ADSP-21000 Family Development Tools

ADDS-210xx-TOOLS

FEATURES

DEVELOPMENT SOFTWARE TOOLS

ASSEMBLER
- Easy-to-Use Algebraic Syntax

LINKER
- Combines Object and Library Files

ASSEMBLY LIBRARY/LIBRARIAN
- Includes Set of Arithmetic and DSP Functions

SIMULATOR
- Reconfigurable, MS Windows GUI Interface
- Full Symbolic Disassembly and On-Line Assembly
- Simulates Memory and Port Configurations
- Plots Memory Graphically

PROM SPLITTER

OPTIMIZING G21K ANSI C COMPILER
- Includes C-Callable Library of ANSI Standard and DSP Functions
- Supports In-Line Assembly Code

CBUG™ C SOURCE LEVEL DEBUGGER
- Integrated with Simulator and Emulator; Uses Same GUI Interface

C RUNTIME LIBRARY
- Includes Over 150 DSP and Mathematical Functions

DEVELOPMENT HARDWARE TOOLS

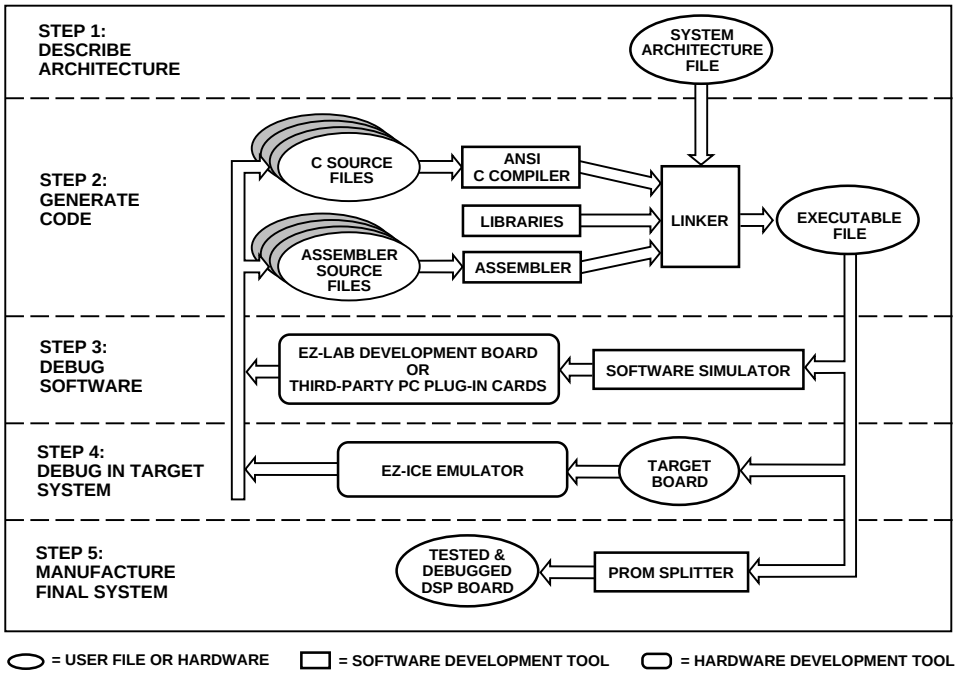
EZ-LAB® DEVELOPMENT BOARD
- Enables Evaluation, Prototyping, and Development of ADSP-21000 Family-Based Systems
- 16-Bit IBM-AT Compatible Plug-In Board

EZ-KIT
- Includes the EZ-LAB Development Board, Development Software Tools, and C Compiler and Runtime Library

EZ-ICE® EMULATOR
- Full Speed, In-Circuit Emulation
- 8-Bit IBM-PC/AT Compatible Plug-In Board with Small 11-Pin JTAG In-Circuit Probe

ICEPAC™ EMBEDDABLE IN-CIRCUIT EMULATOR
- Incorporates Embedded Emulation Functionality in a Plug-In Target Board (ADSP-2106x Only)

SYSTEM DEVELOPMENT DIAGRAM



REV. B

One Technology Way, P.O. Box 9106, Norwood. MA 02062-9106, U.S.A.
Tel: 617/329-4700 **Fax: 617/326-8703**

ADDS-210xx-TOOLS

INTRODUCTION TO DEVELOPMENT TOOLS

The ADSP-21000 Family Development Tools let you design applications for the ADSP-21000 family of Floating-Point DSP processors, including the ADSP-2106x SHARC. These tools enable you to develop hardware architectures together with creating and debugging code for your applications throughout the research, design, development, and test stages. The System Development Diagram illustrates how the tools work together. These tools are compatible with IBM-AT or Sun4 host platforms.

Components of the ADSP-21000 Family Development Tools fall into one of two broad categories: Software Tools and Hardware Tools.

Development Software Tools

- Assembler
- Linker
- Simulator
- PROM Splitter
- Assembly Library/Librarian
- Optimizing G21K ANSI C Compiler with Numeric C Extensions
- CBUG C Source-Level Debugger
- C Runtime Library

The Assembler translates ADSP-21000 Family assembly language source files into object code. The G21K C Compiler compiles C source files into object files or, optionally, into assembly language source files. The Linker then links the multiple object files together with various library files to form an executable program.

The ADSP-21000 Family Simulator runs the program on a software model of the DSP, reproducing the execution of the program by the processor in hardware. The simulator displays different portions of the virtual hardware environment through a reconfigurable windows interface identical to the emulator software interface.

Development Hardware Tools

- EZ-LAB Development Board
- EZ-ICE In-Circuit Emulator
- ICEPAC Embeddable In-Circuit Emulator (SHARC only)
- EZ-KIT and EZ-KIT Plus

The ADSP-2106x and ADSP-21020 EZ-LAB Development Boards are ready-to-run target system and evaluation platforms. They let you download and execute your ADSP-21000 family programs in real time. EZ-ICE, an in-circuit emulator, provides a controlled environment for observing, debugging, and testing by directly connecting to the target processor through its (IEEE 1149.1) JTAG interface. The ICEPAC, a small daughter card, incorporates embedded emulation functionality that effectively adds all of the capabilities of the EZ-ICE to your PC plug-in target board.

The EZ-KIT for the ADSP-2106x SHARC and the EZ-KIT Plus for the ADSP-21020 include the EZ-LAB Development Board, the ADSP-21000 Family Development Software, the C Compiler, C Runtime Library, and the CBUG Source Level Debugger.

Minimum Host Platform Requirements*

IBM-AT	Sun4
• 386-based or greater AT with 4 MB DRAM	• SunOS 4.1.1 for UNIX software; Windows version software compatible in Windows emulation mode
• DOS 3.1 or higher; Windows 3.1 or higher	• High resolution color monitor
• EGA or VGA Monitor and color video card	• 3.5" HD Floppy Disk Drive
• 3.5" HD Floppy Disk Drive	• Minimum 19 MB free hard disk drive space
• Minimum 11 MB free hard disk drive space	

*Some tools may vary.

SOFTWARE TOOLS

Assembler

The Assembler reads ADSP-21000 Family assembly language source files and generates a relocatable object file. It includes a preprocessor that lets you use the C preprocessor directives *#define*, *#include*, *#if*, *#ifdef*, and *#else* in assembly code. Assembler directives define code modules, data buffers, data variables, and memory mapped I/O ports. Both the assembler and C preprocessor have directives to define macros.

Programming in assembly language is eased by the highly readable algebraic syntax of the ADSP-21000 Family instruction set. An add instruction, for example, is written in the same manner as the actual equation: The algebraic statement r = x + y is coded in assembly language as (f0 = f1 + f2).

Linker

The Linker processes separately assembled object and library files to create a single executable program. It assigns memory locations to code and data according to user defined architecture files—text files that describe the memory configuration of the target system. The Linker generates symbols (variable names and program labels) in the processed files that are used by the simulator and emulator to perform symbolic debugging.

Assembly Library/Librarian

The Assembly Library contains standard arithmetic and DSP routines accessible to your programs. You can create libraries of your own functions using the Librarian tool.

Simulator

The Simulator, a software model of the DSP, provides instruction-level simulation of program execution. It models system memory and I/O according to the contents of the system architecture file, and displays hardware registers and memory in separate data windows (see Figure 1). The standard Windows Graphical User Interface (GUI) provides additional reconfigurable windows that display and let you alter register and memory contents, making a powerful debugging environment. The Simulator also reads symbols to perform symbolic debugging. A separate Simulator is provided for ADSP-2106x class DSPs and for ADSP-21020 class DSPs.

Features of the ADSP-21000 Family Simulators:

- Display of all registers, caches, and stacks
- Integration with CBUG C Source-Level Debugger
- Single step execution
- Interrupt simulation
- Plotting memory
- Break points and break conditions
- Simulation of program and data memory.

G21K ANSI C COMPILER

Known for its efficiency and reliability, the GNU-based Optimizing G21K C Compiler supports in-line assembly code, using the *asm*( ) construct, and generates COFF (Common Object Format Files), an industry standard file format for object, library, and executable files.

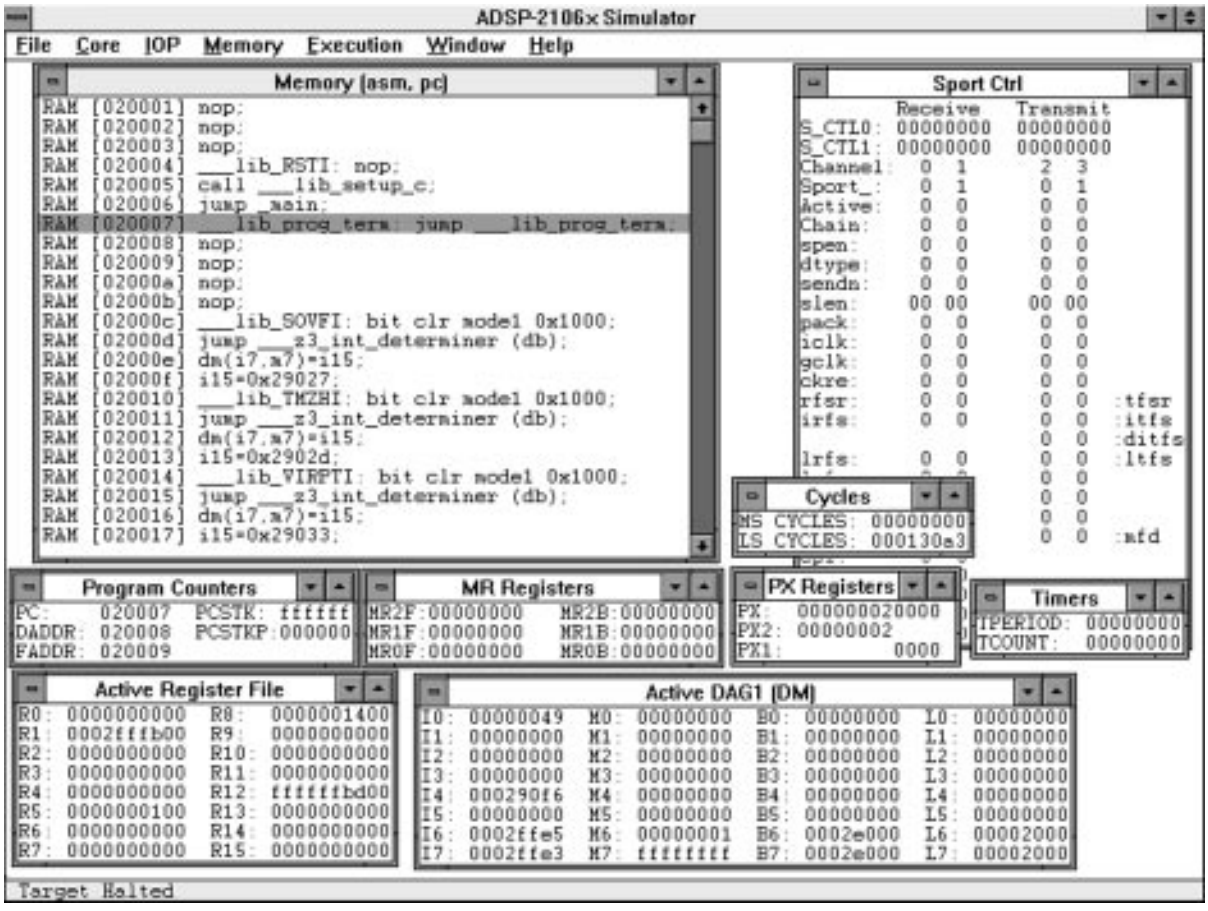


Figure 1. ADSP-21000 Family Simulator and Emulator User Interface

For code portability and development platform flexibility, the G21K C Compiler conforms to ANSI Standard X3J11. Its optimizing features include the following:
- Constant folding
- Common subexpression elimination
- Loop unrolling and strength reduction
- Global and local register allocation
- Flow Analysis
- Pattern combining
- Instruction scheduling
- Global and local register allocation
- Parallelization

NUMERICAL C
Numerical C—extensions to the G21K ANSI C compiler—requires fewer lines of code to perform vector and matrix operations. Developed with the ANSI Numerical C Extensions Group (NCEG), a working committee reporting to ANSI X3J11, Numerical C lets the compiler perform more powerful optimizations. These Numerical C extensions have been adapted by the Free Software Foundation for GNU C compilers (version 2.4).

Numerical C supports iterators used for generating one or more loops out of a single statement. This Numerical C code fragment for DSP application:

```
iter i = N;
A [i] = sin (2 × PI × i/N);
```

is equivalent to this C Code:

```
int i,_ilimit_I = N;
for (i=0; i<_ilimit_I; i++)
    {
    A[i] = sin(2*PI*i/N);
    }
```

A FIR filter with k taps on the array of n numbers notated as follows:

$$y_i = \sum_{j=0}^{k} x_{i-j} a_j$$

is coded as this equation:

```
iter I = n, j = k;
y[i] = sum (x [i-j] * a [j]);
```

CBUG C Source-Level Debugger
CBUG, a full-featured C source-level debugger, is fully integrated with the ADSP-21000 Family Simulator and EZ-ICE Emulator. It performs the following functions:
- Displays Variables and Expressions (Automatically Updated)
- Evaluates Standard ANSI C Expressions
- Stepping and Various other Program Execution Commands
- Breakpoints and Conditional Breaks Based on Expression Evaluation
- Displays Symbol Definition and Values
- Displays Function Calling Tree
- Uses Same Friendly User Interface as the EZ-ICE Simulator

C Runtime Library
The C Compiler has a set of ANSI-standard functions that ease program development. These library routines include ANSI-standard functions commonly used in digital signal processing. The table below contains all the C Runtime Library functions and macros that perform digital signal processing operations.

The ADSP-21000 Family Runtime Library includes the following ANSI-standard function categories:
- Standard Library
- Mathematics
- Signal, Variable and Character Handling

The library also includes signal processing functions in the following categories developed by Analog Devices:
- DSP Filters
- Fast Fourier Transforms
- Matrix Operations
- Interrupt Servicing

PROM Splitter
The PROM splitter translates an ADSP-21000 Family executable program into one of several formats for different PROM configurations or to be downloaded to the target system. The PROM Splitter's output file is generated as a Motorola S Record, Intel Hex Record format, or a stacked format used by the emulators.

Table I. Library Functions

a_compress	A-law compression
a_expand	A-law expansion
abort	abnormal program end
abs	absolute value
acos, acosf	arc cosine
asin, asinf	arc sine
atan, atanf	arc tangent
atan2, atan2f	arc tangent of quotient
atexit	register a function to call at program termination
atoi	convert string to integer
atof	convert string to long integer
autocoh	autocoherence
autocorr	autocorrelation
bsearch	perform binary search in sorted array
biquad	biquad filter section
cabsf	complex absolute value
calloc	allocate and initialize memory
ceil, ceilf	ceiling
cexpf	complex exponential
cos, cosf	cosine
cosh, coshf	hyperbolic cosine
cot, cotf	cotangent
crosscoh	cross-coherence
crosscorr	cross-correlation
div	division
exit	normal program termination
exp, expf	exponential
fabs, fabsf	absolute value
fir	finite impulse response (FIR) filter
floor, floorf	floor
fmod, fmodf	floating-point modulus
free	deallocate memory
frexp, frexpf	separate fraction and exponent
getenv	get string definition from operating system
histo	histogram
idle	execute ADSP-21020 IDLE instruction
ifftN	N-point inverse fast Fourier transform (IFFT)
iir	infinite impulse response (IIR) filter
interrupt	define interrupt handling
isalnum	detect alphanumeric character
isalpha	detect alphabetic character
iscntrl	detect control character
isdigit	detect decimal digit
isgraph	detect printable character, not including whitespace
islower	detect lowercase character
isprint	detect printable character
ispunct	detect punctuation character
isspace	detect whitespace character
isupper	detect uppercase character
isxdigit	detect hexadecimal digit
labs	absolute value
ldexp, ldexpf	multiply by power of 2
ldiv	division
localeconv	get pointer for formatting to current locale
log, logf	natural logarithm
log10, log10f	base 10 logarithm
longjmp	second return from setjmp
malloc	allocate memory
matadd	matrix addition
matmul	matrix multiplication
matscalmut	multiply matrix by scalar
matsub	matrix subtraction
mean	computes mean
memchr	find first occurrence of character
memcmp	compare objects
memcpy	copy characters from one object to another
memmove	copy characters from one object to another
memset	set range of memory to a character
modf, modff	separate integral and fractional parts
mu_compress	μ-law compression
mu_expand	μ-law expansion
poll_flag_in	test input flag
pow, powf	raise to a power
raise	force a signal
rand	random number generator
realloc	change memory allocation
rfftN	N-point fast Fourier transform
set_semaphore	sets semaphore value for ADSP-2106x
set_alloc_type	change memory allocation
set_flag	interface to the input flags of the ADSP-21020
setjmp	label for external linkage
setlocale	set the current locale
sgu	performs a stochastic gradient update on its input (ADSP-2106x)
signal	define signal handling (regular)
signalf	define signal handling (fast dispatchr) (ADSP-2106x)
signals	define signal handling (super dispatchr) (ADSP-2106x)
sin, sinf	sine
sinh, sinhf	hyperbolic sine
sqrt, sqrtf	square root
srand	random number seed
strcat	concatenate strings
strchr	find first occurrence of character in string
strcmp	compare strings
strcoll	compare strings
strcpy	copy from one string to another
strcspn	length of character segment in one string but not the other
strerror	get string containing error message
strlen	string length
strncat	concatenate characters from one string to another
strncmp	compare characters in strings
strncpy	copy characters from one string to another
strpbrk	find character match in two strings
strrchr	find last occurrence of character in string
strspn	length of segment of characters in both strings
strstr	find string within string
strtod	converts an ASCII string to floating point value (ADSP-2106x)
strtok	convert string to tokens
strtol	convert string to long integer
strtoul	convert string to unsigned long integer
strxfrm	transform string using LC_COLLATE
system	send string to operating system
tan, tanf	tangent
tanh, tanhf	hyperbolic tangent
test_and_set_semaphore	tests a value and sets a semaphore for the ADSP-2106x
timer_off	disable timer
timer_on	enable timer
timer_set	initialize timer
tolower	convert from uppercase to lowercase
toupper	convert from lowercase to uppercase
var	variance
zero_cross	count zero crossings

Table II. Library Macros

MIN(X,Y)	Returns the minimum of X and Y
MAX(X,Y)	Returns the maximum of X and Y
ABS(X)	Returns the absolute value of X and Y
SWAP(X,Y)	Swaps the values of X and Y
SUM(var, n, expr)	Returns $\sum_{var=0}^{n} \exp r$
FORALL(var, n, body)	Does body n times
FOREVER( )	ADSP-21020 idling
CIRCULAR_BUFFER(TYPE, DAGREG, name)	Circular buffer declaration macro
BASE(name)	Circular buffer register initialization macro
LENGTH(name)	Circular buffer register initialization macro
CIRC_READ(ptr, step, variable, memory)	Circular buffer access macro
CIRC_WRITE(ptr, step, variable, memory)	Circular buffer access macro
CIRC_MODIFY(ptr, step)	Circular buffer access macro
CLIP(X,Y)	Clip Y by Z: |Rx| < |Ry| then Rx, else if Rx < 0 then –Ry, else Ry
AVG(X,Y)	Returns (x+y)/2

The following provide direct access to actual assembly instructions in the ADSP-210xx Family

SCALB(X,Y)	Returns a value which is a scaled exponent of X added to the fixed-point twos-complement integer Y
MANT(X)	Returns the mantissa from the float X
LOGB(X)	Returns the conversion of the exponent of the float in X to an unbiased twos-complement fixed point integer
FIXBY(X,Y)	Returns the conversion of a floating point operand in X to a twos-complement 32-bit fixed point integer
FLOATBY(X,Y)	Returns the conversion of a fixed-point integer in X to a floating point; Y is a scaling factor which is added to the exponent of the result
RECIPS(X)	Returns an 8-bit accurate seed for 1/X
RSQRTS(X)	Returns a 4-bit accurate seed for 1/sqrt(X)
COPYSIGN(X,Y)	Returns the sign of the floating point value in Y copied to the floating point value in X without changing the exponent
BCLR(X,Y)	Returns a fixed point integer equal to X with the Y bits cleared
BTGL(X,Y)	Returns a fixed point integer equal to X with the Y bits toggled
BTST(X,Y)	Returns a fixed point integer equal to X with the Y bits set
EXP(X)	Returns an exponent of a fixed point integer
LEFTZ(X)	Returns the number of leading zeroes from the fixed point value X
LEFTO(X)	Returns the number of leading ones from the fixed point value X
LOG2I(a)	Calculate log base 2 of a number (int)

HARDWARE DEVELOPMENT TOOLS

Hardware tools for developing ADSP-21000 Family based products are divided into two categories: 1) for the ADSP-2106x SHARC, and 2) for the ADSP-21020 floating-point digital signal processors.

ADSP-2106x SHARC Development Tools

EZ-LAB Development Board Overview: The EZ-LAB Development Board is a 16-bit AT compatible plug-in board that lets you control and observe ADSP-2106x executable programs operating in real-time from on-board RAM. Optional processor and memory expansion modules from third parties let you customize the EZ-LAB. Several demonstration programs accompany EZ-LAB for you to familiarize yourself with and evaluate the ADSP-2106x floating-point DSPs. Figure 2 shows the EZ-LAB Board.

Platform Requirements: The ADSP-2106x EZ-LAB can draw power from the host PC or from an external power source when used in a stand-alone mode. The EZ-LAB's power requirements are: +5 V dc @ 1 A, +12 V dc @ 400 mA, and –12 V dc @ 400 mA. (Note: The use of any SHARCPAC or ICEPAC module will add to the power requirements.)

Memory: The ADSP-2106x EX-LAB is equipped with up to 512Kx8 of Boot PROM for program storage and up to 4 Mbits of SRAM on the DSP itself. This can be supplemented with additional memory through a memory expansion SHARCPAC™ module.

Expansion Connectors: The EZ-LAB has several expansion connectors. These include the MAFE™ expansion connector, the SHARCPAC module connectors, the SHARCNET™ connectors, and the JTAG in-circuit emulator connector.

The Modular Analog Front End (MAFE) connector provides a standard interface for real world analog interface I/O daughterboards. The SHARCPAC connectors provide an interface for optional SHARCPAC modules that may take on a multitude of special functions, including memory expansion and multiprocessing applications. The SHARCNET connectors let you access two link ports from the on-board DSP and to two link ports from the SHARCPAC module connectors. The JTAG connector provides an interface for an in-circuit emulator probe such as the EZ-ICE or the ICEPAC In-Circuit Emulator module.

MAFE Daughtercard: A sample analog interface MAFE daughtercard is supplied with the EZ-LAB. This card contains an Analog Devices AD1847-based sound codec and supporting hardware for audio input and output.

PC Interface: The EZ-LAB can be interfaced to a host PC by plugging the board into a 16-bit ISA expansion slot within the PC. The host PC has access to the on-board resources and the SHARCPAC expansion port through the ISA bus. Jumpers are used for address and interrupt selection, thus minimizing potential conflicts.

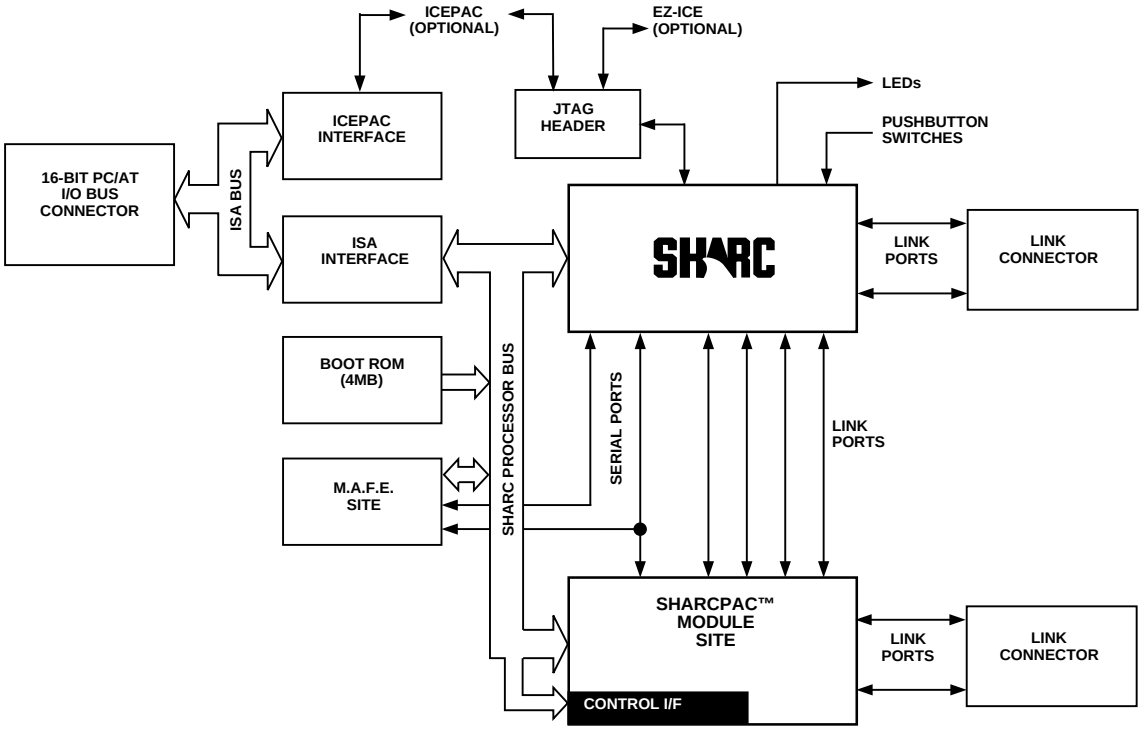


Figure 2. ADSP-2106x EZ-LAB Board

EZ-ICE Emulator

Overview

The ADSP-2106x SHARC EZ-ICE in-circuit emulator provides a controlled environment for observing, debugging and testing real-time activities in a target hardware environment by connecting directly to the target processor through its JTAG interface. The emulator monitors system behavior while running at full speed. It lets you examine and alter memory locations, including processor registers and stacks.

System Configuration Requirements

The EZ-ICE Emulator board is a half-size card that installs in an IBM PC's 8-bit expansion slot. The ICEPAC module is mounted as a daughter card to the EZ-ICE board. And the Test Access Port (TAP) probe is connected to EZ-ICE board through a ribbon cable, and to the target processor through its JTAG interface connector.

The following minimum PC configuration is required for the EZ-ICE:

- 386-based or greater AT with 4 MB DRAM
- EGA or VGA graphic card
- Hard disk with 2.5 MB available
- DOS 3.1 or higher; Windows 3.1 or higher
- An available slot for an 8-bit half-size card
- Mouse or other pointing device

Graphical User Interface

The EZ-ICE interface software uses the same GUI interface design as the simulator software. This same software works with the fully configured EZ-ICE board or with the ICEPAC installed in the EZ-LAB Development Board.

Nonintrusive In-Circuit Emulation

The EZ-ICE emulator does not affect target loading or timing. Nonintrusive, in-circuit emulation is assured because EZ-ICE controls the target system's processor through its IEEE 1149.1 (JTAG) Test Access Port.

EZ-ICE Target System Requirements

The ADSP-2106x SHARC EZ-ICE Emulators use the IEEE 1149.1 JTAG test access port of the processors to monitor and control the target board processor during emulation. The EZ-ICE TAP probe requires the CLKIN, $\overline{\text{EMU}}$, TMS, TCK, $\overline{\text{TRST}}$, TDI, TDO, and GND signals to be accessible on the target system via a 14-pin connector (pin strip header) such as that shown in Figure 3.

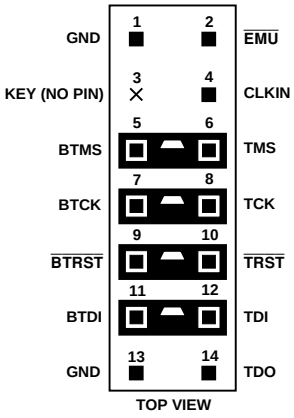


Figure 3. Target Board Connector for ADSP-2106x EZ-ICE (Jumpers in Place)

The EZ-ICE probes plug directly onto these connectors for chip-on-board emulation. You must add a JTAG connector to your target board design if you intend to use the EZ-ICE. It is possible to support multiprocessor SHARC systems using a single JTAG connector and EZ-ICE. Figure 4 shows the dimensions of the ADSP-21060 SHARC EZ-ICE TAP probe. Be sure to allow enough room in your system to fit the probe's cable connector onto the target's JTAG connector.

ADSP-2106x Emulator Connector Specification
The 2-row, 14-pin ADSP-2106x pin strip header is keyed at the Pin 3 location—you must remove Pin 3 from the header. The pins must be 0.025 inch square and at least 0.20 inch long. Pin spacing should be 0.100 × 0.100 inches. Pin strip headers are available from vendors such as 3M, McKenzie, and Samtec.

The length of the traces between the EZ-ICE probe connector and the processor's test access port pins should be as short as possible. Note that the EZ-ICE probe adds two TTL loads to the CLKIN pin.

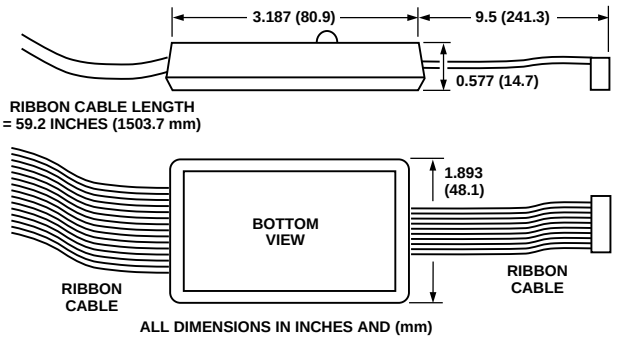


Figure 4. ADSP-2106x SHARC EZ-ICE TAP Probe

The BMTS, BTCK, $\overline{\text{BTRST}}$, and BTDI signals are provided so that the test access port can also be used for board-level testing. When the connector is not being used for emulation, place jumpers between the BXXX pins and the XXX pins as shown in Figure 5. If you are not going to use the test access port for board test, tie $\overline{\text{BTRST}}$ to GND and tie or pull up BTCK to V_{DD}. The $\overline{\text{TRST}}$ pin must be asserted (pulsed low) after power up (through $\overline{\text{BTRST}}$ on the connector) or held low for proper operation of the processor.

ICEPAC Embeddable In-Circuit Emulator
The ICEPAC is a small (business card size) daughter card that contains emulator-specific hardware that incorporates emulation functionality into a plug-in target board. With ICEPAC, you can use standard ADSP-2106x EZ-ICE software for full in-circuit emulation capability.

The ICEPAC interfaces to the PC Host through an 8-bit data bus, and to the target system through its (IEEE 1149.1) JTAG test access port. The ICEPAC connector is a superset of the standard EZ-ICE target board connector.

Using EZ-LAB and EZ-ICE Together
For the ADSP-2106x SHARC EZ-LAB Development System, the only additional component required to have in-circuit emulation is an ICEPAC module. The EZ-ICE board's interface and probe functions are built into the EZ-LAB System. Together, EZ-LAB and ICEPAC combine to form a high speed DSP workstation with an interactive, window-based debugging interface. This setup lets you develop and test your application without any additional time investment in hardware prototyping.

Combined Software and Hardware Packages
EZ-KIT Packages for the ADSP-21000 Family processors offer complete development tools sets at an affordable price.

SHARC EZ-KIT
In addition to the EZ-LAB Development Board, ADSP-2106x EZ-KIT contains the ADSP-2106x EZ-LAB and the ADSP-21000 Family Development Software: Simulator, Assembler, G21K C Compiler, C Source Level Debugger, Linker, Librarian, and PROM Splitter. Also included are abridged versions of software from SHARC Third Party developers. This package creates a complete development environment for programming applications in assembly language.

ADSP-21020 Development Tools
ADSP-21020 EZ-LAB Evaluation Board
Similar to the EZ-LAB SHARC Development Board, the ADSP-21020 EZ-LAB lets you control and observe ADSP-21020 programs executing in real-time from on-board RAM. The large memory space—up to 4 Mbits of SRAM—lets you develop high performance floating-point DSP applications. Unlike the EZ-LAB SHARC Development Board, the ADSP-21020 EZ-LAB is a stand-alone board that interfaces to the host computer through an RS-232 serial link. Several demonstration programs accompany EZ-LAB for you to familiarize yourself with and evaluate the ADSP-21020 floating-point DSPs.

Platform Requirements
The ADSP-21020 EZ-LAB requires a power supply that can deliver +5 V dc @ 1 amp and ± 12 V dc @ 200 mA.

Memory
The ADSP-21020 EZ-LAB contains 32K × 48-bit words of zero-wait state program memory and 32K × 48-bit words of zero-wait state data memory.

Expansion Connectors
Two expansion connectors let you add additional program memory, data memory, and I/O devices to customize the system. The 96-pin expansion connectors accept standard Eurocard prototyping boards (6U or 3U form factor).

PC Control
A host PC controls the ADSP-21020 EZ-LAB board through an RS-232 link. With this connection, you can download and run ADSP-21000 Family programs using interface software that runs on the PC. Program results may be uploaded from EZ-LAB's on-board memory to the host PC. For code debug, plug the ADSP-21020 EZ-ICE Emulator probe into the EZ-LAB's JTAG emulation connection.

Analog Interface
A basic analog interface is provided on-board, based on an AD1849 SoundPort Stereo Codec, for developing speech and audio processing applications. A 16-bit sigma-delta audio codec, the AD1849 integrates two sigma-delta DACs, two sigma-delta ADCs, anti-aliasing filters, digital interpolation filters, attenuators, and analog anti-imaging filters in a single package. It handles multiple channels of stereo input and output, and allows sampling rates from 8 kHz to 48 kHz.

The input signal to the AD1849 can come from a microphone (for speech processing), a signal generator, or any other high-impedance source. The processed signal is output through the standard on-board audio amplifier and a small speaker. Line-level I/O is also provided.

A 2-channel, 8-bit AD7769 general purpose analog interface lets the ADSP-21020 processor work with sampled analog signals. The AD7769 interfaces to the ADSP-21020 processor through four memory-mapped I/O ports (two for input, two for output).

EZ-ICE Emulator

The ADSP-21020 EZ-ICE provides a controlled environment for observing, debugging and testing activities in a target system by connecting directly to the target processor through its JTAG interface. The emulator monitors system behavior while running at full speed; it lets you examine and alter memory locations, including processor registers and stacks.

Nonintrusive In-Circuit Emulation

The emulator does not affect target loading or timing. Non-intrusive, in-circuit emulation is assured because EZ-ICE controls the target system's processor through its IEEE 1149.1 (JTAG) Test Access Port.

Graphical User Interface

The software provides a graphical user interface identical to the simulator's. The emulator connects to an IBM PC host computer through an 8-bit ISA bus plug-in board. *Note: This EZ-ICE ISA bus interface board is not compatible with the ADSP-2106x SHARC class of DSPs.*

System Configuration Requirements

To operate the ADSP-21020 EZ-ICE, you need the following minimum PC configuration:

- 386- or 486-based PC with 2 MB RAM (total)
- 2.5 MB free hard disk space
- graphics card
- DOS 3.1 or higher
- MS-Windows 3.1 or higher
- an available slot for an 8-bit half-size plug-in board
- mouse

EZ-ICE Target System Requirements

The ADSP-21020 EZ-ICE Emulator use the IEEE 1149.1 JTAG test access port of the processors to monitor and control the target board processor during emulation. The ADSP-21020/ADSP-21010 EZ-ICE probe requires the CLKIN, TMS, TCK, $\overline{\text{TRST}}$, TDI, TDO, and GND signals to be accessible on the target system via a 12-pin connector (pin strip header) such as that shown in Figure 5.

Target system boards must have the JTAG connector to interface to EZ-ICE's in-circuit probe. The EZ-ICE probe plugs directly onto this connector for chip-on-board emulation.

Figure 6 shows the dimensions of the ADSP-21020/ADSP-21010 EZ-ICE probe. Be sure to allow enough room in your system to fit the probe onto the connector.

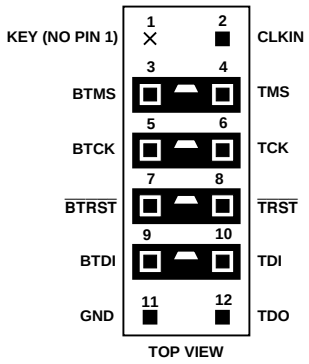


Figure 5. Target Board Connector for ADSP-21020 EZ-ICE (Jumpers in Place)

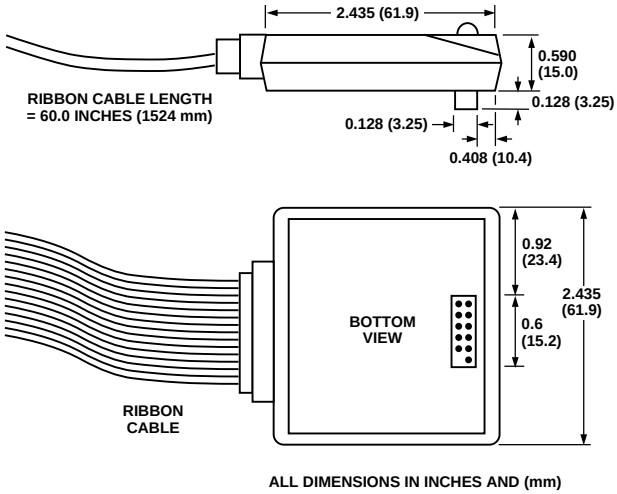


Figure 6. ADSP-21020 EZ-ICE Probe

ADSP-21020 EZ-ICE Probe Connector Specifications

The 2-row, 12-pin ADSP-21020/ADSP-21010 pin strip header is keyed at the Pin 1 location—you must remove Pin 1 from the header. Pin dimensions and availability are similar to those for the ADSP-2106x EZ-ICE TAP probe compatible header. However, the tip of the pins must be at least 0.10 inch higher than the tallest component under the ADSP-21020 emulator's probe to allow clearance for the bottom of the probe.

The length of the traces between the EZ-ICE probe connector and the processor's test access port pins should be as short as possible. Note that the ADSP-21020 EZ-ICE probe adds two TTL loads to the CLKIN pin.

The BMTS, BTCK, $\overline{\text{BTRST}}$, and BTDI signals are provided so that the test access port can also be used for board-level testing. When the connector is not being used for emulation, place jumpers between the BXXX pins and the XXX pins as shown in Figures 6. If you are not going to use the test access port for board test, tie $\overline{\text{BTRST}}$ to GND and tie or pull up BTCK to V_{DD}. The $\overline{\text{TRST}}$ pin must be asserted (pulsed low) after power up (through $\overline{\text{BTRST}}$) on the connector) or held low for proper operation of the processor.

Using EZ-LAB and EZ-ICE Together
Together, EZ-LAB and EZ-ICE combine to form a high-speed DSP workstation with an interactive, window-based debugging interface. In this configuration, EZ-LAB becomes the target system for EZ-ICE. From the EZ-ICE, you can download and execute programs, set breakpoints, and observe and change register and memory contents. The "Emulator port" on EZ-LAB lets EZ-ICE control the lab board's processor through its JTAG (IEEE 1149.1) interface.

Combined Software and Hardware Packages
The EZ-KIT Plus Package for the ADSP-21020 offers a complete development tool set.

EZ-KIT Plus
EZ-KIT Plus contains the ADSP-21000 Family Development Software, EZ-LAB Evaluation Board, the G21K ANSI C Compiler, the C Runtime Library, and the CBUG C Source-Level Debugger for a comprehensive set of tools for developing applications in C or in mixed C and assembly.

ORDERING INFORMATION

The ADSP-21000 Family Development Software is available for IBM-compatible PCs, and SUN4 platforms. Analog Devices offers a sales package: the SW package includes the ADSP-21000 Family simulator, systems builder, assembler, linker, C Compiler, C Runtime Library, CBUG C Source-Level Debugger and PROM splitter.

ADSP-2100 Family Ordering Guide

ADI Part Number	Description[1, 2]
ADDS-210xx-SW-PC	Assembler Tools, Simulator, and C Tools for the IBM-PC/AT
ADDS-210xx-SW-SUN	Assembler Tools, Simulator, and C Tools for the Sun4 Platform
ADDS-21020-EZ-LAB	ADSP-21020 EZ-LAB Evaluation Board
ADDS-21020-EZ-ICE	ADSP-21020 EZ-ICE Emulator
ADDS-21020-EZKITPL	ADSP-21020 EZ-LAB, Assembler Package, Simulator, and C Tools
ADDS-2106x-EZ-LAB	ADSP-2106x SHARC EZ-LAB Evaluation Board
ADDS-2106x-EZ-ICE	ADSP-2106x SHARC EZ-ICE Emulator
ADDS-2106x-ICEPAC	ADSP-2106x SHARC ICEPAC Embeddable In-Circuit Emulator[3]
ADDS-2106x-EZ-KIT	ADSP-2106x SHARC EZ-LAB, Assembler Package, Simulator, and C Tools

NOTES
[1]Assembler Tools = Assembler, Assembly Library/Librarian, Linker, PROM Splitter and Simulator.
[2]C TOOLS = G21K C Compiler, C Runtime Library, and CBUG C Source-Level Debugger.
[3]Available Fall 1995.

PRINTED IN U.S.A.

C1855a–10–7/95