Path CPLD

The output of the video-in-data CPLD feeds the 32 bit video FIFO. The video-in-data CPLD is programmed to select one of the following formats:

- Primary decoder 16-bit pixel mode
- Primary decoder 8-bit pixel mode
- Dual decoder 8-bit pixel CCIR656 mode

4.9.1.1 Primary Decoder 16-bit Pixel Mode

In this mode, the video-in-data CPLD accepts 16-bit video pixels from the Bt829a in a 4:2:2 format. The 16-bit stream is packed into a 32-bit word stream that is written in to the full 32-bit width of the FIFO. Bit 0 in each 16-bit pixel is coded to indicate the frame sync status. The first pixel of each even frame will have bit 0 equal to 1. All other pixels will have bit 0 equal to 0.

4.9.1.2 Primary Decoder 8-bit Pixel CCIR656 Mode

In this mode, video-in-data CPLD accepts 8-bit video pixels from the primary Bt829a in a CCIR656 coded format. The 8-bit CCIR656 coded stream is packed in to a 32-bit word stream that is written in to the full 32-bit width of the FIFO. The CCIR656 codes are decoded to control the rejecting of invalid pixels and the accepting of Hsync, Vsync and FrameSync coded pixels. These pixels, along with valid pixel values, are packed into 32-bit words for writing into the FIFO. Blanked pixels and invalidated pixels that are the result of scaling in the Bt829a decoder are not fed into the FIFO.

The following CCIR656 codes are passed with the scaled pixels.

Table 4-1. CCIR656 Pixel Codes

Pixel Codes	Description
0x06FF or 0x06FE	Start of odd field code
0x02FF or 0x02FE	Start of active video line
0x01FF or 0x01FE	End of active video line

4.9.1.3 Dual Decoder 8-bit Pixel CCIR656 Mode

This mode runs both decoders simultaneously and splits the FIFO into two 16-bit halves. FIFO bits 15:00 store CCIR656 pixels from the primary decoder, while FIFO bits 31:16 store CCIR656 pixels from the secondary decoder.

Each FIFO half operates independently. The half full (HF) interrupts from each FIFO half are “ORed” together to generate a single HF interrupt to the SA-1100. In this mode, the HF interrupt routine must poll the Interrupt Status Register to determine which FIFO half to service. Three separate FIFO address spaces allow for reading the FIFO as:

- a 32-bit word
- an upper 16-bit word
- a lower 16-bit word

The two decoders may be programmed with different scaling factors and different temporal decimation factors. Typically, the primary decoder is programmed for full resolution full screen output, while the secondary decoder is programmed as CIF or QCIF at 30FPS or less for use in a PIP window.

The three video-in-data CPLD input modes are selected by programming the Vmode1:0 bits in SysRegA.

4.9.2 FIFO Video Read Timing; Vidin Cntrl CPLD

This device controls the FIFO read timing and FIFO address decode. To keep bandwidth utilization efficiency to a maximum, burst read cycles from the FIFO to the SA-1100 are a prerequisite. Burst reads can be invoked using the prefetch Read Buffer (RB) supported in the SA-1100. Three FIFO read address spaces are decoded within the FIFO bank select to allow reading of the high 16 bits of the FIFO, the low 16 bits of the FIFO or the entire 32 bits of the FIFO. Address bits SA_A11 and SA_A10 are connected to the CLPD to allow address decoding of the three FIFO read spaces. A32-bit read of the Hi FIFO or Low FIFO will result in unpredictable data in the unreferenced 16-bit half of the 32-bit word.

The CPLD is wired to support the following FIFO burst read schemes:

- Synchronous clocked CS FIFO reads
- Asynchronous address line timed FIFO reads
- CAS clocked FIFO reads

4.9.2.1 Synchronous Clocked CS Scheme–Not Recommended

The synchronous clocked CS FIFO reads scheme, though **not** recommended, uses the SA-1100 RCLK signal, which can be programmed to appear on SA-1100 GPIO26. The RCLK is equal to SA-1100 core clock divided by two. A state machine in the CPLD watches sa_cs_3 from the SA-1100 and sequences the FIFO read lines in step with the SA-1100. The SA-1100 MCS0 and MCS1 registers are programmed to correspond to the FIFO state machine timing.

The major drawback to this scheme is that RCLK is 110 MHz when the SA-1100 is clocked at 220 MHz. In addition, there is no guaranteed setup and hold of RCLK against the CS signal timing. This latter problem can be addressed by ensuring that the state machine synchronizes the CS signal and provides the data one cycle earlier then necessary. Bus capacitance will hold the data for at least 1 RCLK.

4.9.2.2 Asynchronous Address Line Scheme–Not Recommended

The asynchronous address line timed FIFO reads scheme, though **not** recommended, uses the same sa_cs_3 signal as the synchronous clocked CS FIFO read scheme. However, instead of using RCLK to drive a state machine, address line A2 is used as a timing signal to signal when the next FIFO read cycle must happen.

This scheme requires a timed delay to detect the transition of the A2 address line. The A2 signal and a 15 ns delayed version of the A2 signal are XORed together to produce the FIFO read signal during the CS burst cycle. Short, accurate delays of 10 to 15 ns are difficult to design and may require a delay line.

4.9.2.3 CAS Clocked FIFO Read Scheme–Recommended

This scheme, which is the best choice for production designs and is implemented in the CPLD, uses the RAS3 DRAM bank select to allow CAS0 to drive the FIFO read signals. This scheme requires that the main DRAM banks be timed to run burst cycles no faster than the FIFO can read. A CAS timing of 3-core clocks asserted and 3-core clocks deasserted at 220 MHz represents the minimum FIFO timing and is close to the fastest DRAM timing that can be achieved with 50 ns EDO DRAMs.

A complication with this scheme is CAS Before RAS (CBR) refresh cycles generated by the SA-1100. Refresh cycles must be asynchronously detected and used to block FIFO reads during CBR cycles.

During CBR cycles the OE signal is not asserted. OE is anded with RAS3 and CAS0 to prevent a CBR cycle from cycling the FIFOs.

4.9.2.4 Closed Caption and Extended Data Services Data Capture

The Bt829a has a built in CC and EDS decoder which may be accessed via the SCB interface. A 16-byte FIFO in the Bt829a allows for long latencies. Two bytes of CC data and two bytes of ESD data may be captured per frame. Unlike VBI data, the CC and EDS data is decoded in the Bt829a and does not require further software decoding.

4.9.3 VBI Vertical Blanking Interval Data Capture

The Bt829a may be programmed to capture raw VBI data either in the vertical interval or the entire field. This data is the digitized raw video and must be processed to decode the digital VBI data. The SA-1100 multimedia development board provides sufficient video input bandwidth and MIPs to capture and process this VBI data. Typically, a simple slicing of the raw digital data stream is sufficient to decode the serial VBI binary data stream.

Although feasible, the VBI application has not been developed at this time.

4.9.4 Video Output Design

The TV video output design in the board uses an ADV7175 NTSC/PAL encoder. Many encoders will work in this application, however the ADV7175 offers the following advantages:

- 16-bit pixel stream needed by SA-1100 TV video applications.
- Four DACs to allow simultaneous output of CVBS and RGB signals. This is very useful in TV core designs where the ADV7175 will drive the CRT RGB circuitry directly.
- Programmable color subcarrier PLL allows nonstandard video timing while maintaining color lock in the TV. This allows the SA-1100 to use the standard 3.6864 MHz crystal.

The 4.2.2 YUV pixel interface to the ADI7175 from the SA-1100 uses 16 LCD data lines 8 of which are GPIOs.

The clock for the ADI7175 must be twice that of the 12.27 MHz LCD clock supplied by the SA-1100 when it is programmed with TV video timings. A Cypress CY2308 clock device that generates a zero skew, 2X clock is used to multiply by 2 the 12.27 MHz (NTSC square pixel) from the SA-1100 to produce the 24.54 MHz clock required by the ADI7175.

4.9.5 SA-1100 Core Clock Frequency

The following is a discussion of how the SA-1100 core timing is adapted to TV timing. This discussion is provided to improve understanding of how the SA-1100 can be adapted to TV timing.

The goal is to find a 3.xxxx MHz crystal frequency that is as close as possible to the 3.6864 MHz nominal frequency and that has an integer multiple close to the 230 MHz maximum SA-1100 core clock. At the same time this 2XX MHz core clock must have an integer divisor that produces exactly 12.272725 MHz.

First find a multiple of 12.272725 MHz that comes closest to the highest core frequencies specified in the clock chapter of the *SA-1100 Microprocessor Technical Reference Manual*. Then substitute the exact 12.272725 MHz multiple and re-deriving the 3.XXXX MHz crystal required to generate a 2XX MHZ clock that is an exact multiple of 12.272725 MHz. This approach results in a new 3.681818 MHz crystal frequency that is within less than 0.15% of the optimal 3.6864 MHz. 3.681818 MHz times 4 times 15 yields the desired 220.90 MHz core clock while 220.90 MHz divided by 18 yields the 12.272725 MHz required for the TV video encoder.

The initial implementation of the multimedia development board will use the standard 3.6864 MHz crystal. To compensate for the color subcarrier video timing errors that this produces in the encoder color burst generator, color subcarrier PLL control registers in the ADI7176 are programmed to compensate for this error.

4.9.6 Interlaced Video

An NTSC frame of 525 lines displayed at 30 Hz interlaced consists of two fields of 262.5 lines displayed at 60 Hz. Normally this is achieved in a video controller by having the vertical timing generator count half-lines (instead of full lines) and programming it to 525 half-lines. This produces a vsync every 262.5 lines and provides the ½ line offset between fields required for interlaced timing. Since there is no provision for the interlaced timing in the 1100 LCD controller, the interlaced timing must be done externally.

4.9.7 Video Output Logic; VidOut CPLD

The LCD controller is programmed for 525 lines of video displayed at a 30 Hz rate. An external CPLD will be clocked by the 12.27 MHz LCD pixel clock and will maintain a 10-bit pixel counter and a 10-bit line counter based on the LCD hsync and vsync signals. A new interlaced vsync signal is generated by ORing in an extra vsync at the 262.5 line point. The new interlaced vsync and the original hsync signal plus a 24.54 MHz clock are derived from the 12.27 MHz clock by using a clock doubler device to drive the ADV7175 encoder.

Ongoing design development has demonstrated that the ADV7175 encoder does not require the use of the VidOut CPLD to inject the odd field Vsync pulses. The ADV7175 has internal logic that automatically generates the required odd-numbered field vertical synchronizing pulse. The VidOut CPLD is maintained in the design to allow design flexibility when interfacing to encoders other than the ADV7175. When used with the ADV7175, the VidOut CPLD will be programmed to pass the LCD hsync and vsync signals unmodified.

4.9.8 Interlaced Display Buffer

The video image in the SA-1100 DRAM must be written in interlaced order and with regard to where the two fields are normally blanked. Although this complicates the rendering code, it is possible to design rendering routines that are just as efficient as normal line sequential rendering.

The interlaced frame buffer image in SA-1100 DRAM contains YUV components in 4.2.2 format. Each 32-bit word in the DRAM frame buffer contains four 8-bit video components appearing as UYVY.

Table 4-2. Interlaced Frame Buffer

31 24	23 16	15 8	7 0
• PixelA+1 U[7:0]	• PixelA+1 Y[7:0]	• PixelA V[7:0]	• PixelA Y[7:0]

Note: The representation of the color information in SA-1100 memory is an arbitrary selection between UYVY and YUYV. In the board design, the Bt829 devices output Y on Bt829 data bits 15:8, while the ADV7175 inputs Y on bits 7:0. The video-in-data CPLD is programmed to swap the Y and UV bytes from the Bt829 devices before outputting the data to the FIFO. Cost-focused designs that do not implement the video-in-data CPLD should ensure that the YUV data channels are consistent between input and output.

Video out data in a YUV format has an advantage over video processed by an RGB display device. When video is processed to an RGB display device, such as a SVGA LCD display, an extra rendering step to convert from YUV to RGB is necessary. Maintaining the video image in its native YUV color space avoids this computationally intensive step and results in higher frame rates for H324 video conferencing and allows real time 30 FPS CCIR601 video pass through.

4.9.9 Synchronization with Video Out

Since the video output display timing is not locked to the input video timing, synchronization between the two video timing domains is required to eliminate visual artifacts such as tearing. To eliminate tearing artifacts, the video input software implements frame dropping. The software determines the position of the output display pointer by reading the LCD DMA current address register. The software can decide if a frame must be dropped based on the position of the LCD DMA current address register and the current input pointer maintained in the FIFO interrupt service routine. Double or triple buffering of the display output buffer may be required to implement this scheme. Each buffer requires 614400 (640x480x2) bytes. In general, no more than two buffers are needed. The extra memory required for these buffers may need to coexist with the active operating system.

The frequency of the frame dropping is determined by the difference in frame rates between the TV video timing produced by the SA-1100 and the incoming video source. Typically the frame dropping rates are less than 1 frame per hour for a broadcast source and up to 1 frame per minute for a VCR source.

A more precise method of video synchronization that eliminates frame dropping and tearing by real time adjustments of the video output timing is also available from the ADV7175. This solution is most suitable when the ADV7175 is directly driving the RGB inputs of a TV CRT.

4.9.10 Xbus

The Xbus is isolated from the SA-1100 with buffers and transceivers to reduce loading on the SA-1100 address and data busses. The Xbus provides a 32-bit interface to flash ROMs, the system register CPLD, and the DSP. Headers on the Xbus allow for daughter cards that can implement additional user circuits. In addition, the Xbus is used to buffer address and data lines to the PCMCIA buffers on the SA-1101 development daughter card.

SA-1100 static RAM bank select signals sa_cs_0, sa_cs_1, and sa_cs_3 are used to control three Xbus address spaces. SA-1100 sa_cs_0 is programmed for boot flash ROM timing with a basic read/write cycle time of 200 ns, while sa_cs_1 is programmed for application flash ROM timing

with a basic read cycle time of 150 ns. Signal sa_cs_3 is programmed for DSP and SysReg timing with a basic read/write cycle time of 100 ns. The Xbus cycle timing as well as the allocation of devices to the Xbus device select address ranges may be reconfigured by reprogramming the SA-1100 MSC0/MCS1 registers and reprogramming the Xbus Cntrl CPLD.

4.9.11 Xbus Timing and Control; Xbus Cntrl CPLD

The Xbus Cntrl CPLD uses the SA-1100 sa_cs_3, sa_cs_1 and sa_cs_0 signals to decode the entire Xbus address space. Using CS0 allows the ROMSEL pin into the SA-1100 to automatically define the boot ROM space as 16-bit space at power-up time. The Xbus CPLD decodes SA-1100 address bits A25:22 combined with sa_cs_O, sa_cs_1 and sa_cs_3 from the SA-1100 to partition the Xbus address space into three CS banks of 16, 4 MB segments.

Xbus devices less than 4 MB are multiply mapped through the 4 MB address space.

The sa_cs_3, sa_cs_1 and sa_cs_0 spaces can be configured in SA-1100 firmware using registers MCS0 and MCS1 to allow for slow and fast Xbus devices. The initial configuration will allow 200 ns, 16-bit wide devices in CS0 space and 150 ns, 32-bit wide devices in CS1 space and 100 ns, 16-bit wide devices in CS3 space.

The Xbus CPLD also provides support for Xbus daughter cards via the Xbus headers. Spare SA-1100 GPIO pins and spare address decodes are provided for in the CPLD.

The Xbus CPLD can be reprogrammed to use the spare resources as required. The Xbus data transceivers are controlled by the Xbus CPLD. The CPLD monitors buffered versions of the SA-1100 Oe and We as well as the PCMCIA control signals POE and PIOR signals to control the direction and output enables on the Xbus data transceivers. The Xbus CPLD also provides a slow-speed clock (approximately 1 MHz) for use by the Xbus CPLDs, primarily for contact bounce timing and interrupt edge generation. The 1.02 MHz Xclk is derived from the 28.6 MHz oscillator and is used by the Bt829a devices.

Note: The Xbus CPLD may be reprogrammed to combine sa_cs_3 and sa_cs_1 spaces in order to free up sa_cs_3 for a custom application.

4.9.11.1 System Address Space

Table 1-3 through Table 1-8 show the address space for the board's standard Xbus devices including flash memory, DRAM, Video FIFO, DSP and Xbus control registers. The address range field indicates the entire decoded range while the size:width field indicates the actual resource size and width within the decoded range. All decoded resources are address wrapped within their address range.

Table 4-3. Boot Flash ROM Address Space

Xbus CS0 200 ns space 16-bit space (MCS1-0 RBW0=1)	Address Range	Use	Resource Size: Width
0	0000,0000-003F,FFFC	Boot flash ROMspace	64 Kbytes: 16 bits wide
1 to 15	—	Unused	—

Table 4-4. Application Flash ROM Address Space

Xbus CS1 150 ns space 32-bit space (MCS1-0 RBW1=0)	Address Range	Use	Resource Size: Width
0	0800,0000-083F,FFFC	Application flash ROM	4 MB: 32 bits wide
1 to 15	—	Unused	—

Table 4-5. SA-1101 Development Board Registers Address Space

Xbus CS2 150 ns space 32-bit space (MCS1-0 RBW1=0)	Address Range	Use	Resource Size: Width
—	1000,0000-17FF,FFFC	SA-1101 development board registers	128 MB: 32 bits wide

Table 4-6. SA-1100 Multimedia Development Board Registers Address Space

Xbus CS3 100 ns space 16-bit space (MCS1-0 RBW3=1)	Address Range	Use	Resource Size: Width
0	1800,0000-1800,001E	Ct8020 DSP	16 registers: 8 bits wide
1	1840,0000	XbusReg	1 register: 2 bits wide
2	1880,0000-1880,0006	SysRegA	4 registers: 8 bits wide
3	18C0,0000-18C0,0006	SysRegB	4 registers: 8 bits wide
4	1900,0000-193F,FFF6	Spare CPLD A	Spare for SA-1101 development board CPLDs
5	1940,0000-197F,FFF6	Spare CPLD B	Spare for SA-1101 development board CPLDs
6 to 15	—	Unused/Reserved	—

Table 4-7. Video FIFO Address Space

DRAM Bank	Address Range	Use
RAS 0	C000,0000-C7FF,FFFF	DRAM bank 0
RAS 1	C800,0000-CFFF,FFFF	DRAM bank 1
RAS 2	D000,0000-D7FF,FFFF	Unused
RAS 3	D800,0000-D800,03FF	Video FIFO all 32 bits
RAS 3	D800,0400-D800,07FF	Video FIFO low 16 bits
RAS 3	D800,0800-D800,0BFF	Video FIFO high 16 bits
RAS 3	D800,0C00-D800,0FFF	Video FIFO no cycle
RAS 3	D800,1000-DFFF,FFFF	Video FIFO reserved
Zeros Bank	E000,0000-E7FF,FFFF	Cache cleaning/flushing

Table 4-8. PCMCIA Bank System Memory Map

PCMCIA	Address Range
Slot 0	2000,0000-2FFF,FFFF
Slot 1	3000,0000-3FFF,FFFF

4.9.12 Flash Memory

There are two banks of flash memory in the multimedia development board, boot flash and application flash. The use of a small 64 KB, 16-bit wide boot flash and a 4 MB, 32-bit wide application flash, represents a configuration that is best suited for low-cost consumer designs where the application flash could be implemented in a low-cost ROM.

In this type of configuration, the ROM stores compressed application code as well as execute in place code while system patches and nonvolatile storage are kept in the small, low-cost boot flash.

4.9.12.1 Boot Flash

The boot flash memory is a small 64 Kx16 (128 KB) socketed flash memory. The boot flash is selected by the SA-1100 chip select 0 (CS0) signal. In the SA-1100, CS0 is used to access the boot ROM after system reset time. The width of the boot ROM is determined by the ROMSEL pin on the SA-1100, which is tied to V_{DD} in the multimedia development board to indicate a 16-bit wide boot memory. This design allows for a small, low cost, removable, nonvolatile boot memory that can be quickly flashed to allow rapid system development.

Writes to the 16-bit wide boot flash memory on the Xbus must be restricted to lower halfword D15:00 (16-bit) operations. All 32-bit word reads to boot flash will be automatically sequenced as two 16-bit word reads by the SA-1100.

4.9.12.2 Application Flash

The application flash memory is composed of two 1 Mx16 (2 MB) Intel flash memories configured as 1 Mx32 (4 MB). The application flash is selected by the SA-1100 chip select 1 (CS1) signal. Some system development environments, such as Windows CE, may require more than 4 MB of application flash memory. An expanded flash ROM may be implemented with PCMCIA cards (requires the SA-1101 development board) or with a daughter board on the Xbus header. The Xbus control CPLD provides eight Xspare signals that may be coded as chip selects for additional banks of flash memory.

4.9.13 System Registers

The SysReg CPLDs provides extra GPIO pins, which are referred to as XGPIOs. These XGPIO pins are used to consolidate several video control signals, UCB1200 CODEC, soft resets, keypad signals, discrete LEDs, seven segment LEDs and switch pack signals. The SysRegA CPLD controls the following resources:

Table 4-9. System Register A Function Pins

Signal Name	Type	Function
FIFO reset	O	Resets both FIFOs
Vin Reset	O	Resets both decoders
DPS/UCB reset	O	Resets DSP and UCB1200
SK/Vout reset	O	Resets SA-1101 and video out circuits
SA_GPIO_25	O	Interrupt pin for KeyPad
Vidin Mode 1:0	O	Controls Vidin data path
Keypad X3:0	I	Keypad X receive
Keypad Y3:0	O	Keypad Y drive
LED 3:0	O	Discrete LED drive
LED 7:4	O	Discrete LED drive

The SysRegB CPLD controls the following resources:

Table 4-10. SysRegB CPLD Function Pin Descriptions

Signal name	Type	Function
Hex LED D3:0	O	Hex LED data
Hex LED0 strobe	O	Data strobe for Hex LED0
Hex LED1 strobe	O	Data strobe for Hex LED1
IrDA speed	O	Speed select for IrDA Xcvr. Shared with Hex LED D0
TV Irrmt_enb	O	Enables TV remote demodulation. Shared with Hex LED D1
SA GPIO 0, 1	O	Boot select/SW interrupt/DSP interrupt
DSP_Intr	I	Used to interrupt SA-1100 via GPIO 0
SW7:0	I	Switch pack bits 7:0 interrupt via GPIO 1
Video Capture	O	Controls single frame capture function
FIFO_Hi, FIFO_Lo	I	FIFO high half and FIFO low half interrupt polling

The Xbus CPLD controls the following resources.

Table 4-11. Xbus Function Pins

Signal name	Type	Function
SA_GPIO_17	I/O	Xbus spare SA_GPIO
SA_GPIO_16	I/O	Xbus spare SA_GPIO
Xspare_8:0	I/O	Xbus Xspare pins

Table 4-12. System Registers Address Map

Physical Address (16-bit space)	Symbol	Register Name
0x1840 0000	XCR	Xbus control register
0x1880 0000	SRR	System reset register
0x1880 0002	VIMR	Video input mode register
0x1880 0004	KPIOR	Keypad input output register
0x1880 0006	DLEDR	Discrete light emitting diode register
0x18C0 0000	HLEDR	Hex light emitting diode register
0x18C0 0002	SWR	Switch register
0x18C0 0004	XIRR	Xbus interrupt reason register
0x18C0 0006	SARR	SA-1100 multimedia development board revision register

Table 4-13. Xbus Control Register 0x1840,0000 (XCR)

Bit	Name	Description
1:0	SPGPIO[1:0]	SA-GPIO17 SA-GPIO16 (RW) These bits drive SA-GPIO17 and SA-GPIO16 pins on the SA-1100. They can be used for diagnostic test or be allocated to customer functions. These bits are cleared by a system reset. System Reset Register 0x1880 0000 (SRR)

Table 4-14. System Reset Register 0x1880 0000 (SRR)

Bit	Name	Description
3	DSPR	DSP Reset (RW) This bit drives the reset pin on the Ct8020 DSP. 0 – Holds the DSP in reset. 1 – Allows the DSP to run. This bit is cleared by a system reset.
2	VIR	Video In Reset (RW) This bit drives the reset inputs to the two Bt829a decoders and the video-in-data CPLD. 0 – Holds the devices in reset. 1 – Allows the devices to run. This bit is cleared by a system reset.
1	FIFOR	FIFO Reset (RW) This bit drives the reset inputs to the FIFOs and the Vinbst32 CPLD. 0 – Holds the devices in reset. 1 – Allows the devices to run. This bit is cleared by a system reset.
0	VOR	Video Out Reset (RW) This bit drives the reset inputs to the ADV7175 encoder and the Vocntrl CPLD. 0 – Holds the devices in reset. 1 – Allows the devices to run. This bit is cleared by a system reset.

Table 4-15. Video Input Mode Register 0x1880,0002 (VIMR)

Bit	Name	Description
1:0	VIM[1:0]	Video Input Mode (RW) These bits drive the mode pins on the video-in-data CPLD. 00- Main decoder 16-bit pixel, 32-bit FIFO, frame sync in C[0] 01-PIP decoder CCIR656 8-bit pixels, 32-bit FIFO, CCIR656 codes 10-Main decoder CCIR656 8-bit pixels, Lo 16-bit FIFO, CCIR656 codes PIP decoder CCIR656 8-bit pixels, Hi 16-bit FIFO, CCIR656 codes 11- Main decoder CCIR656 8-bit pixels, 32-bit FIFO, CCIR656 codes These bits are cleared by a system reset.

Table 4-16. Keypad Input/Output Register 0x1880,0004 (KPIOR)

Bit	Name	Description
3:0	KPOY[3:0]	Keypad Output Y signals (RW) These bits drive the four keypad column signals. KPOY0 – Drives the column CDEF KPOY1 – Drives the column 369B KPOY2 – Drives the column 2580 KPOY3 – Drives the column 147A These bits drive the block of four discrete LEDs near the bulkhead edge of the board labeled D2 on the silk screen. These bits are cleared by a system reset.
7:4	KPIX[3:0]	Keypad Input X signals (RO) These bits read the four keypad column signals KPIX0 – Receives the row A0BF KPIX1 – Receives the row 789E KPIX2 – Receives the row 456D KPIX3 – Receives the row 123C

Table 4-17. Discrete Light Emitting Diode Register 0x1880,0006 (DLEDR)

Bit	Name	Description
7:0	DLEDR[3:0]	Discrete Light Emitting Diode Register (RW) These bits drive the block of four discrete LEDs near the bulkhead edge of the board labeled D1 on the silk screen. These bits are cleared by a system reset.

Table 4-18. Hex Light Emitting Diode Register 0x18C0,0000 (HLEDR)

Bit	Name	Description
0	IRSPD	IrDA infrared transceiver speed (RW) 0 – Low speed IrDA 1 – High speed IrDA Note that this bit is shared with HLEDD0 This bit is cleared by a system reset.
1	TVRMT	TV Remote enable (RW) 0 – Pass IR receiver data signal undecoded. 1 – Decode and strip off the 30 KHz to 40 KHz TV IR remote carrier from the IR receive data signal. Note that this bit is shared with HLEDD1 This bit is cleared by a system reset.
3..0	HLEDD[3:0]	Hex LED Data (RW) This 4-bit hex code is displayed as a digit (0-F) on the HEX LEDs. These bits are cleared by a system reset.
4	HL0STB	Hex LED 0 Strobe (RW) 0 – Allows the hex LED 0 segment to display data that is present on HLEDD[3:0]. 1 – The data present on HLEDD[3:0] is latched into the hex LED 0 segment. This bit is cleared by a system reset.
5	HL1STB	Hex LED 1 Strobe (RW) 0 – Allows the hex LED 1 segment to display data that is present on HLEDD[3:0] 1 – The data present on HLEDD[3:0] is latched into the hex LED 1 segment. This bit is cleared by a system reset.
6	VIDCAP	Video Capture enable (RW) 0 – Disables single frame video capture FIFO timing. 1 – Enables single frame video capture FIFO timing. This bit is cleared by a system reset.
7	ENBINT	Enable DSP, FIFO and SWR interrupts (RW) 0 – Disable SWR, FIFO and DSP interrupts. Enables SWR1 and SWR0 to drive SA-GPIO1 and SA-GPIO0 after a system reset. SA-GPIO1 and SA-GPIO0 are used by Angel boot software to select one of four different executable images. 1 – Enable SWR, FIFO and DSP interrupts to drive SA-GPIO1 and SA-GPIO0. This bit may be set after system boot time to reallocate SA-GPIO1 and SA-GPIO0 from boot image selection to SWR and DSP interrupt functions. These bits are cleared by a system reset.

Table 4-19. Switch Register 0x18C0,0002 (SWR)

Bit	Name	Description
7:0	SWR[7:0]	Switch Register (RO) These 8 bits read the value in the 8 switches in switch pack S1. The switch register will be used to input system configuration information such as memory type, memory size and the boot vector for SA-GPIO1 and SA-GPIO0. Changes to the switches when the ENBINT=1 can generate interrupts via SA_GPIO1. This may be useful during development of software that requires simulated interrupts from lid openings, off hook or other user actions.

Table 4-20. Xbus Interrupt Reason Register 0x18C0,0004 (XIRR)

Bit	Name	Description
0	FIFO-LO	**FIFO Low half, half full (RO)** 0 – The lower 16-bit half of the video input FIFO is under ½ full. 1 – The lower 16-bit half of the video input FIFO is over ½ full.
1	FIFO-HI	FIFO High half, half full (RO) 0 – The upper 16-bit half of the video input FIFO is under ½ full. 1 – The upper 16-bit half of the video input FIFO is over ½ full.
2	DSPINTR	DSP Interrupt (RO) 0 – DSP interrupt pin true. 1 – DSP interrupt pin false.
3	INTRSEL	Interrupt Select (RW) 0 – SWR interrupts via SA_GPIO_1 and DSP interrupts via SA_GPIO_0 If ENBINT bit in HLEDR is true. 1 – FIFO interrupts via SA_GPIO_1 and SWR interrupts via SA_GPIO_0 If ENBINT bit in HLEDR is true.

Table 4-21. SA-1100 Multimedia Development Board Revision Register 0x18C0,0006 (SARR)

Bit	Name	Description
4:0	CPLDRR[4:0]	CPLD Revision Register (RO) These 5 bits represent the revision level of the multimedia development board CPLDs. The value of the revision level is incremented by 1 for each revision and the current value is recorded in the release notes.
7:5	HWRR[7:5]	HardWare Revision Register (RO) These 3 bits represent the revision level of the multimedia development board hardware. The value of the revision level is incremented by 1 for each revision and the current value is recorded in the release notes.

4.9.14 Ct8020 DSP

The interface to the Ct8020 is programmed I/O and event driven interrupts. The Ct8020 appears as 16 registers which are addressed as data bits 7:0 on lower ½ word (16-bit) boundaries. Byte writes are not supported on the Xbus, all writes must be to bits 7:0 of low ½ word boundaries. A dedicated SA-1100 GPIO pin is used for DSP to SA-1100 interrupts.

4.9.14.1 Ct8020 DSP Bus Timing

The Ct8020 is capable of 100 ns read and write cycles. Because the Ct8020 is faster than the flash ROM, a separate CS bank select from the SA-1100 is used to time the DSP read write timing. sa_cs_3 registers MCS0 and MCS1 are programmed to match the 50 ns read/write pulses and 50 ns read/write recovery times required by the Ct8020 interface.

4.9.14.2 Main DRAM Interface

The main DRAM used in the SA-1100 multimedia development board design is a 72-pin EDO DRAM SODIMM. The SODIMM is connected directly to the SA-1100 address/data/control bus without buffers. This tightly coupled DRAM interface allows for the fastest possible DRAM cycle times. The SODIMM compact style of memory card was chosen to minimize the trace length and loading on the SA-1100 address and data lines. 50 ns and 60 ns EDO DRAM SODIMM memory boards of 4 MB, 8 MB, 16 MB and 32 MB are available.

4.9.15 In-System Programmable CPLDs

The SA-1100 multimedia development board makes extensive use of nonvolatile ISP* CPLDs from Lattice Semiconductor. These devices offer from 1000 gate equivalents in the 2032V and up to 6000 gates in the 1032E.

Extra CPLDs were used with this board to provide a very flexible evaluation board design—most product designs based on this board will be much less complex and require fewer CPLDs. For example, a simple videophone or internet TV would require only one 2032V device.

The nonvolatile ISP feature combined with low cost, high speed as well as 5 V and 3.3 V tolerance in the case of the 2000V parts, allows the board to be modified and adapted to many different customer requirements. In addition ISP offers rapid system debug and development.

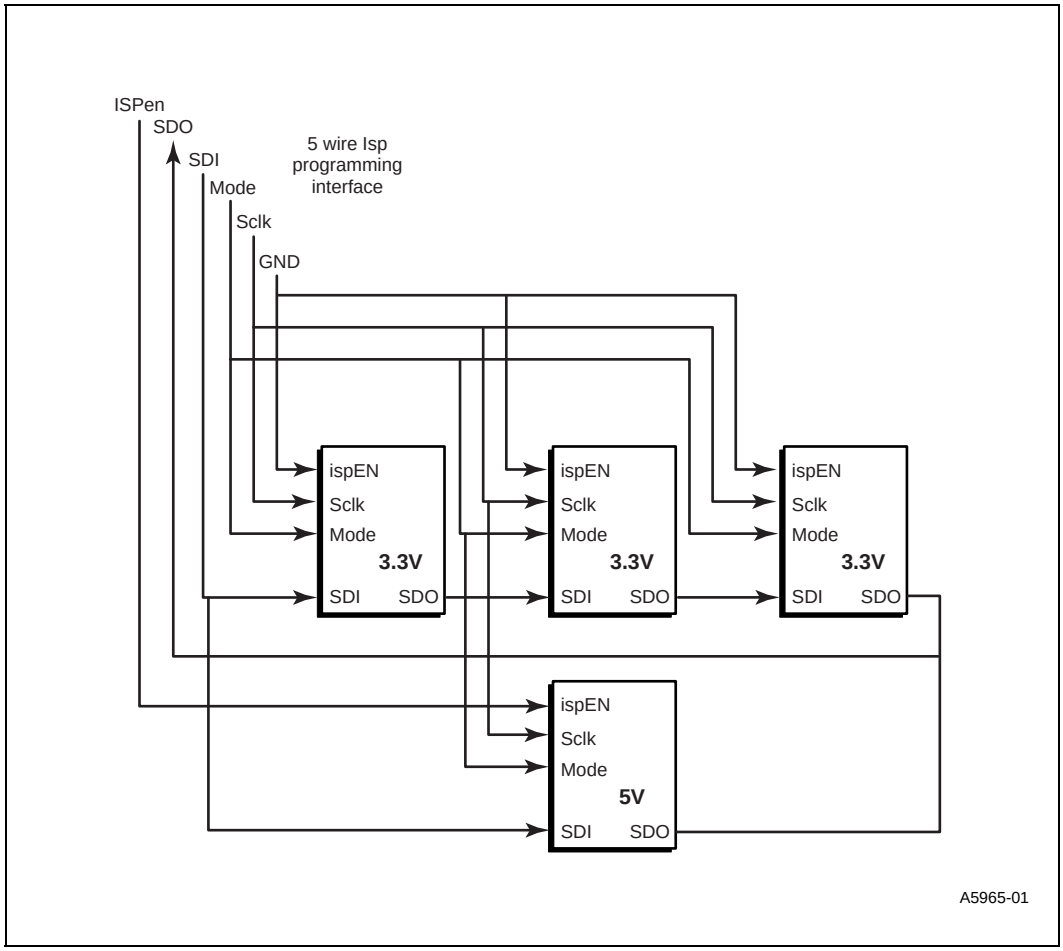
Product designs based on the CPLDs can transfer the logic to a system ASIC or use the low-cost CPLDs in early manufacturing startup and switch to lower cost ASICs later in the product cycle when the design is mature.

The ISP CPLDs have their ISP ports daisy chained to a single programming header. A cable and design tool package available from Lattice connects a PC printer port to the programming header. The parts may be programmed and or verified individually or all together using free ISP software from Lattice.

When daisy chaining 3.3 V and 5 V Lattice CPLDs, it is important to have the 3.3 V parts in a separate serial SDI SDO chain from the 5 V parts. The 3.3 V SDO and 5 V SDO are then connected together and returned to the program head.

Figure 4-9 shows how the ISP devices are daisy chained for programming. Note that the ispEN pins on the 3.3 V devices are connected to ground while the ispEN signal is connected to the ispEN pin on the 5 V parts.

Figure 4-9. ISP Programming Daisy Chain



4.9.16 SA-1100 GPIO Usage

The GPIO pins on the SA-1100 are a precious resource. Many of the 28 GPIO pins are required to perform their alternate functions leaving even fewer GPIOs available for general system implementation.

In order to preserve as many SA-1100 GPIO pins as possible for customer usage, the board design provides additional GPIO signals, called XGPIO, that are accessed directly via programmed IO. Some of these additional XGPIO signals are connected to eight LEDs, two seven-segment displays and eight switches as recommended by the Microsoft Windows CE development board design guide.

The XGPIO signals are implemented in ISP 2064V CPLDs that can be reconfigured to support specific customer needs. Table 4-22 shows the usage of SA-1100 GPIO pins on the SA-1100 multimedia development board as well as the SA-1100 development board and WebPhone reference designs.

Table 4-22. SA-1100 GPIO Usage

GPIO	SA-1100 Development Board[a]	SA-1100 Multimedia Development and SA-1101 Development Boards[b]	Webphone Reference Design
27	32 KHz Out	3.68 MHz Out	DEBUG_LED
26	RCLK_Out	GPIO or RCLK_Out	DEBUG_LED
25	KBC_ATN#	KeyPad IRQ/Xbus_spare	KBC_ATN-
24	KBC_WUKO	SA-1101 development board IRQ	KBC_WUKO
23	KBC_WKUP#	UCB_IRQ	KBC_WKUP-
22	IRQ_C#	nMBREQ	UBC_IRQ
21	IrDA_SD	nMBGNT	IrDA_SD
20	LED_RED#	SCB SDA	DEBUG_NOR (Debug mode or N)
19	SDLC_GPI	SCB SCL	KEYPAD_C4
18	SDLC_HSKI	FIFO_IRQ	KEYPAD_C3
17	SDLC_AAF	Xbus_spare	KEYPAD_C2
16	SDLC_HSKO	Xbus_spare	KEYPAD_C1
15	UART_RXD	UART_RXD	TAD_ACK-
14	UART_TXD	UART_TXD	FLASH_WP-
13	SSP_SFRM	Header SSP_SFRM/spare	KBC_SFRM
12	SSP_SCLK	Header SSP_SCLK/spare	KBC_SCLK
11	SSP_RXD	Header SSP_RXD/spare	KBC_MISO
10	SSP_TXD	Header SSP_TXD/spare	KBC_MOSI
9	LCD_D15/LED_GRN1#	LCD_D15	TPAD_DATA (Touchpad)
8	LCD_D14/LED_GRN2#	LCD_D14	TPAD_CLK (Touchpad)
7	LCD_D13/P1_F1#	LCD_D13	I2C_SCL
6	LCD_D12/P1_IREQ#	LCD_D12	I2C_SDA
5	LCD_D11/P1_STSCHG#	LCD_D11	SPKRDAA_OH
4	LCD_D10/P0_F1#	LCD_D10	SPKRDAA_RI-
3	LCD_D9/P0_IREQ#	LCD_D9	BAT_THRM
2	LCD_D8/P0_STSCHG#	LCD_D8	SPKRDAA_CID
1	SW1	SW1/SW7:0_IRQ/alt FIFO IRQ	HANDSET_OFFHOOK
0	SW0	SW0/DSP_IRQ/SW7:0_IRQ	SWITCH1

a. Order number DE-1S110-OA.
b. Order numbers DE-1S110-OB and DE-1S110-OC respectively.

Table 4-23. UCB 1200 CODEC

GPIO	SA-1100 Development Board[a]	SA-1100 Multimedia Development and SA-1101 Development Boards[b]	Webphone Reference Design
9	ADC_SYNC	ADC_SYNC	ADC_SYNC
8	DAA_OH	DAA_OH	DAA_OH
7	DAA_RI#	DAA_RI#	DAA_RI#
6	RED_LED#	7 segment dot LED	NOT USED
5	GREEN_LED#	NOT USED	NOT USED
4	SEVEN_SEG_BLANK	NOT USED	NOT USED
3	SEVEN_SEG_LED[3]	NOT USED	NOT USED
2	SEVEN_SEG_LED[2]	NOT USED	NOT USED
1	SEVEN_SEG_LED[1]	NOT USED	NOT USED
0	SEVEN_SEG_LED[0]	NOT USED	NOT USED

a. Order number DE-1S110-OA.
b. Order numbers DE-1S110-OB and DE-1S110-OC respectively.

4.9.17 System Reset

A LMC6953CM is used to sense 5 V and 3.3 V rails and generate a $\overline{\text{sys\_reset}}$ signal. A manual reset switch allows warm reset and system booting. Each subsystem in the design has a programmable soft reset control. The SysRegA CPLD provides the following reset controls:

- DSP_Codec_Reset
- SA-1101_Vout_Reset
- Vin_Reset
- FIFO_Reset

All of these soft reset signals are reset at system reset time. The application must clear the reset on any subsystem that is to be used or initialized.

4.9.18 System Power

Three terminal linear regulators are used for all voltages other than V_{CC} 5 V and V_{BB} 12 V. There is no provision for power management.

Name	Voltage	Supplies PowerTo
V_{CORE}	2 V	SA-1100 internal core
V_{DD}	3.3 V	SA-1100 I/O and most onboard logic
V_{CC}	5 V	V_{DD} regulators and some 5 V components
V_{LIN}	8 V	Analog audio signal components
V_{BB}	12 V	LCD header to power a back light and to the 8 V analog signal regulator

4.9.19 LED Displays

There are two seven-segment LED displays and eight discrete general-purpose programmable LEDs on the multimedia development board. Microsoft recommends this combination of indicators for use on all Windows CE development platforms.

Four of the eight discrete LEDs are used for the four keypad XGPIO column-drive signals.

In addition, the two “dot LEDS” on each of the seven segment LED displays are connected to GPIO pins on the UCB1200, main Bt829a, PIP Bt829a and audio DSP devices. This provides a visual verification that these devices are active and responding to their respective program ports. Power on self test (POST) firmware programs the USB1200, main Bt829a, PIP Bt829a and DSP devices to flash the dot LEDs.

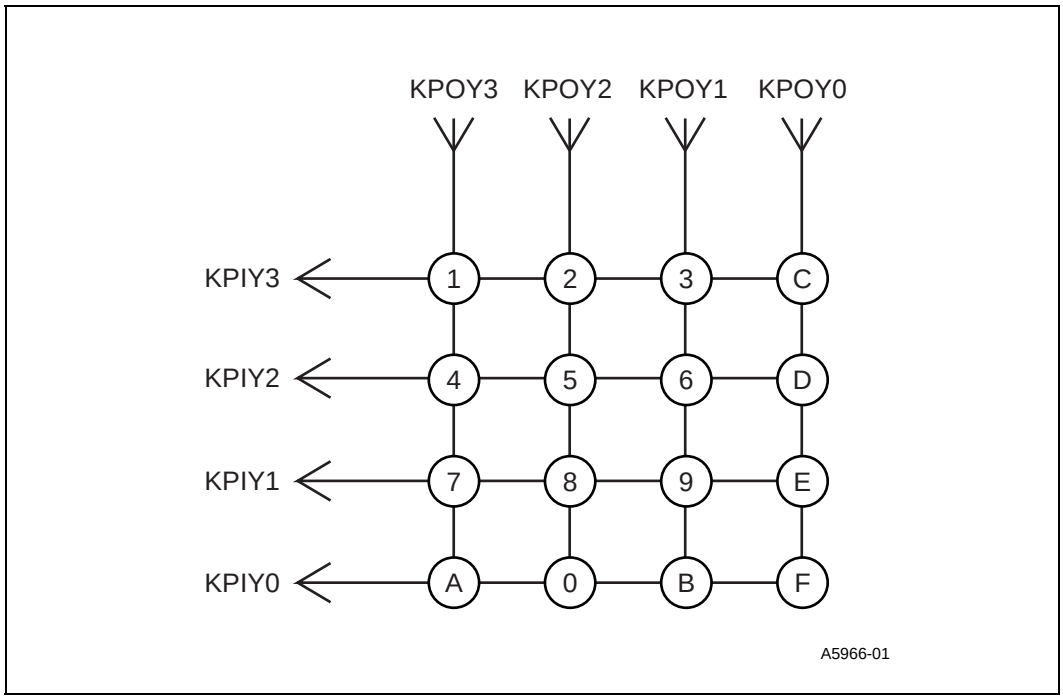
4.9.20 Keypad

A 4x4 matrix keypad suitable for use with telephone applications is supported in the design. The SysRegA CPLD provides eight XGPIO pins that allow writing to the four columns (KPIOY3:0) and reading the four rows (KPIOX3:0) of the switch matrix. The KPIOX signals are pulled up to a logic “1” when the keys are open (not pressed). The SysRegA CPLD is designed to pulse the Xbus interrupt pin SA GPIO25 when it detects a change in the keypad switches over a 16 ms period. Key press or key release causes SA GPIO25 to be driven low then high for about 16 ms. The SA-1100 must be programmed to interrupt on the falling edge of the SA GPIO25 signal. The rising edge of SA_GPIO_25 does not represent key release.

The SA-1100 keypad interrupt service routine should perform the following operations:

1. The four column drive pins KPIOY3:0 writes a 0 to allow a key contact in any row to drive a 0 into the four SysRegA keypad input pins KPIX3:0. The keypad input pins are pulled up by pullups programmed into the SysRegA CPLD.
2. The keypad interrupt service routine reads the SysRegA Xbus Keypad Input Output Register (KPIOR) to determine which row key is depressed.
3. The keypad interrupt service routine determines which column is depressed by writing a 0 to each column drive pin, one at a time, in KPIOY3:0 and re-sampling the row pins KPIX3:0.

Table 4-24. Keypad Matrix



The keypad interrupt service routine should complete the row scan and restore the KPIOY column drive pins in KPIOR to zeros before the next debounce clock occurs (16 ms) to avoid a spurious interrupt. Keypad debouncing for SA_GPIO_25 interrupts is achieved by a low speed clock that samples the keypad switches at a 64 Hz rate. The debounce specification for the 4x4 keypad is 10 ms. The KPIOR data register is not debounced therefore software must debounce the keypad data and detect key press and key release events. The 4x4 keypad is the primary user input device for diagnostics, sample applications, and demonstrations.

4.9.21 Switches

A switch pack containing eight switches is provided. Switch pack debouncing is achieved by a low speed clock that samples the switches at a 64 Hz rate. Changes in the switch settings are debounced and detected in the SysRegA CPLD and can cause an interrupt to the SA-1100 via the SA_GPIO1 pin. Switch bits 7 and 8 can be used to drive BATT_FLT and VDD_FLT. These SW bits and the fault signals are wired to the Vocntrl CPLD, which can be programmed to drive the fault signals, if required. (The default programming of the Vocntrl CPLD does not drive the BATT_FLT or VDD_FLT signals.)

4.9.22 SA-1100 GPIO 0 and 1

The SA-1100 GPIO 0 and 1 pins have special use at system boot time. The Angel micro debug boot software can read SA GPIO 0 and 1 to determine which of four boot images to jump to. After system reset, SA GPIO 0 and 1 are driven through the SysRegA CPLD from switch pack S0 and S1. After boot time, a bit in SysRegA can be set to allow the GPIOs to be used as interrupt pins for the DSP (SA_GPIO_0) and switch pack (SA_GPIO_1).

4.9.23 SCB

The SCB interface is a two-wire serial protocol requiring open collector drivers that allow “wire or” of data on the SCB bus. The SCB SDA (data) and SCL (clock) have pull-up resistors to V_{DD}, such that an undriven SCB pin will be high (one). The GPIO pins on the SA-1100 are tri-state drivers that have programmable control of direction and data as well as interrupt generation from either or both edges of a signal. To use the GPIO pins with the SCB requires the use of the GPIO direction control bits for two of the GPIO pins as SCB data bits.

The GPIO GPCR data register bits for both bits assigned to SCB are always cleared to zero. To drive a low (zero) on to the SCB SDA or SCL lines, the direction bit is set to one which drives the zero data on to the SCB pin. To drive a one, the GPDR bit for the SCB pin is set to zero, which reconfigures the pin as an input and turns off the tri-state driver allowing the pull-up on the SCB to pull the SCB line high to a one. The value of the SCB pins may be read at any time through the GPLR.

The SCB protocol is implemented in SA-1100 software with the assistance of interrupt driven, system timed, programmed I/O. The system timer establishes the SCB clock rate as well as avoiding polled PIO of the SCB GPIO pins. This scheme allows direct connection of two GPIO pins to the two-wire SCB bus with no external glue logic and a minimum of MIPS.

There are four SCB devices on the SA-1100 multimedia development board design. The SCB SDA and SCL signals are interfaced directly to SA-1100 GPIO pins with resistor pullups to 3.3 V.

Table 4-25. SCB Addresses

SCB Device	SCB Address
Main Bt829a video decoder	88
PIP Bt829a video decoder	8 A
ADV7176 video encoder	54 (AD7175 = D4)
TDA9860 analog audio processor	80

4.9.24 Interrupt Latency and Frequency

Three GPIO pins allocated to the video functions are programmed to generate SA-1100 interrupts. The approximate maximum interrupt latency needed to guarantee successful operation for video support is as follows:

- Decoder FIFO half full; 445 µs maximum from HF interrupt true until first read of FIFO for QCIF 15 FPS capture. Approximate interrupt frequency of 2 KHz for QCIF 15FPS.
- Decoder FIFO half full; 55 µs maximum from HF interrupt true until first read of FIFO for CCIR601 30 FPS capture. Approximate interrupt frequency of 2 KHz for CCIR601 30 FPS.
- SCB SDA SCB SCL Asynchronous bus, no maximum latency.

Note: The interrupt frequency and latency requirements for CCIR601 30 FPS may not be compatible with some operating systems, such as Microsoft Windows CE, unless these high frequency interrupts are handled separately.

4.9.25 Phone Codec

The UCB1200 Codec from Philips Semiconductor and a DAA telephone line interface support softmodem and speaker phone. The UCB1200 has two codecs: one for the phone line and one for microphone, speaker or handset.

The CPLDs on the SA-1100 multimedia development board allow the UCB1200 to interface to the SA-1100 synchronous serial port (default) or to the synchronous serial port on the Ct8020 DSP. Both configurations are viable, but because most applications do not require the Ct8020 DSP, the UCB1200 to the SA-1100 is the default setting.

When the CPLDs are installed to connect the UCB1200 to the Ct8020, the SA-1100 MCP signals must be programmed to be in a high-impedance state.

4.9.26 RS232 Ports

The SA-1100 multimedia development board design provides two RS232 ports. The primary port is available via a bulkhead mounted standard 9-pin connector. This port is used for system debug and firmware development. A second port is provided via a 0.1–inch header on the board. This port may be used for debug messages from the development environment.

4.9.27 Logic Analyzer Support

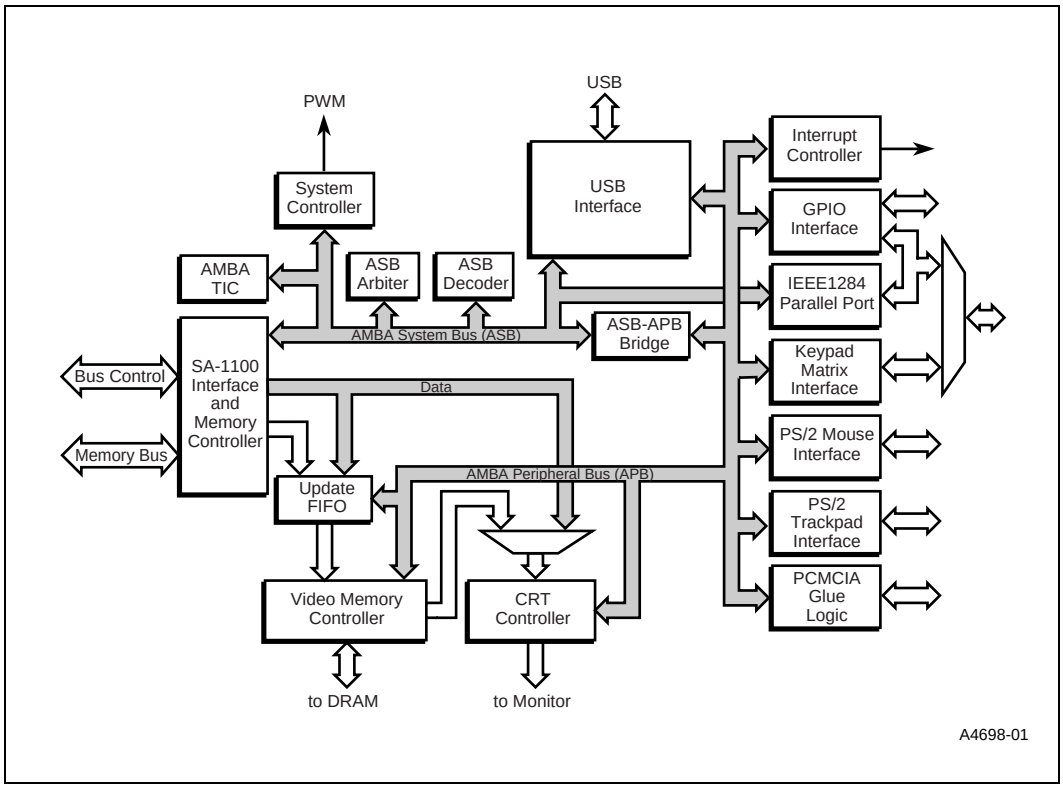
Logic analyzer support is needed on a very small percentage of all development boards. To reduce system complexity and cost, specific logic analyzer test heads are not provided. However, the SA-1101 development board connector provides access to all the SA-1100 memory interface signals.

Logic analyzers may be attached to a SA-1101 development board mating connector that can be hand wired to adapt to any logic analyzer. Alternately, the Xbus header provides a complete set of buffered address and data signals that may be used for logic analyzer connections when the connector is occupied by the daughter board. In addition, the Xspare8:0 signals on the Xbus header can be programmed to route any desired signals that are attached to any of the CPLDs on the SA-1100 multimedia development board or the SA-1101 development board. As an example, the Vinbst32 CPLD monitors SA_RAS3 and SA_CAS0 and routes them to Xspare5 and Xspare6 to allow a logic analyzer to monitor DRAM timing through the Xbus header.

4.10 SA-1101 Development Board Block Diagram

The SA-1101, which is designed around the ARM microcontroller bus architecture, provides a variety of I/O functions that enables complete systems to be built with a small number of components. This section describes these peripheral functions as shown in Figure 4-10.

Figure 4-10. SA-1101 Functional Block Diagram



4.10.0.1 Pulse Width Modulation Outputs

Two pulse width modulation (PWM) outputs are provided. These are intended for use as digital to analog converters with the addition of an external RC network.

4.10.0.2 USB Interface

A single port USB host controller is provided compliant with USB Specification Revision 1.0.

4.10.0.3 Interrupt Controller

Coordinates 63 interrupts that originate either from within the SA-1101 or from an external source connected to the GPIO interface. The interrupts can be enabled or disabled (masked) by setting a register bit in the interrupt controller.

4.10.0.4 GPIO Interface

Fifteen lines of general-purpose digital I/O are provided. Six of these are multiplexed with the keypad interface.

4.10.0.5 IEEE 1284 Parallel Port

An IEEE 1284 compliant parallel port, whose pins are multiplexed with the keypad outputs.

4.10.0.6 Keypad Matrix Interface

An 8 and a 16-bit port are available for use with a keypad. Each port can be configured as an input or output to allow use as either row or column connections to a keypad matrix. These pins are multiplexed with the IEEE 1284 parallel port.

4.10.0.7 PS/2 Ports

Two PS/2 ports are provided for use with keyboard, mice, trackpads or any other PS/2 compliant device.

4.10.0.8 PCMCIA Glue Logic

All the logic necessary for a PCI interface is provided to allow the simple connection to two PCMCIA sockets.

4.10.0.9 CRT Controller

Generates timing for three display resolutions (640x480, 800x600, 1024x768).

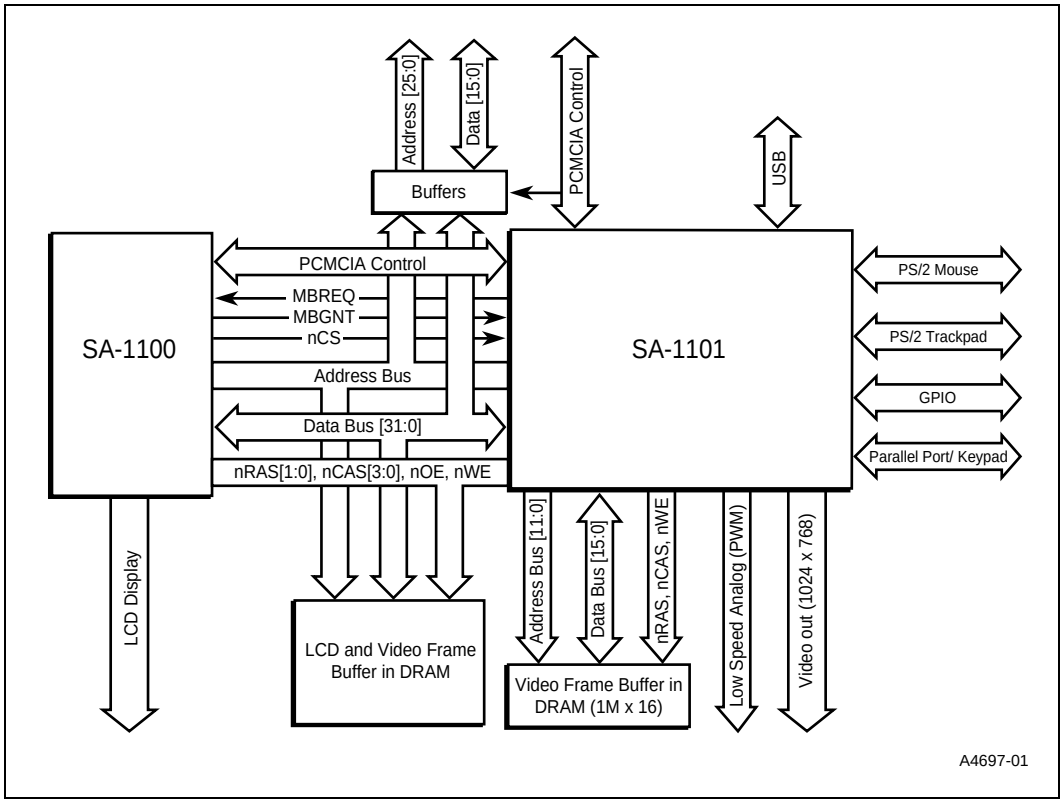
4.10.0.10 Video Memory Controller to DRAM

Generates memory timing to read and write frame buffer for dedicated mode.

4.10.1 SA-1100 and SA-1101 System Diagram

The following section describes how the SA-1100 and SA-1101 function together, as shown in Figure 4-11.

Figure 4-11. SA-1100 and SA-1101 Functional System Diagram



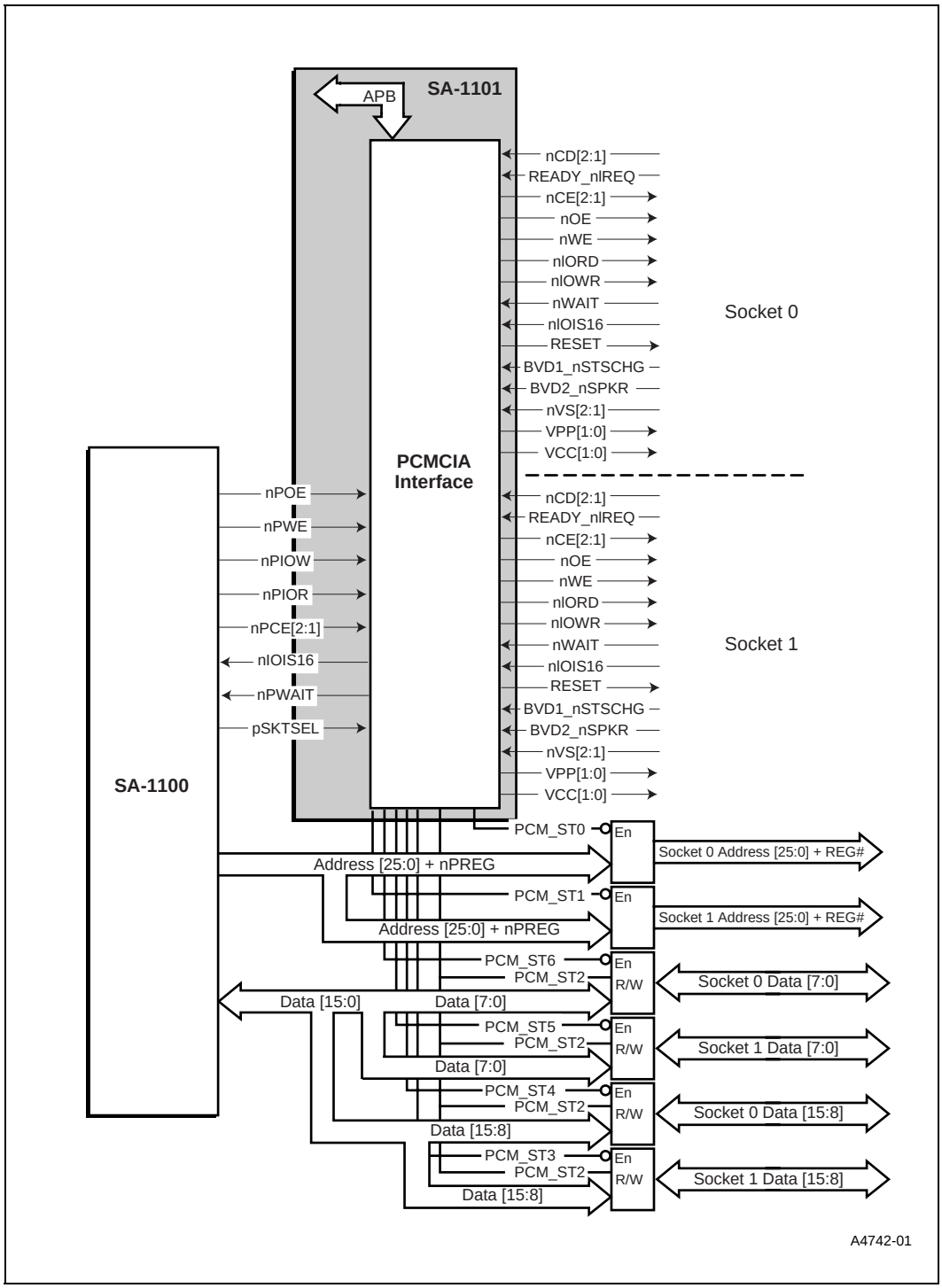
4.10.1.1 PCMCIA

The SA-1101 development daughter board, which supports two PCMCIA slots, is shown in Figure 4-12. Each slot has its own set of address and data buffers that are managed by the PCMCIA control signals generated by the SA-1100 and SA-1101. The SA-1101 receives all SA-1100 PCMCIA signals and generates all of the signals to manage two slots of PCMCIA including external transceivers and power control.

Two CPLDs on the SA-1101 development daughter board allow a PCMCIA interface to function in the absence of the SA-1101 device. The Bypass CPLD is connected to all inputs and outputs of the SA-1101 that relate to PCMCIA control and control of the MAX1600* PCMCIA power controller.

The Control CPLD implements registers in the Xbus address space drive the LED indicators and monitor key SA-1101 signals for diagnostic purposes. When the SA-1101 is installed and functioning, the CPLD devices are disabled or not mounted on the board.

Figure 4-12. SA-1101 PCMCIA Logic Block Diagram



4.10.1.2 VGA Video

The red, green, and blue video signals, plus the horizontal sync, and vertical sync signals are connected to a 15-pin D-Sub standard VGA header. The trimDAC header is aligned to the VGA test header such that a daughter card to a LCD and backlight will fit over the two headers.

4.10.1.3 DRAM

A standard surface mount 1Mx16 EDO DRAM is connected to the SA-1101.

4.10.1.4 USB

A standard host USB connector as well as a 16-pin test header is provided for the SA-1101 USB port. USB 5 V power is provided by a TPS2015 USB power distribution switch. The TPS2015 enable signal is controlled by SA-1101 GPIO_0 while the USB power fail signal is monitored by SA-1101 GPIO_1.

4.10.1.5 Contrast and Brightness PWM DACs

The two Pulse Width Modulated (PWM) DACs are connected to RC integrator circuits and made available to a test header.

4.10.1.6 IEEE1284 Printer Port

The SA-1101 IEEE1284 interface is connected to a standard 25 pin printer port header as well as a 13x2 test header.

4.10.1.7 PS2 Mouse and Touch Pad Ports

The two SA-1101 PS2 ports plus power signals are connected to standard PS2 headers. These signals are also available on test headers. A TPS2015 power controller for the PS2 ports is enabled by SA-1101 GPIO_2 while SA-1101 GPIO_3 monitors the TPS2015 PS2 power fail signal.

4.10.1.8 SA-1101 Development Board GPIO

SA-1101 GPIO A and GPIO B are connected to test headers.

4.10.1.9 Keyboard

The 16 X and 8 Y pins are connected to a 13x2 test header.

4.11 SA-1101 Display Modes

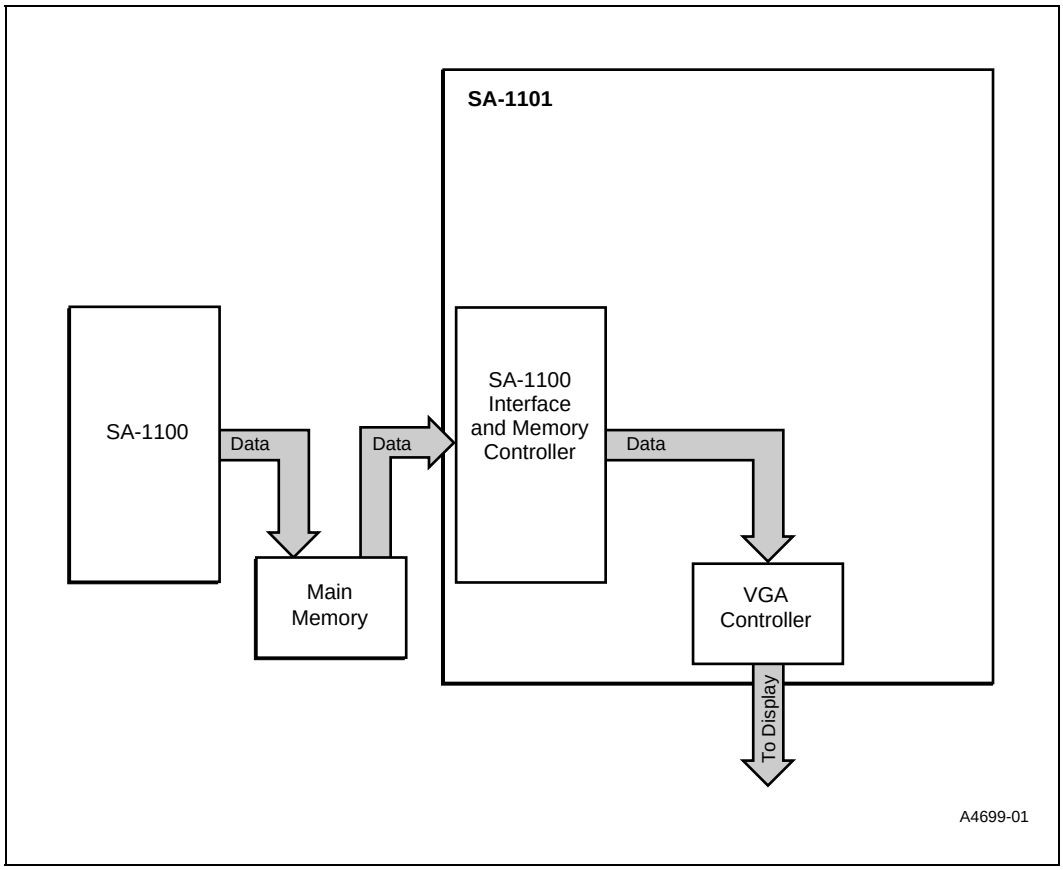
The following methods can be used for displaying video:

- Unified display mode
- Dedicated mode data flow

4.11.0.1 Unified Display Mode

As shown in Figure 4-13, the unified display mode method is applicable for low bandwidth, low resolution designs of 640x480. This design does not require an external frame buffer. However, because the memory is shared, the SA-1100 and the SA-1101 must compete for memory access.

Figure 4-13. Unified Display Mode

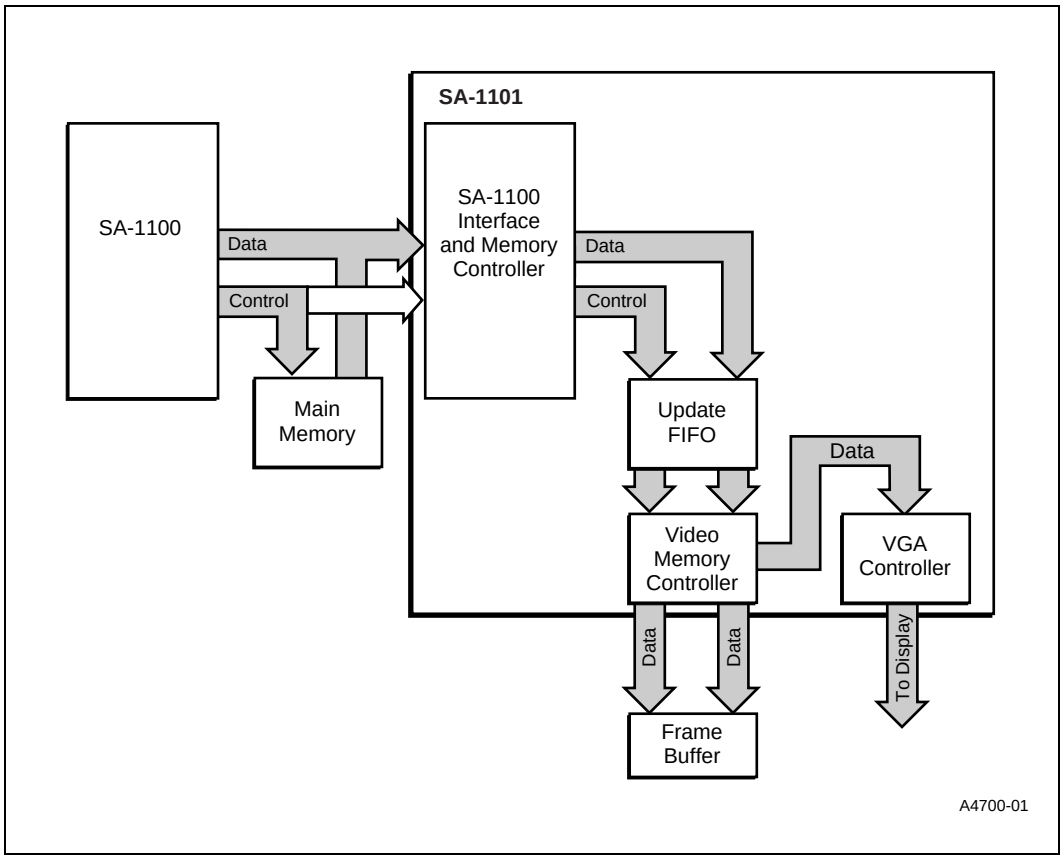


4.11.0.2 Dedicated Mode Data Flow

As shown in Figure 4-14,the dedicated mode data flow supports higher bandwidth and higher resolution displays. However, this method requires a frame buffer (single DRAM) for the refresh traffic for the video display.

In this method, the SA-1101 captures the data written to the main memory and copies it to the frame buffer. Using this method, the SA-1101 does not have to compete with the SA-1100 for memory access.

Figure 4-14. Dedicated Mode Data Flow



4.12 SA-1101 Development Board CPLD Registers

The following register specifications apply to the CPLD implementation of the SA-1101 development board when the SA-1101 device is not installed. Refer to the *StrongARM** SA-1101 Microprocessor Technical Reference Manual* for a complete description of the registers implemented in the SA-1101 device.

4.12.1 PCMCIA CPLD

The SA-1100 and SA-1101 development board combination supports a two-slot PCMCIA interface without the presence of the SA-1101 device. A Lattice Semiconductor 2128V CPLD provides the PCMCIA glue logic as well as four registers that control the PCMCIA slots. When the SA-1101 device is present, these CPLD registers are disabled. The PCMCIA is controlled by the PCMCIA registers in the SA-1101. Refer to the *StrongARM** SA-1101 Microprocessor Technical Reference Manual* for details on the SA-1101 PCMCIA control registers.

The PCMCIA interface supported by the SA-1100 StrongARM device, plus the CPLD glue and control registers, provide the following PCMCIA functions:

- Read PCMCIA power sense pins
- Control PCMCIA power signals: V_{CC} (5 V), V_{DD} (3.3 V), VPP (0 V, 5 V, 12 V)
- Read card detect pins
- Interrupt upon card insertion
- Read RDY/IRQ and STSCHG pins
- Interrupts based on RDY/IRQ and STSCHG pins
- Read and write PCMCIA memory space
- Read and write PCMCIA I/O space

4.12.1.1 Differences Between SA-1100 Development Board and SA-1100/SA-1101 Development Board PCMCIA Implementations

The SA-1100 multimedia development (DE-1S110-OB) and SA-1101 development boards' (DE-1S110-OC) PCMCIA functions are similar to those provided in the SA-1100 development board (DE-1S110-OA) reference design. The main differences between the SA-1100 development and SA-1100 multimedia development board as compared to the SA-1101 development board are as follows.

- Organization and addressing of board level PCMCIA control registers.
- SA-1100 development board uses six SA-1100 GPIO pins (GPIO 7:3) for PCMCIA status and interrupts.
- SA-1100 multimedia development and SA-1101 development board uses SA-1100 GPIO_24 for all PCMCIA interrupts. Register polling is required to determine PCMCIA interrupt reasons.

Table 4-26. PCMCIA CPLD Registers PCMCIA Power Sense Register 0x1940,0000 (PPSR)

Bit	Name	Description
0	S0 VS1	Slot 0 voltage sense 1 (RO)
1	S0 VS2	Slot 0 voltage sense 2 (RO)
2	S0 BVD1	Slot 0 battery voltage sense 1 (RO) The PCMCIA BVD1 signal also functions as the PCMCIA STSCHG status change signal
3	S0 BVD2	Slot 0 battery voltage sense 1 (RO)
4	S1 VS1	Slot 1 voltage sense 1 (RO)
5	S1 VS2	Slot 1 voltage sense 2 (RO)
6	S1 BVD1	Slot 1 battery voltage sense 1 (RO) The PCMCIA BVD1 signal also functions as the PCMCIA STSCHG status change signal
7	S1 BVD2	Slot 1 battery voltage sense 2 (RO)

Table 4-27. PCMCIA Power Control Register 0x1940,0002 (PPCR)

Bit	Name	Description
0	S0 VPP0	Slot 0 voltage programming potential 0 (RW) Refer to PCMCIA Power Control Register control codes table
1	S0 VPP1	Slot 0 voltage programming potential 1 (RW) Refer to PCMCIA Power Control Register control codes table
2	S0 V_{CC0}	Slot 0 voltage main power 0 (RW) Refer to PCMCIA Power Control Register control codes table 3 S0 V_{CC1} Slot 0 voltage main power 1 (RW). Refer to PCMCIA Power Control Register control codes table
4	S1 VPP0	Slot 1 voltage programming potential 0 (RW) Refer to PCMCIA Power Control Register control codes table
5	S1 VPP1	Slot 1 voltage programming potential 1 (RW) Refer to PCMCIA Power Control Register control codes table
6	S1 V_{CC0}	Slot 1 voltage main power 0 (RW) Refer to PCMCIA Power Control Register control codes table
7	S1 V_{CC1}	Slot 1 voltage main power 1 (RW) Refer to PCMCIA Power Control Register control codes table

Table 4-28. PCMCIA Power Control Register Control Codes

Sn V_{CC0}[a]	Sn V_{CC1}	Sn VPP1	Sn VPP0	Sn V_{CC}	Sn V_{PP}	Mode
0	0	0	0	GND	GND	STBY
0	0	0	1	GND	GND	STBY
0	0	1	0	GND	GND	STBY
0	0	1	1	GND	GND	STBY
0	1	0	0	3.3 V	GND	ACTIVE
0	1	0	1	3.3 V	5 V	ACTIVE
0	1	1	0	3.3 V	12 V	ACTIVE
0	1	1	1	3.3 V	Hi Z	ACTIVE
1	0	0	0	5 V	GND	ACTIVE
1	0	0	1	5 V	5 V	ACTIVE
1	0	1	0	5 V	12 V	ACTIVE
1	0	1	1	5 V	Hi Z	ACTIVE
1	1	0	0	3.3 V	GND	ACTIVE
1	1	0	1	3.3 V	5 V	ACTIVE
1	1	1	0	3.3 V	12 V	ACTIVE
1	1	1	1	3.3 V	Hi Z	ACTIVE

a. The order of the Sn V_{cc} have been arranged to correct an error in etch layout. For more information, see the release notes.

The MAX1600 device is used to control the PCMCIA power signals. For more information on the MAX1600, see the device specification at: http://www.nuhorizons.com/new/MXM6.html-ssi

Table 4-29. PCMCIA Status Register 0x1940,0004 (PSR)

Bit	Name	Description
0	S0 CD1	Slot 0 Card Detect 1 (RO) 0 – Slot empty 1 – Card inserted/Interrupt
1	S0 RDY	Slot 0 Ready/Interrupt (RO) 0 – Not ready, /no interrupt 1 – Ready./Interrupt
2	S0 STSCHG	Slot 0 Status Change (RO)
3	S0 Reset	Slot 0 Reset (RW) 0 – Slot reset 1 – Slot reset disabled
4	S1 CD1	Slot 1 Card Detect 1 (RO) 0 – Slot empty 1 – Card inserted/Interrupt

Table 4-29. PCMCIA Status Register 0x1940,0004 (PSR)

5	S1 RDY	Slot 1 Ready/Interrupt (RO) 0 – Not ready, /no interrupt 1 – Ready./Interrupt
6	S1 STSCHG	Slot 1 Status Change (RO)
7	S1 Reset	Slot 1 Reset (RW) 0 – Slot reset 1 – Slot reset disabled

4.12.2 StrongARM** SA-1100 Multimedia Development Board with

Table 4-30. SA-1101 Development Board Discrete LED Register 0x1900,0000 (SKDLR)

Bit	Name	Description
5:0	SKDL[5:0]	SA-1101 development board discrete LEDs (RW) 0 turns LED on 1 turns LED off
7:6	Reserved	Reserved (RO)

Companion SA-1101 Development Board PCMCIA CPLD Interrupts

The CPLD PCMCIA implementation in the SA-1100 and SA-1101 development board design uses a single SA-1100 GPIO pin (GPIO_24) for interrupts. The SA-1100 is programmable to be interrupted by either or both rising and falling edges of a GPIO pin. If PCMCIA interrupts are required, the SA-1100 GPIO_24 must be programmed to generate interrupts on both edges when used with the SA-1100 and SA-1101 development board CPLD PCMCIA logic. Any change in the following signals will result in GPIO_24 toggling.

- Slot 0 or 1 card detect
- Slot 0 or 1 RDY/IRQ
- Slot 0 or 1 STSCHG

The state of these six signals are compared in the CPLD with their previous state at a 3 MHz rate. One or more signals going true or false will result in GPIO_24 toggling and generating an interrupt to the SA-1100. This logic will result in interrupts for both the assertion of a signal as well as the de-assertion of that signal.

PCMCIA interrupts via SA-1100 GPIO_24 may be enabled or disabled via the SA-1100 interrupt control registers.

Software Video-Processing Engine Driver 5

This chapter describes the software video-processing engine for the SA-1100 microprocessor.

Note: The software described in this chapter is the prototype for the SA-1100 multimedia development SDK. The hardware described in this chapter is based on the SA-1100 multimedia development board.

5.1 Overview

Several driver-level functions have been developed to implement video pass through with the SA-1100 multimedia development board. The functions support the following operations:

1. Video hardware configuration
2. Video encoder initialization
3. Video decoder initialization
4. Steady state video capture engine
5. Optimized memory transfers (support full-speed NTSC video)
6. Sample video pass through and multi-resolution demonstration supporting two simultaneous video input streams and video out with PIP.
7. Video development software with a menu driven interface (text rendering) including NTSC mapped video fonts.

Items 1 through 5 provide an initial developer's library for capturing video and enabling digital video output with the SA-1100 multimedia development configuration. This code manipulates the SA-1100 internal clocks, caches, and LCD controller such that the video-output timing coupled with custom CPLDs' drive standard NTSC encoders.

Note: Due to the complexity of video timing issues, Intel recommends using these functions without modification. These software functions should be altered only by those fluent with StrongARM architectural issues and a strong digital video technology background.

The code is designed to deliver captured video into a user-specified frame buffer. The sample code directs video input to a SA-1100 video-output frame buffer. Overlaying the video buffer with the LCDs' frame buffer functionally demonstrates a real-time, digital-video, pass-through application.

In real-world applications, the developer may assign an intermediate data buffer for video input. By assigning a buffer, the video can be compressed and then delivered it to its destination. The destination can be both a remote site and a video window within the SA-1100 video-output buffer. Similarly, compressed video received at a remote host can be decompressed and directed into the video-output buffer. Data written to the video-output buffer is immediately displayed on the video-output monitor. (All video is maintained in YUV space.)

5.1.1 Constraints

An interrupt handler copies unprocessed video input directly into a user-specified buffer. NTSC decoders can be programmed to deliver video in discrete fields or in frames. This implementation uses the discrete field approach when acquiring captured video. Video copied into the user's intermediate buffer is spatially interlaced (that is, field0, followed by field1) in memory. Although the demonstration design is configured to spatially interlace the video, it also supports de-interlacing of the video in real time.

Currently, the video input supports a variety of standard video resolutions. If an unsupported resolution is required, the user must calculate the programming parameters using the Brooktree Bt829 decoder specification. The new parameters must be placed into a decoder data structure similar to those currently supported.

The SA-1100 LCD DMA controller directly accesses the video-output frame buffer. In order for the video encoder to coherently deliver the data to an interlaced NTSC monitor, the video fields must be spatially interlaced in the LCD frame buffer. Additionally, video fields must be separated in memory to compensate for the overscan intervals. The video-output copying algorithms provided with the multimedia tool suite appropriately deal with all these issues. Users who fully understand field and vertical-blanking spacing can provide their own mechanism. However, the routines provided in this application are also optimized for the SA-1100.

5.1.2 Software API

When reviewing the software, either in the provided samples or in the descriptions in this chapter, notice that all API entry points begin with **_sa**. This naming convention is used to prevent naming conflicts between functions provided by Intel and those provided by third parties.

5.2 SA-1100 Multimedia Development Board Configuration Routines:

Use these routines for developing video applications for the SA-1100.

5.2.1 _saInit1100

Syntax	Void _saInit1100 ()
Description:	Calls several multimedia setup functions required to configure the SA-1100 multimedia development board for video (see the called functions listed below).
Remarks:	Should be the first function called in main. Order must be maintained.
Return Value:	None
Called By:	Main(). Must be called prior to any video operation.

Calls:	_saConfigureClocks(CLK_SCALE_TO_148Mhz);
	_sa ConfigureVideoIRQ();
	_sa ConfigureMemorySpace();
	_sa ConfigureMemoryManagement ();

5.2.2 _saConfigureClocks

Syntax	Void _saConfigureClocks(unsigned CLK_SCALE_TO_XXXMhz)
Description:	This function sets up the external clocks for driving video out timing to the CPLD, which in turn drives the analog video encoder.
Remarks:	Called by _saInit1100(); it is not necessary for a programmer to directly call _saConfigureClocks
Return Value:	None
Parameters:	CPU clock speed. This value should be maintained in order to properly drive video out. Changing its value requires adjustments to two other video out timing parameters.
Called By:	saInit1100 ()
Calls:	—

5.2.3 _saConfigureVideoIRQ

Syntax	Void _saConfigureVideoIRQ()
Description:	This function sets up the video in FIFO interrupt.
Remarks:	Called by _saInit1100 (); it is not necessary for a programmer to directly call _saConfigureVideoIrq(). This function indirectly specifies the irq level to be used for the decoder FIFO's. For more details on FIQ/IRQ selection, refer to the SA-1100 architectural specification.
Return Value:	None
Called By:	_saInit1100()
Calls:	SetIrqLevel()—with SA-1100 IRQ level. DisableVideoDecoderIRQ()—to disable interrupts during setup

5.2.4 _saConfigureFIFOTimings

Syntax	Void _saConfigureFIFOTimings()
Description:	Sets up input timing on FIFO's memory space.
Remarks:	Called by _saInit1100(); it is not necessary for a programmer to directly call _ saConfigureFIFOTimings (). Programs FIFO memory timings to support video FIFOs. Formerly called _saConfigureMemorySpace.
Return Value:	None
Called By:	_saInit1100()
Calls:	—

5.2.5 _saConfigureMemoryManagement

Syntax	Void _saConfigureMemoryManagement ()
Description:	Configures the SA-1100's memory management system with video memory considerations and installs the video FIFO's interrupt vector.
Remarks:	Called by _saInit1100(); it is not necessary for a programmer to directly call _saConfigureMemoryManagement (). Because of stale cache issues that can arise after re-programming an interrupt vector, the installation of the video interrupt vector at this phase is crucial.
Return Value:	None
Called By:	_saInit1100()
Calls:	—

5.3 Video Output Routines

These routines implement the interface to the video encoder.

5.3.1 _saInitVideoOut

Syntax	void _saInitVideoOut(struct FrameBuffer *fbp,unsigned nBuffCnt)
Description:	Global video-out initialization utility. Initializes and configures the LCD controller, video-out buffer, and the NTSC encoder. Upon completion, data in the video buffer is displayed on the attached monitor.
Remarks:	Called by main(). This code expects a pointer to a FrameBuffer as defined below.

Note: The frame buffer is an area in off-chip memory that is used to supply enough encoded pixel values to fill the entire screen one or more times. This buffer must be evenly aligned on a 16-byte boundary (address bits 3-0 must be zero).

This code is intentionally called as the second item after the SA-1100 multimedia development board configuration function _saInit1100(). It can be called later, however, enabling video display in the early stages enables the display device for the rest of the setup stages (for example, selecting video-in resolutions).

Return Value:	None

Parameters:	Struct FrameBuffer *fbp;
	typedef unsigned short PIXEL;
	typedef struct FrameBuffer {
	unsigned short palette[DISPLAY_PALLETE_SIZE];
	PIXEL pixel[DISPLAY_MAX_Y +DISPLAY_OVERSCANS*2 +1]
	[DISPLAY_MAX_X];
	int row,col;
	};
	nBuffCnt—specifies the number of video frame buffers

Note: These buffers are consecutively addressed with fbp as frame buffer 0.

Called By: main ()

Calls: _saSetVideoOutFramePtr(fbp,nBuffCnt);
InitVideoDevice(SLAVEADDRESSAMD7176,0,0);
_saSetOddFieldOffset(DISPLAY_MAX_X,DISPLAY_MAX_Y,DISPLAY_OVERSCANS)
LcdSetup();

5.4 Video Input Routines

These routines implement interfaces to standard video decoders.

5.4.1 _saConfigureVideoIn

Syntax: void _saConfigureVideoIn(struct VideoStruct *VidSrc,unsigned vmode,unsigned bMainDecoder,unsigned rate,unsigned pitch)

Description: Configuration routine for video input parameters. This routine programs the video input parameters that are part of the video input data structure. The function of the video input data structure is to concisely group all required parameters related to a single video source. Grouping these parameters results in more concise code and better run-time performance from the efficiencies of SA-1100 relative base addressing.

Remarks: This function can be called multiple times to switch video capture resolutions in real time.

Return Value: None

Constraints: This function should only be called after video-in has been disabled (for example, following a call to _saStopVideoIn())

Parameters: Rate—specifies the frame rate frequency at which the software will check to insure captured video has maintained synchronization. If a loss of synchronization is detected, the system enters frame sync hunt mode. In this mode, software parses (scans) video data until a frame sync is detected, at which point video capture restarts. When no loss of synchronization is detected, a negligible performance penalty is paid for the sync check. It is recommended to set this value based on the expected noise of the video source. Noisier signals should have smaller rate parameters. For instance, a rate value of 300 would force the software to resynchronize the video every 10 seconds for a video stream that was delivering 30 frames a second.
Pitch—the number of pixels per video line. (This code uses a 4/3 aspect ratio)
Vmode—defines the video mode chosen, the following modes are to be supported. The demo code currently uses VIMR_MODE_DUAL_DECODE. The other modes are subsets of the dual decode mode and will require updating the CPLDs.
BMainDecoder—Selects either the main decoder or the PIP decoder

VIMR_MODE_CCIR601:- 1 video source(MAIN), CCIR601, packed[1]
VIMR_MODE_8BIT_MAIN:- 1 video source(MAIN), CCIR656, unpacked
VIMR_MODE_8BIT_PIP:- 1 video source(PIP) CCIR656, unpacked
VIMR_MODE_DUAL_DECODE:- 2 video sources(PIP/MAIN), CCIR656, unpacked

Called By: main ()

Calls: None

[1] In packed mode, both FIFOs are driven by a single video source. Each FIFO read contains 32 bits of valid pixel data. In unpacked mode each FIFO half is clocked independently. The upper half of all FIFO reads in this mode are ignored.

5.4.2 UserForegroundApplication

Syntax: void UserForegroundApplication()

Description: This is where the user application is expected to reside.

Remarks: When UserForegroundApplication() is executed, both the video input and output streams have been fully initialized. However, it is up to the application to actually make the SDK calls to configure and start the video capture process. This is because there are many varieties of video-in and it is the programmer's responsibility to understand and select the appropriate video-in to meet the needs of the application.

Return Value: None

Parameters: None

Called By: main ()

Calls: Fully up to user, but expected to call at a minimum
_saConfigureVideoIn()
_saSetCodecBufPtrs();
_saStopVideoIn();
_saStartVideoIn();

5.4.3 _saStartVideoIn

Syntax: void _saStartVideoIn(),_saStopVideoIn()

Description: These calls are used by the application developer to start and stop the video in.

Remarks: When video-in is active, the background interrupt handler steals a lot of cycles. In instances when the CODEC cannot keep up with the frame rate, it will be necessary to stop incoming video in order to balance CODEC operation with capture operation. Ultimately, it will be the developer's responsibility to determine the optimal balance, which leads to the maximum frame rate. This balance will be subjective not only to the incoming video resolution, but also to the overhead of background threads required by the application. Frame rate can be controlled via hardware or software. To control the frame rate by hardware, the decoder

must be programmed to a specified rate. To change the rate, the developer should refer to the BT829 technical specification, and change the appropriate entry in the decoder data structure (decoder entries are in the file “BT829.h”). Additionally, the user (as is done in our sample application) may need to start and stop video in order to change resolutions in real-time.

_saStopVideoIn() disables video interrupt sources.

_saStartVideoIn() sets up the hardware to enable video, programs the main decoder, and PIP decoder if in dual video mode, resets all video capture parameters to their startup state, resets the FIFOs, and finally enables FIFOs interrupts to the SA-1100

Constraints: _saStartVideoIn can only be called after video in configuration. The video in is configured by the _saConfigureVideoIn() function.

Return Value: None

Parameters: None

Called By: From within UserForegroundApplication().

Calls: Functions called by _saStopVideoIn():

DisableVideoIRQ()—disables video in interrupts

DisableLCDIRQ()—disable LCD frame IRQs

DisableTimerIRQ()—disable timer IRQs

Functions called by _saStartVideoIn():

ResetVideoCaptureParms()—resets all real-time video capture parameters

InitVideoDevice()—programs the onboard video decoders

ResetVideoInFIFO()—performs hardware FIFO reset

EnableVideoDecoderIRQ()—enables the FIFO interrupt

5.4.4 _saSetCodecBufParms

Syntax: void _saSetCodecBufParms(struct VideoStruct *VidSrc,unsigned bufWidth,unsigned bufHeight,unsigned frameSeparation,unsigned interlaced,unsigned uOffset)

Description: Sets global pointers and memory offsets used by the interrupt handler to map the incoming video stream into the CODEC frame buffer.

Remarks: This function operates on interlaced video and rectangular video windows within a rectangular memory buffer. As in the demo, it can specify output pointers directly to the SA1100’s frame buffer so that a coherent CCIR601 video image can be driven to an off-the-shelf analog video encoder. Unlike the demo however, a CODEC generally does not place the video-in stream directly into the LCD controller’s video frame buffer. In this case, the programmer could specify the bufWidth and BufHeight parameters to be identical to the actual video in stream’s pixel row and column widths. Similarly, the bufSeparation and interlaced

parameter could be set to zero. This would cause an interlaced video in stream to be written into a non-interlaced CODEC frame buffer.

Return Value: None

Parameters: Struct Video *VidSrc—pointer to the video sources parameter structure
Unsigned bufWidth—Pixel row count for entire buffer
Unsigned bufHeight—Pixel column count for entire buffer
Unsigned bufSeparation—number of video lines between even and odd field
Unsigned uOffset—pixel offset from upper left-hand corner of CCIR601 memory map where image will start. The sample code programs the offset to center images on the monitor display.
Interlaced—specifies whether captured video is interlaced into the CODEC frame buffer (for example, field 0 followed by field 1 in memory and separated by a number of bytes corresponding to the number of lines that would have occurred during the vertical blanking interval).

Called By: The users application

Constraints: Should only be called after the video in source for this CODEC has been set up by a call to _saConfigureVideoIn()

Calls: _saGetOddFieldOffset()—when interlaced mode is specified, saGetOddFieldOffset is called to do the math to determine the number of bytes which displace the start of field one from the start of field two in the frame buffer. If interlaced is not specified, incoming video is de-interlaced by copying line 1 from field 1 immediately after line 1 from field 0 in the CODEC's frame buffer.

5.4.5 _saGetOddFieldOffset

Syntax: unsigned _saGetOddFieldOffset(unsigned bufWidth,unsigned bufHeight, unsigned frameSeparation)

Description: Calculates the memory offset from the beginning of field 1 in the CODEC's frame buffers memory space, to the beginning of field two.

Remarks: Uses simple math based on the bufWidth parameters to determine the start offset from the beginning of field 1 to the beginning of field 2. The displacement accounts for the line time of the vertical blanking period that is specified by the frameSeparation parameter.

Return Value: Calculated offset.

Parameters: BufWidth—length of output video buffer line (note: this may actually be larger than the line size of the incoming video).
BufHeight—number of lines of video in incoming video stream (note: this may actually be larger then the incoming video's line count).

FrameSeparation—number of lines of blanking between two video fields (gets translated to bytes)

Called By: SaSetCodecBufParms()

Constraints: —

Calls: None

5.4.6 _saSetVideoOutFramePtrs

Syntax	void _saSetVideoOutFramePtrs(struct FrameBuffer *fbp,unsigned nBuffCnt)
Description:	Sets up an array of frame pointers for the video out control software. In the demo application, video-in goes directly to the video-out (pass through) requiring only a single video buffer. In a multi-buffered non-pass through application, the user will have to implement a similar function and separate the CODEC's input buffer from the video output buffer.
Remarks:	All frame buffers are assigned to contiguous spaces relative to the base address specified by fbp. Multiple frame buffers are supported so that incoming and outgoing video streams can be delayed and buffered (for example, to synchronize a video in source with a clock independent video out source).
Return Value:	None
Parameters:	Fbp- specified the address of the frame buffer in physical memory NBuffCnt – specifies the number of contiguous frame buffers to be set up
Called By:	_saInitVideoOut()
Constraints:	—
Calls:	None

5.4.7 _saGetVideoOutFramePtr

Syntax	struct FrameBuffer *_saGetVideoOutFramePtr(unsigned uFrameBufferNum)
Description:	Returns the value of the pointer indexed by uFrameBufferNum.
Remarks:	—
Return Value:	A pointer to a video/CODEC frame buffer.
Parameters:	UFrameBufferNum the index in the frame buffer pointer array.
Calls:	None

5.5 Video Input Theory Of Operation

This section assumes the reader has reviewed the hardware and is familiar with video technology.

5.5.1 Interface Description

Analog video input originates from an NTSC video decoder. The decoder captures analog video, converts it to digital, and delivers it to an onboard 1024-pixel FIFO. This 1K FIFO asserts a digital signal when the FIFO is half full or contains 512 2-byte pixels. This signal is de-asserted only when the FIFO is clocked below the half-full watermark. The FIFO half-full signal is connected to GP[1], which is set up as an interrupt source for the SA-1100. Consequently, available digital video is signaled by an interrupt on GP[1]. If the interrupt is not handled before the FIFO is full, the FIFO overflows, causing video data to be lost.

5.5.2 Interrupt Handling

Initially, the video decoder is programmed (through a software SCB interface) with user parameters to configure a video capture session. Following decoder configuration, software can enable the video interrupt source. Intel recommends that the video FIFOs be reset prior to enabling the interrupt source.

After the decoder is configured, the FIFO cleared, and video interrupts enabled, the SA-1100 CPU begins to receive video interrupts. At this point, video input is sending digital video data, but its operation is asynchronous to the system. In order to synchronize, a frame starting point must be found. To find the frame starting point, the software hunts for a frame synchronization code in the pixel data.

Considering the previous, the following describes video capture from a cold start.

When the first interrupt is received, video is not synchronized. The video engine goes into hunt mode and searches for a frame sync. If the interrupt handler clocks out its allotment of pixels (based on frame size) without detecting frame sync, the interrupt handler exits, and waits to be re-vectored. Eventually, this process results in frame sync detection. Upon detecting frame sync, the handler continues to clock out real pixels and moves them into the codec's first-field frame buffer (specified in the user setup code). The handler header continues to clock out and transfer the pixels until it has clocked out the allotment that caused the interrupt. When this allotment is reached, the handler again exits, and waits for the next interrupt. The interrupt handler maintains a counter of total clocked pixels for the given field, and when it reaches a full-field count, the handler adjusts the memory buffer pointer to the codec's second-field buffer. When completion of the second field is detected, the pointer is switched back to field one, and the process continues indefinitely. (If the user supports multiple buffers, at the end of the second field of video data the frame pointer should be moved to point to the new frame buffer.)

This process runs as described above with the following exception. The SA-1100 multimedia development board SDK requires the programmer to specify a frame re-sync counter. A total frame counter is checked upon completion of each captured frame. When the total frame count equals the user specified re-sync count, total count is reset, and the system is forced back into frame sync hunt mode. For noisy sources the developer may choose to decrease the frame hunt count, which will synchronize the video more frequently.

The preceding paragraphs describe the provided video-transfer engine. It is the user's responsibility to make this engine operate in synchronization with their codec. While the provided code continually updates a single frame buffer, the user must prevent frames from being overwritten as the data they contain is processed. To do this, the user can either implement some form of multiple frame buffering, or use the SDK _saStopVideoIn*()* call to disable incoming video while processing the previous frame. The advantage of calling _saStopVideoIn*()* is that it reduces the CPU cycles required for processing the frame. The disadvantage is that, if adequate cycles are available, fewer frames per second are processed, frame synchronization is required more frequently, and cycles are always wasted when hunting for the next frame.

5.5.3 Video-Out Theory of Operation

The SA-1100 contains an LCD controller peripheral for driving standard off-the-shelf LCDs. To drive NTSC video encoders, glue logic has been added. This glue logic is transparent to the software development effort, and the programmer can implement video as if writing directly to a standard LCD frame buffer. The single requirement for displaying coherent video is that the data in the LCD's frame buffer (specified by the user) must be spatially interlaced. This means that the data must be divided into two separate video fields and separated by a horizontal-blanking field. To find field pointers for a given resolution and position in the CCIR601 video window, the user can reference the calculated variables returned by the SDK call _saSetCodecBufParms*()*.

5.6 Miscellaneous

In addition to providing a video interface, several user interface functions are provided with the toolkit. These functions enable the demo of the video interface to be user configurable and provide other standard functional requirements. These functions include, keyboard entry, text output (to analog video display), and some miscellaneous programming utilities. This section describes these utilities.

5.6.1 Data Entry Functions:

The SA-1100 multimedia development board contains a 16 key keypad. This section describes the keypad and display utilities.

5.7 Keyboard Utilities

These routines implement user I/O functions on the SA-1100 multimedia board.

5.7.1 getcFromKBCTL

Syntax	int getcFromKBCTL(int wait)
Description:	Performs polled I/O while debouncing the keypad.
Remarks:	—
Return Value:	The ASCII hexadecimal equivalent of the pressed key (for example '0' – 'F')

Parameters:	If wait is non-zero this function waits indefinitely until the user presses a key; otherwise if no key is pressed the function returns 0.
Called By:	—
Constraints:	—
Calls:	None of interest

5.7.2 AskYesNo

Syntax	int AskYesNo(char *sPrompt)
Description:	Displays question to user and waits for a yes(1) or a no(0) response.
Remarks:	—
Return Value:	1 if yes, 0 if no
Parameters:	SPrompt – charter string to be displayed to user
Called By:	—
Calls:	None

5.8 Video Out Utilities

These utilities perform high-level video out operations to the display.

5.8.1 lcd_prints

Syntax	void lcd_prints(char m[],unsigned int row,unsigned int col,int color, int bgColor)
Description:	Print the specified text string to the video display at the row and column specified
Remarks:	—
Return Value:	None
Parameters:	M – character string to display Row – starting row of first character Col – starting column of first character Color – Foreground character color BgColor Background color
Called By:	—

5.8.2 putCToFB

Syntax	int putCToFB(char c, unsigned int row, unsigned int col,int color, int bgColor)
Description:	Displays specified character to video display.

Remarks:	—
Return Value:	If row or column out of range returns –1, otherwise 0
Parameters:	C – character to print Row – starting row Col – starting column Color – Foreground character color BgColor Background color
Called By:	—

5.8.3 ClearFrameBuffer

Syntax	void ClearFrameBuffer(unsigned clearChar,PIXEL *fbp)
Description:	Clears out frame buffer.
Remarks:	—
Return Value:	None
Parameters:	ClearChar – value to be written to every character in frame buffer Fbp- pointer to start of frame buffer
Called By:	—

5.8.4 interleave

Syntax	void interleave(unsigned short *pBitMap,unsigned short *pFrameBuf,int nDCols,int nDRows,int nOverScan,int nPixelMask)
Description:	Takes a non-interleaved YUV image, and interleaves it for the video display.
Remarks:	—
Return Value:	None
Parameters:	PBitMap – pointer to source image PFrameBuf – pointer to destination frame buffer NDCols – number of columns in image NDrows – number of rows in image NOverScan – number of overscan rows
Called By:	—

5.9 Timer Utilities

These routines implement timer delays and alarms.

5.9.0.1 init10usTimer

Syntax	void init10usTimer(unsigned timerSel,unsigned us)
Description:	Initializes a 10µs resolution timer using the SA1100's internal clock timers.
Remarks:	—
Return Value:	None
Parameters:	TimerSel – specifies the SA-1100's timer to use (may be 0-3)
Called By:	—
Constraints:	Programmed to run on 10µs resolutions. Higher timer resolutions are available from the SA-1100 but the programmer must explicitly program the timer for them. It is the developer's responsibility to install an interrupt handler that appropriately deals with the timer interrupt when it occurs
Calls:	void TimerSetup(unsigned timerSel,unsigned cnt) – programs the specified Clock register

5.9.1 EnableTimerIRQ

Syntax	void EnableTimerIRQ(unsigned timerSel)
Description:	Enable interrupts on the specified SA-1100 timer
Remarks:	—
Return Value:	None
Parameters:	TimerSel – specifies the SA-1100's timer to use (may be 0-3)
Called By:	—
Constraints:	TimerSel must be legal timer value (0-3)
Calls:	void TimerSetup(unsigned timerSel,unsigned cnt) – programs the specified Clock register

5.9.2 DisableTimerIRQ

Syntax	void DisableTimerIRQ(unsigned timerSel)
Description:	Disable interrupts on the specified SA-1100 timer
Remarks:	—
Return Value:	None
Parameters:	TimerSel – specifies the SA-1100's timer to use (may be 0-3)

Called By:	—
Constraints:	TimerSel must be legal timer value (0-3)
Calls:	void TimerSetup(unsigned timerSel,unsigned cnt) – programs the specified Clock register

5.9.3 Wait

Syntax	void Wait(unsigned Microscont)
Description:	Stall the CPU for specified number of microseconds.
Remarks:	Inefficient use of CPU, as only interrupts will get CPU time when “Wait() is called.
Return Value:	None
Parameters:	Microscont – count in microseconds of delay
Called By:	—
Constraints:	Value can range from 0x0 to 0xFF
Calls:	None

5.10 Miscellaneous Utilities

These following utilities perform high-speed memory operations and control the display of the hexLEDs.

5.10.1 _saFastMemCopy

Syntax	_saFastMemCopy(char *dst,char *src, unsigned byteCount)
Description:	Using buffered reads, and burst writes, transfers data from source to destination.
Remarks:	For best performance dst and source should be aligned on 32 byte boundaries.
Return Value:	None
Parameters:	Dst – address to move data to Src – address to get data from ByteCount – number of bytes to transfer
Called By:	—
Constraints:	For code to get maximum efficiency, the user must enable buffered reads in the translation tables for the source address space. Additionally, to maximize performance, the user may adjust the memory timing registers in the SA-1100 to meet the minimum setup/transfer times of the installed memory components.
Calls:	None

Note: The _saFastMemCopy is a template for a more efficient copying algorithm that demonstrates buffered reading and writing. It does not solve all memory copying cases, such as overlapping buffers.

5.10.2 _saSetHexLEDs

Syntax	_saSetHexLEDs(unsigned val)
Description:	Outputs specified valued on two hex digit led.
Remarks:	—
Return Value:	None
Parameters:	Val – Hex value to be displayed on SA-1100 multimedia development LEDs
Called By:	—
Constraints:	A value from 0x00 to 0xFF
Calls:	—

SCB Library 6

6.1 Overview

The following section describes the StrongARM SCB library. The SCB library provides a set of utilities for using the SA-1100 as an SCB bus master.

6.2 Functional Description

SCB is a two-wire serial hardware bus and protocol that enables multiple devices to reside on the same bus and communicate with each other. SCB is also a relative simple protocol and implementation for low-cost devices.

Each SCB device is required to have a unique address. The SCB device that initiates communication is the master, and the receiver is the slave. A device can be both a master and a slave at different times. The software provided with this kit has been engineered to program all devices as slaves with the exception of the SA-1100 device, which is programmed to be the master.

The SCB implementation for the SA-1100 evaluation board uses GPIOs 19 & 20 as the SCB clock and data lines. On the SA-1100 device, SCB is the interface to the two digital video decoders and the analog video encoder.

For more information on SCB bus operations or other SCB devices, see the data sheets for SCB devices. All SCB functions names on the SA-1100 device have the prefix "_saIIC_" followed by the operation type.

Action	Section	Command
SCB bus Initialization	Driver I/O Operations	void_saIIC_Init(void)
SCB bus data line		void_saIIC_setSDA(WORD level)
SCB bus clock line		void_saIIC_setSCL(WORD level)
Start sequence	SCB Primitive Operations	void_saIIC_StartSeq(void)
Stop sequence		void_saIIC_StopSeq(void)
ACK sequence		void_saIIC_ACK(void)
NAK sequence		void_saIIC_NAK(void)
Wait for NAK from slave		void_saIIC_GetACK(void)
Address of slave device	Slave I/O Operations	void_saIIC_SetSlaveAddr(UCHAR uSlaveAddress, UCHAR RWbit)
Reads byte from slave		UCHAR_saIIC_ReadByte(void)
Writes byte to slave		UCHAR_saIIC_WriteByte(void)
Writes byte to addressed device		void_saIIC_RandomByteWrite(UCHAR SlaveAddress, UCHAR SubAddress, UCHAR data)

Action	Section	Command
Burst read from slave	Burst I/O Operations	void_saIIC_BurstRead(UCHAR uSlaveAddress, UCHAR *pBuffer, UCHAR ucOffset, WORD iNumberBytes)
Burst write from slave		void_saIIC_BurstWrite(UCHAR uSlaveAddress, struct RegistersTableInfo *RegisterTable, long tableSize)
Burst write then read to verify		void_saIIC_BurstWriteVerify(uSlaveAddress, RegistersTable, long tableSize)
Burst read with compare		void_saIIC_BurstReadVerify(uSlaveAddress, RegistersTable, long tableSize)

6.3 Driver I/O Operations

The following commands are used for initializing and driving clock and data lines on the SCB bus.

6.3.1 _saIIC_Init

Syntax	void _saIIC_Init(void)
Description:	Initializes the GPIO 19 and 20 clock and data lines for SCB bus operations. Upon return, the SCB bus is ready to be driven with high or low signals on the clock and data lines.
Calling Sequence:	This function must be called prior to any other SCB operations.
Parameters:	None.
Return Value:	None.

6.3.2 _saIIC_SetSDA

Syntax	void _saIIC_SetSDA(WORD level)
Description:	This function drives SDA to the specified high or low level.
Calling Sequence:	Can be called at any time.
Parameters:	HIGH and LOW are definitions which determine the resulting output level on the SCB SDA line.
Return Value:	None.

6.3.3 _saIIC_SetSCL

Syntax	void _saIIC_SetSCL(WORD level)
Description:	This function drives SCL to the specified level.
Calling Sequence:	Can be called at any time.
Parameters:	HIGH and LOW are defines which determine the resulting output level on the SCB SCL line.
Return Value:	None.

6.4 SCB Primitives Operations

The following commands are used for invoking SCB bus events.

6.4.1 _saIIC_StartSeq

Syntax	void _saIIC_StartSeq(void)
Description:	Send bus master start sequence on the SCB bus.
Calling Sequence:	Called by the bus master (SA-1100) to initiate a communication session with an SCB slave device.
Parameters:	None.
Return Value:	None.

6.4.2 _saIIC_StopSeq

Syntax	void _saIIC_StopSeq(void)
Description:	Sends a bus master stop sequence on the SCB bus.
Calling Sequence:	Called by bus master (SA1100) to terminate a communication session with an SCB slave device.
Parameters:	None.
Return Value:	None.

6.4.3 _saIIC_SetSlaveAddr

Syntax	void _saIIC_SetSlaveAddr(UCHAR uSlaveAddress,UCHAR RWbit)
Description:	Puts the device address on the SCB bus.
Calling Sequence:	Called after saIICStartSeq(); to address slave device for communication.
Parameters:	USlaveAddress – address of SCB device.
	Rwbit – indicates direction of i/o operation (1=read, 0=write).
Return Value:	None.

6.4.4 _saIIC_ACK

Syntax	void _saIIC_ACK(void)
Description:	Drives the SCB ACK sequence on bus.
Calling Sequence:	During data transfers to separate consecutive data to same device (for more information see the SCB specification).
Parameters:	None.
Return Value:	None.

6.4.5 _saIIC_NAK

Syntax	void _saIIC_NAK(void)
Description:	Drives the SCB NAK sequence on bus.
Calling Sequence:	None.
Parameters:	None.
Return Value:	None.

6.4.6 _saIIC_GetACK

Syntax	void _saIIC_GetACK(void)
Description:	Turns bus around to wait for slave NAK.
Calling Sequence:	None.
Parameters:	None.
Return Value:	None.

6.4.7 Slave I/O Operations

The following commands are used for reading and writing information to and from the slave.

6.4.8 _saIIC_ReadByte

Syntax	UCHAR _saIIC_ReadByte(void)
Description:	Clocks a byte from a pre-addressed slave device on SCB bus.
Calling Sequence:	After device has been addressed and set up for read.
Parameters:	None.
Return Value:	Data read from device.

6.4.9 _saIIC_WriteByte

Syntax	void _saIIC_WriteByte(UCHAR data)
Description:	Clocks a byte to a pre-addressed slave device on SCB bus.
Calling Sequence:	After device has been addressed and set up for write.
Parameters:	Data—byte to be sent.
Return Value:	None.

6.4.10 _saIIC_RandomByteWrite

Syntax	void _saIIC_RandomByteWrite(UCHAR SlaveAddress,UCHAR SubAddress,UCHAR data)
Description:	Transmits data to addressed device on SCB bus.
Calling Sequence:	At any time.
Parameters:	Data—byte to be sent.
	SlaveAddress—device's SCB address.
	SubAddress—device's subaddress.
Return Value:	None.

6.5 Burst I/O Operations

The following commands are used for reading and writing information in a burst sequence.

6.5.1 _saIIC_BurstRead

Syntax	void _saIIC_BurstRead(UCHAR uSlaveAddress,UCHAR *pBuffer,UCHAR ucOffset,WORD iNumberBytes)
Description:	Performs a burst read from specified device into specified buffer.
Calling Sequence:	At any time.
Parameters:	PBuffer—points to destination buffer.
	SlaveAddress—device's SCB address.
	ucOffset—offset into buffer and device.
	iNumberBytes—byte count.
Return Value:	None.

6.5.1.1 Table Burst Operations

The following SCB table burst operations are designed to work from predefined configuration tables which a user builds to implement table driven SCB operations. The tables contain the operations device subaddress, the data to be downloaded, and a verification parameter. Burst table structures are defined using the following C data structure:

```
enum RegTypes {ReadWrite,ReadOnly,WriteOnly,Verify,Reserved};
struct RegistersTableInfo {
       int addr;
       int data;
       int code;
       char *text;
       } ;
```

6.5.2 _saIIC_BurstWrite

Syntax	void _saIIC_BurstWrite(UCHAR uSlaveAddress,struct RegistersTableInfo *RegisterTable,long tableSize)
Description:	Performs a burst write using the addresses and data specified in the register table.
Calling Sequence:	At any time.
Parameters:	SlaveAddress—device's SCB address.
	RegisterTable—data structure with device addresses, data, and verification parameters.
	TableSize—number of items to write from table.
Return Value:	None.

6.5.3 _saIIC_BurstWriteVerify

Syntax	WORD _saIIC_BurstWriteVerify(uSlaveAddress, RegisterTable,long tableSize)
Description:	Performs a burst write followed by a read back to verify written data (see burst write).
Calling Sequence:	At any time.
Parameters:	SlaveAddress—device's SCB address.
	RegisterTable—data structure with device addresses, data, and verification parameters.
	TableSize—number of items to write from table.
Return Value:	Error count, 0 if no error.

6.5.4 _saIIC_BurstReadVerify

Syntax	WORD _saIIC_BurstReadVerify(uSlaveAddress, RegisterTable,long tableSize)
Description:	Performs a burst read and then compares to a known data table.
Calling Sequence:	At any time
Parameters:	SlaveAddress—device's SCB address
	RegisterTable—data structure with device addresses, data, and verification parameters
	TableSize—number of items to read from table
Return Value:	Error count, 0 if no Error

Firmware Modification 7

7.1 Sa-1100 Firmware Introduction

This chapter describes the contents of the firmware and applications available for the SA-1100 multimedia development platform.

7.2 Firmware Contents

The firmware is supplied in three parts:

- Angel Debug Monitor
- SA-1100 multimedia development board and SA-1101 development board diagnostics
- StrongARM application

7.3 SA-1100 Multimedia Development Board Specific Software

Angel is a remote debug monitor which communicates with a debug host running the ARM Software Development Toolkit version 2.11a via the Angel Debug Protocol (ADP). It allows the developer to upload images onto the board and to debug them. Angel is usually held in flash part E3 on the SA-1100 multimedia development board, however, it can alternately be held in E1 and E4 for upgrades. The 1.05 firmware release contains a new Angel/bootloader image and, if you are not already running this image, you must upgrade to it. Angel also includes bootloader software that determines which image to run at boot time. By default Angel will run, but images held in the socketed (E3) and unsocketed (E1, E4) flash parts can be selected with Switch Position 0 and Switch Position 1 of component S1 (for more information on selecting boot images, see Table 3.6 “Bootloader Control” on page 3-11).

Please refer to the README.TXT file supplied for details of the contents of this firmware and information on how to build it. For more information on building and using uHAL, please refer to the documentation supplied with the uHAL software.

7.4 The Flash Management Utility

Flash device E3 is an Atmel A29LV1024 component organized as 64Kx16-bits. This component is read twice to provide 32 bits of data. Flash devices E1 and E4 are Intel 28F016 components organized as 2Mx16-bits. E1 is mapped into data bus lines 0-15, while E4 is mapped onto data lines 16-31.

The Flash Management Utility (FMU) runs from the ARM Debugger. It keeps a RAM copy of the flash bank to manipulate. It keeps track of which blocks are changed then updates the flash accordingly. This version of FMU allows the user to program any binary format file into any flash location.

7.4.1 FMU Commands

The Flash Management Utility supports the following commands.

Command	Description
List	no parameters List all of the images currently programmed into the flash and ROM
Listblocks	no parameters Lists the contents of the first 4 words of each flash block and the image, if any, associated with that block.
Testblock	<block number> Writes a test pattern into the given flash block in order to verify its functionality. Note: This destroys the block contents
Delete	<image number> Delete the image from the flash.
Deleteblock	<block number> Delete the block. This writes 0xFF to the entire flash block. It will not allow deletion of a block that is part of an image.
Deleteall	<no parameters> Deletes all of the flash blocks.
Program	<Image No> <Name> <Filename> <Block no> [<Noboot>] Programs an image into flash. Image no is the tag that FMU will use to label the image. 'Name' is the label that will appear in the list function. 'Filename' is the full pathname to the AIF file to put into flash. The user may optionally supply a block number into which the image will go, otherwise FMU will find the first empty block and put it there. The image may span multiple blocks. The 'Noboot' option allows the user to specify that image will not execute but will be loaded.
Quit	This calls the exit routine which is in fact a SWI call that uses semi hosting to communicate to the host that the program has terminated.

7.5 Booting an Image From Flash

uHAL 1.0 contains example images that can either be loaded and run via the Angel Debug Monitor (Angel images) or run at boot time (stand-alone images). For a description of the memory maps, see Section 4.9.11.1, “System Address Space” on page 4-20 and Section 4.9.12, “Flash Memory” on page 4-22.

Use the following procedure after you have a stand-alone image in a flash device.

1. Establish a connection with the Angel Debug Monitor; upload fmu.axf and run it:

```
$ armsd -remote -port 2 -adp -line 115200
A.R.M. Source-level Debugger vsn 4.48 (Advanced RISC Machines SDT 2.11a) [Dec  2 1997]
Angel Debug Monitor for SideARM (FIQ),  MMU on, Caches enabled,  Clock Switching on (serial)
1.02 (Advanced RISC Machines SDT 2.11a) rebuilt on Aug 20 1998 at 17:18:31
Serial Rate 11520
armsd: load fmu.axf
armsd: go
Flash Management Utility [v0.1] (Angel)
Entering SVC mode...Done
Disabling caches...Done
Searching for flash device
man is 0X00890089
ID is 0X66A066A0
Flash found at 0x08000000 (32 blocks of size 0X20000)
Scanning Flash blocks for usage
FMU>
```

2. Program ping.axf into flash as image 2 in flash block 4:

```
FMU> program 2 ping standalone/ping.axf 4
Caching the file in memory
Attempting to get aif header
Writing standalone/ping.axf into flash block 4
Deleting blocks ready to program:
Deleting block 4
Delete flash block at offset 0x0 words
Attempting Flash Sync
Sync’d flash.
Calculating checksum
Writing Flash header at 0x40000
Writing flash image header
Image is non-executable AIF file
Image will execute from Flash
padding is 0 bytes
Writing image file
############################################################
Attempting Flash Sync
Sync’d flash.
Scanning Flash blocks for usage
FMU>
```

3. Select image 2 to run at boot time by placing switch 1 up (1) and switch 0 down (0) on S1.
4. Quit from FMU and the Angel Debug Monitor.

5. Reset the SA-1100 multimedia development board. The following output will be on the second serial port (J22) and on the 8 bit LCD panel:

```
TargetInit() called
Setting up the soft vectors
Timer init
TargetInit() complete
Ping.c, calling OSStart()
IRQInstall() called 0x1C
1+[1-2] 1+[1-2] 1+[1-2] 1+[1-2] 1+[1-2]
```

7.6 Building Angel Images

The SA-1100 multimedia development board specific platform includes the system specific sources to the Angel Debug Monitor. These sources need to be combined with those in the 2.11a release of the ARM SDT. The ARM SDT 2.11a Angel source tree's angel subdirectory contains the system independent angel sources (for example angel/startrom.s contains the first code executed when Angel is run). Within this directory there are two directories for each system that has had angel ported to it. For the SA-1100 multimedia development boards these are sidearm and sidearm.b. The sidearm directory contains the SA-1100 multimedia development board-specific sources and the sidearm.b directory contains files used to build (hence the b suffix) angel images.

Starting with a clean ARM SDT 2.11a angel source tree, copy over the two directories sidearm/angel/sources/sidearm and sidearm/angel/sources/sidearm.b. Any non-system specific angel source files must be replaced with those from sidearm/angel/sources (cdefs.h and startrom.s). If you already have an angel source tree, for example to build angel images for the EBSA-285 StrongARM Evaluation Board, you must be careful that the versions of these common files are the same.

Once this has been completed, new angel images can be built either using the GNU make file in angel/sidearm.b/gccsunos/makefile or the ARM project file in angel/sidearm.b/apm/angelsa.apj. Both of these build the same images: angel.axf, angel.srec. The flash management utility must be used to program the angel.axf into flash. The angel.srec can be programmed using an external flash programmer.

7.7 Updating the CPLDs

Use the following procedure to download new code into the CPLDs. For information about programming the CPLDs, see Appendix C.

1. Power down the SA-1100 multimedia board
2. Connect the ISP download cable to the printer port of the PC
3. Connect the 8-pin header on the ISP download cable to J23 on the SA-1100 multimedia board
4. Power up the SA-1100 multimedia board
5. Launch the ISP download software on the PC
6. In the ISP download software, select the configuration file that describes the SA-1100 multimedia board and the CPLD programming file list (*.JED files)
7. Select *go* from the function menu

8. Disconnect the ISP cable from J23 on the SA-1100 multimedia board
9. Re-boot the SA-1100 multimedia board

Note: The ISP software on the PC also supports CPLD verification and checksums.

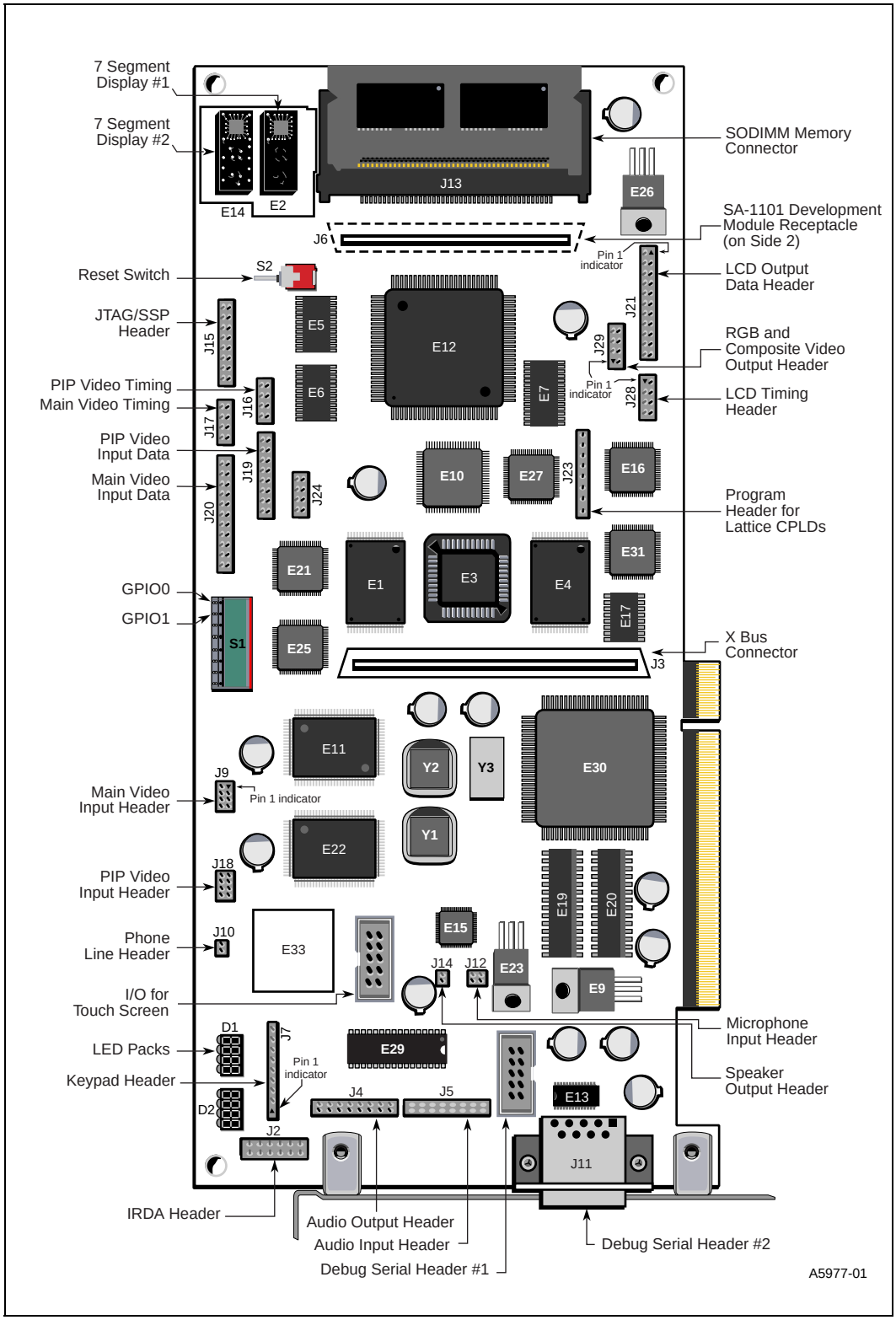
SA-1100 Board Configuration Guide A

This appendix describes the header pins, switches, and LEDs on the SA-1100 multimedia development board.

A.1 SA-1100 Components

Figure A-1 illustrates the location of set up switches, seven-segment display, indicator LEDS, keypad header, SODIMM header, connectors, and reset switch of the SA-1100 multimedia development board. The SA-1100 connectors and headers are described in Table A-1.

Figure A-1. SA-1100 Multimedia Development Board



A.2 Description of Headers and Connectors

Table A-1 provides a brief description of all the headers and connectors for the SA-1100 multimedia development board.

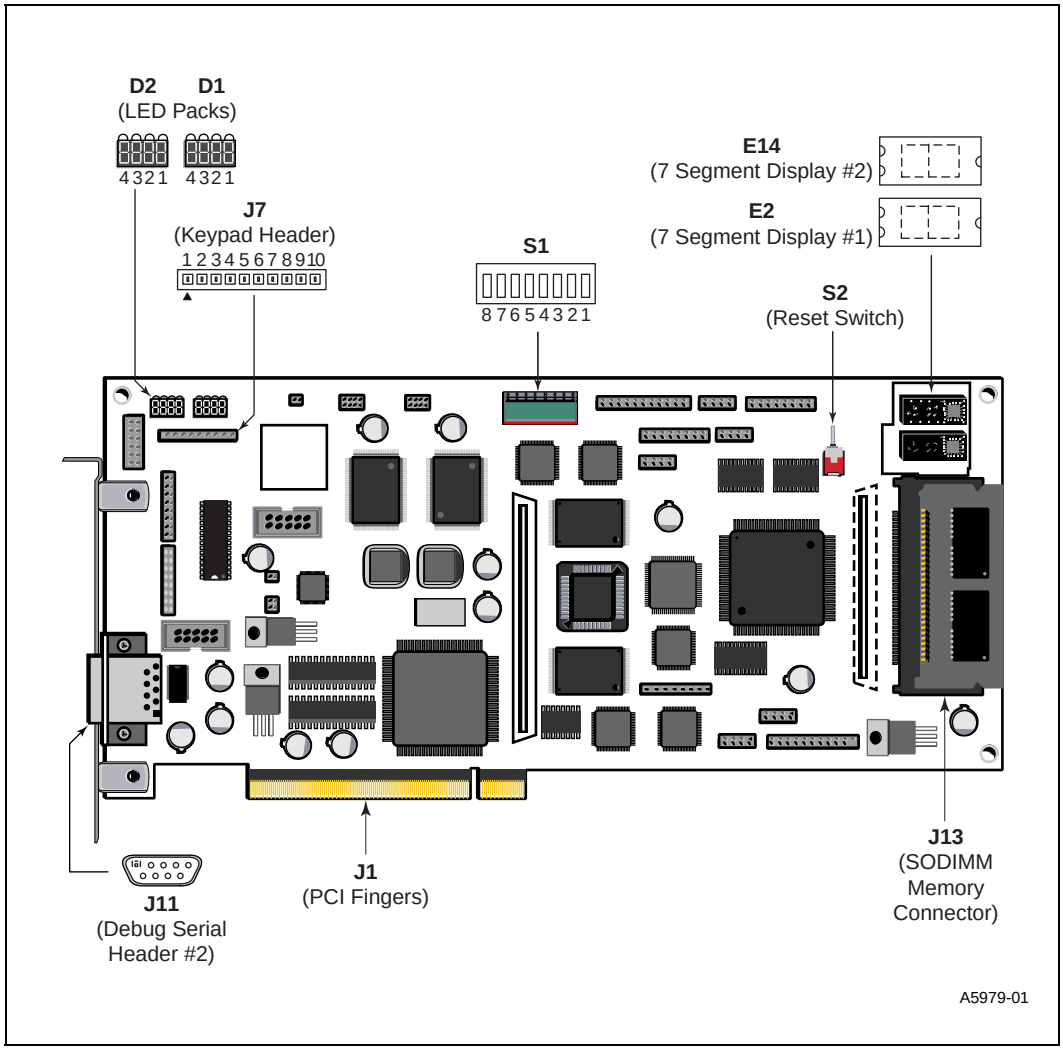
Table A-1. SA-1100 Headers and Connectors

Ref	Description
J2	IrDA Header
J3	Xbus connector
J4	Audio output header
J5	Audio input header
J6	SA-1101 development board receptacle (side 2)
J7	Keypad header
J8	Debug serial header
J9	Main video input header
J10	Phone line header
J11	Debug serial header #2
J12	Microphone input header
J13	SODIMM memory connector
J14	Speaker output header
J15	JTAG/SSP header
J16	PIP video timing
J17	Main video timing
J18	PIP video input header
J19	PIP video input data
J20	Main video input data
J21	LCD output data header
J23	Program header for lattice CPLD's
J28	LCD timing header
J29	RGB and composite video output header

A.2.1 Switches, Displays, and I/O Functions

Figure A-2, illustrates the location of set up switches, seven-segment display, indicator LEDS, keypad header, SODIMM header, and reset switch of the SA-1100 multimedia development board.

Figure A-2. S1, S2, LEDs, Keypad and Debug Headers and SODIMM Connector



As described in Table A-2, configuration switch S1 determines ROM image selection, boot selection, and which video FIFO interrupt to disable.

Table A-2. Configuration Switch S1

Switch Position	Signal Name	Function	CPLD Connection	Switch Open (Up)	Setting: Closed (Down)	User Defined Purpose
#1	SW0	ROM image selection	E21 pin 31	—	—	—
#2	SW1	ROM image selection	E21 pin 32	—	—	—
#3	SW2	Boot ROM Select	E21 pin 33	Boot from boot flash (E3)	Boot from application flash (E1, E4)	Factory defined
#4	SW3	User defined	E21 pin 34	—	—	—
#5	SW4	User defined	E21 pin 35	—	—	—
#6	SW5	User defined	E21 pin 36	—	—	—
#7	SW6	Disable video FIFO interrupts	E21 pin 37	Main video FIFO interrupt disabled	Main video FIFO interrupt enabled	—
#8	SW7	Disable video FIFO interrupts	E21 pin 38	PIP video FIFO interrupt disabled	PIP video FIFO interrupt enabled	—

A.2.2 Seven Segment Displays: E2, E14

The seven-segment displays of E2 and E14 provide power-up status information about the power-up self test and error information for Angel and customer applications.While the SA-1100 multimedia board is being initialized, E2 and E14 display “00” and change dynamically while Angel is starting. For information on the reset switch, see Section A.2.3.

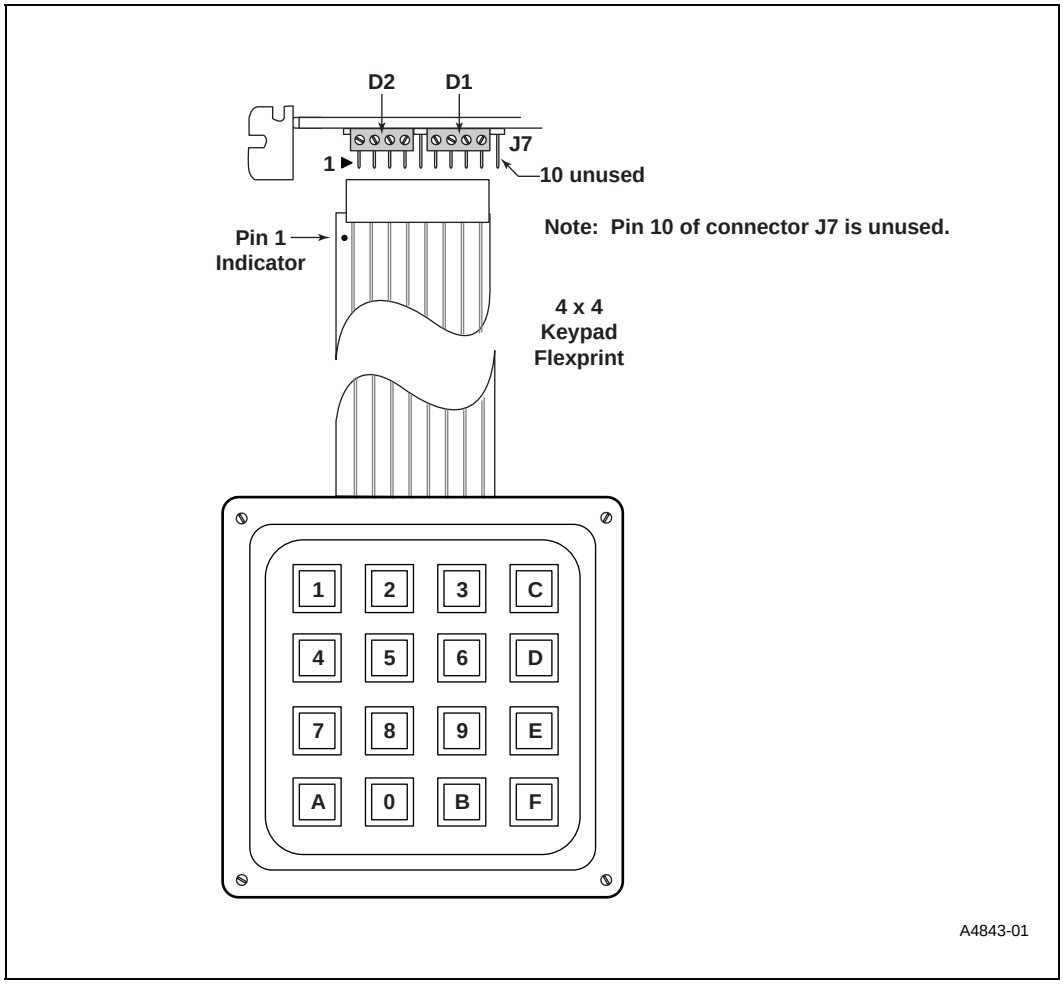
A.2.3 Reset Switch: S2

The reset switch S2 is used to initialize the SA-1100 multimedia board. While the board is being initialized, the seven-segment displays E2 and E14 indicate “00” and the LEDs D1 and D2 both light up. While Angel is starting, the seven-segment display changes dynamically and the LEDs provide error information. For information on the seven segment displays, see Section A.2.2 and for information on the display LEDs see Section A.2.5.

A.2.4 4x4 Keypad Header: J7

The 4x4 Keypad (9 pin), as shown in Figure A-3, connects to the 4x4 Keypad header J7 (10 pin). The flex cable for the keypad has a pin one indicator (a circle on the flex print) that must be aligned with pin one of J7 on the SA-1101 multimedia development board.

Figure A-3. Connecting the 4X4 Keypad



A.2.5 Display LEDS: D1, D2

The display LEDs D1 and D2 provide status information about the power-up self test and error information for Angel and customer applications.

While the board is being initialized, the LEDs D1 and D2 both remain lit and flash while Angel is starting (for information on the reset switch, see Section A.2.3). Table A-1indicates the CPLD pin locations.

Table A-3. Display LEDs D1 and D2 Connections

Display LED	LED Name	CPLD connection
D2- 4	KP_OUT_3	E25 pin 34
D2- 3	KP_OUT_2	E25 pin 33
D2- 2	KP_OUT_1	E25 pin 32
D2- 1	KP_OUT_0	E25 pin 31
D1- 4	LED7	E25 pin 26
D1- 3	LED6	E25 pin 25
D1- 2	LED5	E25 pin 24
D1- 1	LED4	E25 pin 23

A.2.6 SODIMM Header: J13

SODIMM components that are inserted into this Header J13 are 72 Pin SODIMM components with gold contacts.

A.2.7 Debug Connector Number 1: J11

This connector accommodates the serial line input of the 9-pin null modem cable connected to the serial port of the host computer.

A.2.8 PCI Fingers

The PCI fingers, J1, only supplies power to the SA-1100 multimedia development board, which requires +12 volts and +5 volts with ground. Table A-4 describes the voltages and associated pins for J1.

Table A-4. PCI Voltage Connections

Voltage	Name	J1 Pin(s)
+12 V	PWRP12	A2
+5 V	VCC	A5, A8, A10, A16, A59, A61, A62, B5, B6, B19, B59, B61, B62
Ground	GND	A12, A13, A18, A24, A30, A35, A37, A42, A48, A56, B12, B13, B15, B17, B22, B28, B34, B38, B46, B49, B57

A.2.9 Video Input

The video headers provide the connections for receiving two complete video sources, main and PIP, and the video out signal. Each video source is comprised of video functions (composite, luminance, and chrominance), data, and timing. Both video sources are supported by the BT829 video capture processor, while output is supported by the ADV7176 video encoder.

Figure 7-1. Video Headers

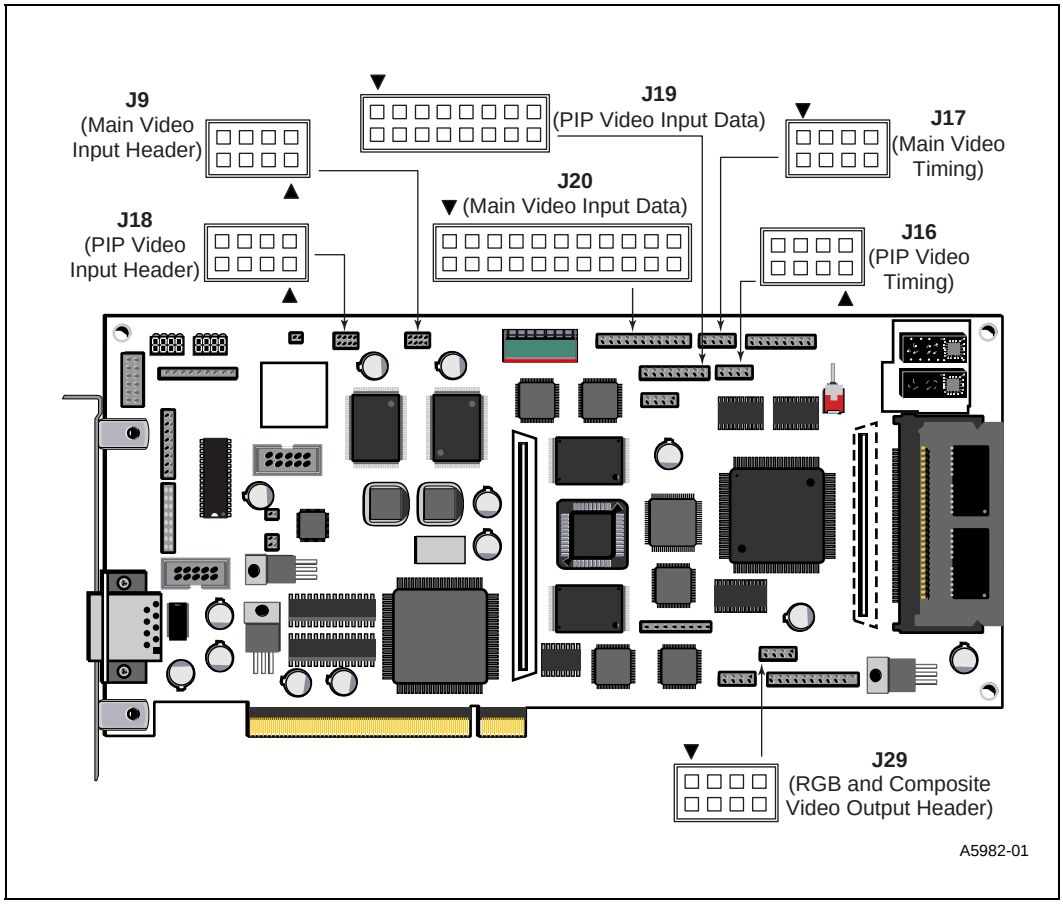


Table A-5. Main: J9

Connector	Signal Name	Video Controls
J9 pin 1	Ground	
J9 pin 2	COMP_IN	Composite
J9 pin 3	Ground	
J9 pin 4	Y_IN	Luminance
J9 pin 5	Ground	
J9 pin 6	spare	
J9 pin 7	Ground	
J9 pin 8	C_IN	Chrominance

Table A-6. Main Video Input Data: J20

Header	Signal Name	CPLD
J20 Pin 1	BT_VD_0	E18 pin 20
J20 Pin 2	BT_VD_4	E18 pin 28
J20 Pin 3	BT_VD_1	E18 pin 21
J20 Pin 4	BT_VD_5	E18 pin 29
J20 Pin 5	BT_VD_2	E18 pin 22
J20 Pin 6	BT_VD_6	E18 pin 30
J20 Pin 7	BT_VD_3	E18 pin 23
J20 Pin 8	BT_VD_7	E18 pin 31
J20 Pin 9	Ground	
J20 Pin 10	Ground	
J20 Pin 11	BT_VD_8	E18 pin 46
J20 Pin 12	BT_VD_12	E18 pin 48
J20 Pin 13	BT_VD_9	E18 pin 45
J20 Pin 14	BT_VD_13	E18 pin 42
J20 Pin 15	IIC_SDA	
J20 Pin 16	IIC_SCL	
J20 Pin 17	BT_VD_10	E18 pin 54
J20 Pin 18	BT_VD_14	E18 pin 40
J20 Pin 19	BT_VD_11	E18 pin 47
J20 Pin 20	BT_VD_15	E18 pin 53
J20 Pin 21	Ground	
J20 Pin 22	VCC	

Table A-7. Main Video Timing: J17

Connector	Signal Name	CPLD
J17 pin 1	BT_CLKX2	E18 pin 41
J17 pin 2	Ground	
J17 pin 3	BT_FIELD	E18 pin 44
J17 pin 4	Ground	
J17 pin 5	#BT_VRESET	E18 pin 43
J17 pin 6	Ground	
J17 pin 7	BT_QCLK	E18 pin 33
J17 pin 8	Ground	

Table A-8. PIP Video Input Data: J19 (Sheet 1 of 2)

Header	Signal Name	CPLD
J19 Pin 1		
J19 Pin 2		
J19 Pin 3	Ground	
J19 Pin 4	Ground	
J19 Pin 5	BT_VD_24	E18 pin 3

Table A-8. PIP Video Input Data: J19 (Sheet 2 of 2)

Header	Signal Name	CPLD
J19 Pin 6	BT_VD_28	E18 pin 7
J19 Pin 7	BT_VD_25	E18 pin 4
J19 Pin 8	BT_VD_29	E18 pin 8
J19 Pin 9	IIC_SDA	
J19 Pin 10	IIC_SCL	
J19 Pin 11	BT_VD_26	E18 pin 5
J19 Pin 12	BT_VD_30	E18 pin 9
J19 Pin 13	BT_VD_27	E18 pin 6
J19 Pin 14	BT_VD_31	E18 pin 17
J19 Pin 15	Ground	
J19 Pin 16	VCC	

Table A-9. PIP: J18

Connector	Signal Name	Video Function
J18 pin 1	Ground	
J18 pin 2	COMP_IN_U	Composite
J18 pin 3	Ground	
J18 pin 4	Y_IN_U	Luminance
J18 pin 5	Ground	
J18 pin 6	Spare	
J18 pin 7	Ground	
J18 pin 8	C_IN_U	Chrominance

Table A-10. PIP Video Timing: J16

Connector	Signal Name	CPLD
J16 pin 1	BT_CLKX2_U	E18 pin 18
J16 pin 2	Ground	
J16 pin 3	BT_FIELD_U	
J16 pin 4	Ground	
J16 pin 5	# BT_VRESET_U	
J16 pin 6	Ground	
J16 pin 7	BT_QCLK_U	
J16 pin 8	Ground	

Table A-11. Video Out: J29

Connector	Signal Name
J29 pin 1	Red
J29 pin 2	Ground
J29 pin 3	Green
J29 pin 4	Ground
J29 pin 5	Blue
J29 pin 6	Ground
J29 pin 7	Composite
J29 pin 8	Ground

A.3 Telephone, Speaker, and Audio Connections

The telephone, codec, microphone, speaker connections, and touch screen analog to digital inputs are controlled by the UCB1200, modem and audio analog front-end device. The audio connections are connected to the Hi-Fi audio processor, which controls tone, base, treble, and volume.

Figure 7-2. Telephone Input

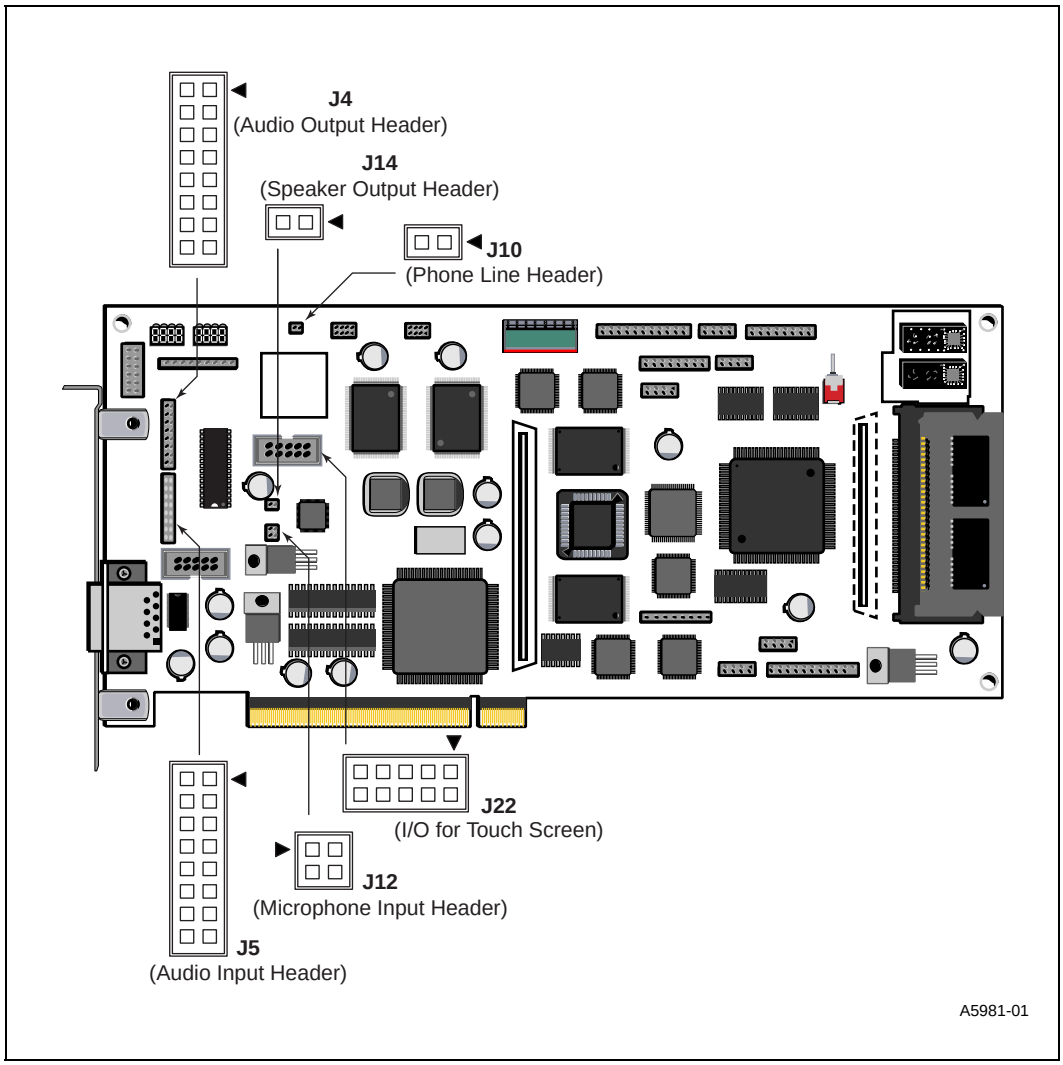


Table A-12. Telephone: J10

Telephone	Signal Name
J10 Pin 1	Tip
J10 Pin 2	Ring

Table A-13. Codec Audio Output: J14

Codec Audio Speaker	Signal Name
J14 Pin 1	SPKRP / CODEC_SPKR
J14 Pin 2	SPKRN

Table A-14. Microphone: J12

Codec Audio Speaker	Signal Name
J12 Pin 1	VCORE (phantom voltage for microphone)
J12 Pin 2	MICP
J12 Pin 3	Ground
J12 Pin 4	MICGND

Table A-15. Touch Screen / Analog to Digital input: J22

Touch Screen / Analog	Signal Name	Description
J22 Pin 1	VCC	
J22 Pin 2	Ground	
J22 Pin 3	AD0	Analog Voltage input
J22 Pin 4	TSPY	Positive Y-plate touch screen
J22 Pin 5	AD1	Analog Voltage input
J22 Pin 6	TSMX	Negative X-plate touch screen
J22 Pin 7	AD2	Analog Voltage input
J22 Pin 8	TSMY	Negative Y-plate touch screen
J22 Pin 9	AD3	Analog Voltage input
J22 Pin 10	TSPX	Positive X-plate touch screen

Table A-16. Audio Input: J5 (Sheet 1 of 2)

TDA9860 Inputs	Signal Name	Signal Inputs
J5 Pin 1	Ground	
J5 Pin 2	AUX_L	Auxiliary Left
J5 Pin 3	Ground	
J5 Pin 4	AUX_R	Auxiliary Right
J5 Pin 5	Ground	
J5 Pin 6	CODEC_SPKRB	Codec Speaker / SCART Left
J5 Pin 7	Ground	
J5 Pin 8	SCART_R	SCART Right
J5 Pin 9	Ground	
J5 Pin 10	MAIN_L	Main Left
J5 Pin 11	Ground	
J5 Pin 12	MAIN_R	Main Right
J5 Pin 13	Ground	

Table A-16. Audio Input: J5 (Sheet 2 of 2)

TDA9860 Inputs	Signal Name	Signal Inputs
J5 Pin 14	VCC	
J5 Pin 15	Ground	
J5 Pin 16	Ground	

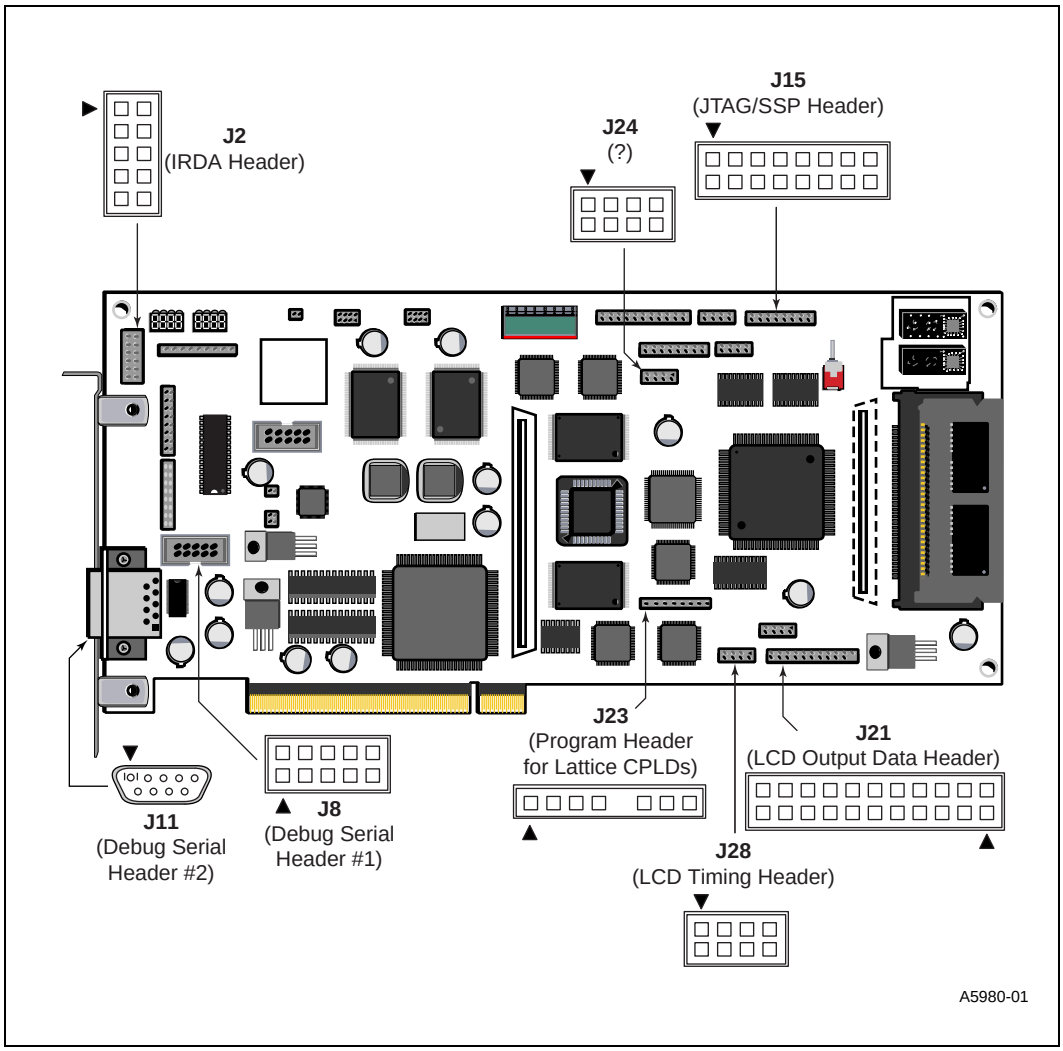
Table A-17. Audio Outputs: J4

TDA9860 Outputs	Signal Name	Signal Outputs
J4 Pin 1	HD_OUT_L	Headphone Left
J4 Pin 2	Ground	
J4 Pin 3	HD_OUT_R	Headphone Right
J4 Pin 4	Ground	
J4 Pin 5	S_OUT_L	SCART Output Left
J4 Pin 6	Ground	
J4 Pin 7	S_OUT_R	SCART Output Right
J4 Pin 8	Ground	
J4 Pin 9	SPK_L	Loudspeaker Left
J4 Pin 10	Ground	
J4 Pin 11	SPK_R	Loudspeaker Right
J4 Pin 12	Ground	
J4 Pin 13	VCC	
J4 Pin 14	Ground	
J4 Pin 15	Ground	
J4 Pin 16	Ground	

A.4 Debug, Program, IrDA, JTAG, USB, LCD Headers

These connections provide access to the test, debug, infrared transmit and receive, CPLD programming, and LCD display headers.

Figure 7-3. IrDA, JTAG, LCD, and Debug Ports and Headers



J11 is a 9-pin, 2-row D-Sub header. This port is used as a serial line input.

Table A-18. Debug Port Number 1: J11 (Sheet 1 of 2)

Connector	Signal Name
J11 pin 1	
J11 pin 2	CON_RX
J11 pin 3	CON_TX
J11 pin 4	
J11 pin 5	Ground
J11 pin 6	

Table A-18. Debug Port Number 1: J11 (Sheet 2 of 2)

Connector	Signal Name
J11 pin 7	
J11 pin 8	
J11 pin 9	

J8 is a 10-pin header (2x5).

Table A-19. Debug Port Number 2: J8

Connector	Signal Name
J8 pin 1	
J8 pin 2	DEBUG_RX
J8 pin 3	DEBUG_TX
J8 pin 4	
J8 pin 5	Ground
J8 pin 6	
J8 pin 7	
J8 pin 8	
J8 pin 9	
J8 pin 10	VCC

J23 is an 8-pin header (1x8). This header is used for the programming the CPLD components on the board.

Note: Pin 5 is missing to ensure the correct mounting of the keyed programming cable. Once parts have been factory programmed, this cable does not need to be installed unless changes to the CPLD components are required.

Table A-20. Programming Header: J23

Connector	Signal Name
J23 pin 1	VCC
J23 pin 2	SDO_FINI
J23 pin 3	SDI / INO
J23 pin 4	# ISP_EN
J23 pin 5	—-
J23 pin 6	MODE / IN2
J23 pin 7	Ground
J23 pin 8	ISP_SCLK

J2 is a 10-pin header (2x5).

Table A-21. IrDA Header: J2 (Sheet 1 of 2)

Connector	Signal Name	Description	CLPD
J2 pin 1	XSPARE_4	Spare	E27 pin 44, E28 pin 38
J2 pin 2	SA_IrDA_TXD		
J2 pin 3	X_IrDA_RXD		E28 pin 2
J2 pin 4	IRSPD_S7LED_D0		E21 pin 23
J2 pin 5			
J2 pin 6			

Table A-21. IrDA Header: J2 (Sheet 2 of 2)

Connector	Signal Name	Description	CLPD
J2 pin 7	S7LED_D3		E21 pin 26
J2 pin 8	VCC		
J2 pin 9	S7LED_D2		E21 pin 25
J2 pin 10	Ground		

Table A-22. JTAG / SSP Header: J15

Position	Signal Name	CPLD
J15 Pin 1	JTAG_TCK	
J15 Pin 2	XSPARE_7	E28 pin 34, E27 pin 38
J15 Pin 3	JTAG_TDI	
J15 Pin 4	XSPARE_6	E28 pin 35, E27 pin 37
J15 Pin 5	JTAG_TDO	
J15 Pin 6	SA_GPIO_10	E28 pin 15
J15 Pin 7	JTAG_TMS	
J15 Pin 8	SA_GPIO_11	E28 pin 13
J15 Pin 9	# JTAG_TRST	
J15 Pin 10	SA_GPIO_12	E28 pin 16
J15 Pin 11	Ground	
J15 Pin 12	SA_GPIO_13	E28 pin26
J15 Pin 13	Ground	
J15 Pin 14	XSPARE_5	E28 pin 37, E27 pin 43
J15 Pin 15	Ground	
J15 Pin 16	VCC	

J24 is an 8-pin header (2x4).

Table A-23. USB Header: J24

Connector	Signal Name
J24 pin 1	USB_MINUS
J24 pin 2	Ground
J24 pin 3	USB_PLUS
J24 pin 4	Ground
J24 pin 5	Ground
J24 pin 6	Ground
J24 pin 7	
J24 pin 8	Ground

Table A-24. LCD Timing: J28 (Sheet 1 of 2)

Connector	Signal Name	CLPD
J28 pin 1	ISP_SPARE_0	E31 pin 33
J28 pin 2	# ISP_VSYNC	E31 pin 25
J28 pin 3	ISP_SPARE_1	E31 pin 34

Table A-24. LCD Timing: J28 (Sheet 2 of 2)

Connector	Signal Name	CLPD
J28 pin 4	# ISP_VSYNC	E31 pin 24
J28 pin 5	ISP_SPARE_2	E31 pin 35
J28 pin 6	SA_LCD_PCLK	E31 pin 11
J28 pin 7	Ground	
J28 pin 8	# ISP_CBLANK	E31 pin 26

Table A-25. LCD Output Data: J21

Header	Signal Name
J21 Pin 1	SA_LCD_D_0
J21 Pin 2	Ground
J21 Pin 3	SA_LCD_D_1
J21 Pin 4	SA_LCD_D_9
J21 Pin 5	SA_LCD_D_2
J21 Pin 6	SA_LCD_D_10
J21 Pin 7	SA_LCD_D_3
J21 Pin 8	SA_LCD_D_11
J21 Pin 9	SA_LCD_D_4
J21 Pin 10	SA_LCD_D_12
J21 Pin 11	SA_LCD_D_5
J21 Pin 12	SA_LCD_D_13
J21 Pin 13	SA_LCD_D_6
J21 Pin 14	SA_LCD_D_14
J21 Pin 15	SA_LCD_D_7
J21 Pin 16	SA_LCD_D_15
J21 Pin 17	SA_LCD_D_8
J21 Pin 18	VCC
J21 Pin 19	IIC_SDA
J21 Pin 20	IIC_SCL
J21 Pin 21	PP12 (+ 12 Volts)
J21 Pin 22	Ground

SA-1101 Board Configuration Guide B

This appendix describes the header pins on the SA-1101 development board.

Figure B-1illustrates the location of header and connectors of the SA-1101 multimedia development board, which are described in Table B-1.

Figure B-1. SA-1101 Development Board

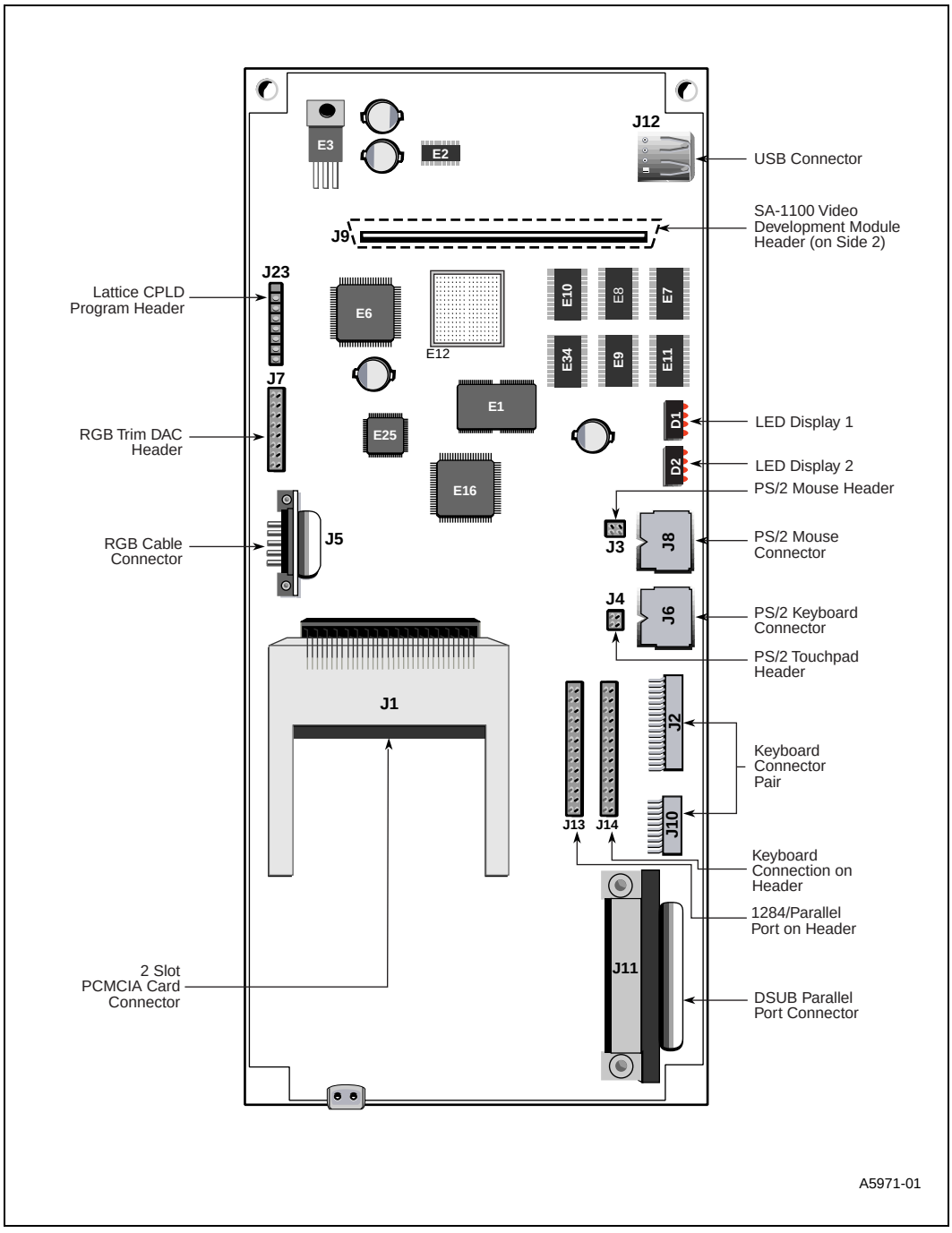


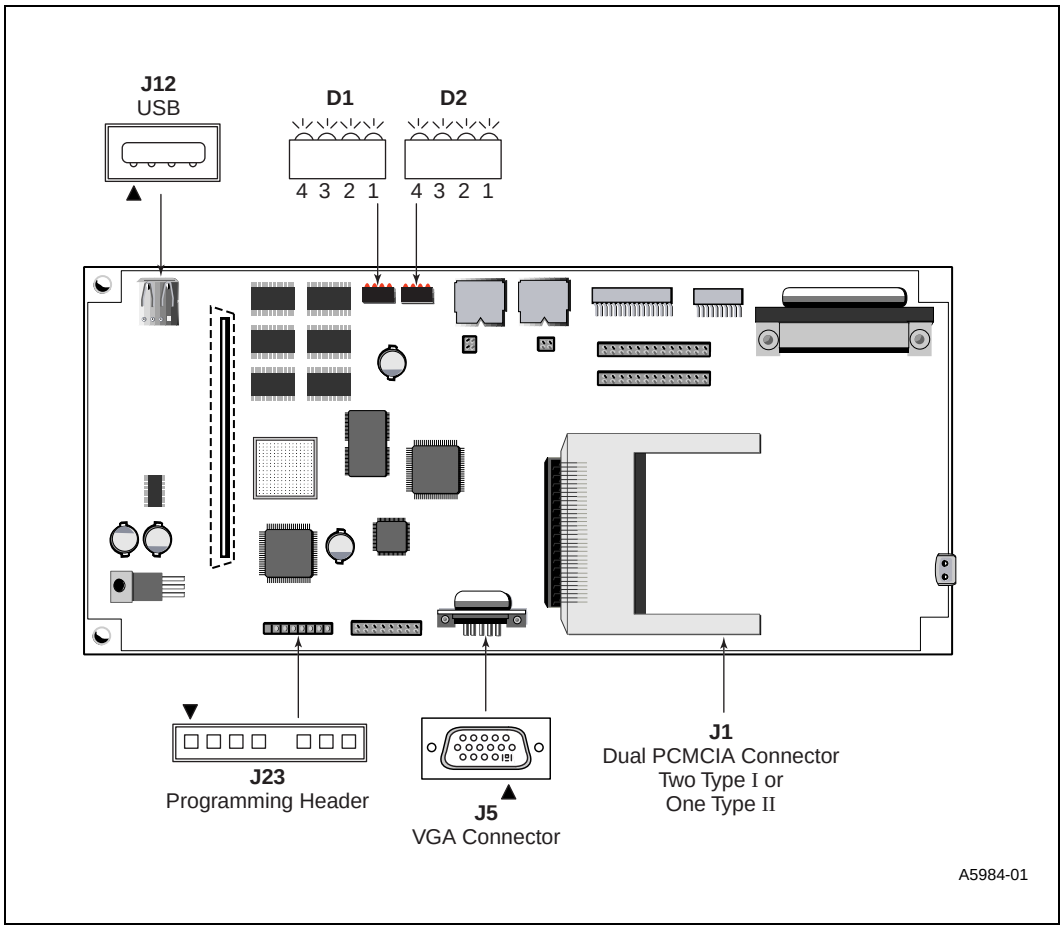
Table B-1 provides a brief description of all headers and connectors for the SA-1101 development board.

Table B-1. SA-1101 Headers and Connectors

Ref	Description
J1	PCMCIA card connector (2 slot)
J2	Keyboard connector pair (see J10)
J3	PS/2 mouse header
J4	PS/2 touch pad header
J5	RGB cable connector
J6	PS/2 keyboard connector
J7	RGB/Trim DAC header
J8	PS/2 mouse connector
J9	128 pin connector to SA-1101 board (side 2)
J10	Keyboard connector pair (see J2)
J11	DSUB parallel port connector
J12	USB connector
J14	Keyboard connection on header
J13	1284/Parellel port header
J23	Lattice CPLD program header

The configuration diagram, B-1, illustrates the location of the USB Port, indicator LEDs, Programming header, VGA Connector, and the Dual PCMCIA connector.

Figure 7-4. USB Port, LEDs, Programming Header, VGA and Dual PCMCIA Connector



B.1 Display LEDs: D1, D2

The display LEDs D1 and D2, as described in Table B-2, are connected directly to the CPLDs E16 and E25.

Table B-2. LEDs and CPLD Connection

Display LED	LED Name	CPLD connection
D1- 4	S0_RESET_LED7	E16 pin 99, E25 pin 4
D1- 3	S1_RESET_LED6	E16 pin 1, E25 pin 3
D1- 2	LED5	E16 pin 2, E25 pin 2
D1- 1	LED4	E16 pin 3, E25 pin 1
D2- 4	LED3	E16 pin 5, E25 pin 44
D2- 3	LED2	E16 pin 6, E25 pin 43
D2- 2	LED1	E16 pin 7, E25 pin 42
D2- 1	LED0	E16 pin 8, E25 pin 41

B.2 USB Connector

J12 is a 4-pin Universal Serial Bus connector (1x4), USB Type A.

Table B-3. USB Connector: J12

Connector	Signal Name	CPLD Connection
J12 pin 1	VCC	—
J12 pin 2	USB_MINUS	E25 pin 37
J12 pin 3	USB_PLUS	E25 pin 36
J12 pin 4	Ground	—

B.3 CPLD Programming Header

J23 is an 8-pin header (1x8). This header is used for the programming of the CPLD components on the board.

Note: Pin 5 is missing to insure the correct mounting of the keyed programming cable. Once parts have been factory programmed, this cable is not installed unless changes to the CPLD components have been made.

Table B-4. CPLD Programming Header: J23

Connector	Signal Name	CPLD Connection
J23 pin 1	VCC	
J23 pin 2	SDO_FINI	E16 pin 87
J23 pin 3	SDI / INO	E25 pin 8
J23 pin 4	# ISP_EN	—
J23 pin 5	—	—
J23 pin 6	MODE / IN2	E6 pin 37, E16 pin 37, E25 pin 30
J23 pin 7	Ground	—
J23 pin 8	ISP_SCLK	E6 pin 59, E16 pin 59, E25 pin 27

B.3.1 VGA Connector

J5 is an 15-pin D-Sub connector (3x5). This connector is for cable input for a VGA monitor.

Table B-5. VGA Connector: J5

Header	Signal Name	CPLD Connection
J5 Pin 1	R	—
J5 Pin 2	G	—
J5 Pin 3	B	—
J5 Pin 4	—	—
J5 Pin 5	Ground	Ground
J5 Pin 6	Ground	Ground
J5 Pin 7	Ground	Ground
J5 Pin 8	Ground	Ground
J5 Pin 9	—	—
J5 Pin 10	Ground	Ground
J5 Pin 11	—	—
J5 Pin 12	—	—
J5 Pin 13	HSYNCB	—
J5 Pin 14	VSYNCB	—
J5 Pin 15	—	—

B.4 PCMCIA

This PCMCIA connector is dual input supporting two Type II or one Type III PCMCIA cards.

Table B-6. PCMCIA Dual Position: J1 (Sheet 1 of 3)

PCMCIA SLOT 1	Signal Name	CPLD
J1 Pin A1	Ground	Ground
J1 Pin A2	S0_D_3	—
J1 Pin A3	S0_D_ 4	—
J1 Pin A4	S0_D_ 5	—
J1 Pin A5	S0_D_6	—
J1 Pin A6	S0_D_7	—
J1 Pin A7	# S0_CE1	E6 pin 30
J1 Pin A8	S0 _A_10	—
J1 Pin A9	# S0_OE	E6 pin 43
J1 Pin A10	S0_A_11	—
J1 Pin A11	S0_A_9	—
J1 Pin A12	S0_A_8	—
J1 Pin A13	S0_A_13	—
J1 Pin A14	S0_A_14	—
J1 Pin A15	# S0_WE	E6 pin 45
J1 Pin A16	# S0_RDY_IREQ	E6 pin 29
J1 Pin A17	S0_VCC	—
J1 Pin A18	S0_VPP	—
J1 Pin A19	S0_A_16	—
J1 Pin A20	S0_A_15	—

Table B-6. PCMCIA Dual Position: J1 (Sheet 2 of 3)

PCMCIA SLOT 1	Signal Name	CPLD
J1 Pin A21	S0_A_12	—
J1 Pin A22	S0_A_7	—
J1 Pin A23	S0_A_6	—
J1 Pin A24	S0_A_5	—
J1 Pin A25	S0_A_4	—
J1 Pin A26	S0_A_3	—
J1 Pin A27	S0_A_2	—
J1 Pin A28	S0_A_1	—
J1 Pin A29	S0_A_0	—
J1 Pin A30	S0_D_0	—
J1 Pin A31	S0_D_1	—
J1 Pin A32	S0_D_2	—
J1 Pin A33	# S0_IOIS16	E6 pin 34
J1 Pin A34	Ground	Ground
J1 Pin A35	Ground	Ground
J1 Pin A36	# S0_CD1	E6 pin 28
J1 Pin A37	S0_D_11	—
J1 Pin A38	S0_D_12	—
J1 Pin A39	S0_D_13	—
J1 Pin A40	S0_D_14	—
J1 Pin A41	S0_D_15	—
J1 Pin A42	# S0_CE2	E6 pin 32
J1 Pin A43	# S0_VS1	E6 pin 41
J1 Pin A44	# S0_IORD	E6 pin 46
J1 Pin A45	# S0_IOWR	E6 pin 47
J1 Pin A46	S0_A_17	—
J1 Pin A47	S0_A_18	—
J1 Pin A48	S0_A_19	—
J1 Pin A49	S0_A_20	—
J1 Pin A50	S0_A_21	—
J1 Pin A51	S0_VCC	—
J1 Pin A52	S0_VPP	—
J1 Pin A53	S0_A_22	—
J1 Pin A54	S0_A_23	—
J1 Pin A55	S0_A_24	—
J1 Pin A56	S0_A_25	—
J1 Pin A57	# S0_VS2	E6 pin 42
J1 Pin A58	S0_RESET_LED7	E6 pin 48, E16 pin 99, E25 pin 4
J1 Pin A59	# S0_WAIT	E6 pin 33
J1 Pin A60	S0_VCC	—
J1 Pin A61	S0_REG	—
J1 Pin A62	# S0_BVD2_SPKR	E6 pin 40
J1 Pin A63	# S0_BVD1_STSCHG	E6 pin 35
J1 Pin A64	S0_D_8	—
J1 Pin A65	S0_D_9	—
J1 Pin A66	S0_D_10	—

Table B-6. PCMCIA Dual Position: J1 (Sheet 3 of 3)

PCMCIA SLOT 1	Signal Name	CPLD
J1 Pin A67	# S0_CD2	
J1 Pin A68	Ground	Ground

Table B-7. PCMCIA Connector: J1 Slot 2 (Sheet 1 of 2)

PCMCIA SLOT 2	Signal Name	CPLD
J1 Pin B1	Ground	Ground
J1 Pin B2	S1_D_3	—
J1 Pin B3	S1_D_ 4	—
J1 Pin B4	S1_D_ 5	—
J1 Pin B5	S1_D_6	—
J1 Pin B6	S1_D_7	—
J1 Pin B7	# S1_CE1	E6 pin 57
J1 Pin B8	S1_A_10	—
J1 Pin B9	# S1_OE	E6 pin 74
J1 Pin B10	S1_A_11	—
J1 Pin B11	S1_A_9	—
J1 Pin B12	S1_A_8	—
J1 Pin B13	S1_A_13	—
J1 Pin B14	S1_A_14	—
J1 Pin B15	# S1_WE	E6 pin 76
J1 Pin B16	# S1_RDY_IREQ	E6 pin 56
J1 Pin B17	S1_VCC	—
J1 Pin B18	S1_VPP	—
J1 Pin B19	S1_A_16	—
J1 Pin B20	S1_A_15	—
J1 Pin B21	S1_A_12	—
J1 Pin B22	S1_A_7	—
J1 Pin B23	S1_A_6	—
J1 Pin B24	S1_A_5	—
J1 Pin B25	S1_A_4	—
J1 Pin B26	S1_A_3	—
J1 Pin B27	S1_A_2	—
J1 Pin B28	S1_A_1	—
J1 Pin B29	S1_A_0	—
J1 Pin B30	S1_D_0	—
J1 Pin B31	S1_D_1	—
J1 Pin B32	S1_D_2	—
J1 Pin B33	# S1_IOIS16	E6 pin 68
J1 Pin B34	Ground	Ground
J1 Pin B35	Ground	Ground
J1 Pin B36	# S1_CD1	E6 pin 55
J1 Pin B37	S1_D_11	—
J1 Pin B38	S1_D_12	—
J1 Pin B39	S1_D_13	—
J1 Pin B40	S1_D_14	—

Table B-7. PCMCIA Connector: J1 Slot 2 (Sheet 2 of 2)

PCMCIA SLOT 2	Signal Name	CPLD
J1 Pin B41	S1_D_15	—
J1 Pin B42	# S1_CE2	E6 pin 58
J1 Pin B43	# S1_VS1	E6 pin 72
J1 Pin B44	# S1_IORD	E6 pin 77
J1 Pin B45	# S1_IOWR	E6 pin 78
J1 Pin B46	S1_A_17	—
J1 Pin B47	S1_A_18	—
J1 Pin B48	S1_A_19	—
J1 Pin B49	S1_A_20	—
J1 Pin B50	S1_A_21	—
J1 Pin B51	S1_VCC	—
J1 Pin B52	S1_VPP	—
J1 Pin B53	S1_A_22	—
J1 Pin B54	S1_A_23	—
J1 Pin B55	S1_A_24	—
J1 Pin B56	S1_A_25	—
J1 Pin B57	# S1_VS2	E6 pin 73
J1 Pin B58	S1_RESET_LED7	E6 pin 79, E16 pin 1, E25 pin 3
J1 Pin B59	# S1_WAIT	E6 pin 67
J1 Pin B60	S1_VCC	—
J1 Pin B61	S1_REG	—
J1 Pin B62	# S1_BVD2_SPKR	E6 pin 70
J1 Pin B63	# S1_BVD1_STSCHG	E6 pin 69
J1 Pin B64	S1_D_8	—
J1 Pin B65	S1_D_9	—
J1 Pin B66	S1_D_10	—
J1 Pin B67	# S1_CD2	—
J1 Pin B68	Ground	Ground

B.4.1 KEYBOARD Connectors for Fujitsu - J2, J10

The Connector pair J2 and J10 service the portable Fujitsu Keyboard model N860-1406-T001.

Note: The flex print cables for the keyboard are center justified with connectors J2 and J10, resulting in unused pins on the left side of J2 and on the right side of J10. Please seeFigure B-2 for proper insertion of the keyboards two flex print cables.

Figure B-2. Keyboard Connectors

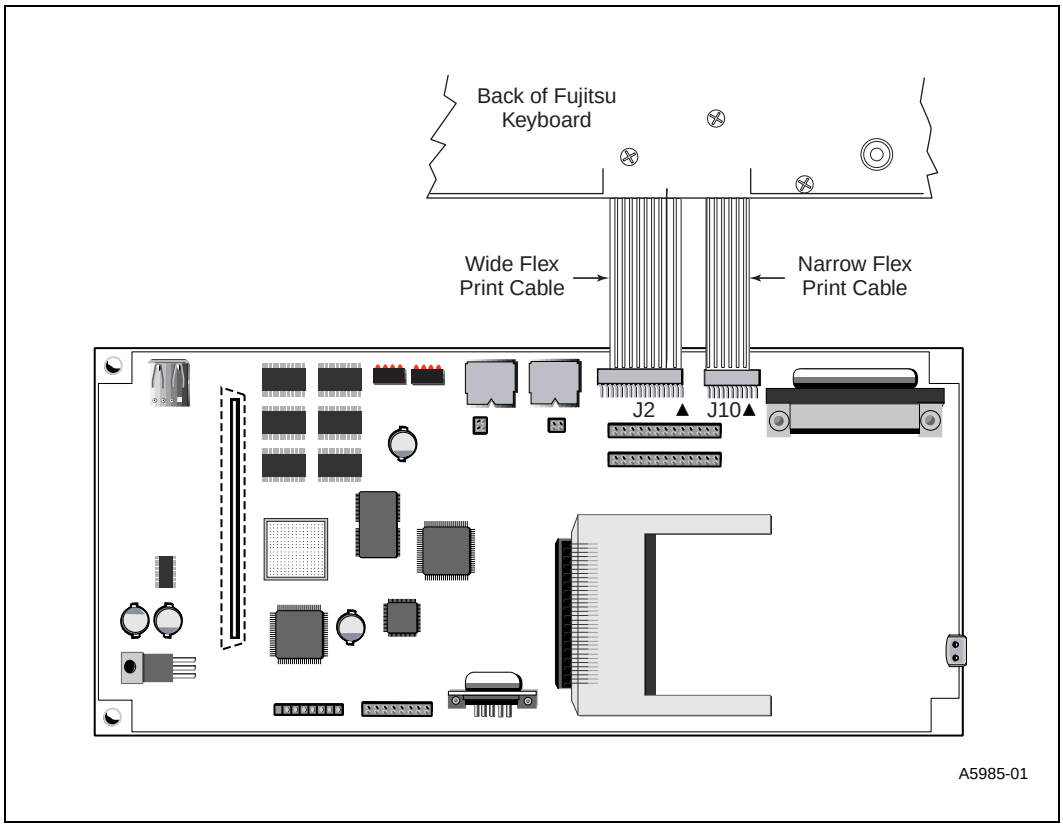


Table B-8. KEYBOARD INPUT, FUJITSU: J2, J10 (Sheet 1 of 2)

Connector	Signal Name	CPLD connection
J2 pin 1	# SKPPSTROBE_KPY0	E16 PIN 1, E25 PIN 29
J2 pin 2	# SKPPSELECTIN _KPY1	E16 PIN 68
J2 pin 3	# SKPPINIT_KPY2	E16 PIN 67
J2 pin 4	# SKPPAUTOFD _KPY3	E16 PIN 57
J2 pin 5	# SKPPACK_KPY4	E16 PIN 52
J2 pin 6	SKPPSELECT_KYP5	E16 PIN 56
J2 pin 7	SKPPPERROR_KYP6	E16 PIN 55
J2 pin 8	# SKPPFAULT_KYP7	E16 PIN 58
J2 pin 9	SKPPBUFFEN_KYP8	E16 PIN 69
J2 pin 10	SKPPBUSY_KYP9	E16 PIN 53
J2 pin 11	GPB_0_KYP10	E16 PIN 91

Table B-8. KEYBOARD INPUT, FUJITSU: J2, J10 (Sheet 2 of 2)

Connector	Signal Name	CPLD connection
J2 pin 12	GPB_1_KYP11	E16 PIN 92
J2 pin 13	GPB_2_KYP12	E16 PIN 93
J2 pin 14	GPB_3_KYP13	E16 PIN 95
J2 pin 15	GPB_4_KYP14	E16 PIN 96
J2 pin16	GPB_5_KYP15	E16 PIN 97
J2 pin 17	Unused	—

Table B-9. Second Fujitsu Keyboard Connector: J10

Connector	Signal Name	CPLD connection
J10 pin 1	UNUSED	—-
J10 pin 2	UNUSED	—-
J10 pin 3	SKPPDATA_KPX0	E16 PIN 35
J10 pin 4	SKPPDATA_KPX1	E16 PIN 34
J10 pin 5	SKPPDATA_KPX2	E16 PIN 33
J10 pin 6	SKPPDATA_KPX3	E16 PIN 32
J10 pin 7	SKPPDATA_KPX4	E16 PIN 30
J10 pin 8	SKPPDATA_KPX5	E16 PIN 29
J10 pin 9	SKPPDATA_KPX6	E16 PIN 28
J10 pin 10	SKPPDATA_KPX7	E16 PIN 27

B.4.2 Keyboard, Printer and Mouse Headers

This header is an alternate connection or monitoring point for the Fujitsu Keyboard.

Figure 7-5. Keyboard, Printer Port, Mouse Connector and Headers

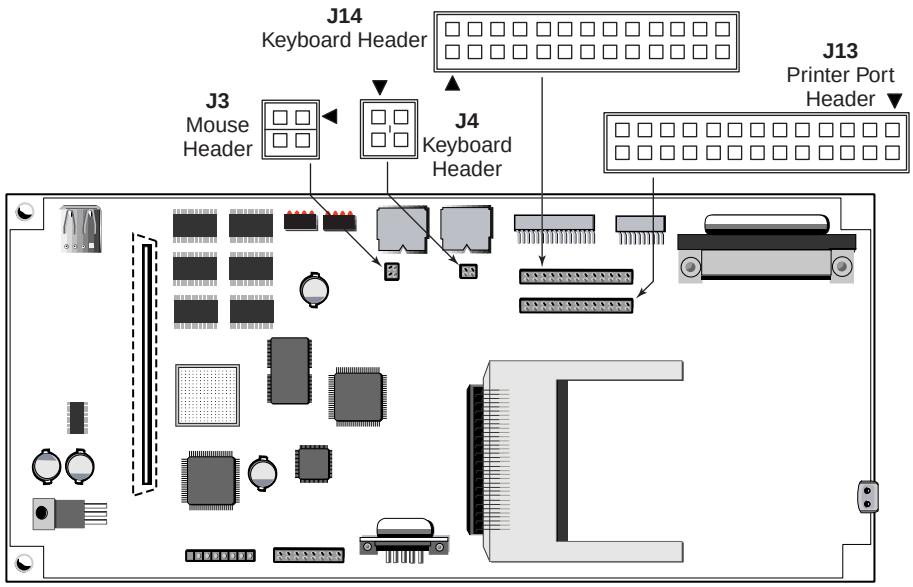


Table B-10. KEYBOARD HEADER: J14

Header	Signal Name	CPLD Connection
J14 Pin 1	GPB_5_KYB_15	E16 pin 97
J14 Pin 2	GROUND	—
J14 Pin 3	GPB_4_KYP14	E16 pin 96
J14 Pin 4	VCC	—
J14 Pin 5	GPB_3_KYP13	E16 pin 95
J14 Pin 6	SKPPDATA_KPY0	E16 pin 35
J14 Pin 7	GPB_2_KYP12	E16 pin93
J14 Pin 8	SKPPDATA_KPX1	E16 pin 34
J14 Pin 9	GPB_1_KYP11	E16 pin 92
J14 Pin 10	SKPPDATA_KPX2	E16 pin 33
J14 Pin 11	GPB_0_KYP10	E16 pin 91
J14 Pin 12	SKPPDATA_KPX3	E16 pin 32
J14 Pin 13	SKPPBUSY_KYP9	E16 pin 53
J14 Pin 14	SKPPDATA_KPX4	E16 pin 30
J14 Pin 15	SKPPBUFFEN_KYP8	E16 pin 69
J14 Pin 16	SKPPDATA_KPX5	E16 pin 29
J14 Pin 17	SKPPFAULT_KPY7	E16 pin 58
J14 Pin 18	SKPPDATA_KPX6	E16 pin 28
J14 Pin 19	SKPPPERROR_KPY6	E16 pin 55
J14 Pin 20	SKPPDATA_KPX7	E16 pin 27
J14 Pin 21	SKPPSELECT_KPY5	E16 pin 56
J14 Pin 22	SKPPSTROBE_KPY0	E16 pin 51
J14 Pin 23	SKPPACK_KPY4	E16 pin 52
J14 Pin 24	SKPPSELECTIN_KPY1	E16 pin 68
J14 Pin 25	SKPPAUTOFD_KPY3	E16 pin 57
J14 Pin 26	SKPPINIT_KPY2	E16 pin 67

J4 is an alternate PS/2 Keyboard connection. J4 can also be used to monitor PS/2 Connector J6.

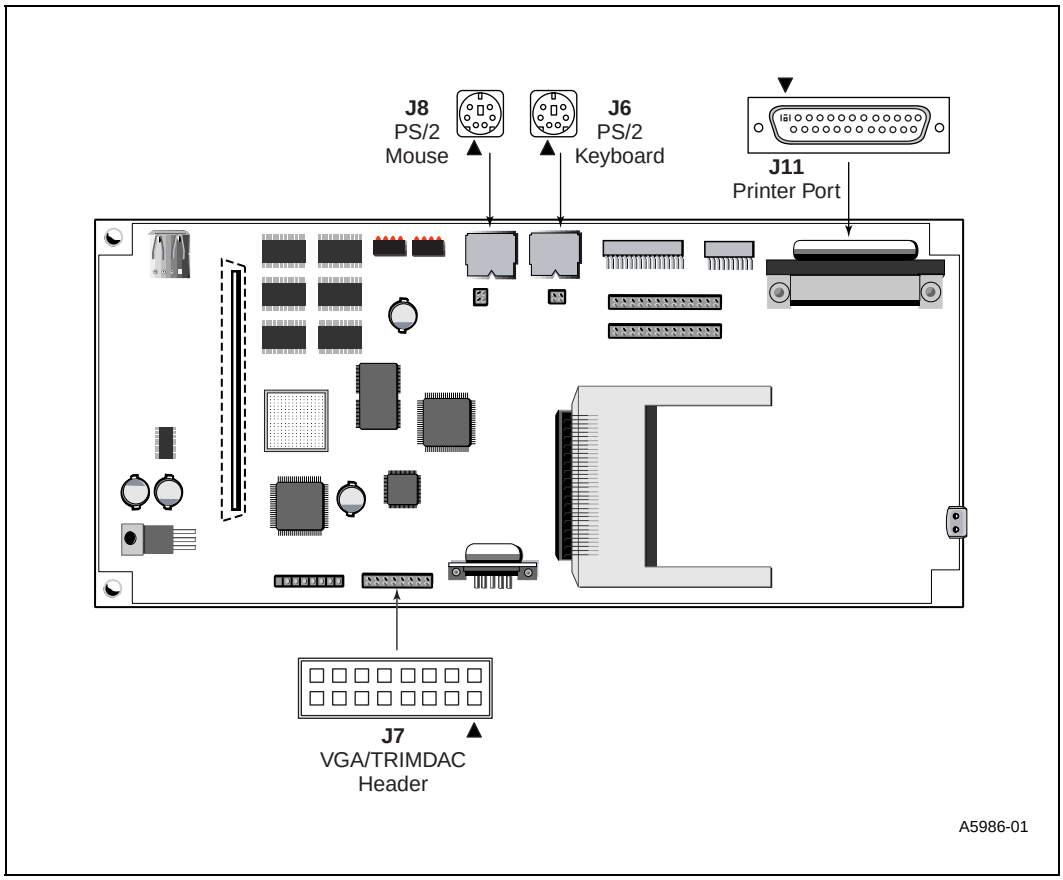
Table B-11. KEYBOARD Connector: J4

Connector	Signal Name	CPLD Connection
J4 pin 1	TPCLK	E25 pin 26
J4 pin 2	TPCLK	E25 Pin 31
J4 pin 3	VCC	VCC
J4 pin 4	Ground	Ground

B.4.3 PS/2 Mouse, Keyboard, Printer Port Connector, and VGA/TRIMDAC Header

The standard mouse, keyboard, and printer port connectors are described in this section, along with the VGA/TrimDAC header, which is an alternate to J5 a 15-pin D-Sub connector.

Figure 7-6. PS/2 Mouse, Keyboard, Printer Port, and VGA/TrimDAC Header



Header J7 is a combination of two connectors, an alternate VGA connection and TRIMDAC connections.

Table B-12. VGA/TRIMDAC Header: J7 (Sheet 1 of 2)

Connector	Signal Name	CPLD Connection
J7 pin 1	R (RED)	—
J7 pin 2	Ground	—
J7 pin 3	G (GREEN)	—
J7 pin 4	Ground	—
J7 pin 5	B (BLUE)	—
J7 pin 6	Ground	—
J7 pin 7	HSYNC	E25 pin 35
J7 pin 8	Ground	—
J7 pin 9	VSYNC	E25 pin 34
J7 pin 10	Ground	—

Table B-12. VGA/TRIMDAC Header: J7 (Sheet 2 of 2)

Connector	Signal Name	CPLD Connection
J7 pin 11	TRIMDAC0	E25 pin 24
J7 pin 12	Ground	—
J7 pin 13	TRIMDAC1	E25 pin 25
J7 pin 14	Ground	—
J7 pin 15	—	—
J7 pin 16	VCC	—

J13 is an alternate printer port attachment for J11 or can be used to monitor J11.

Equation B-1. PRINTER PORT HEADER: J13

Header	Signal Name	CPLD Connection
J13 Pin 1	# SKPPSTROBE_KPY0	E16 pin 51
J13 Pin 2	SKPPDATA_KPX0	E16 pin 35
J13 Pin 3	SKPPDATA_KPX1	E16 pin 34
J13 Pin 4	SKPPDATA_KPX2	E16 pin 33
J13 Pin 5	SKPPDATA_KPX3	E16 pin 32
J13 Pin 6	SKPPDATA_KPX4	E16 pin 30
J13 Pin 7	SKPPDATA_KPX5	E16 pin 29
J13 Pin 8	SKPPDATA_KPX6	E16 pin 28
J13 Pin 9	SKPPDATA_KPX7	E16 pin 27
J13 Pin 10	# SKPPACK_KPY4	E16 pin 52
J13 Pin 11	SKPPBUSY_KYP9	E16 pin 53
J13 Pin 12	SKPPPERROR_KPY6	E16 pin 55
J13 Pin 13	SKPPSELECT_KPY5	E16 pin 56
J13 Pin 14	SKPPAUTOFD_KPY3	E16 pin 57
J13 Pin 15	# SKPPFAULT_KPY7	E16 pin 58
J13 Pin 16	# SKPPINIT_KYP2	E16 pin 67
J13 Pin 17	# SKPPSELECTIN_KPY1	E16 pin 68
J13 Pin 18	Ground	Ground
J13 Pin 19	Ground	Ground
J13 Pin 20	Ground	Ground
J13 Pin 21	Ground	Ground
J13 Pin 22	Ground	Ground
J13 Pin 23	Ground	Ground
J13 Pin 24	Ground	Ground
J13 Pin 25	Ground	Ground
J13 Pin 26	VCC	VCC

J6 is a PS/2 Keyboard connection.

Equation B-2. PS/2 KEYBOARD Connector: J6 (Sheet 1 of 2)

Connector	Signal Name	CPLD Connection
J6 pin 1	TPDATA	E25 pin 31
J6 pin 2	N/C	—
J6 pin 3	Ground	—
J6 pin 4	VCC	—

Equation B-2. PS/2 KEYBOARD Connector: J6 (Sheet 2 of 2)

Connector	Signal Name	CPLD Connection
J6 pin 5	TPCLK	E25 Pin 26
J6 pin 6	N/C	—

J8 is a standard PS/2 Mouse input connector

Table B-13. PS/2 KEYBOARD Connector: J8

Connector	Signal Name	CPLD Connection
J8 pin 1	MSDATA	E25 pin 33
J8 pin 2	N/C	—
J8 pin 3	Ground	—
J8 pin 4	VCC	—
J8 pin 5	MSCLK	E25 Pin 32
J8 pin 6	N/C	—

J3 is a header and can be used as an alternate to the PS/2 Mouse connector or as a means to monitor the PS/2 connector.

Table B-14. KEYBOARD Header: J3

Connector	Signal Name	CPLD Connection
J3 pin 1	MSDATA	E25 pin 33
J3 pin 2	MSCLK	E25 Pin 32
J3 pin 3	VCC	VCC
J3 pin 4	Ground	Ground

J11 is a 25 pin (3x5) DSub connector for a printer cable.

Table B-15. PRINTER PORT Connector: J11 (Sheet 1 of 2)

Connector	Signal Name	CPLD Connection
J11 Pin 1	# PPSTROBE	E16 pin 40
J11 Pin 2	PPD0	E16 pin 26
J11 Pin 3	PPD1	E16 pin 24
J11 Pin 4	PPD2	E16 pin 23
J11 Pin 5	PPD3	E16 pin 22
J11 Pin 6	PPD4	E16 pin 20
J11 Pin 7	PPD5	E16 pin 19
J11 Pin 8	PPD6	E16 pin 18
J11 Pin 9	PPD7	E16 pin 17
J11 Pin 10	# PPACK	E16 pin 46
J11 Pin 11	PPBUSY	E16 pin 42
J11 Pin 12	PPPERROR	E16 pin 43
J11 Pin 13	PPSELECT	E16 pin 45
J11 Pin 14	# PPAUTOFD	E16 pin 46
J11 Pin 15	# PPFAULT	E16 pin 47
J11 Pin 16	# PPINIT	E16 pin 48
J11 Pin 17	# PPSELECTIN	E16 pin 49
J11 Pin 18	Ground	Ground
J11 Pin 19	Ground	Ground

Table B-15. PRINTER PORT Connector: J11 (Sheet 2 of 2)

Connector	Signal Name	CPLD Connection
J11 Pin 20	Ground	Ground
J11 Pin 21	Ground	Ground
J11 Pin 22	Ground	Ground
J11 Pin 23	Ground	Ground
J11 Pin 24	Ground	Ground
J11 Pin 25	Ground	Ground

CPLD Source Code Process C

This appendix describes the required resources and the process to program the CPLDs on the SA-1100 multimedia development and the SA-1101 development boards.

C.1 Resources Required to Program CPLDs

The following components are required to program the CPLDs:

- 486 or Pentium-compatible PC
- Mouse and mouse driver
- Windows NT or Windows 95
- 16 megabytes of RAM
- SVGA resolution display (800 x 600)
- 35 megabytes of free hard disk space
- Lattice's ISP Daisy Chain Download Software
- The ispDOWNLOAD Cable

The Lattice's ISP* Daisy Chain Download Software and the ispDOWNLOAD cable may be ordered from the Lattice Semiconductor Corporate web site at http://www.latticesemi.com. For information on updating the CPLDs, see Section 7.7.

C.2 Developing and Modifying CPLD Designs

The CPLD sources are created and edited on standard editors and then input to the Lattice Semiconductor PC based CPLD tools. The CPLD sources are processed by the Lattice tools to become .JED files that can be downloaded to the CPLDs.

As an example, if an application required a different address decode for the application flash bank than that provided in the original SA-1100 multimedia board Xbus CPLD, a simple change to the CPLD source file followed by re-compiling the Xbus.JED file and downloading it to the SA-1100 multimedia board would affect that change.

To obtain the latest CPLD sources, see Intel's website for developers at:

http://developer.intel.com

Support, Products, and Documentation

If you need technical support, a *Product Catalog*, or help deciding which documentation best meets your needs, visit the Intel World Wide Web Internet site:

http://www.intel.com

Copies of documents that have an ordering number and are referenced in this document, or other Intel literature may be obtained by calling **1-800-332-2717** or by visiting Intel's website for developers at:

http://developer.intel.com

You can also contact the Intel Massachusetts Information Line or the Intel Massachusetts Customer Technology Center. Please use the following information lines for support:

For documentation and general information:	
Intel Massachusetts Information Line	
United States:	1–800–332–2717
Outside United States:	1–303-675-2148
Electronic mail address:	techdoc@intel.com

For technical support:	
Intel Massachusetts Customer Technology Center	
Phone (U.S. and international):	1–978–568–7474
Fax:	1–978–568–6698
Electronic mail address:	techsup@intel.com

StrongARM